Title
Numerical Continuation on a GPU for Kinematic Synthesis

Author
Glabe, Jeffrey

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Numerical Continuation on a GPU for Kinematic Synthesis

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Mechanical and Aerospace Engineering

by

Jeffrey Glabe

Dissertation Committee:
Professor J. Michael McCarthy, Chair
Assistant Professor Haithem Taha
Assistant Professor Edwin Peraza Hernandez
Professor David Reinkensmeyer

2020

DEDICATION

To all my friends and family, without whom this dissertation would not be possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Professor J. Michael McCarthy for his steadfast support and his extensive knowledge of kinematics. I would like to thank the developers of Bertini: Daniel Bates, Jonathan Hausenstein, Andrew Sommese and Charles Wampler for producing such a useful and large-scaled program. Their work was largely the inspiration for my research.

I would like to thank Professors Haithem Taha, David Reinkensmeyer, and Edwin Peraza Hernandez for their continual interest in my research. I thank my labmates: Mark Plecnik, Kaustubh Sonawale, Peter Wang, Brandon Tsuge, Yang Liu, Shramana Ghosh and Dina Abulon for the great conversation and input over the past four years.

Finally, I would like to thank the National Science Foundation who provided the funding for my Ph.D. studies.

This material is based on work supported by the National Science Foundation under Grant No. 1636017.

VITA

Jeffrey Glabe

EDUCATION

Doctor of Philosophy in Mechanical and Aerospace Engineering **2020**
University of California, Irvine *Irvine, California*

Master of Science in Mechanical and Aerospace Engineering **2018**
University of California, Irvine *Irvine, California*

Bachelor of Science in Chemistry **2008**
University of California, Davis *Davis, California*

RESEARCH EXPERIENCE

Graduate Research Assistant **2016–2020**
University of California, Irvine *Irvine, California*

Undergraduate Research Assistant **2006–2008**
University of California, Davis *Davis, California*

TEACHING EXPERIENCE

Teaching Assistant **2016–2020**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Numerical Continuation on a GPU for Kinematic Synthesis **2019**
Journal of Computing and Information Science in Engineering

A GPU Homotopy Path Tracker and End Game for Mechanism Synthesis **2020**
International Journal of Mechanisms and Robotic Systems

REFEREED CONFERENCE PUBLICATIONS

Six-bar Linkage Design System with a Parallelized Polynomial Homotopy Solver **2018**
Advances in Robot Kinematics Conference

Five Position Synthesis of a Planar Four-Bar Linkage **2019**
IFToMM World Congress

A GPU Homotopy Path Tracker and End Game for Mechanism Synthesis **2020**
USCToMM Symposium on Mechanical Systems and Robotics

SOFTWARE

Mechanism Generator `https://mechanicaldesign101.com/product/mechanism-generator-3-0/`
Windows program that interfaces with Solidworks to design planar and spherical four and six-bar mechanisms.

MechGen FG `https://mechanicaldesign101.com/product/mechgen-fg/`
iOS app that allows users to design a four-bar function generator.

MechGen SP `https://mechanicaldesign101.com/product/mechgen-sp/`
iOS app that designs double wishbone suspension linkage for a given camber gain and displacement profile.

ABSTRACT OF THE DISSERTATION

Numerical Continuation on a GPU for Kinematic Synthesis

By

Jeffrey Glabe

Doctor of Philosophy in Mechanical and Aerospace Engineering

University of California, Irvine, 2020

Professor J. Michael McCarthy, Chair

Kinematic synthesis, at its heart, involves finding the zero-dimensional solution set of a system of polynomials. The degree of these polynomials increases rapidly as more complex designs are considered. The computation time required to find the solution set has traditionally been the bounding factor for what can been achieved in kinematic synthesis. Homotopy continuation is typically used to find solutions to these polynomials. Homotopy continuation is itself an inherently parallelizable method. Graphics processing units (GPUs) were developed were developed with a structure that makes them optimal to solve problems in parallel. This dissertation explores the use of homotopy continuation running a GPU in order to decrease the computation time required for kinematic synthesis.

First, we discuss the development of an algorithm for homotopy continuation that is ideal to run on a GPU. The traditional path tracking algorithm is analyzed and then modified to better perform on a GPU. Additionally, the endgame methods are analyzed and the more ideal method is identified and implemented in CUDA. We outline the drawbacks of such modifications and discuss why they are admissible in the context of kinematic synthesis.

The implementation of the new homotopy continuation algorithm on a GPU is demonstrated by solving the four-bar linkage synthesis problem. This is the first development of a GPU-accelerated four-bar linkage design system. Novel (non-Burmester) loop equations are derived such that the entire mechanism is solved in one computation. These equations are then reduced and implemented into CUDA. The entire program is outlined and then demonstrated on a sample design problem. The results are compared with a similar CPU implemented system and a GPU speedup of around 120 times was observed.

The results of the four-bar linkage design system were then extended to the problem of six-bar linkage synthesis. A system was developed utilizing the same GPU-based path tracker and endgame. This has resulted in the first known GPU-accelerated six-bar linkage design system that presents new opportunities for multi-GPU systems capable of designing even more complicated mechanisms.

Chapter 1

Introduction

1.1 Kinematic Synthesis

Kinematica synthesis is the process of transforming a specific set of performance requirements known as a task into the dimensions of a mechanism that meets the specified requirements. There are three main types of kinematic synthesis, which vary in the tasks specified. Path generation focuses on finding a mechanism to guide an end effector through a series of specified task points. Motion generation extends path generation to include an orientation parameter at each of the task points. Finally, function generation seeks to coordinate pairs of input and output angles specified as the task.

The kinematic synthesis of four-bar linkages for five required task positions was formulated by Shoenflies [1] and Burmester[2], and and solved using graphical constructions. Freudenstein[3] formulated kinematic synthesis of these linkages using their loop equations which he solved using a computer–also see Erdman et al.[4] and McCarthy and Soh[5]. Wampler et al.[6] used polynomial homotopy to solve the synthesis equations to find a four-bar linkage with a coupler curve passing through nine points required the designer. Brake et al.[7] used polynomial homotopy to solve the path generation problem for four-bar linkages.

Computer-based dimensional synthesis provides effective tools for transforming a task requirement into linkage dimensions, see Glabe and McCarthy [8], Purwar et al. [9], Brake et al. [7]. However, if the number of task requirements is selected to completely specify the system of equations, then the resulting finite number of solutions may not yield useful designs—a useful design is a linkage that moves smoothly through the required task positions. McCarthy and Choe [10] demonstrated that 10,000 randomly selected five-position tasks had on average one task in 38 that yielded a useful four-bar linkage.

Synthesis techniques have additionally been extended to six-bar linkages. Denavit, Hartenberg [11], and McLarnan [12] formulated the loop equations for Stephenson II, Stephenson III, and Watt II six-bar linkages to solve the function generation problem for 8 input/output angles. Rao and Sandor [13] extended Freudenstein's work to algebraically synthesize six-bar linkages for 6 input/output angles. Dhingra et al[14] later were able to synthesize Watt II, Stephenson II, and III six-bar linkages utilizing homotopy continuation and burmester-ball theory. Plecnik and McCarthy [15, 16, 17, 18] used polynomial homotopy continuation and isotropic coordiantes to solve the six-bar synthesis problem.

Kinematic synthesis has also been applied to even more complex linkages. Soh et al [19, 20] were able to design eight-bar linkages by constraining a variety of problems. Sonawale and McCarthy [21] created eight bar linkages by constraining 6R loops. Mueller [22, 23] developed procedures for solving the function generation problem for eight and ten-bar linkages. Hamid and Soni [24] presented a hybrid approach of designing eight-bar linkages using motion, path, and function generation techniques.

This has lead to a computational synthesis strategy that starts with the task required by the designer and then searches nearby tasks for useful solutions, which can require thousands of iterations of the solution to the synthesis equations. For this reason, rapid solution of the synthesis equations is critical to effective linkage synthesis algorithms.

Computer-based dimensional synthesis requires finding the solution set of the loop equations parameterized by the task requirements. There are many ways of finding the solution set: Newton's method, interval analysis, sylvester resultant, groebner basis, and homotopy continuation to name a few. This dissertation focuses solely on the use of homotopy continuation as the method used to rapidly find the solution of the synthesis equations.

1.2 Homotopy Continuation

The kinematic synthesis of six, eight, or ten-bar linkages is limited by the size of the polynomial systems that must be solved to determine the link dimensions. Polynomial homotopy continuation systems, such as *Bertini* [25], PHCpack [26] and POLSYS-GLP [27], have demonstrated the ability to reliably solve large systems of linkage design equations, [28, 27, 29].

Homotopy continuation finds the solution set of a system of polynomials follows the known roots of a starting polynomial system as its coefficients are smoothy changed into the coefficients of the target polynomial system, Sommese and Wampler (2005)[30]. The process of following the transformation of these known roots into the roots of the target system is known as path-tracking, and can be parallelized for distributed computation, Su et al. (2006)[27].

The numerical solution of mechanism synthesis equations is found by transforming a start system of equations with known solutions into a target system equations can be traced to Freudenstein and Roth (1963)[31], who formulated the parameter-perturbation procedure to trace the set of known solutions of the start system to the find the solutions of the target system. They used this method to solve the nine-point path synthesis equations for a four-bar linkage, Roth and Freudenstein (1963)[32]. Wampler et al. (1992)[6] returned to this nine point path synthesis problem, and identified Roth and Freudenstein's approach as a type of numerical homotopy continuation.

Zangwill and Garcia (1981)[33] described a wide range of problems that can be solved by tracking the paths of a homotopy, such as nonlinear programming, economic equilibria, and game theory. Morgan (1983)[34] applied this approach to finding all of the solutions of a system of polynomial equations. Tsai and Morgan (1985)[35] used numerical polynomial continuation to solve for the inverse kinematics of a general 6R robot, which was was an important outstanding problem at the time. Additionally, homotopy continuation has been used to solve the forward kinematics problem for parallel robots [36, 37].

Since then numerical continuation has improved in capabilities, see Sommese et al. (2005). Now a variety of software packages for numerical continuation are available, such as Bertini[25] and PHCpack[38], HOM4PS[39] and POLSYS_GLP[27].

Polynomial homotopy has proven to be an important tool in the kinematic synthesis of six and eight-bar linkages. Plecnik and McCarthy[29, 40] obtain a range of new linkage designs by using polynomial homotopy to solve the synthesis equations obtained from the loop equations of a six-bar linkage. Plecnik[41] designed an eight-bar robot leg that achieved vertical straight-line movement with high mechanical advantage. The result was a high-performance jumping robot, see Haldane et al.[42]. More recently, homotopy continuation techniques for kinematic synthesis have been developed focusing only on tracking paths whose solutions are finite [43, 44].

One of the key properties of homotopy continuation is that there is no interdependence between the paths as they are being tracked. This means that the method is readily parallelizable. Graphics processing units, or GPUs, are known to be used to accelerate computations by executing steps in parallel. This makes the GPU an attractive host for performing homotopy continuation.

1.3 Graphics Processing Units

Graphics processing units (GPUs) were originally developed to accelerate rendering calculations in computer graphics, Parker(2017)[45]. These devices execute the same instruction set for each pixel in a display at very high speed. This capability has been deployed in other applications where identical sets of instructions are executed for a large number of cases. Examples are finite element analysis [46], genetic algorithms [47], computational fluid mechanics, Hori et al. (2011)[48], Verma et al(2019) [49], Lai et al (2019) [50], robot motion planning, Pan et al. (2010)[51] and Ichter et al. (2017)[52], and deformable body modeling for computer graphics, Dick et al. (2011)[53].

In the area of kinematics, GPUs have been used to accelerate the calculation of the inverse kinematics problem [54, 55, 56] for serial chain problems.

Concerning homotopy continuation, Verschelde and Yoffe [57, 58] introduced the use of a GPU for polynomial homotopy, using it to evaluate the polynomials and their derivatives with extended precision mathematics. The result was speeds of almost 20 times the speed of computation on a single CPU, but necessarily required polynomials with many mononomial terms to achieve sufficient speedup. Verschelde and Yu(2015)[59, 60] additionally used a GPU to offset the increased computation cost of of working with higher precision arithmetic, allowing for roots to be more accurately tracked and produce sharper solutions.

1.4 Contribution

This dissertation presents a novel method of homotopy continuation adapted specifically for a GPU for the probelm of kinematic synthesis, with a goal of decreasing computation time. While reasearch has previously been done on GPU acceleration of homotopy continuation

[59, 60], the focus was on completeness of solution set and enhanced precision. For the engineering problem of kinematic synthesis, these constraints can be relaxed to obtain increased computation speed.

Chapter 2 presents the mathematical background of the technique of homotopy continuation, focusing specifically on path tracking and endgame methods. Chapter 3 describes the hardware structure of a GPU, the GPU programming model as described by NVIDIA, and considerations to make when attempting to maximize performance of a program running on a GPU. Chapter 4 serves as a "how-to" guide in setting up a computer to perform complex GPU calculations as well as describing pitfalls to watch out for and debugging techniques.

Chapter 5 outlines one of the major contributions to this dissertation, which is developing an algorithm for homotopy continuation that is ideal to run on a GPU. The traditional path tracking algorithm is analyzed and then modified to better perform on a GPU. Additionally, the endgame methods are analyzed and the more ideal method is identified and implemented in CUDA. This section also discusses the drawbacks of such modifications and why they are admissible in the context of kinematic synthesis.

Chapter 6 presents the implementation of the new homotopy continuation algorithm on a GPU by solving the four-bar linkage synthesis problem. This is the first development of a GPU-accelerated four-bar linkage design system. Novel (non-Burmester) loop equations are derived such that the entire mechanism is solved for in one shot. These equations are then reduced and implemented into CUDA. The entire program is outlined and then demonstrated on a sample design problem. The results are compared with a similar CPU implemented system and a GPU speedup of around 120 times was observed.

Chapter 7 extends the findings of chapter 6 to the more complex six-bar (Watt 1) linkage design problem. The loop equations are derived and then algebraically reduced to form a simpler polynomial system. A six-bar design system is then developed in CUDA and C++.

The program is outlined and compared to a similar CPU implementation for the same design problem. This is the first known GPU-accelerated six-bar linkage design system.

Chapter 2

Mathematical Background

2.1 Homotopy Continuation

Numerical polynomial continuation obtains the roots of a system of n polynomials, $\mathcal{P}(\mathbf{x}) = (p_1, p_2, \ldots, p_n)$ in n unknowns $\mathbf{x} \in \mathbb{C}^{nx1} = (x_1, x_2, \ldots, x_n)$ by starting with a system $\mathcal{S}(\mathbf{x})$ that has the same total degree, M. The start system $\mathcal{S}(\mathbf{x})$ is constructed so that these roots have known values $\mathbf{y}_k$, $k = 1, \ldots, M$, that is

$$\mathcal{S}(\mathbf{y}_k) = \begin{Bmatrix} s_1(\mathbf{y}_k) \\ s_2(\mathbf{y}_k) \\ \vdots \\ s_n(\mathbf{y}_k) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \quad k = 1, \ldots, M. \tag{2.1}$$

The M roots $\bar{\mathbf{x}}_k$ of $\mathcal{P}(\mathbf{x})$ are obtained by smoothly transforming the roots of $\mathcal{S}$ into those of $\mathcal{P}$ and tracking the paths $\mathbf{v}_k(t) = (v_{1k}, v_{2k}, \ldots, v_{nk})$ as t varies from 1 to 0; the initial positions of the roots are $\mathbf{y}_k = \mathbf{v}_k(1)$ and their final positions are $\bar{\mathbf{x}}_k = \mathbf{v}_k(0)$. Note that while $\mathcal{S}(\mathbf{x})$ starts with M roots, the target system $\mathcal{P}(\mathbf{x})$ may have fewer than M roots. The

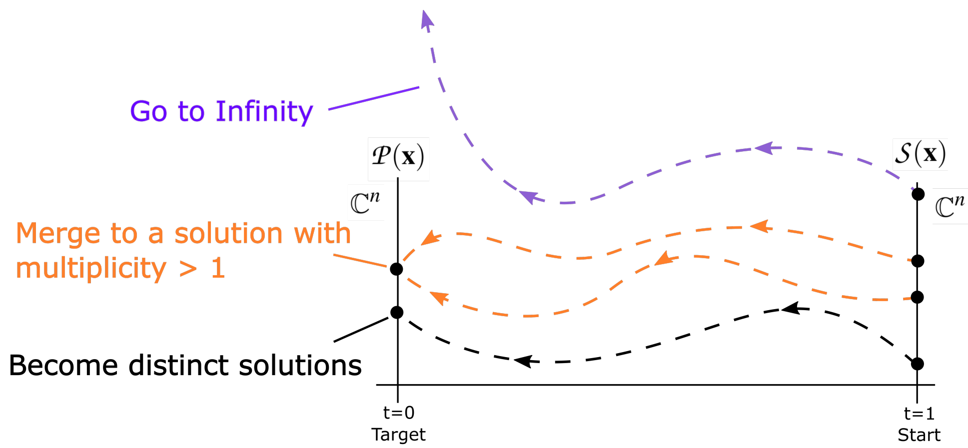


Figure 2.1: The three possible outcomes of paths during homotopy continuation.

number of roots is $\mathcal{P}(\mathbf{x})$ is upper bounded by M.

The homotopy $\mathcal{H}(\mathbf{x}, t)$ that transforms $\mathcal{S}$ into $\mathcal{P}$ is given by the polynomial system,

$$\mathcal{H}(\mathbf{x}, t) = \mathcal{S}(\mathbf{x})\, t + \mathcal{P}(\mathbf{x})\, (1 - t). \tag{2.2}$$

A path $\mathbf{v}_k(t)$ exists for each root of the start system and is a solution of the polynomial system $\mathcal{H}(\mathbf{x}, t)$, that is

$$\mathcal{H}(\mathbf{v}_k, t) = \begin{Bmatrix} s_1(\mathbf{v}_k)t + p_1(\mathbf{v}_k)(1-t) \\ s_2(\mathbf{v}_k)t + p_2(\mathbf{v}_k)(1-t) \\ \vdots \\ s_n(\mathbf{v}_k)t + p_n(\mathbf{v}_k)(1-t) \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix}, \quad k = 1, \ldots, M. \tag{2.3}$$

There are three possible outcomes for each path $\mathbf{v_k}(t)$:

1. The path can fail to converge and go to infinity.

2. The path can merge with another path at $t = 0$ to become a solution with multiplicity > 1.

3. The path can become a distinct solution at $t = 0$.

Figure 2.1 shows what each of these particular outcomes looks like. The path tracking also depends on the homotopy function $H(x, t)$ not having a root with multiplicity > 1 between $t = 1$ and $t = 0$. However, it can be shown using the Theorem of Sard, that if $\mathcal{S}(x)$ is chosen with random coefficients, then there is a very high probability that $H(x, t)$ will not have a root with multiplicity > 1 between $t = 1$ and $t = 0$. These coefficients are typically chosen from the unit circle, as numerical methods work better with coefficients near zero (this is because floating point numbers are more accurate near the origin).

2.1.1 Parameter Homotopy

If the number of start points equals the degree of $\mathcal{P}(x)$, then this is known as a total degree homotopy. In a total degree homotopy the whole start system is continuously transformed into the target system. However, if the structure of the two systems does not change (as often happens in engineering problems), then it suffices to instead only transform the coefficients between the start and target system. This method is known as the parameter homotopy, coefficient-parameter homotopy, or cheater's homotopy method.

Let p be the parameters of the target system, $\mathcal{P}(x; p)$ and q be the parameters of the start system, $\mathcal{P}(x; q)$. We can form a new homotopy distinct from Eq. (2.2),

$$\mathcal{H}(x; (p, q), t) = \mathcal{P}(\mathbf{q}\, t + \mathbf{p}\,(1 - t), \mathbf{x}). \tag{2.4}$$

Equation 2.4 moves the parameters q and p from start to finish, rather than the whole systems themselves. This distinction is important as it allows for the N solutions from a previous system to be used, rather than having to test all M paths as in a total degree for a degree M polynomial ($N << M$). This results in much fewer paths that have to be tracked.

Similar to total degree homotopy, care must be taken to ensure that there is no multiple root of $\mathcal{H}(x;(p,q),t)$ between $0 < t < 1$. For a parameter homotopy, the start parameters q are randomly chosen from the unit circle, and then a total degree calculation is performed to find the N roots of $\mathcal{P}(x;q)$. This total degree calculation is typically called the *Ab Initio* step and is a one-time calculation. For subsequent problems, the N roots form the paths that need to be tracked as the coefficients q are transferred to p.

2.2 Path Tracking

In order to describe a strategy to track these paths, consider the Taylor series expansion of the homotopy $\mathcal{H}(\mathbf{x},t)$ in $\mathbf{x}$ and t, given by

$$\mathcal{H}(\mathbf{x}+\Delta\mathbf{x}, t+\Delta t) = \mathcal{H}(\mathbf{x},t) + \frac{\partial \mathcal{H}}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathcal{H}}{\partial t}\Delta t + \text{higher order terms.} \tag{2.5}$$

Introduce the notation,

$$\mathbf{J}_{\mathbf{x}} = \frac{\partial \mathcal{H}}{\partial \mathbf{x}} \qquad \text{and} \qquad \mathbf{H}_t = \frac{\partial \mathcal{H}}{\partial t}, \tag{2.6}$$

where $\mathbf{J}_{\mathbf{x}}$ is an $n \times n$ matrix and $\mathbf{H}_t$ an $n \times 1$ vector with elements J_{ij} and H_i, respectively, given by

$$J_{ij} = \frac{\partial s_i}{\partial x_j}\, t + \frac{\partial p_i}{\partial x_j}\,(1-t), \quad H_i = s_i - p_i, \qquad i,j = 1,\ldots,n. \tag{2.7}$$

Now consider a point $(\mathbf{x},t)$ that is sufficiently close to a path $\mathbf{v}(t)$ of the homotopy, such that $\mathcal{H}(\mathbf{x}, t) \approx 0$. Then, we can predict a new point $\mathbf{p} = \mathbf{x} + \Delta\mathbf{x}$ at $t + \Delta t$ by setting $\mathcal{H}(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) = 0$ and solving for the first order terms of Eq. (2.5) to obtain

$$\mathbf{p} = \mathbf{x} - \mathbf{J}_{\mathbf{x}}^{-1}(\mathbf{x}, t)\mathbf{H}_t(\mathbf{x})\Delta t. \tag{2.8}$$

This is the Davidenko equation and the solution is known as the "prediction" step of the path-tracker and is achieved using a solver for ordinary differential equations.

If $\mathcal{H}(\mathbf{x}, t)$ is not sufficiently close to a path, then $\mathcal{H}(\mathbf{x}, t) \neq 0$, and we compute $\mathbf{c} = \mathbf{x} + \Delta\mathbf{x}$ for $\Delta t = 0$ such that $\mathcal{H}(\mathbf{x} + \Delta\mathbf{x}, t) \approx 0$. From Eq. (2.5), we obtain

$$\mathbf{c} = \mathbf{x} - \mathbf{J}_{\mathbf{x}}^{-1}(\mathbf{x}, t)\mathcal{H}(\mathbf{x}, t). \tag{2.9}$$

This is called the "correction" step of the path-tracker and is an example of Newton's method for root finding. Path-tracking executes a sequence of these prediction and correction steps to track the paths from $\mathcal{S}(\mathbf{x})$ to $\mathcal{P}(\mathbf{x})$. Figure 2.2 outlines how these prediction and correction steps keep the points along the solution curve $\mathcal{H}(\mathbf{x}(t), t) = 0$.

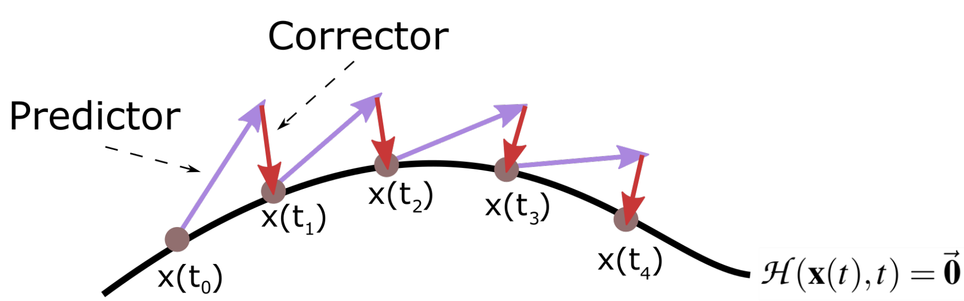


Figure 2.2: Prediction and correction steps executed to keep a path on the solution curve.

Newton's method is a popular choice for the correction step as it has quadratic convergence. However it should be noted that convergence depends on the predicted point $\mathbf{p}$ being in the convergence zone of the solution curve, which is inversely proportional to the degree of the polynomial system.

Path tracking depends on the Jacobian matrix $\mathbf{J}_{\mathbf{x}}$ not being ill-conditioned. Ill-conditioning in path tracking occurs when two paths near each other. Note that the two paths should not merge, as choosing a start system with random coefficients makes this very improbable as given by the theorem of sard. An ill-conditioned Jacobian matrix can introduce numerical error in calculating $\mathbf{p}$ as well as prevent the corrected point $\mathbf{c}$ from converging to the solution curve.

For high degree polynomial systems, paths nearing each other occurs as t approaches 0. This results in being unable to path-track all the way to 0 to find a solution. Instead, new methods to approximate the solution at $t = 0$ are necessary. This has produced a family of methods known as Endgames.

2.3 Endgame

For relatively low degree polynomial systems, it is possible to track the paths all the way to $t = 0$ without encountering an ill-conditioned Jacobian matrix. However as the degree of these polynomial systems increases, there is a high chance that the condition number will blow up as t approaches zero. This necessitates stopping path tracking at some end game boundary, $0 < R < 1$, and then switching to a different method to approximate a solution for that particular path.

All endgame methods operate on the fact that a solution $\mathcal{H}(\mathbf{x}^*, 0)$ is a point lying on a complex algebraic curve $\mathcal{H}(\mathbf{x}, t) = 0$. The endgame methods assume that the homotopy function $\mathcal{H}(\mathbf{x}, t)$ is a holomorphic function. A holomorphic function is a complex-valued function that is differentiable within a local neighborhood of every point.

The endgames work on the assumption that the solution curve $\mathcal{H}(\mathbf{x}, t)$ is the analytic image of a disk Δ_R of radius R in which the transformation $s = t^{\frac{1}{c}}$ yields a new system $h(x, s) := \mathcal{H}(\mathbf{x}, t^{\frac{1}{c}})$. By convention, this transformation sends $\mathbf{x}(t)$ to $\phi(s)$ and then endgame now works by finding $x^* = \phi(0)$. The number $c \in \mathbb{Z}$ is an important term called the cycle number or ***winding number***. Both endgame methods discussed in this dissertation will not work until the winding number is found. The winding number c is upper bounded by the multiplicity of the root that the current path is attempting to converge to. The multiplicity of the root ends up as the sum of the winding numbers of all paths converging to that root.

2.3.1 Operating Range

One important consideration is choosing an appropriate value of R. If R is too large, then the endgame methods may fail to converge to a solution as it is technically not in the local neighborhood of $\mathcal{H}(\mathbf{x}, 0)$. If R is too small, however, there is a good chance that the endgame will operate within the ill-conditioned Jacobian area near $t = 0$ and also fail to converge. This is partially because both Endgame methods use path tracking to a degree.

There are methods to determine whether or not one has entered the endgame operating range [61]. The simplest way is to track t from 1 to R and then conjecture that you are now in the correct operating range and proceed with the endgame. This method can work for problems where the structure of the polynomial system does not change and thus repeated performance can be expected.

An alternative is also to simply increase the precision of the path tracking. A Jacobian matrix with a condition number of 10^Y will lose roughly Y digits of accuracy when performing one iteration of Newton's method. Therefore, for a desired accuracy of k digits, the accuracy d can be increased such that $d - Y > k$. Increasing d effectively increase the size of the endgame operating zone at the cost of increased memory usage and the decreased computation speed resulting from using increased accuracy.

When in the proper endgame operating range, there are two endgame methods discussed

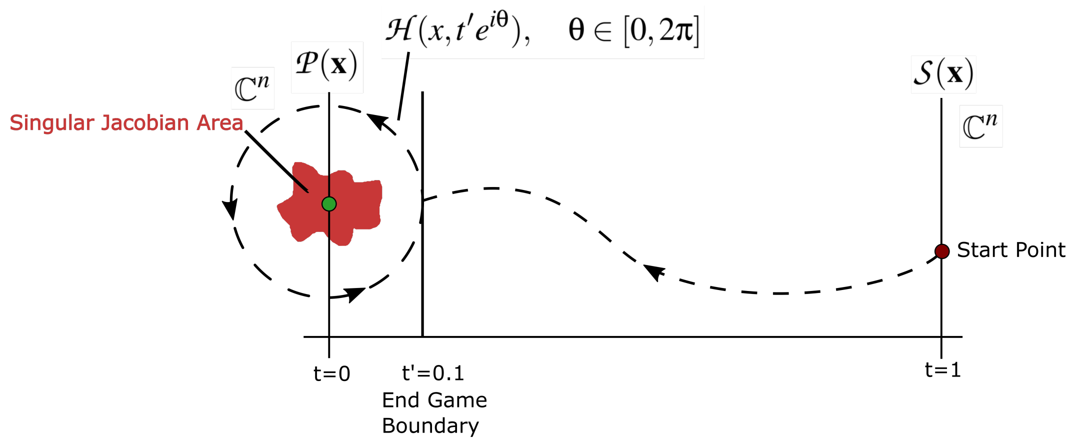


Figure 2.3: Using Cauchy's integral to approximate a solution.

to approximate a solution. These are the Cauchy's Integral method and the Power Series method.

2.3.2 Cauchy's Integral

The Cauchy's Integral method operates using Cauchy's integral formula from complex analysis. For the purposes of our endgame, Cauchy's integral formula states that the holomorphic function $\phi(s)$ discussed earlier is completely determined by the values on the boundary of the disk Δ_R, where R is the endgame boundary. That is, any point a in ϕ can be determined by,

$$\phi(a) = \frac{1}{2\pi i} \oint_{\Delta_R} \frac{\phi(s)}{s - a} \, \mathrm{d}s \tag{2.10}$$

Numerically this means first tracking the path to the endgame boundary $t = R$, then tracking around Δ_R by transforming $t = Re^{i\theta}, \theta \in [0, 2\pi c]$. The value $\mathcal{H}(\mathbf{x}^*, 0) = \phi(0)$ can then be

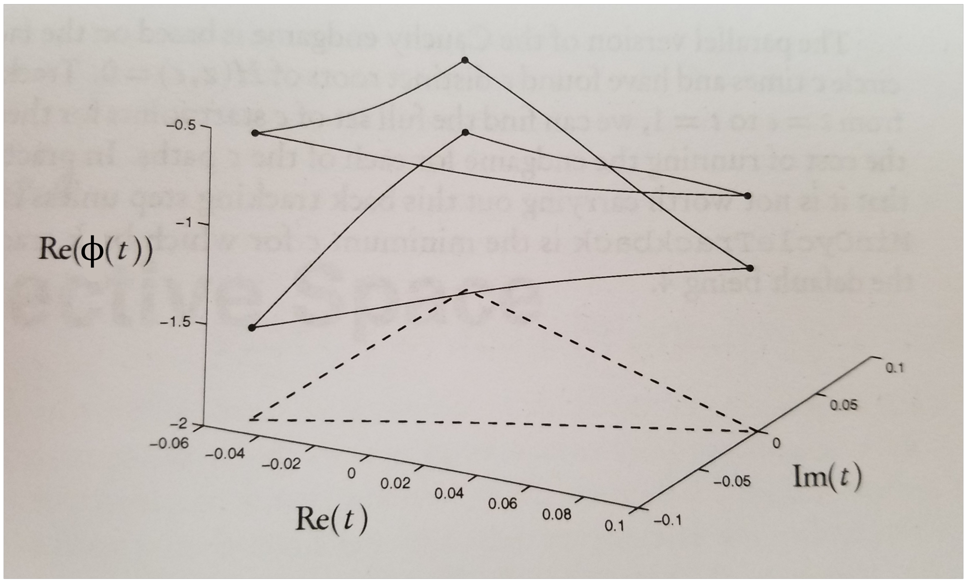


Figure 2.4: A path closing after tracking t around the triangular sampling path twice, resulting in a winding number $c = 2$.

found by,

$$\mathcal{H}(\mathbf{x}^*, 0) = \frac{1}{2\pi c}\int_0^{2\pi c} \mathcal{H}(\mathbf{x}, Re^{i\theta})d\theta \tag{2.11}$$

Figure 2.3 outlines this process.

Cauchy's integral formula requires that the curve defined by $\phi(Re^{i\theta})$ found while path tracking be closed. That is, the $\phi(Re^{i\theta}) = \phi(R)$. This means that the paths must be tracked around the disk Δ_R, stopping every time the path returns to the point $\theta = 2\pi j, \quad j = 1, 2, ...$ until the end point $\phi(Re^{i\theta j})$ is the same as the point found when beginning the endgame $\phi(R)$. When this occurs, then the winding number $c = j$ is known and the solution can be found using equation (2.11). The winding number cannot be found a priori.

Numerically, the integral can be computed by selecting a sample of M points around Δ_R at equal angle intervals, $s_k = Re^{2\pi i \frac{kc}{M}}$ and then finding the solution x^* as the average of the sample points,

$$x^* = \frac{1}{M}\sum_{k=1}^{M}\phi(s_k) \tag{2.12}$$

As path tracking works better when tracking in a straight line, an M-gon is fit to Δ_R and the sampling is taken at the verticies of the polygon. Figure 2.4 shows a path closing at $c = 2$ for $M = 3$ samples.

2.3.3 Power Series

The power series endgame method seeks to approximate $\phi(s)$ by a truncated power series. If $\phi(s)$ is a vector-valued function, then there exists a separate truncated power series for each row of ϕ. Therefore each element of x_j is its own power series $x_j^* + s^k(a_{j,0} + a_{j,1}s + ...)$, where $s = t^{\frac{1}{c}}$.

Similar to the Cauchy Integral method, the power series endgame also depends on finding the correct winding number in order for the truncated power series to converge to a solution. While the Cauchy method has to track paths in a circle around the origin until the path closes to find c, the power series method simply executes a series of guesses $c = 1, 2, ...c_{max}$ and tests them to see which is the proper winding number. This means that the power series method will not find a solution with a winding number $> c_{max}$.

When choosing c_{max} its important to consider the numerical precision the program is working with. When working with double precision, there is little reason to test c values > 4 or 5 due to the lack of accuracy.

The power series method first selects a sample $\kappa + 1$ t values $(t_1, t_2, ..., t_{\kappa+1})$, and then uses path tracking to find the corresponding path values $(\bar{\mathbf{x}}(t_1), \bar{\mathbf{x}}(t_2), ..., \bar{\mathbf{x}}(t_{\kappa+1}))$. Sample values are typically collected using the geometric sequence $(t_1, t_2, ...,) = (R, \lambda R, \lambda^2 R, ...)$ for some $\lambda \in (0, 1)$.

For each value c, the path values are transformed to

$$(\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_{\kappa+1}) = (\mathbf{t}_1^{1/c}, \mathbf{t}_2^{1/c}, ..., \mathbf{t}_{\kappa+1}^{1/c}). \tag{2.13}$$

We compute an interpolating polynomial $\bar{\phi}_c(\mathbf{s})$ to fit to the first κ values. The final value $\bar{\mathbf{x}}_{\kappa+\mathbf{1}}$ is predicted from $\phi_c(\mathbf{s}_{\kappa+\mathbf{1}})$ for each c. The c which minimizes $||\bar{\mathbf{x}}(t_{\kappa+1}) - \bar{\phi}_c(\mathbf{s}_{\kappa+\mathbf{1}})||$ is used to predict the solution $\mathbf{x}^* = \bar{\phi}_c(0)$.

2.3.4 Polynomial Interpolation

The power series endgame method uses polynomial interpolation to fit the interpolating function $\bar{\phi}_c(\mathbf{s})$ that maps s to $\bar{\mathbf{x}}$. There are many methods that can be used to express the interpolating polynomial, but this dissertation will go over two methods: Newton and Hermite interpolation.

Newton and Hermite interpolation are similar in the sense that Hermite interpolation is an extension of Newton interpolation to include not only s values but their derivatives ds as well.

Newton interpolation works by, given a table of s and $\bar{\mathbf{x}}(s)$ values such as,

s	s_0	s_1	...	s_κ
$\bar{\mathbf{x}}(s)$	$\bar{\mathbf{x}}(s_0)$	$\bar{\mathbf{x}}(s_1)$	...	$\bar{\mathbf{x}}(s_\kappa)$

Where both the s and $\bar{\mathbf{x}}(s)$ are distinct, Newtonian interpolation states that the interpolating polynomial can be expressed in the form,

$$\bar{\phi}_c(\mathbf{s}) = a_0 + a_1(s - s_0) + a_2(s - s_0)(s - s_1) + ... + a_\kappa(s - s_0)...(s - s_\kappa) \tag{2.14}$$

An important property of Newtonian interpolation is that the coefficients a_0, a_1, ..., $a_{\kappa-1}$ do not depend on κ. Alternatively, the next coefficient in the expression can be found using only the previous coefficient. This yields a systematic way of assigning coefficient values,

$$\begin{aligned}
\bar{\mathbf{x}}(s_0) &= a_0 \\
\bar{\mathbf{x}}(s_1) &= a_0 + a_1(s_1 - s_0) \\
\bar{\mathbf{x}}(s_2) &= a_0 + a_1(s_2 - s_0) + a_2(s_2 - s_1) \\
&...
\end{aligned} \tag{2.15}$$

or, equivalently,

$$\bar{\mathbf{x}}(s_k) = \sum_{i=0}^{k} a_i \prod_{j=0}^{i-1} (s_k - s_j) \tag{2.16}$$

for $0 \leq k \leq \kappa$. Solving (2.15) for these coefficients therefore results in,

$$\begin{aligned}
a_0 &= \bar{\mathbf{x}}(s_0) \\
a_1 &= \frac{\bar{\mathbf{x}}(s_1) - a_0}{(s_1 - s_0)} = \frac{\bar{\mathbf{x}}(s_1) - \bar{\mathbf{x}}(s_0)}{(s_1 - s_0)} \\
a_2 &= \frac{\bar{\mathbf{x}}(s_2) - a_0 - a_1(s_2 - s_0)}{(s_2 - s_0)(s_2 - s_1)} = \frac{\frac{\bar{\mathbf{x}}(s_2)-\bar{\mathbf{x}}(s_1)}{s_2-s_1} - \frac{\bar{\mathbf{x}}(s_1)-\bar{\mathbf{x}}(s_0)}{(s_1-s_0)}}{s_2 - s_0}
\end{aligned} \tag{2.17}$$

the terms $\frac{\bar{\mathbf{x}}(s_1)-\bar{\mathbf{x}}(s_0)}{(s_1-s_0)}$ as well as the expression for a_2 are known as the **divided difference** and is crucial to finding the Newton form of the interpolating polynomial. By traditional convention, the notation,

$$a_k = \bar{\mathbf{x}}[s_0, s_1, ..., s_k] \tag{2.18}$$

refers to the divided difference $\bar{\mathbf{x}}[s_0, s_1, ..., s_k]$ of order k. Finding an expression for $\bar{\mathbf{x}}[s_0, s_1, ..., s_k]$ can be found by solving (2.16) for a_k,

$$\begin{aligned}
\bar{\mathbf{x}}(s_k) &= a_k \prod_{j=0}^{k-1}(s_k - s_j) + \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1}(s_k - s_j) \\
\Rightarrow a_k &= \frac{\bar{\mathbf{x}}(s_k) - \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1}(s_k - s_j)}{\prod_{j=0}^{k-1}(s_k - s_j)}
\end{aligned} \tag{2.19}$$

By way of (2.18), this yields,

$$\bar{\mathbf{x}}[s_0, s_1, ..., s_k] = \frac{\bar{\mathbf{x}}(s_k) - \sum_{i=0}^{k-1} a_i \prod_{j=0}^{i-1}(s_k - s_j)}{\prod_{j=0}^{k-1}(s_k - s_j)} \tag{2.20}$$

Newtonian interpolation requires filling out a table of these divided differences to find the coefficients of the interpolating polynomial. The interpolating polynomial can be found by substituting a_i in (2.16) with $\bar{\mathbf{x}}[s_0, s_1, ..., s_i]$.

Hermite interpolation extends Newtonian interpolation to include derivatives as well as the values of s. If the first k derivatives can be computed for a particular s, then s is equivalent to $k+1$ points without derivatives when computing the power series expansion. This is attractive for the endgame, as the first derivative $\frac{dx}{ds}$ can be easily found at each point. Additionally, if too many sample points are needed, then there is a high risk of entering the ill-conditioned zone which would cause the endgame to fail.

In terms of the endgame, the derivative $\frac{dx}{ds}$ can be computed by first computing $\frac{dx}{dt} = -\mathbf{J_x}^{-1}\mathbf{H_t}$ (from 2.8) and then, using the chain rule,

$$\frac{dx}{ds} = \frac{dx}{dt}\frac{dt}{ds} = (\frac{dx}{dt})cs^{c-1} \tag{2.21}$$

Note that $t = s^c$. Hermite interpolation extends the sample table used to calculate the divided differences by repeating the sample points k times before calculating the differences. Naturally, this violates the unique values requirement which would result in dividing by zero in (2.20) when $s_k = s_j$. However, when this would occur, then the value in the divided difference table is replaced instead by $\frac{dx}{ds}$.

Regardless of the method used, the value of $\bar{\mathbf{x}}(s_{\kappa+1})$ and $\bar{\mathbf{x}}(0)$ can be found by using (2.16) once the coefficients of the divided difference table have been found.

2.3.5 De Moivre's Theorem

Since computing powers of complex numbers is not the same as computing powers of real numbers a discussion of how to compute $s = t^{1/c}$ is warranted. Named after French mathematician Abraham de Moivre, he discovered that, given a complex number $z = a + bi$, it can be converted to polar form $z = r(cos\theta + isin\theta)$, where,

$$r = \sqrt{a^2 + b^2} \qquad\qquad \theta = atan(b/a) \tag{2.22}$$

It follows then that squaring z results in.

$$\begin{aligned} z^2 &= (r(cos\theta + isin\theta))^2 \\ &= r^2(cos\theta + isin\theta)^2 \\ &= r^2(cos^2\theta + isin\theta cos\theta + icos\theta sin\theta + i^2 sin^2\theta) \\ &= r^2((cos^2\theta - sin^2\theta) + i(sin\theta cos\theta + sin\theta cos\theta)) \\ &= r^2(cos2\theta + isin2\theta) \end{aligned} \tag{2.23}$$

It can be shown that De Moivre's theorem further generalizes to the n-th degree as,

$$z^n = r^n(cos(n\theta) + isin(n\theta)) \tag{2.24}$$

This provides a convenient way of calculating $t^{1/c}$ during the endgame methods.

Chapter 3

GPU Hardware

In this section we outline the basic hardware that makes up a GPU. This section references the CUDA C Programming Guide [62] published by NVIDIA.

3.1 Multiprocessors

A GPU is made up of an array of streaming multiprocessors (SM). Each multiprocessor is designed to run hundreds of threads concurrently. The execution of the threads is managed by what is known as Single Instruction, Multiple-Thread (SIMT) architecture. Under SIMT architecture, each SM executes a group of 32 threads known as a warp (a term originating from the weaving of threads into fabric). The SM processes one warp at a time using a warp scheduler. Figure 3.1 outlines the overall structure of a GPU.

Each SM is further divided into an array of Streaming Processors (SP). Each SP executes instructions on data stored in the register file. The register file is a set of local memory (each 32 bits in size at the time of this dissertation) that makes up the workspace for the SM. Data is loaded into the register, operated on, and then sent out to other memory locations

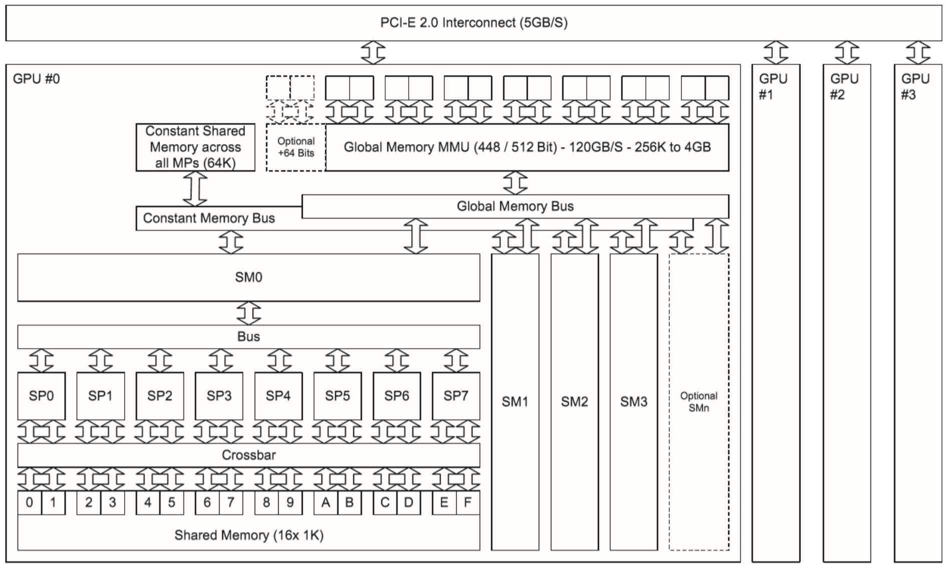


Figure 3.1: Overall architecture of a GPU showing the array of Streaming Multiprocessors (SM).

(L1/L2 Cache, global memory, etc).

3.2 Instruction Sets and Branch Divergence

CUDA code is converted by the CUDA compiler into PTX (Parallel Thread eXecution) machine level code analogous to assembly code on CPUs. The PTX code forms what is known as the instruction set, which is the machine level commands that each warp is meant to execute. Examples of these commands are ADD, COMPARE, SET, etc.

Branch divergence occurs whenever two threads have to execute different instruction sets. This commonly occurs wherever there are data-driven conditional statements such as (IF, ELSE, WHILE, etc). One of the big differences between CPUs and GPUs is how they handle branch divergence. A lot of CPU technology revolves around handling branch divergence: branch prediction, out of order execution, etc. A GPU forgoes this technology in order to maximize the number of threads that can be executed.

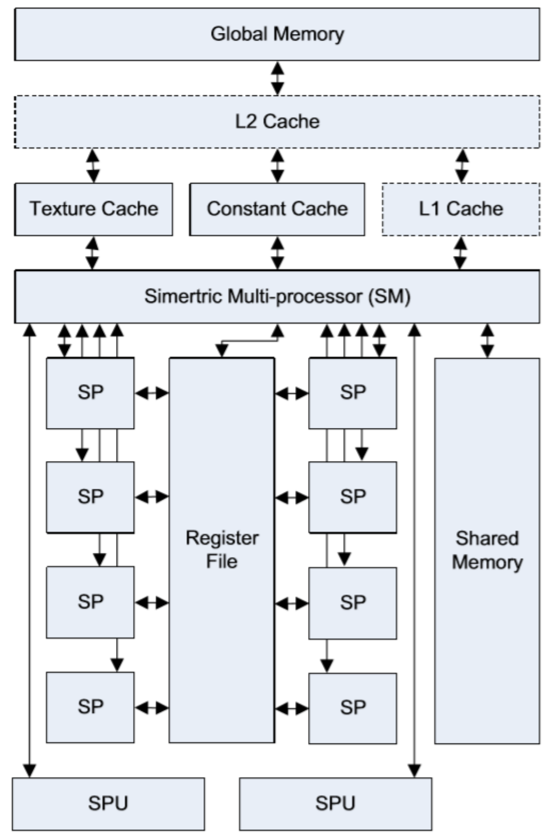


Figure 3.2: Architecture of a streaming multiprocessor.

The result is that while branch divergence results in minimal performance loss on a CPU, it can cause massive performance degradation on a GPU.

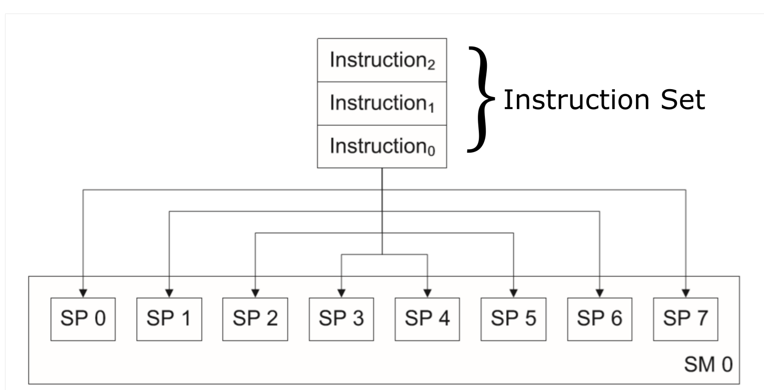


Figure 3.3: An SM broadcasting instructions from an instruction set to a warp.

On a GPU, an SM broadcasts one common instruction at a time to each warp. When any branch divergence occurs, the SM will simply execute both branches serially one at a time.

While the instruction set associated with one branch is being executed, the threads that are not on that path are disabled. This has the effect of under-utilizing the amount of active threads on a GPU at any point in time. On a CPU, branch divergence does not have the effect of blocking other threads.

Conditional statements are useful and necessary however, so when writing CUDA code they should not be avoided all together. Instead, it is important to consider the size of the instruction sets associated to the different paths of a diverged branch and attempt to keep it to a minimum to maximize GPU performance.

3.3 GPU Program Structure

CUDA follows that NVIDIA calls a heterogeneous programming model. This consists of the GPU, known in CUDA as the ***device*** acting as a coprocessor to the CPU, known in CUDA as the ***host***. The host runs either C or C++ code and then calls the CUDA code in what is known as a kernel call. The programming model assumes both the host and device have their separate memory spaces, known as host memory and device memory respectively. NVIDIA has provided a CUDA runtime API to use in the host code to manage device memory from the host.

A typical CUDA program begins with the host initializing all the data in the host memory. Device memory is then allocated using the *cudaMalloc* function from the CUDA runtime API. Once the device memory has been successfully initialized, the host then transfers the data to the device using the *cudaMemcpy* function. The host code then calls the device kernel function and the GPU then executes the code on the data stored in device memory. Once the device has finished modifying the data, the results can be transferred back to the host using the same *cudaMemcpy* function.

It is important to mention that device memory is not reset between kernel calls. Therefore it is possible to have multiple kernel calls one after the other without having to worry about the memory state between them. Contrasting with a CPU, the stack memory is not guaranteed to persist through multiple function calls.

When executing a kernel call, the programmer must specify two additional parameters: the number of blocks and the number of threads per block. The device will then enumerate these blocks and distribute them to the multiprocessors. The threads of the block then run in parallel on the multiprocessor (one warp at a time) until the code is completed. If more blocks are available, they run on the completed multiprocessor until no blocks remain.

3.4 Maximizing GPU Performance

When considering GPU performance, the CUDA C Programming guide [62] outlines three basic strategies:

1. Maximize GPU utilization.
2. Maximize memory throughput.
3. Maximize instruction throughput.

Choosing which strategy to pursue depends on the problem at hand and where the bottleneck is. Attempting to maximize GPU utilization by ensuring all SMs have blocks of threads to process will not work if branch divergence is causing most of the warps to be executing instructions on only a few of the 32 threads at a time.

3.4.1 GPU Utilization and Occupancy

The amount of GPU utilization at any point of time is directly proportional to the number of active warps. The number of warps that can be processed on a multiprocessor depends on the amount of registers and shared memory available. If a multiprocessor lacks enough registers or shared memory to execute at least one block, the kernel will fail to launch. During compilation, one can view the amount of registers and shared memory used by a kernel function by compiling with the -ptxas-options=-v option.

Nvidia has coined the term *occupancy* to describe how many resident warps are being processed at any time. For maximum GPU utilization, it is important to choose a number of block and number of threads per block that achieves full occupancy. It is difficult to predict what number of blocks and threads per block will result in maximum performance, as those numbers depend heavily upon the kernel code. At the time of this dissertation, NVIDIA has released a spreadsheet to help developers estimate which settings will maximize occupancy.

3.4.2 Memory Throughput

Following CUDA's heterogeneous programming model, the device will always receive its data directly from the host. This transfer of data from the host to device is much slower than the transfer of data between a device and the L1/L2/Shared memory caches of a multiprocessor. Therefore it is imperative to minimize the amount of data transfer between the host and device.

Developer wise this means moving as much code as possible from the host to the device, even if the code isn't particularly parallelizable. This has the effect of decreasing the frequency of data transfer between host and device.

On the device level, maximizing memory throughput deals with attempting to store the most used variables in as low a memory level as possible. This means having variables stored in the L1/L2 cache or shared memory as opposed to the global memory which takes a lot longer to read and write.

3.4.3 Instruction Throughput

Maximizing instruction throughput come in two forms: using arithmetic instructions with high throughput and minimizing the number of divergent warps by avoiding the use of control flow instructions, when possible.

Arithmetic instructions (such as multiplying and adding) typically have a higher throughput when associated with lower precision. This means using floating point numbers instead of doubles, when possible. The CUDA C Programming guide has a table which details the throughput of various operations at different precision levels.

Control flow instructions are those that can cause branch divergence (IF, ELSE, WHILE, DO, FOR, etc). As discussed in section 3.2, the GPU handles paths that diverge by simply executing each path serially and having threads that are not following the path currently being executed idle. Idle threads are not executing any instructions, so maximizing instruction throughput corresponds to minimizing branch divergence by avoiding control flow instructions when possible.

Chapter 4

GPU Implementation

This section serves as a "how to" guide on how to set up a computer to perform GPU calculations. This process is not entirely straightforward and there are some pitfalls to be careful to avoid. This section addresses those issues and how to fix them.

4.1 Setup and Development Environment

Before considering writing CUDA code, it is important to verify that one has a CUDA capable GPU. Any graphics card that is listed in `http://developer.nvidia.com/cuda-gpus` has been deemed by NVIDIA to be a CUDA capable GPU.

The development platform chosen depends on the operating system and availability of different development software packages. All of the CUDA code presented in this dissertation was written on the Windows 10 operating system. NVIDIA's website refers to Microsoft's Visual Studio `https://visualstudio.microsoft.com/` as the chosen development environment for Windows. It is important to note that the CUDA toolkit is only available from the Enterprise or Professional version of Visual Studio, not the free Community version.

Visaul Studio Enterprise was chosen for this research, it was available for free through the University of California, Irvine.

After installing an appropriate version of Visual Studio, the next step is to download the CUDA toolkit installer `https://developer.nvidia.com/cuda-downloads` from NVIDIA's website and install the CUDA toolkit. As of writing this dissertation, CUDA has three versions available: 8,9, and 10. The four-bar synthesis code was developed in CUDA 8, while the six-bar synthesis code was developed using CUDA 10. This was because the CUDA 8 compiler was unable to compile the necessary code for six-bar synthesis, due to its complexity, while the CUDA 10 compiler was. Additionally, as of the time of this dissertation, CUDA versions are backwards compatible so code written for CUDA 8 can still be compiled using a CUDA 10 compiler.

The CUDA toolkit comes with many samples one can use to verify the successful toolkit installation. One particularly useful program located with the samples is the *deviceQuery* function. After compiling the device query project in Visual Studio, one can run the *deviceQuery.exe* executable. Figure 4.1 shows the output of the function executed on the PC used to develop this dissertation's research.

The *deviceQuery.exe* executable lists the details of all CUDA capable devices it detects. There are many important things to note from the output. Some important things to note includes the device number (if there are multiple GPUs on the same PC), the CUDA capability version, the number of multiprocessor and maximum threads per block, the number of registers available per block, and the total amount of global memory.

If you can compile the device query sample and *deviceQuery.exe* shows at least one device in its output, then you should have everything you need to begin developing CUDA code. This leads naturally into the first pitfall of GPU programming: the watchdog timer.

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v8.0\bin\win64\Debug>deviceQuery.exe
deviceQuery.exe Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 2 CUDA Capable device(s)

Device 0: "Quadro M2000"
  CUDA Driver Version / Runtime Version          10.1 / 8.0
  CUDA Capability Major/Minor version number:    5.2
  Total amount of global memory:                 4096 MBytes (4294967296 bytes)
  ( 6) Multiprocessors, (128) CUDA Cores/MP:     768 CUDA Cores
  GPU Max Clock rate:                            1163 MHz (1.16 GHz)
  Memory Clock rate:                             3303 Mhz
  Memory Bus Width:                              128-bit
  L2 Cache Size:                                 786432 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  CUDA Device Driver Mode (TCC or WDDM):         WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA):      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 2 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

Figure 4.1: Sample output of the device query executable.

4.2 Watchdog Timer

Generally speaking, watchdog timers are used in operating systems to recover from faults. They are useful in having systems automatically recover from problems and return to a safe state. Concerning GPUs, the watchdog timer checks any function running on the ***display*** GPU and kills the function if it does not return within a specified number of seconds. A display GPU is any GPU that is connected to a display device (such as a computer monitor) by a cable. On Windows operating systems the limit for display devices is two seconds. The watchdog timer is intended to prevent display hangups from freezing the screen for an extended period of time. The CUDA code presented in this these takes minutes to hours to run, depending on the size of the problem. Therefore the watchdog timer must be bypassed.

The watchdog timer can be bypassed by executing a function on a GPU that is not connected to any display. This means installing a second GPU to the computer and then running the code on that. The *cudaSetDevice()* function can be used to specify which device to execute the code on, using the device number found from the output of *deviceQuery.exe*.

4.2.1 Adding a Second GPU

Adding a second GPU to a computer is a relatively straightforward process. To begin, first decide whether the second GPU you are purchasing is going to be the display GPU or the compute GPU. The display GPU does not need to be particularly powerful, so a cheaper GPU can be used. It is recommended that you purchase another NVIDIA GPU to simplify device driver installation.

Before purchasing the GPU however, verify that your system has an available slot and space for multiple GPUs. At the time of writing this dissertation, most GPUs require a PCI Express 3.0 slot, though this will likely be different in the future. The Lenovo P-series workstations used for this dissertation research had slots for up to four GPUs.

After purchasing the GPU it can be installed in a free slot on the motherboard of the PC. It is important to verify that the GPU intended to be the display GPU is the one that is connected to the display devices. After installing the new GPU, the ”display adapters” tab in Windows' Device Manager should show two devices. Next, install the latest driver from NVIDIA that is compatable with **both** devices. NVIDIA drivers can be found at `https://www.nvidia.com/Download/index.aspx`.

However, even adding a second GPU on Windows isn't necessarily enough to begin running complex CUDA code. This is because Windows specifically uses a system called the Time-out Detection and Recovery (TDR). The TDR system will still monitor and kill any GPU function lasting more than two seconds, even if the function is is not running on the display device. To run more extensive GPU code on Windows, this system must be modified.

4.2.2 Modifying the Watchdog Timer on Windows

Modifying the TDR system on Windows requires editing the Windows registry. To do this, use the Registry Editor app that comes with Windows 10. Using the app, locate the

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\GraphicDrivers`

directory within the registry, right click, and select New > DWORD(32 bit) value. Figure 4.2 shows the adding of a new DWORD to the registry.

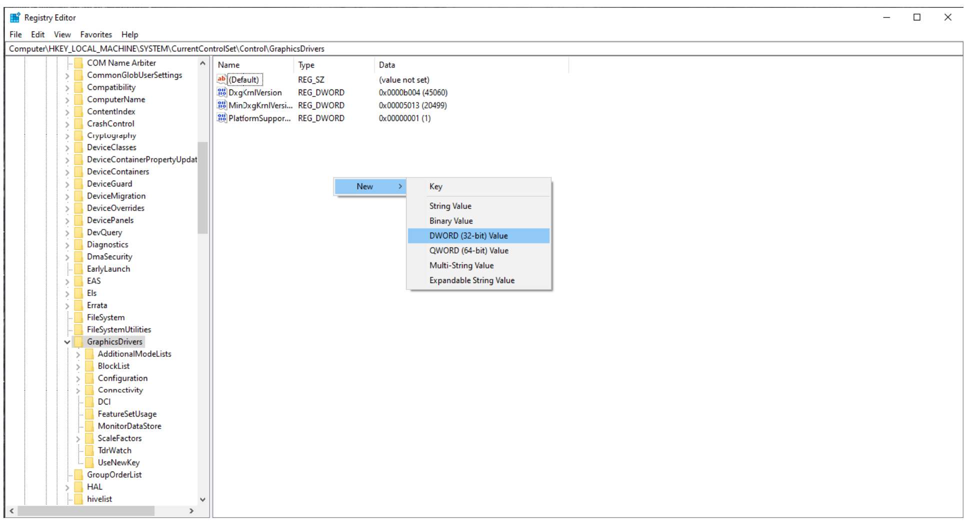


Figure 4.2: Adding a new DWORD to the Windows registry.

Once the new DWORD has been added, change the name to "TdrLevel" and then right click and select "Modify" (Figures 4.3 and 4.4).

In the box that pops up, set the value of the "TdrLevel" key to be 0 in either Hex or Decimal as it is the same value before hitting Ok (Figure 4.5). The effect of this is to disable the TDR system, allowing for computations lasting longer than two seconds to run on a GPU. If turning this off presents problems, the timeout threshold for the TDR system can be increased, particularly if the computation time is known beforehand.

Using the same procedure listed above, add a second DWORD with the name "TdrDelay"

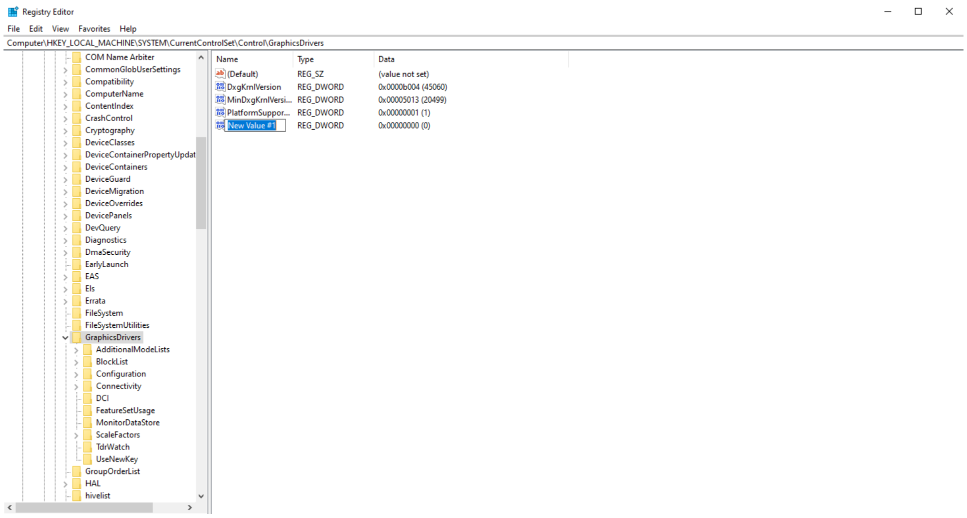


Figure 4.3: New DWORD added to the registry.

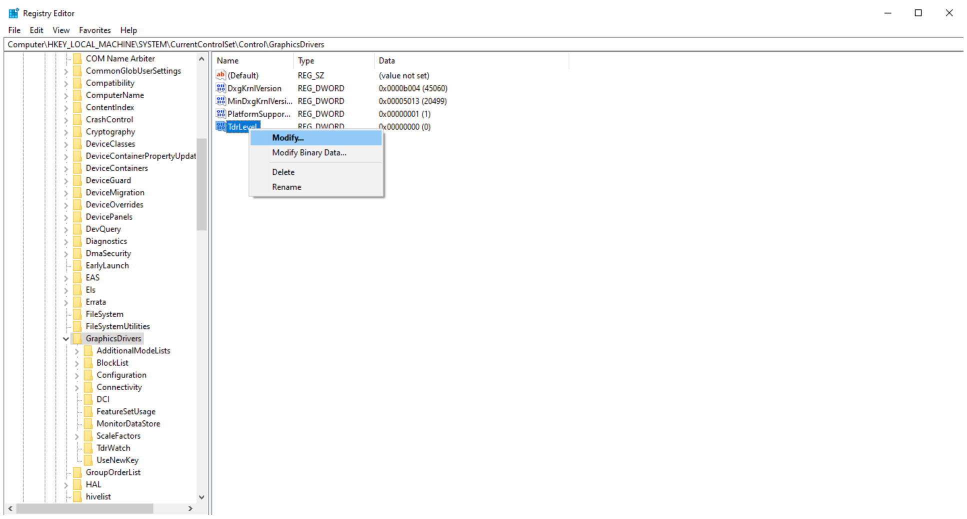


Figure 4.4: Modifying the added DWORD.

and a value of 10 (in decimal, NOT hex). Figure 4.6 shows this second DWORD.

Once the new registry keys have been added, reset the PC and now more intensive computations can be ran on any GPU on the machine. Specifics on what these keys and settings do can be found on Microsoft's web page at `https://docs.microsoft.com/en-gb/windows-hardware/drivers/display/tdr-registry-keys`.

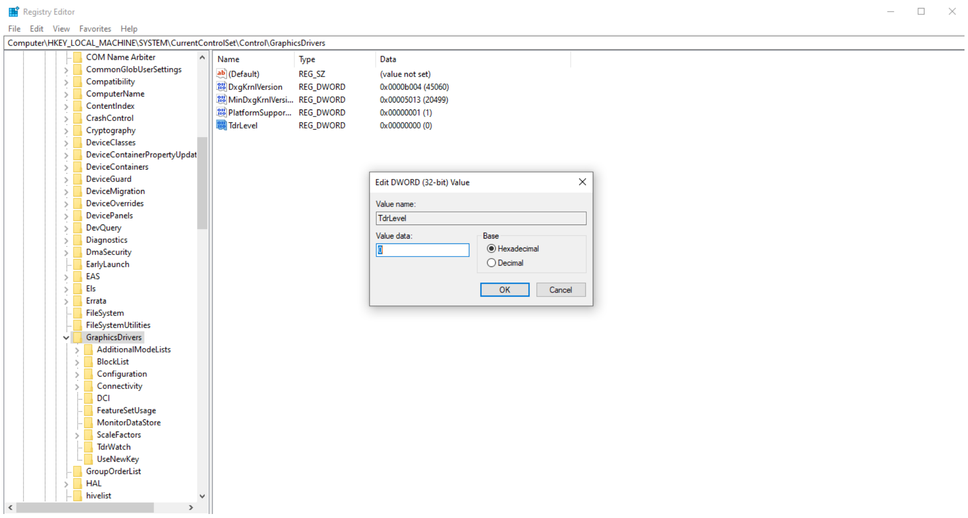


Figure 4.5: Setting the DWORD value.

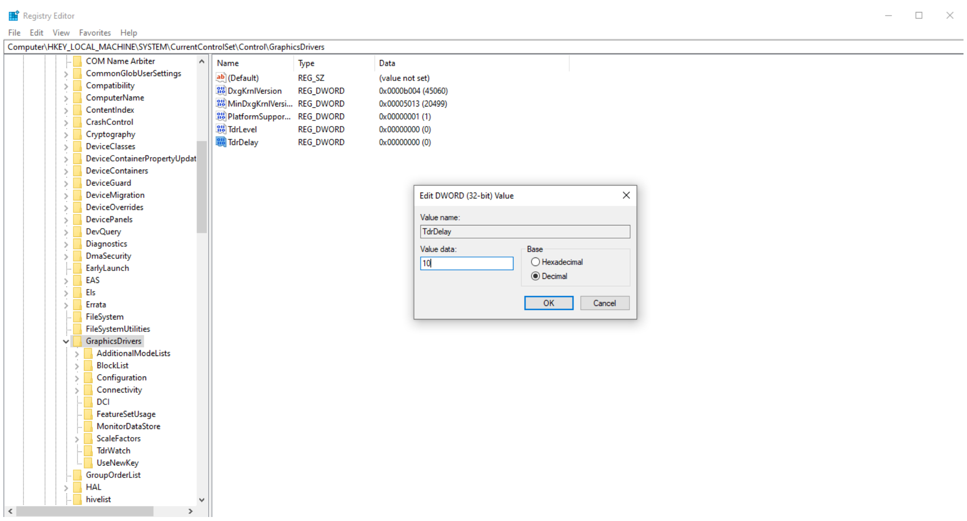


Figure 4.6: Increasing the TDR timeout threshold.

4.3 CUDA Debugging

As with any development projects, there will always be necessary debugging. For CUDA debugging NVIDIA has released a couple of important tools that can be used. The first tool is NVIDIA's Nsight, which can be used to add break points to check values, stack tracing, and disassembly of the PTX code to name a few. Nsight must be used with Visual Studio and operates in a similar fashion to Visual Studio's own debugging toolkit. However, not every GPU is Nsight compatible, a list of Nsight compatible GPUs can be found at `https://`

`developer.nvidia.com/nsight-visual-studio-edition-supported-gpus-full-list`.

For devices that are not Nsight compatible, there still are alternatives. For checking intermediate values of a kernel, the *printf()* function from C will work on devices with compute capability 2.x or higher. There are a couple caveats to using *printf()*. First, *printf()* writes the output to a buffer, which does not get flushed ot the output console until the kernel function completes. If the kernel function fails to complete, then the output will be discarded and the console will show nothing. Second, due to the multithreaded nature of GPU programming, every single thread will print the results of *printf()* to console. The order the threads print in is not necessarily the absolute order of the threads as well, the warps that finish first are the ones whose threads will print first. This can result in a chaotic and hard to interpret result.

To simplify printing, it is often preferable to have only one thread print to the console. To do this, use the following code:

```
if (blockIdx.x * blockDim.x + threadIdx.x == 0)
{
    printf();
}
```

The *blockIdx.x * blockDim.x + threadIdx.x* calculation has a special meaning in CUDA. This is a way of calculating the absolute index of a thread. Recall in section 3.3 that the developer has to specify the number of blocks and the number of threads per block for each kernel call. Each block will have its own threads ranging from 0 to the number of threads per block. Therefore there will be (number of blocks) threads with the index of 0. Getting the absolute index of a thread is useful to figure out what data the thread needs to be pointed to, so getting familiar with this statement is important.

In a similar vein, another debugging trick is to insert a *printf()* statement at the end of all kernel calls to let the developer know that that particular kernel call has completed. CUDA has a *cudaerror_t* struct which can be used to check for any major errors following a kernel call or the use of any of the functions in the CUDA API such as *cudaMalloc()*. If there is an error, the *cudaGetErrorString()* function will return a string that details the error in question. Certain errors such as a launch error from attempting to execute a kernel function with the wrong parameters (too many threads, too many resources requested, etc) can fail to report any error when checking for error codes in the *cudaerror_t* struct. This will cause the kernel function to be skipped, if there are multiple kernel functions, then the GPU program will continue calling those functions without indicating that any error has occured. Therefore it is important to have kernel functions print that they are finished when they have completed their execution.

The errors that sometimes are not flagged can also be found by using the *cuda-memcheck* tool that also comes in the CUDA toolkit. To use it, locate the directory where the executable of the program you have written is located, open up a cmd window (in Windows), then change to that directory and use the command

```
cuda-memcheck <your CUDA program>.exe
```

This should launch the CUDA Memcheck tool, which is similar to the Valgrind tool used by C programmers. The cuda memcheck tool can precisely detect and report out of bounds memory accesses as well as report hardware reported errors. It is very a very useful tool for debugging hard to find problems.

Chapter 5

Homotopy Continuation on a GPU

5.1 Path Tracking Algorithm

A GPU consists of an array of streaming multiprocessors (SM), each of which is analogous to the core of a CPU. However, unlike a CPU which can execute different instruction sets concurrently, the SMs of a GPU executes the same instruction set concurrently. An important consequence of this aspect of SM computation is that conditional instructions degrade the performance on a GPU. This can occur wherever there is an IF statement or a WHILE statement, because when the instructions of some of the threads in a warp are different from the rest, the SM must execute the two different instruction sets one at a time, which results in stalled threads and under-utilization of the GPU. The amount of under-utilization depends on the size of the instruction sets to be used. IF statements are quite useful so they should not be avoided all together, rather the amount of divergence should be considered when implementing code on a GPU, particularly if maximum performance is desired.

In polynomial numerical continuation, branch divergence can occur when the path tracking step size Δt is changed, and when the number of correction steps is changed to achieve convergence. The path tracking algorithms of numerical polynomial continuation solvers implement adaptive step size and convergence checking to ensure the accuracy of each path. Unfortunately, these features can have a large performance hit when implemented on a GPU, because when path tracking step is changed in one thread all the other threads in the warp are paused until the computation is completed. This happens when the number of correction steps is changed as well as can be seen in Algorithm 1. The step sized is decreased whenever a bad step has been taken, where the predictor has put the path out of the range of convergence of the correction step. This area of convergence is inversely proportional to the degree of the polynomial system. Additionally, typical path trackers also check if the predicted point is within the area of convergence, but fails to converge to an acceptable degree. Both of these issues are influenced by the condition number of the Jacobian matrix. Ill-conditioned Jacobians require smaller steps to be taken and more correction steps to achieve an acceptable level of convergence.

Algorithm 1: Path tracking with variable step size

Data: x_0, a solution to $g(x)$;

Δt_0, an inital step size;

ϵ, an error tolerance;

c, the step adjustment factor;

Result: x, a solution to $f(x)$

begin
    $x \leftarrow x_0, t \leftarrow 1$
    while $t - \Delta t > 0$ **do**
        $x_p \leftarrow predict(H, x, t, \Delta t)$
        $x_c \leftarrow correct(H, x_p, t, \Delta t)$
        if $H(x_c, t) > \epsilon$ **then**
            $\Delta t \leftarrow c^{-1} \Delta t$
        end
        else
            $x \leftarrow x_c$
            $\Delta t \leftarrow min(c\Delta t, t)$
            $t \leftarrow t - \Delta t$
        end
    end
end

Our approach is to introduce a new path tracking algorithm that is a better match to the SIMT requirements of a GPU, Algorithm 2. Our algorithm is has a fixed number N of equal steps Δt along the path, as well as a fixed number of iterations C of the Newton correction.

Algorithm 2: New path tracking algorithm with fixed step size Δt.

Data: x_0, a solution to $\mathcal{S}(\mathbf{x})$;

Δt, the fixed step size;

C, the number of iterations of newtons method to execute;

N, the number of steps to take;

Result: x, a solution to $\mathcal{P}(\mathbf{x})$

begin
- $x \leftarrow x_0, t \leftarrow 1$
- **for** i = *1:N* **do**
  - $t \leftarrow t - \Delta t$
  - $p \leftarrow \mathit{predict}(\mathcal{H}, \mathbf{x}, t, \Delta t)$
  - $c \leftarrow \mathit{correct}(\mathcal{H}, \mathbf{p}, t, \Delta t, C)$
  - $x \leftarrow c$
- **end**

end

It is possible to implement a path tracker with an adaptive step size on a GPU, however, this causes the all the threads of the GPU to run at the speed of the most difficult path. Rather than parallelize the path tracking, it is also useful to use the GPU to speed up the computation of the elements of both the polynomial system and its Jacobian matrix, see Verschelde and Yu(2015) [59, 60]. Our goal is to use the GPU to minimize computation time, so we fix the step size to minimize conditional branching.

Tracking paths with a fixed step size can mean that we cannot avoid the zones around areas singularities which would cause the Jacobian matrix, $\mathbf{J_x}$, to become ill-conditioned. In these zones, the path will almost certainly not result in a solution to the target system of polynomials. Similarly, if the number of iterations of Newton's correction is not enough to assure convergence the path will not result in a solution.

Both of these issues can be mitigated by an evaluation kernel call that checks all of the roots computed for $\mathcal{P}(\mathbf{x})$, and identifying those paths that do not yield accurate roots. Those that fail can be recalculated using a smaller step size and larger values for N and C. This strategy sacrifices individual path tracking accuracy for increased GPU computational performance associated with a single instruction set.

5.2 Path Tracking Implementation

In this section, we present the series of computations that perform path tracking for numerical polynomial continuation that will be executed on a GPU. Because the initial step of the path tracker begins with a known root of the start system, we begin by predicting the next value using the Runge-Kutta-Fehlbreg method. See Cheney and Kincaid (2012)[63]. For convenience, introduce the notation $\mathbf{f}(\mathbf{x}, t)$ for the vector function in (2.8), so we have,

$$\Delta\mathbf{x} = -\mathbf{J}_{\mathbf{x}}^{-1}(\mathbf{x}, t)\mathbf{H}_t(\mathbf{x})\Delta t = \mathbf{f}(\mathbf{x}, t). \tag{5.1}$$

For step size Δt, the next prediction point $\mathbf{p}$ of a path can be calculated using the Runge-Kutta fourth order formulas,

$$\mathbf{p}(t + \Delta t) = \mathbf{x}(t) + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4). \tag{5.2}$$

where,

$$
\begin{aligned}
\mathbf{K}_1 &= \Delta t\, \mathbf{f}(\mathbf{x}, t),\\
\mathbf{K}_2 &= \Delta t\, \mathbf{f}(\mathbf{x} + \frac{1}{2}\mathbf{K}_1, t + \frac{1}{2}\Delta t),\\
\mathbf{K}_3 &= \Delta t\, \mathbf{f}(\mathbf{x} + \frac{1}{2}\mathbf{K}_2, t + \frac{1}{2}\Delta t),\\
\mathbf{K}_4 &= \Delta t\, \mathbf{f}(\mathbf{x} + \mathbf{K}_3, t + \Delta t)
\end{aligned}
\tag{5.3}
$$

This calculation of $\mathbf{p}$ involves four evaluations of $\mathbf{f}$ for different arguments, each of which requires finding the inverse of the $n \times n$ Jacobian matrix $\mathbf{J}_{\mathbf{x}}$.

5.2.1 LU decomposition

An effective algorithm for calculating the inverse of $\mathbf{J}_{\mathbf{x}}$ is known as LU decomposition [63]. This is achieved by permuting $\mathbf{J}_{\mathbf{x}}$, so that it can be factored into the product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$, that is

$$\mathbf{P}\mathbf{J}_{\mathbf{x}} = \mathbf{L}\mathbf{U}. \tag{5.4}$$

Where $\mathbf{P}$ is an $n \times n$ matrix that permutes the rows of $\mathbf{J}_{\mathbf{x}}$. Write Eq 2.8 in the form,

$$\mathbf{J}_{\mathbf{x}}\Delta\mathbf{x} = -\mathbf{H}_t\,\Delta t, \tag{5.5}$$

and substitute the LU decomposition to obtain,

$$\mathbf{P}\mathbf{J}_{\mathbf{x}}\Delta\mathbf{x} = \mathbf{L}\mathbf{U}\Delta\mathbf{x} = -\mathbf{P}\mathbf{H}_t\,\Delta t. \tag{5.6}$$

We solve this equation by introducing $\mathbf{z} = \mathbf{U}\Delta\mathbf{x}$, and use sequential elimination by rows to solve

$$\mathbf{L}\mathbf{z} = -\mathbf{P}\mathbf{H}_t\,\Delta t, \tag{5.7}$$

for $\mathbf{z}$. Then, back-substitution is used to solve

$$\mathbf{U}\Delta\mathbf{x} = \mathbf{z}. \tag{5.8}$$

for $\Delta\mathbf{x}$.

This solution $\Delta\mathbf{x}$ is used to calculate each of the four terms $\mathbf{K}_i$, $i = 1, 2, 3, 4$ in the Runge-Kutta calcuation for $\mathbf{p}$.

5.2.2 Newton's correction

We assume the calculation of the point $\mathbf{p}(t + \Delta t)$ along a path $\mathbf{v}(t)$ takes the point away from the homotopy hypersurface $\mathcal{H}(\mathbf{x}, t) = 0$, so we use Newton's method to find the nearby root $\mathbf{c}(t + \Delta t)$. Write (2.9) in the form,

$$\mathbf{J}_{\mathbf{x}}(\mathbf{p}, t + \Delta t)\Delta\mathbf{x} = \mathcal{H}(\mathbf{p}, t + \Delta t). \tag{5.9}$$

This equation can be solved using LU decomposition to calculate $\Delta\mathbf{x}$, which yields the correction,

$$\mathbf{c} = \mathbf{p} + \Delta\mathbf{x}. \tag{5.10}$$

The usual implementation of a path tracker for numerical polynomial continuation iterates

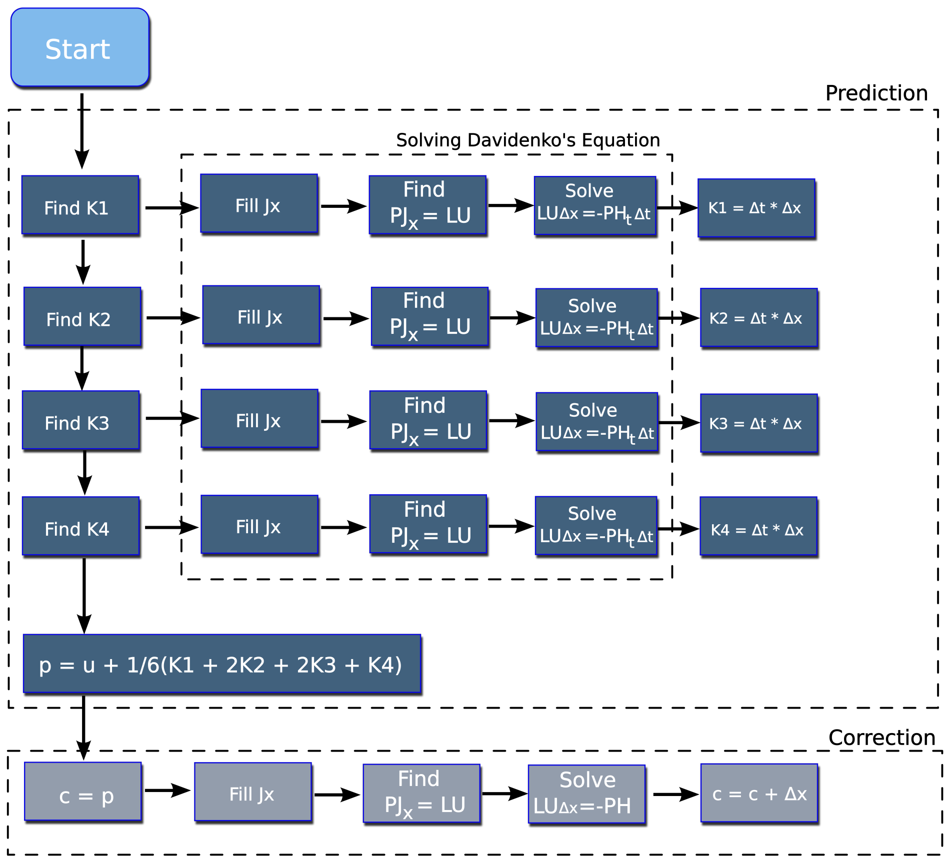


Figure 5.1: The solution procedure of the ordinary differential equations for prediction and Newton's method for correction at each path tracking step.

Newton's method to ensure convergence to the homotopy hypersurface. The correction step can be repeated multiple times until a desired level of convergence is achieved. The computation flow of the path-tracking method is shown in Figure 5.1.

5.3 Endgame Algorithm

When considering which endgame method to use it is important to determine how parallel each of the methods are. Cauchy's integral requires additional path tracking around the circle $t' = Re^{i\theta}, \quad \theta \in [0, 2\pi c]$, where c is the winding number for that particular solution. Convergence of a solution using Cauchy's integral depends on selection of an appropriate radius, which is the endgame boundary R. If the radius chosen is too large, then it is possible for the circle to encapsulate more than one solution. If the radius is too small, the numerics of the path tracking will break down and fail to find a solution. Therefore it is possible to select multiple radii $R_i = \{R_1, R_2, ..., R_N\}$ and then track all of the N circles in parallel. This would result in (N * the number of paths) threads.

The power series method of endgame relies on fitting an interpolating polynomial to a set of sample points, transforming the t values of the samples $s = t^{1/c}$ for a set of potential winding numbers $c = \{1, 2, ..., c_{max}\}$. This presents the first axis of parallelism for the power series endgame, as the candidate solution for each of the c values can be found independent of the other c values.

The power series method additionally has a second axis of parallelization however. One of the properties of interpolation with vector-valued functions is that each element of the resulting vector can be calculated independently of the others. In other words, the interpolation can be treated as a series of separate problems, where each unknown has its own interpolation function. This means that each of the unknowns being solved for can be found independently of one another. This results in ($c_{max}*$ number of unknowns $*$ number of paths) threads. It is this second axis of parallelization that makes the power series method much more parallelizable than Cauchy's integral and thus a better candidate for the GPU.

One of the drawbacks of the power series method is that it will fail to find a solution with winding number $c > c_{max}$. Additionally, potential winding numbers larger than 4 or so require precision higher than that of a double. While GPUs do have multi-precision libraries like that of the CPU, the throughput of mathematical operations using such precision would have a significant impact on the performance of the GPU.

However Cauchy's integral has a worse drawback: the convergence of finding a solution depends on finding the exact winding number c for each path. Because the exact winding number is not known a priori, each path would have to be tracked around the origin until the path closed, revealing the correct c and predicted solution. On a GPU this would have the effect of upper bounding each warp to the path that had the highest winding number. This would result in a much higher performance impact on the GPU than the power series method would.

The power series therefore sacrifices the ability to find solutions with high winding numbers, reducing the amount of required computation to approximate a solution in return. If all solutions absolutely must be found, Cauchy's integral method would be preferable to the power series endgame. However, for the problem of kinematic synthesis, completeness of the solution set is not necessary. This is because 1) an incomplete solution results in missed potential designs, which while not the best is neither catastrophic for the problem and 2) when designing more complex linkages, it is known that very few (0.5 percent of solutions for the watt 1 six-bar synthesis problem) solutions will result in a successful design.

These details combined with the fact that the power series method is inherently more parallelizable means that the method is the preferred endgame chosen to be run on a GPU for kinematic synthesis.

5.4 Endgame Implementation

To implement the power series endgame two steps are necessary: 1) A sampling of points must be found and 2) An interpolating polynomial of those points is found for each value of c. The sampling of points can be found by using the same path tracking algorithm listed in Algorithm 2.

The sampling begins at the chosen endgame boundary value $0 < R < 1$. The power series method selects a sampling t values $(t_1, t_2, ..., t_{\kappa+1})$, and then uses path tracking to find the corresponding path values $(\bar{\mathbf{x}}(t_1), \bar{\mathbf{x}}(t_2), ..., \bar{\mathbf{x}}(t_{\kappa+1}))$.

For each value c, the path values are transformed to

$$(\mathbf{s}_1, \mathbf{s}_2, ...\mathbf{s}_{\kappa+1}) = (\mathbf{t}_1^{1/c}, \mathbf{t}_2^{1/c}, ..., \mathbf{t}_{\kappa+1}^{1/c}). \tag{5.11}$$

We compute an interpolating polynomial $\phi_c(\mathbf{s})$ to fit to the first κ values. As the number of samples increases, the values of $\mathbf{s}$ become very small. This affects the numerical stability of the interpolation, so the $\mathbf{s}$ values are normalized before proceeding.

As for finding the interpolating polynomial, many methods exist to find $\phi_c(\mathbf{s})$. There are two methods considered for our GPU implementation: Newton and Hermite. For the purposes of this section the only difference between the two is that Hermite uses the sample points and their corresponding derivatives while Newton only uses the sample points themselves.

The final value $\bar{\mathbf{x}}_{\kappa+\mathbf{1}}$ is predicted from $\phi_c(\mathbf{s}_{\kappa+\mathbf{1}})$ for each c. The c which minimizes $||\bar{\mathbf{x}}(t_{\kappa+1}) - \phi_c(\mathbf{s}_{\kappa+\mathbf{1}})||$ is used to predict the solution $\mathbf{x}^* = \phi_c(0)$.

The power series endgame can test each candidate c in parallel. Furthermore, each of the n rows of $\bar{\mathbf{x}}$ requires its own interpolating polynomial. Therefore each ϕ_{ic} can be found in parallel for rows $i = 1, ..., n$.

The power series endgame can be executed on $c_{max} * n$ threads for each path and is more well suited to run on a GPU. For an outline of the entire endgame, see Figure 5.2.

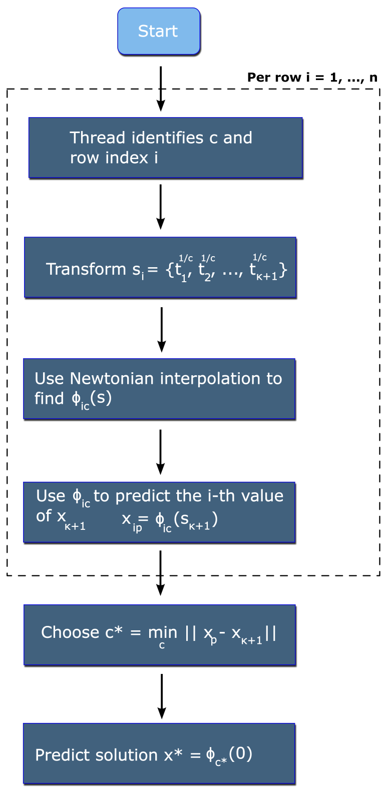


Figure 5.2: Computational flow of the endgame for a single path.

Chapter 6

Four-Bar Linkage Synthesis

In this section, we formulate the synthesis equations for a four-bar linkage that we solve using our GPU implementation of numerical polynomial continuation. The goal is to compute the dimensions of a four-bar linkage that guides its coupler through five task positions. We formulate the design problem following Glabe and McCarthy (2019)[64] using the loop equations of the linkage. This is different from the usual approach known as Burmester theory which uses the constraint equations of a crank, McCarthy and Soh (2010) [5]. We use this approach because it can be generalized to design more complex linkage systems [29].

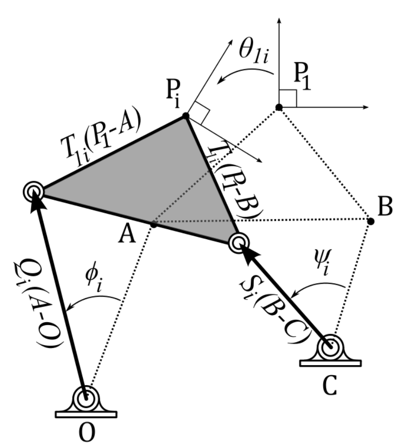


Figure 6.1: A four-bar linkage moving from position P_1 to P_i, showing the complex vectors that form the loop equations.

6.1 Loop Equations

The synthesis equations that we will solve are obtained from the loop equations of the four-bar linkage. Let the coordinates of the fixed pivots of the linkage be denoted $O = O_x + iO_y$ and $C = C_x + iC_y$ and let the moving pivot coordinates be $A = A_x + iA_y$ and $B = B_x + iB_y$.

The loop equation defines the relationship that is preserved among these variables throughout the movement of linkage linkage. They obtained as the complex vector equations,

$$P = O + Q_\phi(A - O) + T_\theta(P_1 - A) = C + S_\psi(B - C) + T_\theta(P_1 - B), \tag{6.1}$$

where

$$Q_\phi = e^{i\phi}, \quad S_\psi = e^{i\psi}, \quad T_\theta = e^{i\theta}. \tag{6.2}$$

This loop equation can be used to formulate synthesis equations for the dimensions of the four-bar linkage. First, identify five positions that are to be achieved by the end-effector of the linkage, denoted, $\Gamma_j = (\theta_j, P_j)$, $j = 1, \ldots, 5$, where P_j is the position of the origin and θ_j is the orientation of a desired end effector pose with respect to the x-axis. Then, evaluate the loop equations and their conjugates for each of these task positions. The result is

$$\begin{aligned} O + Q_j(A - O) + T_j(P_1 - A) =& P_j, \\ C + S_j(B - C) + T_j(P_1 - B) =& P_j, \quad j = 1, ..., 5. \end{aligned} \tag{6.3}$$

and

$$\begin{aligned} \bar{O} + \bar{Q}_j(\bar{A} - \bar{O}) + \bar{T}_j(\bar{P}_1 - \bar{A}) =& \bar{P}_j, \\ \bar{C} + \bar{S}_j(\bar{B} - \bar{C}) + \bar{T}_j(\bar{P}_1 - \bar{B}) =& \bar{P}_j, \quad j = 1, ..., 5. \end{aligned} \tag{6.4}$$

Then, introduce the normal conditions for the angles Q_j and T_j,

$$Q_j \bar{Q}_j = 1, \quad \text{and} \quad S_j \bar{S}_j = 1, \quad j = 1, ..., 5. \tag{6.5}$$

The result is a set of polynomial equations in the coordinates O, A, B, and C and their conjugates which define the coordinates of the pivots of the linkage in the reference position and the relative angles ϕ_j and ψ_j, $j = 1, \dots, 5$ that define the movement of the linkage through the five task positions.

6.2 Algebraic Reduction

We can simplify these equations to eliminate the unknowns Q_j, $\bar{Q}_j$, S_j and $\bar{S}_j$ by solving Eq. 6.3 for Q_j and S_j and Eq. 6.4 for $\bar{Q}_j$ and $\bar{S}_j$, and then substitute the results into Eq. 6.5. This yields,

$$\begin{aligned}
(P_j - T_j P_1 + T_j A - O)(\bar{P}_j - \bar{T}_j \bar{P}_1 + \bar{T}_j \bar{A} - \bar{O}) - (A - O)(\bar{A} - \bar{O}) &= 0, \\
(P_j - T_j P_1 + T_j B - C)(\bar{P}_j - \bar{T}_j \bar{P}_1 + \bar{T}_j \bar{B} - \bar{C}) - (B - C)(\bar{B} - \bar{C}) &= 0 \quad j = 1, ..., 5.
\end{aligned} \tag{6.6}$$

which form a system of 10 quadratic equations.

These equations can be further simplified by selecting the first task position as the reference frame, such that $P_1 = (0, 0)$ and $\theta_1 = 0$ and measuring the remaining four task positions relative to this frame. To do this, compute the five homogeneous transformation matrices

H_j associated with the given task positions, $\Gamma_j = (\theta_j, P_j)$, $j = 1, \ldots, 5$,

$$\mathbf{H}_j = \begin{bmatrix} \cos\theta_j & -\sin\theta_j & Px_j \\ \sin\theta_j & \cos\theta_j & Py_j \\ 0 & 0 & 1 \end{bmatrix}, \quad j = 1, ..., 5. \tag{6.7}$$

Then, transform these matrices to the first task frame of Γ_1,

$$\mathbf{K}_{1j} = \mathbf{H}_1^{-1}\mathbf{H}_j, \qquad j = 2, \ldots, 5. \tag{6.8}$$

Obtain the new relative task positions $\Gamma_{1i} = (\theta_{1i}, W_{1i})$ as,

$$\theta_{1i} = \arctan(k_{21}/k_{11}), \quad W_{1i} = (k_{13} + ik_{23}), \quad i = 2, \ldots, 5. \tag{6.9}$$

This yields two sets of loop equations relative to P_1,

$$\begin{aligned} \mathcal{P}: \quad & (W_{1i} + T_{1i}A - O)(\bar{W}_{1i} + \bar{T}_{1i}\bar{A} - \bar{O}) - (A - O)(\bar{A} - \bar{O}) = 0, \quad i = 2, \ldots, 5, \\ & (W_{1i} + T_{1i}B - C)(\bar{W}_{1i} + \bar{T}_{1i}\bar{B} - \bar{C}) - (B - C)(\bar{B} - \bar{C}) = 0, \quad i = 2, \ldots, 5. \end{aligned} \tag{6.10}$$

Where $T_{1i} = e^{i\theta_{1i}}$. Figure 6.1 outlines the geometry of the relative displacement of the four-bar linkage. Equation 6.10 is a system of eight polynomials in eight unknowns $(O, \bar{O}, A, \bar{A}, B, \bar{B}, C, \bar{C})$. Each solution of this set of equations is a candidate for a four-bar linkage that guides its coupler link through the given set of task positions. This polynomial system has a Bezout degree of $2^8 = 256$.

The synthesis equations $\mathcal{P}$ in equation (6.10) are linear combinations of monomials formed from the variables $\mathbf{x} = (O, \bar{O}, A, \bar{A}, B, \bar{B}, C, \bar{C})$. These equations include the 16 parameters

$$\mathbf{p} = (T_{12}, T_{13}, T_{14}, T_{15}, \bar{T}_{12}, \bar{T}_{13}, \bar{T}_{14}, \bar{T}_{15}, W_{12}, W_{13}, W_{14}, W_{15}, \bar{W}_{12}, \bar{W}_{13}, \bar{W}_{14}, \bar{W}_{15}), \quad (6.11)$$

which are constants that define the task for the linkage to be designed.

We use the numerical continuation software Bertini to solve start system $\mathcal{S}$ that can be used for parameter continuation to solve a polynomial system $\mathcal{P}(\mathbf{p}, \mathbf{x})$ for different values of the parameters $\mathbf{p}$. Bertini chooses a generic set of parameters $\mathbf{q}$ so that the start system is $\mathcal{S}(\mathbf{q}, \mathbf{x})$ has known roots $\mathbf{y}_k$. We used the numerical continuation solver Bertini [25] to compute a start system for this polynomial system and found that it had a multi-homogeneous degree of 25 as a result of this special mononomial structure

6.3 Randomization

A multi-homogeneous degree of 25 means that there are 25 paths to track. A GPU typically needs thousands of threads to be running in order to see a significant speedup, 25 threads (one per path) is simply not enough. Fortunately, there is another property of kinematic synthesis that can be used to increase the number of concurrent threads into the thousands: randomization.

It is the nature of this linkage design problem that for a given task $T : \Gamma_j = (\theta_j, P_j)$, $j = 1, \ldots, 5$, the resulting four-bar linkage may have one or more of a set of various defects, Chase and Mirth (1993)[65] and Beloiu and Gupta (1997)[66]. To address this Plecnik and McCarthy (2012) [67] introduced tolerance zones around the specified task positions, and randomly selected small variations within these zones. The result is a successful set of linkages that reach task positions close to the originally specified positions. See also Tsuge and McCarthy (2016) [68].

We implement this strategy by reading the specified task positions $\Gamma_j = (\theta_j, P_j)$, $j = 1, \ldots, 5$ and a set of tolerance zones specified by the designer, $(\Delta\theta, \Delta x, \Delta y)_i$, $i = 1, \ldots, 5$ and writing L new tasks,

$$T_m : \quad \Gamma_{jm} = (\theta_j + \rho_{jm}\Delta t, P_j + \sigma_{jm}\Delta x + i\tau_{jm}\Delta y)_m, \;\; j = 1, \ldots, 5, \;\; m = 1, \ldots, L, \quad (6.12)$$

where ρ_{jm}, σ_{jm} and τ_{jm} are randomly generated constants between -1 and 1.

This results in $L{*}25$ threads, which can be arbitrarily scaled up or down until GPU occupancy has been achieved. This randomization property of kinematic synthesis is another feature that makes it particularly attractive to be performed on a GPU.

6.4 Four-Bar Synthesis Implementation

Following the heterogeneous computation model describe in Section 3.3, it is important to consider what parts are executed by the host (CPU) and which are executed by the device (GPU). For this synthesis problem, the CPU first reads in the task data and tolerances specified in text files, transfers the data to the GPU, and then next three GPU kernel functions are executed sequentially: Task Generator, Path Tracker, and Solution Filter.

6.4.1 Task Generator

The Task Generator reads the data transferred from the CPU and writes L different tasks to the GPU memory. After generating the L new sets of task positions as described in Section 6.3, Task Generator then converts them into the set of relative task positions using the procedure described in Section 6.2, Equations (6.7, 6.8, and 6.10). These new relative task position sets are also written to the GPU memory forming the final parameters $\mathbf{p}$ for the parameter homotopy.

6.4.2 Path Tracker

Path Tracker is the implementation of the algorithm outlined in Section 5.1. As outlined in Section 2.1, path tracking involves integrating the Davidenko differential equation (2.8) derived in Section 2.2 and then correcting using Newton's method.

We use the parameters $\mathbf{q}$ computed by Bertini to construct the parameter homotopy,

$$\mathcal{H} = \mathcal{P}(\mathbf{q}\,t + \mathbf{p}\,(1-t), \mathbf{x}). \tag{6.13}$$

This is computed symbolically in Mathematica (Wolfram (2003)[69]), where we also compute symbolic equations for 8×8 Jacobian matrix $\mathbf{J_x}$ and the 8×1 vector $\mathbf{H}_t$; see (2.6). These symbolic equations are converted into the CUDA code used by Path Tracker.

Path Tracker also includes the algorithm for LU decomposition, Runge-Kutta prediction and Newton correction. Organizing the calculations in this way uses the advantages of the GPU for rapid computation, however, it means that some of the paths may not converge to roots of our target system. Our formulation of the linkage synthesis problem reduces the importance of finding any particular root.

6.4.3 Solution Filter

The Solution Filter kernel call checks solutions found by Path Tracker to ensure that the requirements of $(O, \bar{O}), (A, \bar{A}), (B, \bar{B})$, and $(C, \bar{C})$ are true complex conjugate pairs. Only solutions that meet this requirement have a true physical meaning, for more info, see

[6]. True physical solutions are flagged for transferring to the CPU for further analysis.

6.4.4 Linkage Analysis

The physical linkages identified in the GPU must be evaluated to determine they are defect-free, which we call effective solutions. For the Four-bar problem, linkages with branch and circuit defects are rejected, but those with order defects are allowed. Examples of this analysis can be found in references such as Uicker et al (2016)[70] or McCarthy and Soh (2010)[5]. Effective solutions are then written to a file along with the task positions that generated that particular solution as the output of the design system. Figure 6.2 outlines the flow of the whole program.

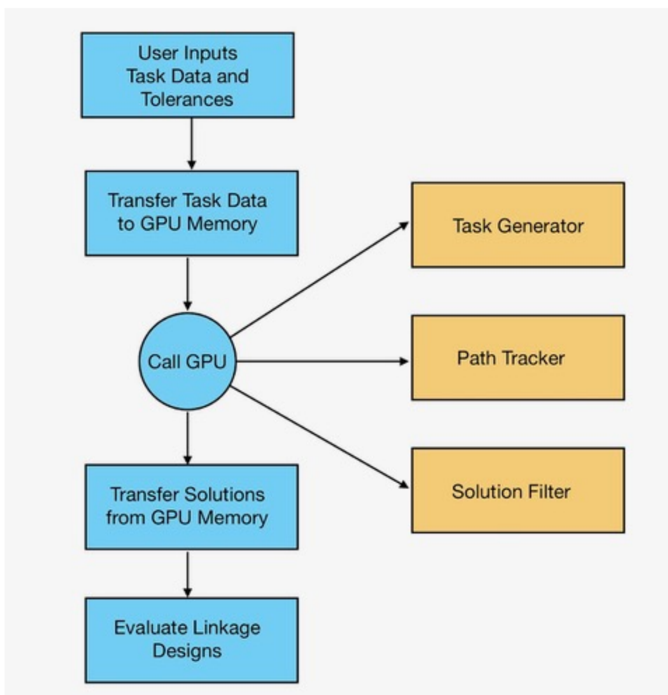


Figure 6.2: Flow of calculations on the CPU (on the left) with calls to the GPU on the right.

6.5 Demonstration

In order to demonstrate this algorithm, we use a Lenovo workstation with an Intel Xeon 2.10 GHz CPU, running Windows 10 with an NVIDIA Quadro M2000 GPU. The five task positions together are listed in Table 6.1 and shown in Figure 6.3. The tolerance zones chosen were $\Delta\theta = 0.5$ degrees, $\Delta x = \Delta y = 0.1$. The original transformed task positions that are used with our synthesis equations are listed in Table 6.2. Because the randomization can produce two different sets of task positions with differing numbers of solutions to analyze, we generated one standard randomized set and ran both the CPU and GPU code on it. The time to calculate 200 iterations, or 5000 threads, with the GPU is shown in Table 6.4 to be 63 seconds.

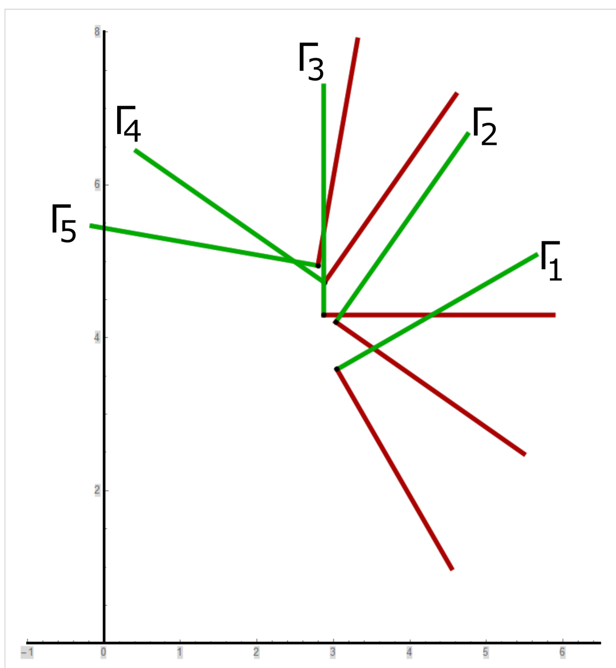


Figure 6.3: The five task positions in the global frame.

For comparison, we used Mathematica 11.1 to define the synthesis equations for 200 randomized tasks for computation using Bertini v1.5.1 on the Lenovo workstation. The Lenovo workstation has multiple cores which allows Bertini to be ran in parallel using 8 CPU threads. Bertini by default performs path tracking using an adaptive step size. The speed up using

Table 6.1: Task position coordinates in the global frame.

j	θ_j (deg)	P_j
1	80	(2.7, 4.9)
2	55	(2.8, 4.6)
3	0	(2.9, 4.4)
4	-35	(3.0, 4.2)
5	-60	(3.0, 4.0)

Table 6.2: Task positions relative to the first task frame.

j	θ_j (deg)	P_j
1	0	(0, 0)
2	-25	(-0.28, -0.15)
3	-80	(-0.46, -0.28)
4	-115	(-0.64, -0.42)
5	-140	(-0.83, -0.45)

the GPU was 120 times compared to the 8 CPU thread computation. One example solution computed by the GPU is given by the coordinates in Table 6.3 and shown in each of the task positions in Figure 6.4.

Table 6.3: Joint coordinates in global frame for a selected solution.

Point	(x, y) coordinates
O	(2.37, 4.43)
A	(2.54, 4.13)
B	(2.46, 4.58)
C	(2.73, 4.29)

A comparison of the results of the two calculations shows the impact of adaptive step size and convergence test for the Newton corrector in Bertini as opposed to the fixed step size and fixed number of Newton iterations. Bertini calculated fewer physical solutions as our GPU code, but more effective designs. This is likely due to the fact that Bertini checks for

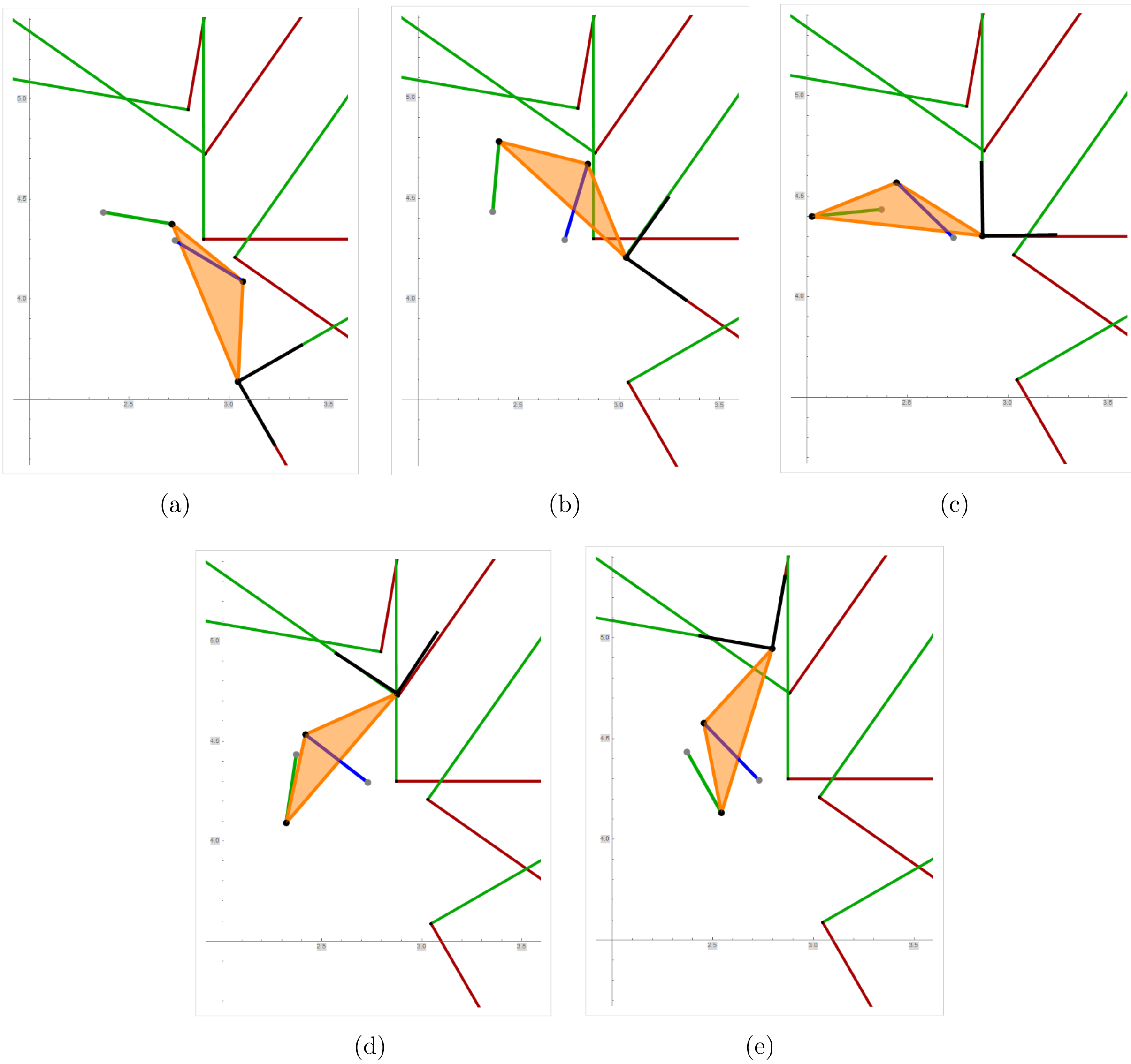


Figure 6.4: An example four-bar linkage design moving the end-effector through the five task positions.

paths crossing, whereas the GPU does not. If two paths cross, it is possible for one path to jump to the other, resulting in a repeated solution. When calculating the number of effective solutions in the GPU we removed duplicate solutions. Thus, the GPU calculation provides 85 effective designs in 63 seconds compared to 455 effective designs in just over two hours of computation.

While it might be tempting to assume using 5000 threads should be 625 times faster than

Hardware	Time (sec)	Solutions	Physical Sols.	Effective Sols.
CPU	7588	3774	1510	455
GPU	63	4425	2382	85

Table 6.4: Comparison of kinematic synthesis of 8 parallel threads on a workstation CPU with our algorithm on 5000 threads on a GPU.

8 threads, it is important to note that GPU and CPU threads are not the same. The base clock speed for an Nvidia M2000 GPU is 1126 MHz while the clock speed for an Intel Xeon CPU is 2.10 GHz. Additionally, CUDA requires threads to be grouped up into blocks, where each block is processed by an SM, one warp at a time. The current GPU algorithm implementation uses only 25 threads per block, whereas on the Maxwell architecture 1024 threads can be executed simultaneously per block. This results in under-utilization of the GPU. More research into this area should be performed to further increase GPU performance.

Additionally, it is important to note that Bertini is not optimized to be executed on instance of the same problem with different parameters. Bertini as a program has many file input/output operations that invariably slows the computation time down. It is important to mention that there is a program being developed called Paramotopy which is intended to be ran on the same problem with different parameters, but the program was unavailable at the time of this dissertation.

Chapter 7

Six-Bar Linkage Synthesis

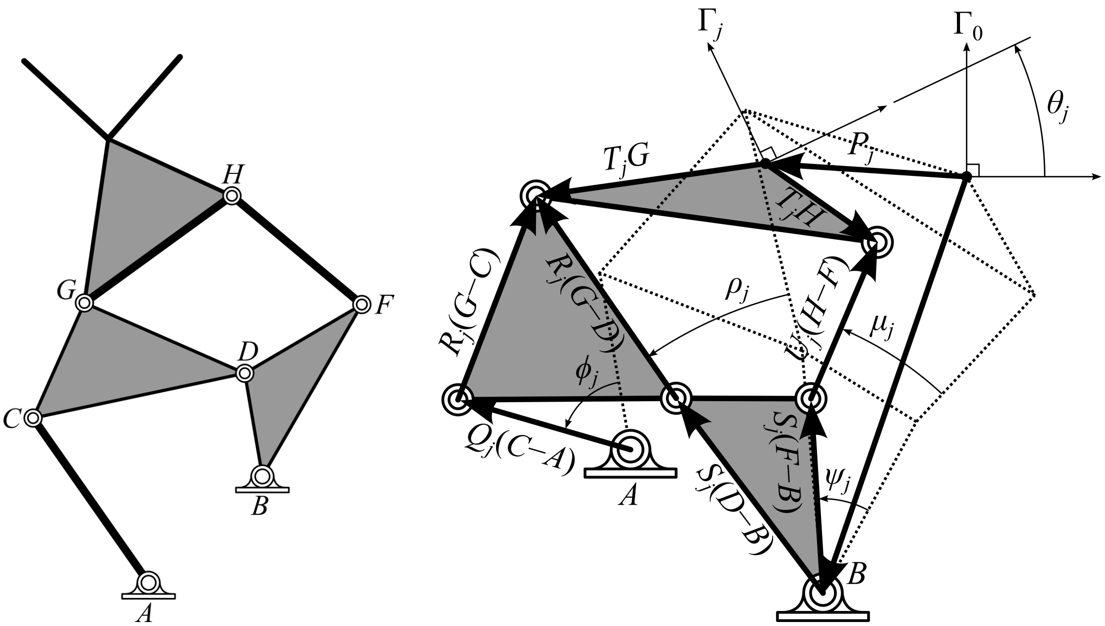
This section extends the results of the four-bar linkage synthesis section to create a GPU-accelerated six-bar linkage design system. This section solves the watt 1 six-bar linkage synthesis problem with two fixed pivot locations specified.

7.1 Loop Equations

The goal of the design system is to compute of the dimensions of Watt 1 six-bar linkages that guide an end-effector through a user-defined set of six task positions with fixed pivots in user-specified regions. This linkage consists of two four-bar loops, Figure 7.1. The first loop, $ACDB$, has fixed pivots A and B and coupler CGD, and the second $GHFD$ has the end-effector attached to the link GH. In order to define the design equations, we follow [71].

The six task frames are defined by the world frame Γ_0 and five relative positions, $\Gamma_j, j = 1, \ldots, 5$, where P_j is the origin of the task frame and θ_j is its angle relative to the x-axis,

$$\Gamma_j = (x_j + iy_j, e^{i\theta_j}) = (P_j, T_j). \tag{7.1}$$



(a) Watt 1 six-bar linkage.

(b) Displaced position.

Figure 7.1: A Watt 1 six-bar motion generator displaced relative to the initial task position.

The design equations are obtained from the three vector loop equations that define the geometry of the six-bar linkage. Each task position defines three sets of loop equations $\mathcal{A}_j$, $\mathcal{B}_j$, and $\mathcal{C}_j$,

$$\mathcal{A}_j = \begin{cases} A + Q_j(C - A) + R_j(G - C) - T_jG & = P_j \\ \bar{A} + \bar{Q}_j(\bar{C} - \bar{A}) + \bar{R}_j(\bar{G} - \bar{C}) - \bar{T}_j\bar{G} & = \bar{P}_j \end{cases} \quad j = 1, \dots, 5, \tag{7.2}$$

$$\mathcal{B}_j = \begin{cases} B + S_j(D - B) + R_j(G - D) - T_jG & = P_j \\ \bar{B} + \bar{S}_j(\bar{D} - \bar{B}) + \bar{R}_j(\bar{G} - \bar{D}) - \bar{T}_j\bar{G} & = \bar{P}_j \end{cases} \quad j = 1, \dots, 5, \tag{7.3}$$

$$\mathcal{C}_j = \begin{cases} B + S_j(F - B) + U_j(H - F) - T_jH & = P_j \\ \bar{B} + \bar{S}_j(\bar{F} - \bar{B}) + \bar{U}_j(\bar{H} - \bar{F}) - \bar{T}_j\bar{H} & = \bar{P}_j \end{cases} \quad j = 1, \dots, 5. \tag{7.4}$$

where Q_j, R_j, S_j, and U_j are the rotations of each joint relative to the initial configuration of the linkage, Figure 7.1, and $\bar{Q}_j, \bar{R}_j, \bar{S}_j, \bar{U}_j$ are their respective complex conjugates,

$$
\begin{aligned}
(Q_j, \bar{Q}_j) &= (e^{i\phi_j}, e^{-i\phi_j}), \quad (R_j, \bar{R}_j) = (e^{i\rho_j}, e^{-i\rho_j}),\\
(S_j, \bar{S}_j) &= (e^{i\psi_j}, e^{-i\psi_j}), \quad (U_j, \bar{U}_j) = (e^{i\mu_j}, e^{-i\mu_j}).
\end{aligned} \tag{7.5}
$$

The relative rotation operators and their conjugates must further satisfy the normalization conditions

$$
Q_j\bar{Q}_j = 1, \quad R_j\bar{R}_j = 1, \quad S_j\bar{S}_j = 1, \quad U_j\bar{U}_j = 1, \qquad j = 1, \ldots, 5. \tag{7.6}
$$

This yields 50 equations in the 10 unknown joint coordinates C, D, F, G and H, and the 40 joint angles Q_j, R_j, S_j, and $U_j, j = 1, \ldots, 5$ and their conjugates.

7.2 Algebraic Reduction

These equations can be simplified by solving for the joint rotation angles, $(Q_j, \bar{Q}_j)$, in the loop equations $\mathcal{A}_j$, and substituting the results into the normalization conditions to obtain,

$$
(P_j - A + R_j(C-G) + T_jG)(\bar{P}_j - \bar{A} + \bar{R}_j(\bar{C}-\bar{G}) + \bar{T}_j\bar{G}) = (C-A)(\bar{C}-\bar{A}), j = 1, \ldots 5. \tag{7.7}
$$

A similar calculation computes $(R_j, \bar{R}_j)$ using its loop equations $\mathcal{C}_j$ and normalization conditions to define,

$$
(P_j - B + S_j(B-D) + T_jG)(\bar{P}_j - \bar{B} + \bar{S}_j(\bar{B}-\bar{D}) + \bar{T}_j\bar{G}) = (G-D)(\bar{G}-\bar{D}), j = 1, \ldots 5. \tag{7.8}
$$

Also substitute $(R_j, \bar{R}_j)$ into (7.7) to obtain,

$$((G-D)(P_j-A+T_jG)+w_j(C-G))((\bar{G}-\bar{D})(\bar{P}_j-\bar{A}+\bar{T}_j\bar{G})+\bar{w}_j(\bar{C}-\bar{G})) = (C-A)(\bar{C}-\bar{A})(G-D)(\bar{G}-\bar{D}), j=1,\ldots 5, \tag{7.9}$$

where

$$w_j = P_j - B + S_j(B-D) + T_jG, \quad \bar{w}_j = \bar{P}_j - \bar{B} + \bar{S}_j(\bar{B}-\bar{D}) + \bar{T}_j\bar{G}. \tag{7.10}$$

Finally, solve the loop equations $\mathcal{B}_j$ for $(U_j, \bar{U}_j)$ and substitute the result into its normalization conditions. This yields the equations,

$$(P_j-B+S_j(B-F)+T_jH)(\bar{P}_j-\bar{B}+\bar{S}_j(\bar{B}-\bar{F})+\bar{T}_j\bar{H}) = (H-F)(\bar{H}-\bar{F}), j=1,\ldots 5. \tag{7.11}$$

The Equations 7.7, 7.9 and 7.11 and the normalization conditions for $(S_j, \bar{S}_j)$ form a system of 20 polynomials in the 10 unknown joint coordinates and the five joint angles S_j and their five conjugates. This polynomial system has a total degree of $\approx 5.90 \times 10^9$ and a multi-homogeneous degree of $1,998,720$.

This polynomial system is converted to design equations for our design system by using *Bertini* to solve a generic system with randomly assigned task parameters. This calculation yielded a generic set of 5,743 roots that can be used in a parameter homotopy. This one-time calculation required seven hours on a Linux cluster assembled from four Lenovo workstations, each with dual Intel Xeon 2GHz processors running 16 threads each for a total of 128 threads. These 5,743 roots form the start points of the paths that must be tracked to solve subsequent six-bar synthesis problems.

7.3 Randomization

Similar to four-bar synthesis, the process of generating and evaluating solutions is iterated for variations of the task parameters obtained by randomize selection from within tolerance zones around the nominal task parameters. The size of the tolerance zones and the number of iterations are specified by the designer. Both the task positions and the coordinates of the fixed pivot locations are randomly perturbed.

On the GPU, the main difference between four and six-bar randomization is that while randomization allowed enough threads to be generated for the four-bar problem, the six-bar problem has to track 5743 paths, which means 5743 threads running concurrently. Particularly these threads use more resources those of the four-bar problem. This means that, rather than solving multiple sets of task positions concurrently on the GPU, for the six-bar problem we must solve these sets of task positions one at a time.

7.4 CPU-based Parallel Six-Bar Synthesis System

In this section we present a system of clustered workstations that allow for rapid design of a watt 1 six-bar linkage. This presents an alternative way of accelerating the computation of the solution set to the design equations, but requires multiple computers to operate.

7.4.1 Parallel Solution of the Design Equations

The parameter homotopy obtained from the design equations was integrated into our design system for parallel computation on a Linux cluster. A Python script populates this system with the user-defined task, and manages the execution of *Bertini* to obtain solutions to the design equations. This consists of distributing the data necessary for path tracking from the starting point to a solution, and then collecting the solution data.

The Linux cluster communicates between the computer nodes using the University of California, Irvine, local network. Each computer generates SSH keys, which are sent to every other computer in the network to allow each node to communicate autonomously with every other node without a password. One node is selected as the head to coordinate the other nodes by distributing starting points and collecting solutions, while simultaneously performing its own path tracking.

Bertini generates intermediate output files as it solves a parameter homotopy, which results in a large number of file operations. This is managed by a shared network hard drive on a separate computer system running OS/X. Figure 7.2 shows the cluster's structure.

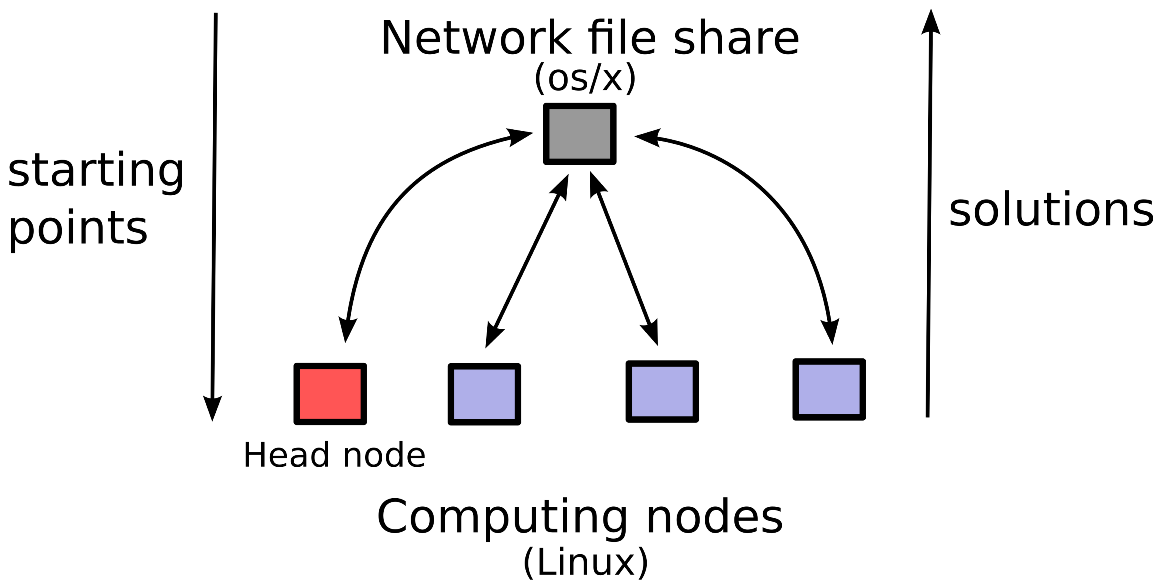


Figure 7.2: Parallel computation is managed through network file sharing. Each not receives starting points and returns the resulting solutions.

Bertini was installed separately on each node of the cluster using *flex* and *bison* programs, as well as the *gmp*, *mpfr* and the *openmpi* libraries that are part of the Ubuntu Linux Package Management System [72]. *Bertini* is configured to use the make build automation tool to compile.

7.4.2 Filter and Iterate

Each solution obtained from the parameter homotopy provides the joint coordinates $\{A, B, C, D, F, G, H\}$ of the linkage in the first task position and the set of relative crank angles $\{S_1, S_2, ..., S_5\}$ that position the system in the other task positions. The size of each solutions is measured against packaging constraints $\{x_{min}, x_{max}\}$ and $\{y_{min}, y_{max}\}$ provided by the designer. That is, the set of joint coordinates are evaluated to ensure

$$x_{min} \leq x_i \leq x_{max} \quad y_{min} \leq y_i \leq y_{max}, \qquad (x_i, y_i) \in \{A, B, C, D, F, G, H\} \tag{7.12}$$

This is a rapid calculation that eliminates designs that are impractical.

Solutions that fit the packaging constraints are analyzed to verify smooth movement through the task positions. We determine the initial crank angle S_0 of the crank BD relative to the x-axis, and add it to the relative angles S_j to obtain

$$\{\Psi_1, \Psi_2, \Psi_3, ..., \Psi_6\} \quad = \quad \{S_0, S_1 + S_0, S_2 + S_0, ..., S_5 + S_0\} \tag{7.13}$$

Then, for each set of Ψ_i, the linkage is analyzed to determine the position of the end-effector in each assembly. Realizable linkages are those that reach each of the task positions in a single assembly. This required for successful operation of the linkage.

This process of generating and evaluating solutions is iterated for variations of the task parameters obtained by randomize selection from within tolerance zones around the nominal task parameters. The size of the tolerance zones and the number of iterations are specified by the designer.

7.4.3 Candy Coating Mechanism

In order to demonstrate the performance of this design system, we seek a linkage that selects an item, moves it through three stations, and returns it to the original location. Furthermore the location of the base of the device is specified. See Figure 7.3. The data for this design is presented in Table 7.1. We chose mounting pivots for the coating mechanism to be $A =$ (228.15mm, 584.34mm) and $B =$ (104.31mm, 463.84mm) with a tolerance of $\pm$10mm in x and y directions.

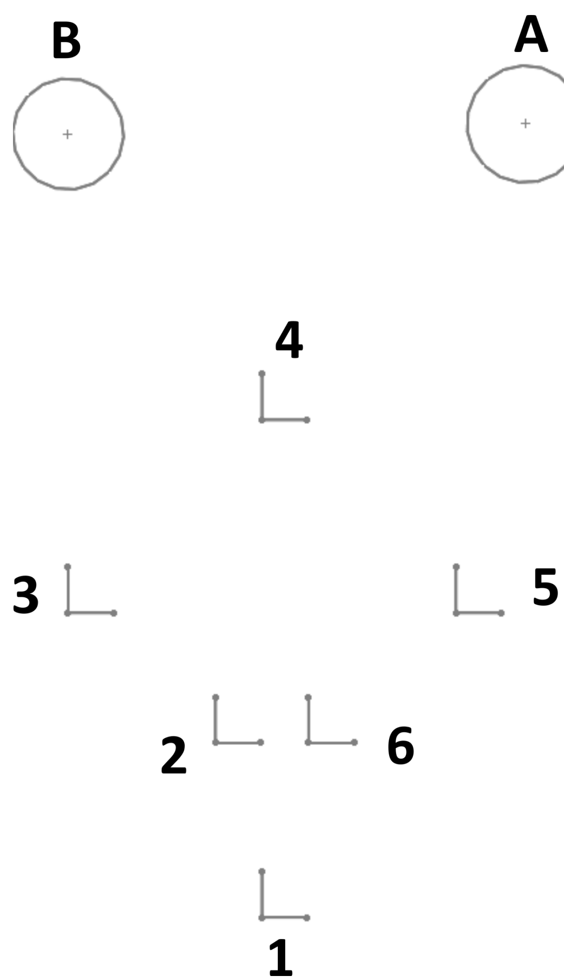


Figure 7.3: The six task positions and fixed pivot locations A and B of the candy coating machine

For this calculation, the design system ran on the cluster using only two Lenovo Workstations. Each solution of the parameter homotopy required approximately three minutes of

Table 7.1: Prescribed task positions of the candy coater.

Task Position	Angle (in degrees)	Angle Tolerance ($\pm$ degrees)	Location (x, y) in mm.	Coordinate Tolerance ($\pm$ mm)
1	0.060	0.1	(20.00, -12.27)	2.0
2	0.034	0.1	(-39.99, 127.77)	2.0
3	359.99	0.1	(-127.21, 192.00)	2.0
4	359.99	0.1	(19.06, 318.34)	2.0
5	0.058	0.1	(129.14, 211.72)	2.0
6	359.9	0.1	(38.21, 120.79)	2.0

Table 7.2: Joint coordinates of the candy coater.

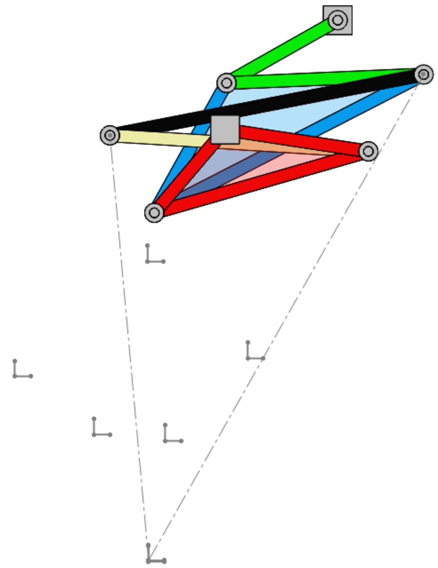
Pivot	Location (x, y) in mm.
A	(228.15, 584.34)
B	(104.31 , 463.84)
C	(127.55, 682.36)
D	(25.32, 55.46)
F	(56.88, 311.73)
G	(342.81, 655.56)
H	(-2.61, 590.24)

computation time. 220 iterations produced 55,130 solutions of which only 12,581 candidates met the packaging constraints. The performance of each these candidates was evaluated and yielded 26 realizable designs. The time required for these design calculations was 41 hours, the majority of which is devoted to the verification calculations.

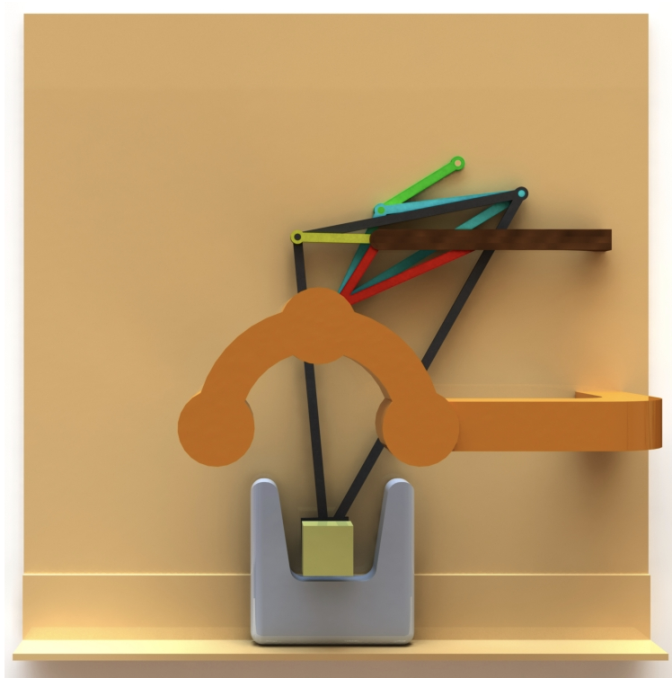
Nine of the verified designs were found to have rotating driving cranks, which is convenient, and were of similar in size and shape. Table 7.2 lists the coordinates of one of these nine designs and Figure 7.4 is a solid model illustrating the device. Figure 7.5 illustrates the motion of this device.

7.4.4 Conclusion

This section presents a six-bar linkage design system that includes the parallel execution of polynomial homotopy solver. A Linux cluster of Lenovo workstations was implemented to



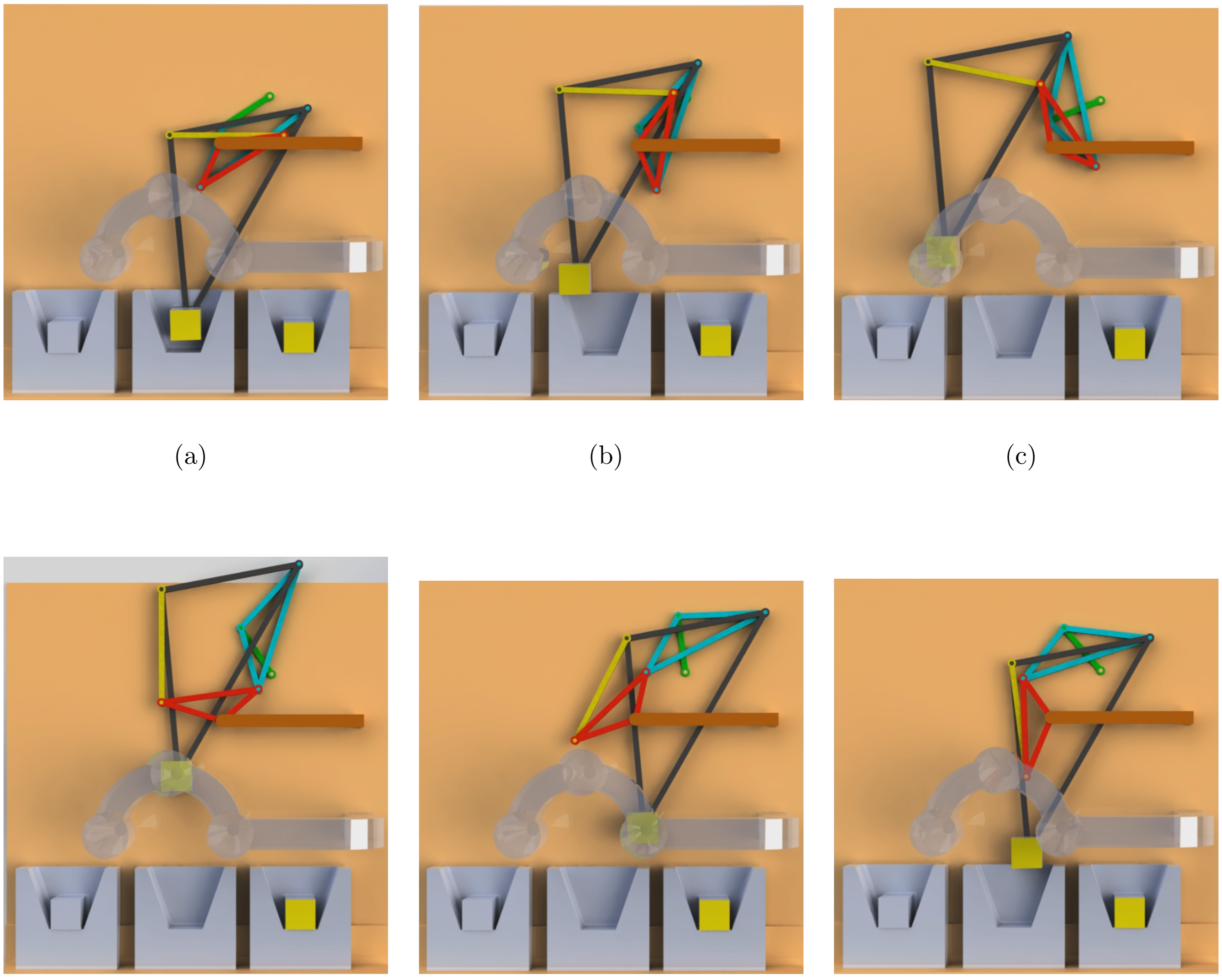
(a) Candy coater, first task position



(b) Solid model of the candy coater

Figure 7.4: Solid models of the candy coater six-bar linkage.

solve this parameter homotopy, and verify the resulting designs. An example design task required 41 hours to compute 26 realizable designs.



(a) (b) (c)

(d) (e) (f)

Figure 7.5: Single frames from an animation demonstrating the linkage in each of the six task positions

7.5 GPU-Accelrated Six-Bar Synthesis Implementation

Similar to the four-bar synthesis implementation, in this section we detail which parts are executed by host (CPU) and the device (GPU). The six-bar implementation is similar, but not the same as the four-bar implementation. The six-bar implementation first begins by reading in the task data as specified by the designer, at which point it transfers the data to the GPU.

7.5.1 Task Generator

As stated above, the six-bar synthesis problem is complex enough to generate a large enough amount of threads (5743 paths per task position set). Only one set of task positions can be processed at a time. Therefore the kernel function Task Generator can be omitted from the six-bar implementation.

7.5.2 Path Tracker

Similar to Path Tracker discussed in section 6.4.2, we used the parameters $\mathbf{q}$ computed by Bertini to form the parameter homotopy,

$$\mathcal{H} = \mathcal{P}(\mathbf{q}\,t + \mathbf{p}\,(1-t), \mathbf{x}). \tag{7.14}$$

where $\mathbf{p}$ is the set of task positions as specified by the designer. $\mathcal{H}$ is first computed by Mathematica [73] as well as the symbolic expressions for the 20x20 Jacobian matrix $\mathbf{J_x}$ and the 20x1 vector $\mathbf{H_t}$. These expressions are then converted into the CUDA code used by Path Tracker. It is important to mention that these equations are much more complex than those found in the four-bar implementation. A value of $L = 400$ steps from $t = 1$ to $t = R$ and 3 iterations of Newton's method were found to give a good number of solutions before beginning the engame.

7.5.3 Endgame

Once the paths have been tracked from $t = 1$ to $t = R$, where $R = 0.1$ was the endgame boundary tuned for the six-bar design problem, the design system switches into the endgame. This value of R was shown to be within the endgame convergence zone. At this point the endgame begins by sampling the sequence $(t_1, t_2, ...,) = (R, \lambda R, \lambda^2 R, ..., \lambda^\kappa R)$ for $\kappa + 1$ samples. 25 samples with a value of $\lambda = 0.8$ were found to give good results for this problem.

One the sampling has been completed, the endgame then begins interpolation. While it is possible to parallelize on both the guesses of the cycling number c as well as each of the predicted values of the unknown vector $\phi_c(\mathbf{s}_{\kappa+1})$ as described in section 5.4, this resulted in too many threads for the GPU to handle. Instead, the endgame sequentially checks $c = 1, 2, ..., c_m ax$ and then calculates the values of $\phi_c(\mathbf{s}_{\kappa+1})$ as well as the predicted solution $\phi_c(0)$ in parallel using polynomial interpolation. After calculating $\phi_c(\mathbf{s}_{\kappa+1})$, the host then checks the residual error $||\bar{\mathbf{x}}(t_{\kappa+1}) - \phi_c(\mathbf{s}_{\kappa+\mathbf{1}})||$ and stores the predicted value $\phi_c(0)$ if the residual is minimal for the current c.

7.5.4 Solution Filter

Similar to the derivation of the four-bar synthesis equations, for the solution to have true physical meaning, the solutions must be true complex conjugate pairs. This means that the unknown solution coordinates $(C, \bar{C}), (D, \bar{D}), (F, \bar{F}), (G, \bar{G}), (H, \bar{H})$ must be complex conjugate pairs. Additionally, the relative crank angle pairs $(S_1, \bar{S_1}), (S_2, \bar{S_2}), ..., (S_5, \bar{S_5})$ must also be complex conjugate pairs as described in (7.6) . The kernel function Solution Filter checks to see if this condition is satisfied and flags solutions that pass the check so they can be transferred back to the host for analysis. Figure 7.6 outlines the computational flow of these kernel functions within the design system as a whole.

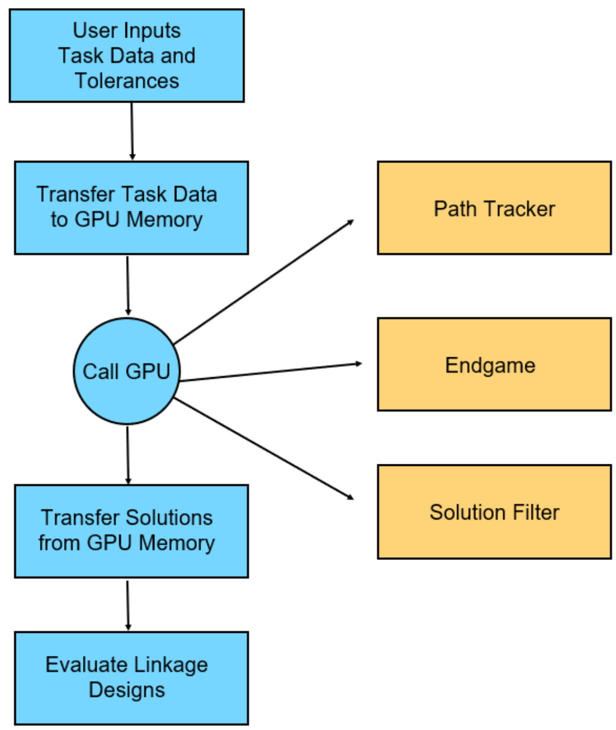


Figure 7.6: Flow of calculations on the CPU (on the left) with calls to the GPU on the right for the six-bar watt 1 design problem.

7.5.5 Linkage Analysis

Each solution obtained from the parameter homotopy provides the joint coordinates $\{A, B, C, D, F, G, H\}$ of the linkage in the first task position and the set of relative crank angles $\{S_1, S_2, ..., S_5\}$ that position the system in the other task positions.

Solutions with physical meaning are analyzed to verify smooth movement through the task positions. We determine the initial crank angle S_0 of the crank BD relative to the x-axis, and add it to the relative angles S_j to obtain

$$\{\Psi_1, \Psi_2, \Psi_3, ..., \Psi_6\} = \{S_0, S_1 + S_0, S_2 + S_0, ..., S_5 + S_0\} \tag{7.15}$$

Then, for each set of Ψ_i, the linkage is analyzed to determine the position of the end-effector in each assembly. Realizable linkages are those that reach each of the task positions in a

Table 7.3: Task positions for the six-bar design problem.

j	θ_j (deg)	P_j
1	-0.32	(0.03, 0.04)
2	-3.23	(0.04, 32.22)
3	3.89	(-12.59, 19.78)
4	-0.5	(12.57, 19.72)
5	3.11	(-3.01, 11.29)
6	0.95	(2.98, 11.35)

single assembly. This required for successful operation of the linkage.

7.6 Demonstration

In this section, we demonstrate the algorithm and compare performance between a CPU implementation of the same system using a Lenovo workstation with an Intel Xeon 2.10GHz CPU running Windows 10 and a GPU implementation with an NVIDIA Quadro M2000 GPU. The six task positions are shown in Table 7.3 and pictured in Figure 7.7. We chose the two fixed pivot locations to be $A = -12.58 + 50.85i$ and $B = 17.14 + 51.48i$.

The task position set was solved using a Python script and Bertini v1.5.1 for the CPU implementation as well as the GPU-accelerated design system using CUDA and C++. The CPU implementation ran Bertini in parallel on 8 threads while the GPU ran on 5743 threads, one for each starting point.

The results of both the CPU and GPU implementations can be found in Table 7.4. The CPU implementation was roughly 14x faster than the GPU implementation, requiring 550 seconds of computation time compared to 7337 seconds on the GPU. Additionally the CPU obtained more solutions out the endgame, 5544 versus the GPU with 142. This resulted in the CPU obtaining more solutions that passed the Solution Filter check at 395 versus

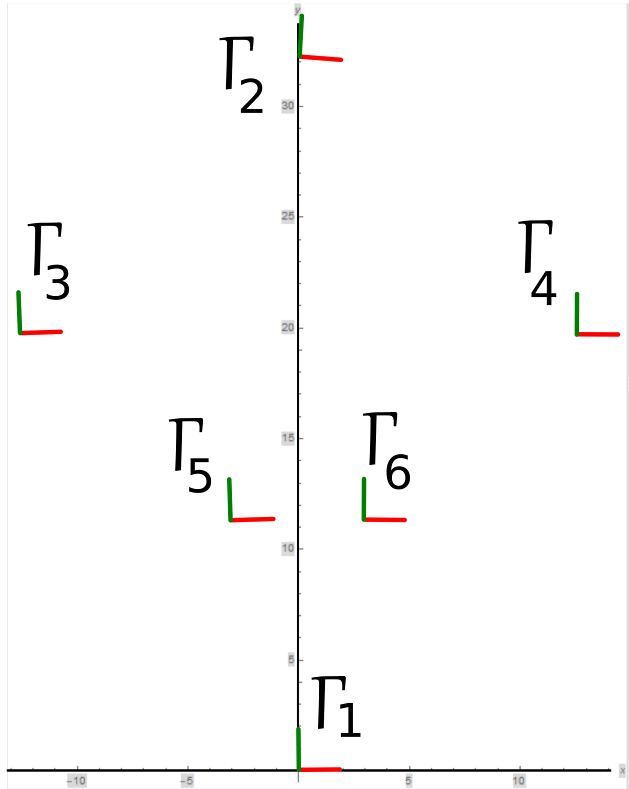


Figure 7.7: The six task positions in the global frame.

Hardware	Time (sec)	Solutions	Physical Sols.	Effective Sols.
CPU	550	5544	395	1
GPU	7337	142	2	1

Table 7.4: Comparison of kinematic synthesis of 8 parallel threads on a workstation CPU with our algorithm on 5743 threads on a GPU for the six-bar design problem.

2 on the GPU. However, both the CPU and GPU found the same effective solution. This solution's joint coordinates are listed in Table 7.5 and a 3D model of the solution is shown in each of the task positions in Figure 7.8.

7.6.1 Discussion

There are a few possible reasons for the poor performance of the GPU implementationn. The first is the low number of solutions resulting from the endgame. At the point $t = R$, the

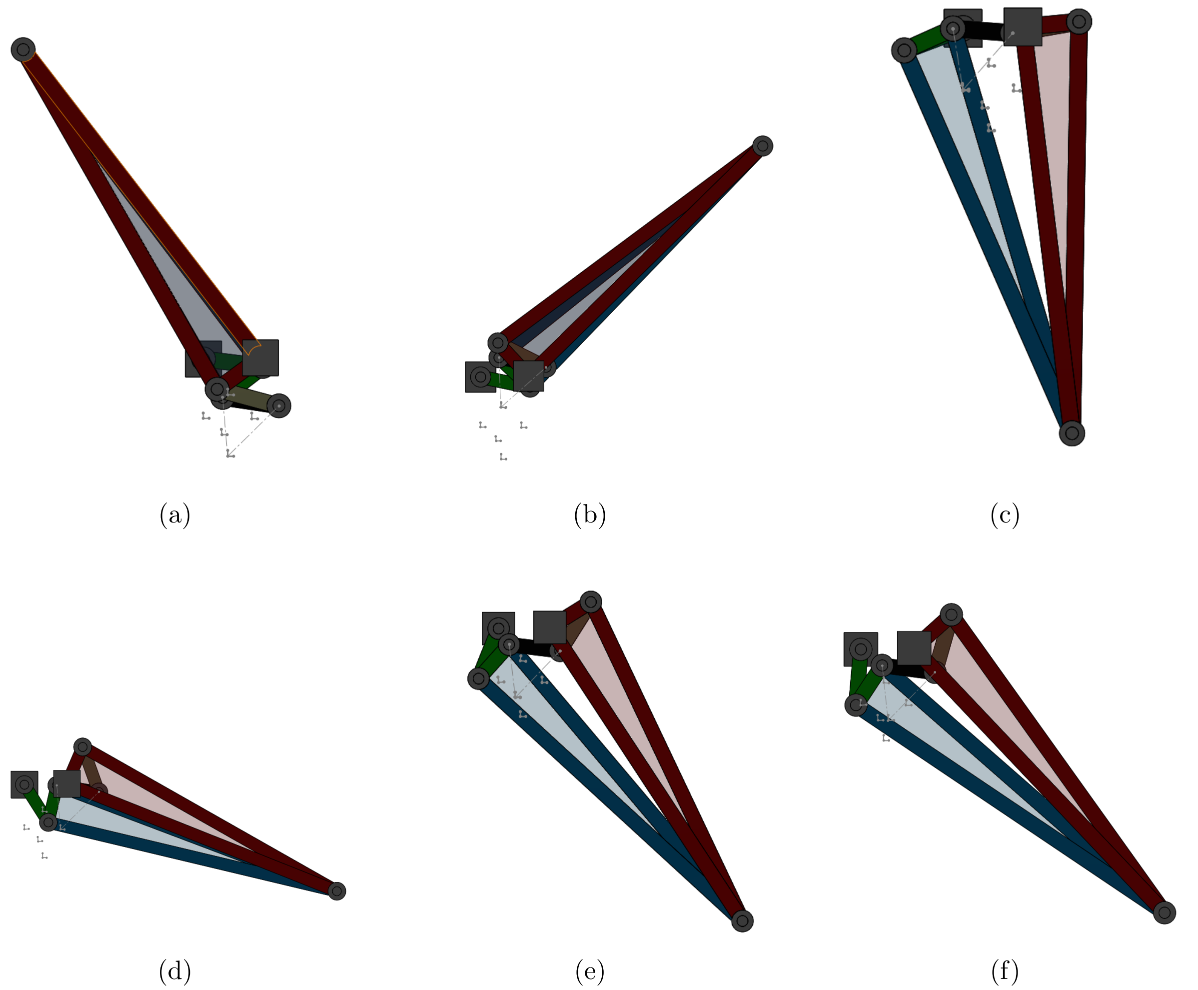


Figure 7.8: An example six-bar linkage design moving the end-effector through the six task positions.

Table 7.5: Joint coordinates in global frame for the six-bar solution.

Point	(x, y) coordinates
A	(-12.58, 50.85)
B	(17.14, 51.48)
C	(18.34, 46.97)
D	(-107.35, 211.72)
F	(5.35, 34.96)
G	(-2.56, 30.74)
H	(26.82, 26.24)

path tracker had successfully tracked 3324 paths such that $\mathbf{H}(x, R) = 0$. This means that only about 4 percent of potential solutions were successfully found from the endgame.

The current implementation of the endgame uses Newtonian interpolation. This can be improved, however, by saving the derivatives as well as the sample points and performing Hermite interpolation to improve the accuracy of the interpolating polynomial. This could potentially increase the number of successful solutions found by the endgame.

The next issue to discuss is register usage. As mentioned in Section 7.5.2, the expressions for $\mathbf{H}, \mathbf{H_x}$, and $\mathbf{H_t}$ are quite complex. When compiling these expressions into a CUDA executable, the compiler ends up using a lot of register space to hold the intermediate values of the resulting expressions. This has a result of using up all of the available registers in the SM, forcing it to use global memory. There is so much memory used in these expressions, in fact, that the GPU runs out of available global memory. This certainly decreases the occupancy and results in under utilization of the GPU. Possible fixes for this issue would be to use automatic differentiation to sufficiently approximate $\mathbf{H_x}$ and $\mathbf{H_t}$ as well as accelerating the computation of these expressions by using methods similar to those in [57].

Finally, it is important to note that there are advantages of a working implementation on a GPU. Currently, GPU performance is increasing much more rapidly than CPU performance. The Maxwell architecture released in 2015 that the NVIDIA M2000 GPU runs on has 6 SM units, while the most current Ampere architecture released in 2020 has 128 SMs per GPU while also having double the amount of memory. Running the code on a more modern GPU would certainly decrease the computation time. Additionally, GPU computing performance scales better than CPU performance. It is easier to add more GPU units to a computer than to add CPU units and CUDA code scales to multi-GPU systems relatively easy. More research should be performed in these areas to further increase GPU performance.

Chapter 8

Conclusion

This dissertation has presented a novel GPU-accelerated method for kinematic synthesis. The problem of solving the design equations found in kinematic synthesis is known to be very computationally intensive, involving finding the solution set to a system of polynomials, typically of high degree. The method of homotopy continuation is able to find the solution set to systems of polynomials by splitting the problem into several sub-problems with little to no interdependence. This makes the method easily parallelizable. Graphics Processing Units (GPU) are pieces of hardware developed to execute code in parallel on large sets of data. In this dissertation, we have shown how to organize a naturally parallelizable computation for efficient execution on a GPU to yield good results.

First, the method of path tracking for homotopy continuation was considered and optimized to perform on a GPU suitably for kinematic synthesis. Next, the endgame methods were considered and the power series method chosen as the method best suited for GPU execution. Both the path tracking algorithm and power series endgame were then implemented into CUDA code and shown to work.

This has resulted in the first known GPU-accelerated four-bar linkage design system. This system takes advantage of the randomization property of kinematic synthesis to generate thousands of threads, making it ideal to be executed on a GPU. The design system was compared to a similar CPU-based system and was shown to produce 4425 solutions, of which 2382 had physical meaning, and 85 solutions that met design requirements. The GPU-accelerated system was able to do this in 63 seconds which compares favorably to the runtime of 7588 seconds for the CPU-based system.

The results of the four-bar linkage design system were then extended to the problem of six-bar linkage synthesis. A system was developed utilizing the same GPU-based path tracker and endgame. This has resulted in the first known GPU-accelerated six-bar linkage design system. While not able to perform favorably against a similar CPU based system, this new design system presents new opportunities for multi-GPU systems capable of designing even more complicated mechanisms.

8.1 Future Work

One of the natural ways of extending this research is to apply the methods described to multi-GPU systems. NVIDIA has provided a relatively easy way to allow code to be ran on multiple GPUs. With the advent of methods such as bitcoin mining, multi-GPU systems have become ubiquitous. This would take advantage of the better scaling of GPU performance to allow for even faster computation.

Additionally, the development of the Watt 1 six-bar linkage design system can be extended to the other six-bar topologies with relative ease. The loop equations for the Watt II and Stephenson topologies can be placed into the developed Mathematica code and then be automatically converted to CUDA code. This would allow for design of all six-bar topologies and could even be extended to more complex mechanisms such as eight-bar or spatial linkages.

Finally, being able to rapidly find solutions could allow for insight into the solution space for the kinematic synthesis problem. These GPU-accelerated design systems could be used to rapidly generate training sets for the development of reduced-order models via machine learning. These models could reduce the necessity of high performance computation systems and allow for software to be developed to allow for complex mechanism design on a wide range of computers.

Bibliography

[1] Arthur Schoenflies. Geometrie der bewegung in synthetischer darstellung. *Leipzig, B. G. Teubner, 1886. 8vo, pp. vi + 194.*

[2] L. Burmester. *Lehrbuch der Kinematic.* Verlag Von Arthur Felix, Leipzig, Germany, 1886.

[3] F Freudenstein. An analytical approach to the design of four-link mechanisms. *Trans. ASME*, 76(3):483–492, 1954.

[4] A. Erdman, G. Sandor, and S. Kota. Mechanism design: Analysis and synthesis. 4, 2001.

[5] J. M. McCarthy and G. S. Soh. *Geometric Design of Linkages.* Springer-Verlag, New York, 2nd edition, 2011.

[6] C. W. Wampler, A. P. Morgan, and A. J. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *ASME J. Mech. Des*, 114:153–159, 1992.

[7] D. A. Brake, J. D. Hauenstein, and A. P Murray. Myszka, d. *H., Wampler, C.W., "The Complete Solution of Mixed Burmester Synthesis Problems for Four-Bar Linkages," ASME. J. Mechanisms Robotics*, 8(4):8–8, 2016.

[8] J. Glabe and J. M. McCarthy. Six-bar linkage design system with a parallelized polynomial homotopy solver. In J. Lenarcic and V. Parenti-Castelli, editors, *Advances in Robot Kinematics 2018. ARK 2018.* Springer Proceedings in Advanced Robotics, vol 8. Springer, Cham, 2018.

[9] A. Purwar, S. Deshpande, and Q. J. Ge. *MotionGen: Interactive Design and Editing of Planar Four-Bar Motions for Generating Pose and Geometric Constraints.* Journal of Mechanisms and Robotics , 2017.

[10] J. M. McCarthy and Choe J.. Difficulty of kinematic synthesis of usable constrained planar 6r robots. In M. Stanisic, editor, *Lenarcic J*, pages 978–90. Motion in Man and Machine. Springer, Dordrecht, Advances in Robot Kinematics, 2010.

[11] R. S. Hartenberg and J. Denavit. *Kinematic synthesis of linkages.* McGraw-Hill, 1964.

[12] CW McLarnan. Synthesis of six-link plane mechanisms by numerical analysis. *Journal of Engineering for Industry*, 85(1):5–10, 1963.

[13] AV Mohan Rao and GN Sandor. Extension of freudenstein's equation to geared linkages. *Journal of Engineering for Industry*, 93(1):201–210, 1971.

[14] AK Dhingra, JC Cheng, and D Kohli. Synthesis of six-link, slider-crank and four-link. mechanisms for function, path and motion generation using homotopy with m-homogenization. *Journal of Mechanical Design*, 116:1122–1122, 1994.

[15] Mark M Plecnik and J Michael McCarthy. Numerical synthesis of six-bar linkages for mechanical computation. *Journal of Mechanisms and Robotics*, 6(3):031012, 2014.

[16] Mark M Plecnik and J Michael McCarthy. Design of stephenson linkages that guide a point along a specified trajectory. *Mechanism and Machine Theory*, 96:38–51, 2016.

[17] Mark M Plecnik and J Michael McCarthy. Kinematic synthesis of stephenson iii six-bar function generators. *Mechanism and Machine Theory*, 97:112–126, 2016.

[18] Mark M Plecnik and J Michael McCarthy. Computational design of stephenson ii six-bar function generators for 11 accuracy points. *Journal of Mechanisms and Robotics*, 8(1):011017, 2016.

[19] G. S. Soh and J. M. McCarthy. Synthesis of eight-bar linkages as mechanically constrained parallel robots. *In Proceedings of the 12th IFToMM world congress*, June 2007.

[20] Gim Song Soh and Fangtian Ying. Dimensional synthesis of planar eight-bar linkages based on a parallel robot with a prismatic base joint. In *Volume 6A: 37th Mechanisms and Robotics Conference*. American Society of Mechanical Engineers, aug 2013.

[21] Kaustubh H. Sonawale and J. Michael McCarthy. Synthesis of useful eight-bar linkages as constrained 6r loops. In *Volume 5A: 38th Mechanisms and Robotics Conference*. American Society of Mechanical Engineers, aug 2014.

[22] J. Mueller. Design procedures for the determination of dimensions of eight-bar and ten-bar linkages. *Dissertation, Technische Universitat Dresden.*

[23] J. Mueller. On the design of eight-bar linkages without using four-bar linkages. *ournal of Engineering for Industry, 3:213217,*, 1954.

[24] S. Hamid and A. H. Soni. Synthesis of an eight-link mechanisms for varieties of motion programs. journal of engineering for industry. *Journal of Engineering for Industry, (72-Mech-49):744750*, August 1973.

[25] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. *Numerically Solving Polynomial Systems with Bertini*. SIAM Press, 2013.

[26] N Bliss, J Sommars, J Verschelde, and Yum X. Solving polynomial systems in the cloud with polynomial homotopy continuation. In V Gerdt, W Koepf, W. Seiler, and E Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, Lecture Notes in Computer Science. Springer, Cham, 2015.

[27] H. Su, J. M. McCarthy, M. Sosonkina, and L.T. Watson. Algorithm 857: Polsys-glp—a parallel general linear product homotopy code for solving polynomial systems of equations. *ACM Transactions on Mathematical Software*, 32(4):561–579, December 2006.

[28] H. Su, J. M. McCarthy, and L.T. Watson. Generalized linear product homotopy algorithms and the computation of reachable surfaces. *Journal of Computing and Information Science in Engineering*, 4:226–234, September 2004.

[29] M. M. Plecnik and J. M. McCarthy. Computational design of stephenson ii six-bar function generators for 11 accuracy points. *Journal of Mechanisms and Robotics*, 8, February 2016.

[30] A. J. Sommese and C. W. Wampler. The numerical solution of systems of polynomials arising in engineering and science. *World Scientific*, 2005.

[31] F. Freudenstein and B. Roth. Numerical solution of systems of nonlinear equations. *Journal of the ACM*, 10(4):55–556, 1963.

[32] B. Roth and F. Freudenstein. Synthesis of path-generating mechanisms by numerical methods. *Journal of Engineering for Industry*, pages 298–304, August 1963.

[33] W. I. Zangwill and C. B. Garcia. *Pathways to Solutions, Fixed Points and Equilibria.* Prentice-Hall, Inc, 1981.

[34] A. P. Morgan. A method for computing all solutions to systems of polynomials equations. *ACM Transactions on Mathematical Software*, 9(1):1–17, 1983.

[35] L. W. Tsai and A. P. Morgan. Solving the kinematics of the most general six- and five-degree-of- freedom manipulators by continuation methods. *Journal of Mechanisms, Transmissions and Automation in Design*, 107, June 1985.

[36] Naser Mostashiri, Alireza Akbarzadeh, and Amir Rezaei. Implementing the homotopy continuation method in a hybrid approach to solve the kinematics problem of spatial parallel robots. *Intelligent Service Robotics*, 10(3):257–270, mar 2017.

[37] Jaime Gallardo-Alvarado. An application of the newton-homotopy continuation method for solving the forward kinematic problem of the 3-RRS parallel manipulator. *Mathematical Problems in Engineering*, 2019:1–6, jan 2019.

[38] J. Verschelde. Polynomial homotopy continuation with phcpack. *ACM Communications in Computer Algebra*, 44(3/4):217–220, 2010.

[39] T. Y. Li and C.-H. Tsai. Hom4ps-2.0para: Parallelization of hom4ps-2.0 for solving polynomial systems. *Parallel Computing*, 35(4):226–238, 2009.

[40] M. M. Plecnik and J. M. McCarthy. Design of stephenson linkages that guide a point along a specified trajectory. *Mechanism and Machine Theory*, 96, Part 1:38–51, February 2016.

[41] M. M. Plecnik, D. W. Haldane, J. K. Yim, and R. S. Fearing. Design exploration and kinematic tuning of a power modulating jumping monopod. *Journal of Mechanisms and Robotics*, 9(011009):1, 2017.

[42] D. W. Haldane, M. M. Pecnik, J. K. Yim, and R. S. Fearing. Robotic vertical jumping agility via series-elastic power modulation. *Science Robotics*, 1:1, December 2016.

[43] Mark M. Plecnik and Ronald S. Fearing. Finding only finite roots to large kinematic synthesis systems. *Journal of Mechanisms and Robotics*, 9(2), mar 2017.

[44] Mark M. Plecnik and Ronald S. Fearing. A study on finding finite roots for kinematic synthesis. In *Volume 5B: 41st Mechanisms and Robotics Conference*. American Society of Mechanical Engineers, aug 2017.

[45] M. Parker. *Digital Signal Processing 101: Everything you need to know to get started*. Elsevier Inc, 2017.

[46] Quansheng Liu, Weiqin Wang, and Hao Ma. Parallelized combined finite-discrete element (FDEM) procedure using multi-GPU with CUDA. *International Journal for Numerical and Analytical Methods in Geomechanics*, 44(2):208–238, dec 2019.

[47] Jia Luo and Didier El Baz. A dual heterogeneous island genetic algorithm for solving large size flexible flow shop scheduling problems on hybrid multicore CPU and GPU platforms. *Mathematical Problems in Engineering*, 2019:1–13, mar 2019.

[48] C. Hori, H. Gotoh, H. Ikari, and A. Khayyer. Gpu-acceleration for moving particle semi-implicit method. *Computers and Fluids*, 51:174–183, 2011.

[49] Kevin Verma, Christopher McCabe, Chong Peng, and Robert Wille. A PCISPH implementation using distributed multi-GPU acceleration for simulating industrial engineering applications. *The International Journal of High Performance Computing Applications*, page 109434202090619, feb 2020.

[50] Jianqi Lai, Hua Li, Zhengyu Tian, and Ye Zhang. A multi-GPU parallel algorithm in hypersonic flow computations. *Mathematical Problems in Engineering*, 2019:1–15, mar 2019.

[51] J. Pan, C. Lauterbach, and D. Manocha. g-planner: Real-time motion planning and global navigation using gpu. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI, 2010.

[52] B. Ichter, E. Schmerling, A. Agha-mohammadi, and M. Pavone. Real-time stochastic kinodynamic motion planning via multiobjective search on gpus. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[53] C. Dick, J. Georgii, and R. Westermann. A real-time multigrid finite hexahedra method for elasticity simulation using cuda. *Simulation Modelling Practice and Theory*, 19(2):801–816, 2011.

[54] Nicolas H. Lehment, Dejan Arsic, Moritz Kaiser, and Gerhard Rigoll. Automated pose estimation in 3d point clouds applying annealing particle filters and inverse kinematics on a GPU. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. IEEE, jun 2010.

[55] Omar Alejandro Aguilar and Joel Carlos Huegel. Inverse kinematics solution for robotic manipulators using a CUDA-based parallel genetic algorithm. In *Advances in Artificial Intelligence*, pages 490–503. Springer Berlin Heidelberg, 2011.

[56] Pawan Harish, Mentar Mahmudi, Benoît Le Callennec, and Ronan Boulic. Parallel inverse kinematics for multithreaded architectures. *ACM Transactions on Graphics*, 35(2):1–13, may 2016.

[57] J. Verschelde and G. Yoffe. Evaluating polynomials in several variables and their derivatives on a gpu computing processor. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum*. IEEE Computer Society, 2012.

[58] Jan Verschelde and Xiangcheng Yu. Polynomial homotopy continuation on GPUs. *ACM Communications in Computer Algebra*, 49(4):130–133, feb 2016.

[59] J. Verschelde and X. Yu. Tracking many solution paths of a polynomial homotopy on a graphics processing unit in double double and quad double arithmetic. In *2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC)*. IEEE, 2015.

[60] Jan Verschelde and Xiangcheng Yu. Accelerating polynomial homotopy continuation on a graphics processing unit with double double and quad double arithmetic. *CoRR*, abs/1501.06625, 2015.

[61] A. P. Morgan, A. J. Sommese, and C. W. Wampler. A power series method for computing singular solutions to nonlinear analytic systems. *Numerische Mathematik*, 63:391–409, 1992.

[62] NVIDIA Corporation. *CUDA C Programming Guide*. 2018.

[63] W. Cheney and D. R. Kincaid. *Numerical Mathematics and Computing*. Brooks Cole Pub. Co., 2012.

[64] J. Glabe and J. M. McCarthy. Five position synthesis of a planar four-bar linkage. In Springer International Publishing, editor, *Advances in Mechanism and Machine Science*, pages 599–606. 2019.

[65] T. R. Chase and J. A. Mirth. Circuits and branches of single degree-of-freedom planar linkages. *Journal of Mechanical Design*, 115:223–230, June 1993.

[66] A. S. Beloiu and K. C. Gupta. A unified approach for the investigation of branch and circuit defects. *Mechanism and Machine Theory*, 32(4):539–557, 1997.

[67] M M Plecnik and J M McCarthy. Design of a 5-ss spatial steering linkage. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conferences*, volume Mechanisms and Robotics Conference. ASME, August 2012.

[68] Brandon Y Tsuge, Mark M Plecnik, and J Michael McCarthy. Homotopy directed optimization to design a six-bar linkage for a lower limb with a natural ankle trajectory. *Journal of Mechanisms and Robotics*, 8(6):061009, 2016.

[69] Stephen Wolfram. *The Mathematica Book, Fifth Edition.* Wolfram Media, 2003.

[70] J. J. Uicker, G. R. Pennock, and J. E. Shigley. *Theory of Machines and Mechanisms.* Oxford University Press, fifth edition, 2016.

[71] Mark Plecnik, J. Michael McCarthy, and Charles W. Wampler. Kinematic synthesis of a watt i six-bar linkage for body guidance. In Jadran Lenarčič and Oussama Khatib, editors, *Advances in Robot Kinematics*, pages 317–325, Cham, 2014. Springer International Publishing.

[72] Xenial Xerus. *Ubuntu 16.04.* Canonica Ltd., 2016.

[73] Inc. Wolfram. High-performance numeric solution of polynomial systems. 2018.

Appendix A

Mathematica Code

This section contains the code used to formulate the parameter homotopy as well calculate the values of $\mathbf{J_x}$ and $\mathbf{H_t}$. It exports all the values to text files that can be then copied to CUDA.

A.1 Four-Bar Synthesis

```
Four-bar motion synthesis using complex numbers - Parameter Homotopy
derivation
Jeffrey Glabe, January 2019
Constraint Equations
$Assumptions = Element[{O, Oc, A, Ac, B, Bc, C, Cc, P1, Pc1, P2,
Pc2, P3, Pc3, P4, Pc4, T1, Tc1, T2, Tc2, T3, Tc3, T4, Tc4}, Complexes];
f1 =(P1 + T1* A - O)(Pc1 + Tc1* Ac - Oc) - (A - O)(Ac - Oc);
f2 = (P1 + T1*B - C)(Pc1  + Tc1* Bc - Cc) - (B - C)(Bc - Cc);
```

```
f3 = (P2 + T2*A - O)(Pc2 + Tc2* Ac - Oc) - (A - O)(Ac - Oc);
f4 = (P2 + T2*B - C)(Pc2 + Tc2* Bc - Cc) - (B - C)(Bc - Cc);
f5 = (P3 + T3*A - O)(Pc3 + Tc3* Ac - Oc) - (A - O)(Ac - Oc);
f6 = (P3 + T3*B - C)(Pc3 + Tc3*Bc - Cc) - (B - C)(Bc - Cc);
f7 = (P4 + T4* A - O)(Pc4 + Tc4* Ac - Oc) - (A - O)(Ac - Oc);
f8 = (P4 + T4*B - C)(Pc4 + Tc4*Bc - Cc) - (B - C)(Bc - Cc);
ConstraintEquations = {f1, f2, f3, f4, f5, f6, f7, f8};
UnknownVars = {O, Oc, A, Ac, B, Bc, C, Cc};
Parameters = {P1, Pc1, P2, Pc2, P3, Pc3, P4, Pc4, T1, Tc1, T2, Tc2,
T3, Tc3, T4, Tc4};
GenerateParameterRule[var_] := Module[{varString, varStringStart,
varStringFinal, resubRuleString, resubRuleExpr},
varString = ToString[var];
varStringStart = varString <> "Start";
varStringFinal = varString <> "Final";
resubRuleString = varString <> "␣->␣" <> varStringStart <>
"*␣(t*gamma)␣+␣" <> varStringFinal <> "␣*␣(1␣-␣t)";
resubRuleExpr = ToExpression[resubRuleString];

Return[resubRuleExpr];
]
parameterResubRules = Map[GenerateParameterRule, Parameters];
HomotopySystem = Map[# /. parameterResubRules&, ConstraintEquations];
Calculating the Jacobian of the System (JH) and the partial
with respect to t (Ht).
JH = D[HomotopySystem, {UnknownVars}];
Ht = D[HomotopySystem , t];
```

```
Finding the re−substitution rules for JH
BoolMatrix = Table[ Map[Boole[!PossibleZeroQ[#]]&, JH[[i]]],
{i, Length[JH]}]; BoolMatrix // MatrixForm;
AssignSymbol[row_, rowNum_, unknowns_] := Module[{rowVars},
rowVars = Table[
If[PossibleZeroQ[row[[i]]], 0,
ToExpression[ToString[unknowns[[i]]] <> ToString[rowNum]]],
{i, Length[row]}];
rowVars]
SymbolMatrix = Table[AssignSymbol[BoolMatrix[[i]], i, UnknownVars],
{i, Length[BoolMatrix]}] ; SymbolMatrix // MatrixForm;
ResubRules[jsrow_, jrow_, rowNum_] := Module[{rules},
rules = Table[
If[PossibleZeroQ[jsrow[[i]]], 0,
(*ToExpression[ToString[jsrow[[i]]] <> " −> " <> ToString[jrow[[i]]]]],*)
Rule[jsrow[[i]], jrow[[i]]]],
{i, Length[jsrow]}];
rules]
Resubs = Table[Select[ResubRules[SymbolMatrix[[i]], JH[[i]], i],
!PossibleZeroQ[#]&], {i, Length[SymbolMatrix]}];
Exporting Resub Rules
SetDirectory[NotebookDirectory[]];
jacobianCPP = StringDrop["JMat << " <> Map[ToString[#] <> ","&,
Flatten[SymbolMatrix]], −1] <> ";"
Export["JacobianMatrixExpr.txt", jacobianCPP];
JacobianResubRule[rule_] := Module[{matlabCode},
matlabCode = ToString[rule[[1]]] <> " = " <> ToString[rule[[2]]]
```

```
// CForm] <> ";";
Return[matlabCode];
]
ResubSymbols = Table[Map[First, Resubs[[i]]], {i, Length[Resubs]}]
// Flatten;
JacobianRules = Table[Map[JacobianResubRule, Resubs[[i]]],
{i, Length[Resubs]}];
JacobianRules // Dimensions;
With[{file=OpenWrite@FileNameJoin@{NotebookDirectory[],
"JacobianResubs.txt"}},
Do[
Do[
WriteLine[file,JacobianRules[[row, col]]],{col,Length[JacobianRules[[row]]]}
],
{row, Length[JacobianRules]}
];
Close@file];
HtMatLab = Table["Ht" <> ToString[i] <> " = " <> ToString[Ht[[i]] // CForm],
Export["HtResubs.txt", HtMatLab];
HomotopySystemExport = Table["f" <> ToString[i] <> " = " <>
ToString[HomotopySystem[[i]] // CForm ] <> ";", {i, Length[HomotopySystem]}];
With[{file=OpenWrite@FileNameJoin@{NotebookDirectory[],"HomotopySystem.txt"}}
Do[
WriteLine[file,HomotopySystemExport[[i]]],{i,Length[HomotopySystemExport]}
];


Close@file];
```

A.2 Six-Bar Synthesis

```
Six−Bar two fixed pivot parameter homotopy
Jeffrey Glabe, February 2019
SetDirectory[NotebookDirectory[]];
Six−Bar Watt 1 Equations (Two Fixed Pivot) and Homotopy System formulation
f1=2.*B*Bc−1.*Bc*DD−1.*B*DDc+DDc*G+DD*Gc−1.*Bc*P1−1.*B*Pc1+
P1*Pc1−1.*B*Bc*T31+Bc*DD*T31+B*Pc1*T31−1.*DD*Pc1*T31−1.*Bc*
G*T51+G*Pc1*T51−1.*B*Bc*Tc31+B*DDc*Tc31+Bc*P1*Tc31−1.*DDc*
P1*Tc31+Bc*G*T51*Tc31−1.*DDc*G*T51*Tc31−1.*B*Gc*Tc51+Gc*P1*
Tc51+B*Gc*T31*Tc51−1.*DD*Gc*T31*Tc51;

f2=2.*B*Bc−1.*Bc*DD−1.*B*DDc+DDc*G+DD*Gc−1.*Bc*P2−1.*B*Pc2+
P2*Pc2−1.*B*Bc*T32+Bc*DD*T32+B*Pc2*T32−1.*DD*Pc2*T32−1.*Bc*
G*T52+G*Pc2*T52−1.*B*Bc*Tc32+B*DDc*Tc32+Bc*P2*Tc32−1.*DDc*
P2*Tc32+Bc*G*T52*Tc32−1.*DDc*G*T52*Tc32−1.*B*Gc*Tc52+Gc*P2*
Tc52+B*Gc*T32*Tc52−1.*DD*Gc*T32*Tc52;

f3=2.*B*Bc−1.*Bc*DD−1.*B*DDc+DDc*G+DD*Gc−1.*Bc*P3−1.*B*Pc3+
P3*Pc3−1.*B*Bc*T33+Bc*DD*T33+B*Pc3*T33−1.*DD*Pc3*T33−1.*Bc*
G*T53+G*Pc3*T53−1.*B*Bc*Tc33+B*DDc*Tc33+Bc*P3*Tc33−1.*DDc*
P3*Tc33+Bc*G*T53*Tc33−1.*DDc*G*T53*Tc33−1.*B*Gc*Tc53+Gc*P3*
Tc53+B*Gc*T33*Tc53−1.*DD*Gc*T33*Tc53;

f4=2.*B*Bc−1.*Bc*DD−1.*B*DDc+DDc*G+DD*Gc−1.*Bc*P4−1.*B*Pc4+
P4*Pc4−1.*B*Bc*T34+Bc*DD*T34+B*Pc4*T34−1.*DD*Pc4*T34−1.*Bc*
G*T54+G*Pc4*T54−1.*B*Bc*Tc34+B*DDc*Tc34+Bc*P4*Tc34−1.*DDc*
```

```
P4*Tc34+Bc*G*T54*Tc34-1.*DDc*G*T54*Tc34-1.*B*Gc*Tc54+Gc*P4*
Tc54+B*Gc*T34*Tc54-1.*DD*Gc*T34*Tc54;

f5=2.*B*Bc-1.*Bc*DD-1.*B*DDc+DDc*G+DD*Gc-1.*Bc*P5-1.*B*Pc5+
P5*Pc5-1.*B*Bc*T35+Bc*DD*T35+B*Pc5*T35-1.*DD*Pc5*T35-1.*Bc*
G*T55+G*Pc5*T55-1.*B*Bc*Tc35+B*DDc*Tc35+Bc*P5*Tc35-1.*DDc*
P5*Tc35+Bc*G*T55*Tc35-1.*DDc*G*T55*Tc35-1.*B*Gc*Tc55+Gc*P5*
Tc55+B*Gc*T35*Tc55-1.*DD*Gc*T35*Tc55;

f6=-1.*A*Bc*CCc*DD-1.*Ac*B*CC*DDc+Ac*CC*DD*DDc+A*CCc*DD*DDc+
A*Bc*CCc*G+Ac*B*DDc*G-1.*Ac*CC*DDc*G-1.*A*CCc*DDc*G-1.*CCc*
DD*DDc*G+CCc*DDc*G^2+Ac*B*CC*Gc+A*Bc*DD*Gc-1.*Ac*CC*DD*Gc-1.*
A*CCc*DD*Gc-1.*CC*DD*DDc*Gc-1.*Ac*B*G*Gc-1.*A*Bc*G*Gc+Ac*CC*G*
Gc+A*CCc*G*Gc+2.*DD*DDc*G*Gc-1.*DDc*G^2*Gc+CC*DD*Gc^2-1.*DD*G*
Gc^2+Bc*CCc*DD*P1+Ac*CC*DDc*P1-1.*Ac*DD*DDc*P1-1.*Bc*CCc*G*
P1-1.*Ac*CC*Gc*P1+Ac*DD*Gc*P1-1.*Bc*DD*Gc*P1+Bc*G*Gc*P1+A*CCc*
DD*Pc1+B*CC*DDc*Pc1-1.*A*DD*DDc*Pc1-1.*A*CCc*G*Pc1+A*DDc*G*
Pc1-1.*B*DDc*G*Pc1-1.*B*CC*Gc*Pc1+B*G*Gc*Pc1-1.*CCc*DD*P1*Pc1-1.*
CC*DDc*P1*Pc1+DD*DDc*P1*Pc1+CCc*G*P1*Pc1+CC*Gc*P1*Pc1-1.*G*Gc*
P1*Pc1+Ac*B*CC*DDc*T31-1.*Ac*CC*DD*DDc*T31-1.*Ac*B*DDc*G*T31+
Ac*DD*DDc*G*T31-1.*Ac*B*CC*Gc*T31+Ac*CC*DD*Gc*T31+Ac*B*G*Gc*
T31-1.*Ac*DD*G*Gc*T31-1.*B*CC*DDc*Pc1*T31+CC*DD*DDc*Pc1*T31+
B*DDc*G*Pc1*T31-1.*DD*DDc*G*Pc1*T31+B*CC*Gc*Pc1*T31-1.*CC*DD*
Gc*Pc1*T31-1.*B*G*Gc*Pc1*T31+DD*G*Gc*Pc1*T31+Bc*CCc*DD*G*T51+
Ac*CC*DDc*G*T51-1.*Ac*DD*DDc*G*T51-1.*Bc*CCc*G^2*T51-1.*Ac*CC*
G*Gc*T51+Ac*DD*G*Gc*T51-1.*Bc*DD*G*Gc*T51+Bc*G^2*Gc*T51-1.*CCc*
DD*G*Pc1*T51-1.*CC*DDc*G*Pc1*T51+DD*DDc*G*Pc1*T51+CCc*G^2*Pc1*
```

```
T51+CC*G*Gc*Pc1*T51-1.*G^2*Gc*Pc1*T51+A*Bc*CCc*DD*Tc31-1.*A*
CCc*DD*DDc*Tc31-1.*A*Bc*CCc*G*Tc31+A*CCc*DDc*G*Tc31-1.*A*Bc*
DD*Gc*Tc31+A*DD*DDc*Gc*Tc31+A*Bc*G*Gc*Tc31-1.*A*DDc*G*Gc*Tc31-1.*
Bc*CCc*DD*P1*Tc31+CCc*DD*DDc*P1*Tc31+Bc*CCc*G*P1*Tc31-1.*CCc*
DDc*G*P1*Tc31+Bc*DD*Gc*P1*Tc31-1.*DD*DDc*Gc*P1*Tc31-1.*Bc*G*Gc*
P1*Tc31+DDc*G*Gc*P1*Tc31-1.*Bc*CCc*DD*G*T51*Tc31+CCc*DD*DDc*G*
T51*Tc31+Bc*CCc*G^2*T51*Tc31-1.*CCc*DDc*G^2*T51*Tc31+Bc*DD*G*
Gc*T51*Tc31-1.*DD*DDc*G*Gc*T51*Tc31-1.*Bc*G^2*Gc*T51*Tc31+DDc*
G^2*Gc*T51*Tc31+A*CCc*DD*Gc*Tc51+B*CC*DDc*Gc*Tc51-1.*A*DD*DDc*
Gc*Tc51-1.*A*CCc*G*Gc*Tc51+A*DDc*G*Gc*Tc51-1.*B*DDc*G*Gc*Tc51-1.*
B*CC*Gc^2*Tc51+B*G*Gc^2*Tc51-1.*CCc*DD*Gc*P1*Tc51-1.*CC*DDc*Gc*
P1*Tc51+DD*DDc*Gc*P1*Tc51+CCc*G*Gc*P1*Tc51+CC*Gc^2*P1*Tc51-1.*G*
Gc^2*P1*Tc51-1.*B*CC*DDc*Gc*T31*Tc51+CC*DD*DDc*Gc*T31*Tc51+B*DDc*
G*Gc*T31*Tc51-1.*DD*DDc*G*Gc*T31*Tc51+B*CC*Gc^2*T31*Tc51-1.*CC*
DD*Gc^2*T31*Tc51-1.*B*G*Gc^2*T31*Tc51+DD*G*Gc^2*T31*Tc51;

f7=-1.*A*Bc*CCc*DD-1.*Ac*B*CC*DDc+Ac*CC*DD*DDc+A*CCc*DD*DDc+A*Bc*
CCc*G+Ac*B*DDc*G-1.*Ac*CC*DDc*G-1.*A*CCc*DDc*G-1.*CCc*DD*DDc*G+CCc*
DDc*G^2+Ac*B*CC*Gc+A*Bc*DD*Gc-1.*Ac*CC*DD*Gc-1.*A*CCc*DD*Gc-1.*CC*
DD*DDc*Gc-1.*Ac*B*G*Gc-1.*A*Bc*G*Gc+Ac*CC*G*Gc+A*CCc*G*Gc+2.*DD*DDc*
G*Gc-1.*DDc*G^2*Gc+CC*DD*Gc^2-1.*DD*G*Gc^2+Bc*CCc*DD*P2+Ac*CC*DDc*
P2-1.*Ac*DD*DDc*P2-1.*Bc*CCc*G*P2-1.*Ac*CC*Gc*P2+Ac*DD*Gc*P2-1.*Bc*
DD*Gc*P2+Bc*G*Gc*P2+A*CCc*DD*Pc2+B*CC*DDc*Pc2-1.*A*DD*DDc*Pc2-1.*A*
CCc*G*Pc2+A*DDc*G*Pc2-1.*B*DDc*G*Pc2-1.*B*CC*Gc*Pc2+B*G*Gc*Pc2-1.*
CCc*DD*P2*Pc2-1.*CC*DDc*P2*Pc2+DD*DDc*P2*Pc2+CCc*G*P2*Pc2+CC*Gc*P2*
Pc2-1.*G*Gc*P2*Pc2+Ac*B*CC*DDc*T32-1.*Ac*CC*DD*DDc*T32-1.*Ac*B*DDc*
G*T32+Ac*DD*DDc*G*T32-1.*Ac*B*CC*Gc*T32+Ac*CC*DD*Gc*T32+Ac*B*G*Gc*
```

T32−1.∗Ac∗DD∗G∗Gc∗T32−1.∗B∗CC∗DDc∗Pc2∗T32+CC∗DD∗DDc∗Pc2∗T32+B∗DDc∗G∗
Pc2∗T32−1.∗DD∗DDc∗G∗Pc2∗T32+B∗CC∗Gc∗Pc2∗T32−1.∗CC∗DD∗Gc∗Pc2∗T32−1.∗
B∗G∗Gc∗Pc2∗T32+DD∗G∗Gc∗Pc2∗T32+Bc∗CCc∗DD∗G∗T52+Ac∗CC∗DDc∗G∗T52−1.∗Ac∗
DD∗DDc∗G∗T52−1.∗Bc∗CCc∗G^2∗T52−1.∗Ac∗CC∗G∗Gc∗T52+Ac∗DD∗G∗Gc∗T52−1.∗
Bc∗DD∗G∗Gc∗T52+Bc∗G^2∗Gc∗T52−1.∗CCc∗DD∗G∗Pc2∗T52−1.∗CC∗DDc∗G∗Pc2∗T52+
DD∗DDc∗G∗Pc2∗T52+CCc∗G^2∗Pc2∗T52+CC∗G∗Gc∗Pc2∗T52−1.∗G^2∗Gc∗Pc2∗T52+
A∗Bc∗CCc∗DD∗Tc32−1.∗A∗CCc∗DD∗DDc∗Tc32−1.∗A∗Bc∗CCc∗G∗Tc32+A∗CCc∗DDc∗
G∗Tc32−1.∗A∗Bc∗DD∗Gc∗Tc32+A∗DD∗DDc∗Gc∗Tc32+A∗Bc∗G∗Gc∗Tc32−1.∗A∗DDc∗
G∗Gc∗Tc32−1.∗Bc∗CCc∗DD∗P2∗Tc32+CCc∗DD∗DDc∗P2∗Tc32+Bc∗CCc∗G∗P2∗Tc32−1.∗
CCc∗DDc∗G∗P2∗Tc32+Bc∗DD∗Gc∗P2∗Tc32−1.∗DD∗DDc∗Gc∗P2∗Tc32−1.∗Bc∗G∗Gc∗
P2∗Tc32+DDc∗G∗Gc∗P2∗Tc32−1.∗Bc∗CCc∗DD∗G∗T52∗Tc32+CCc∗DD∗DDc∗G∗T52∗
Tc32+Bc∗CCc∗G^2∗T52∗Tc32−1.∗CCc∗DDc∗G^2∗T52∗Tc32+Bc∗DD∗G∗Gc∗T52∗
Tc32−1.∗DD∗DDc∗G∗Gc∗T52∗Tc32−1.∗Bc∗G^2∗Gc∗T52∗Tc32+DDc∗G^2∗Gc∗T52∗
Tc32+A∗CCc∗DD∗Gc∗Tc52+B∗CC∗DDc∗Gc∗Tc52−1.∗A∗DD∗DDc∗Gc∗Tc52−1.∗A∗CCc∗
G∗Gc∗Tc52+A∗DDc∗G∗Gc∗Tc52−1.∗B∗DDc∗G∗Gc∗Tc52−1.∗B∗CC∗Gc^2∗Tc52+B∗G∗
Gc^2∗Tc52−1.∗CCc∗DD∗Gc∗P2∗Tc52−1.∗CC∗DDc∗Gc∗P2∗Tc52+DD∗DDc∗Gc∗P2∗
Tc52+CCc∗G∗Gc∗P2∗Tc52+CC∗Gc^2∗P2∗Tc52−1.∗G∗Gc^2∗P2∗Tc52−1.∗B∗CC∗DDc∗
Gc∗T32∗Tc52+CC∗DD∗DDc∗Gc∗T32∗Tc52+B∗DDc∗G∗Gc∗T32∗Tc52−1.∗DD∗DDc∗G∗
Gc∗T32∗Tc52+B∗CC∗Gc^2∗T32∗Tc52−1.∗CC∗DD∗Gc^2∗T32∗Tc52−1.∗B∗G∗Gc^2∗
T32∗Tc52+DD∗G∗Gc^2∗T32∗Tc52;

f8=−1.∗A∗Bc∗CCc∗DD−1.∗Ac∗B∗CC∗DDc+Ac∗CC∗DD∗DDc+A∗CCc∗DD∗DDc+A∗Bc∗CCc∗
G+Ac∗B∗DDc∗G−1.∗Ac∗CC∗DDc∗G−1.∗A∗CCc∗DDc∗G−1.∗CCc∗DD∗DDc∗G+CCc∗DDc∗G^2+
Ac∗B∗CC∗Gc+A∗Bc∗DD∗Gc−1.∗Ac∗CC∗DD∗Gc−1.∗A∗CCc∗DD∗Gc−1.∗CC∗DD∗DDc∗Gc−1.∗
Ac∗B∗G∗Gc−1.∗A∗Bc∗G∗Gc+Ac∗CC∗G∗Gc+A∗CCc∗G∗Gc+2.∗DD∗DDc∗G∗Gc−1.∗DDc∗G^2∗
Gc+CC∗DD∗Gc^2−1.∗DD∗G∗Gc^2+Bc∗CCc∗DD∗P3+Ac∗CC∗DDc∗P3−1.∗Ac∗DD∗DDc∗P3−1.∗
Bc∗CCc∗G∗P3−1.∗Ac∗CC∗Gc∗P3+Ac∗DD∗Gc∗P3−1.∗Bc∗DD∗Gc∗P3+Bc∗G∗Gc∗P3+A∗CCc∗

```
DD*Pc3+B*CC*DDc*Pc3-1.*A*DD*DDc*Pc3-1.*A*CCc*G*Pc3+A*DDc*G*Pc3-1.*B*DDc*
G*Pc3-1.*B*CC*Gc*Pc3+B*G*Gc*Pc3-1.*CCc*DD*P3*Pc3-1.*CC*DDc*P3*Pc3+DD*DDc*
P3*Pc3+CCc*G*P3*Pc3+CC*Gc*P3*Pc3-1.*G*Gc*P3*Pc3+Ac*B*CC*DDc*T33-1.*Ac*
CC*DD*DDc*T33-1.*Ac*B*DDc*G*T33+Ac*DD*DDc*G*T33-1.*Ac*B*CC*Gc*T33+Ac*CC*
DD*Gc*T33+Ac*B*G*Gc*T33-1.*Ac*DD*G*Gc*T33-1.*B*CC*DDc*Pc3*T33+CC*DD*DDc*
Pc3*T33+B*DDc*G*Pc3*T33-1.*DD*DDc*G*Pc3*T33+B*CC*Gc*Pc3*T33-1.*CC*DD*Gc*
Pc3*T33-1.*B*G*Gc*Pc3*T33+DD*G*Gc*Pc3*T33+Bc*CCc*DD*G*T53+Ac*CC*DDc*G*
T53-1.*Ac*DD*DDc*G*T53-1.*Bc*CCc*G^2*T53-1.*Ac*CC*G*Gc*T53+Ac*DD*G*Gc*T53
-1.*Bc*DD*G*Gc*T53+Bc*G^2*Gc*T53-1.*CCc*DD*G*Pc3*T53-1.*CC*DDc*G*Pc3*T53+DD*
DDc*G*Pc3*T53+CCc*G^2*Pc3*T53+CC*G*Gc*Pc3*T53-1.*G^2*Gc*Pc3*T53+A*Bc*CCc*
DD*Tc33-1.*A*CCc*DD*DDc*Tc33-1.*A*Bc*CCc*G*Tc33+A*CCc*DDc*G*Tc33-1.*A*Bc*
DD*Gc*Tc33+A*DD*DDc*Gc*Tc33+A*Bc*G*Gc*Tc33-1.*A*DDc*G*Gc*Tc33-1.*Bc*CCc*
DD*P3*Tc33+CCc*DD*DDc*P3*Tc33+Bc*CCc*G*P3*Tc33-1.*CCc*DDc*G*P3*Tc33+Bc*
DD*Gc*P3*Tc33-1.*DD*DDc*Gc*P3*Tc33-1.*Bc*G*Gc*P3*Tc33+DDc*G*Gc*P3*Tc33-1.*
Bc*CCc*DD*G*T53*Tc33+CCc*DD*DDc*G*T53*Tc33+Bc*CCc*G^2*T53*Tc33-1.*CCc*DDc*
G^2*T53*Tc33+Bc*DD*G*Gc*T53*Tc33-1.*DD*DDc*G*Gc*T53*Tc33-1.*Bc*G^2*Gc*T53*
Tc33+DDc*G^2*Gc*T53*Tc33+A*CCc*DD*Gc*Tc53+B*CC*DDc*Gc*Tc53-1.*A*DD*DDc*Gc*
Tc53-1.*A*CCc*G*Gc*Tc53+A*DDc*G*Gc*Tc53-1.*B*DDc*G*Gc*Tc53-1.*B*CC*Gc^2*
Tc53+B*G*Gc^2*Tc53-1.*CCc*DD*Gc*P3*Tc53-1.*CC*DDc*Gc*P3*Tc53+DD*DDc*Gc*P3*
Tc53+CCc*G*Gc*P3*Tc53+CC*Gc^2*P3*Tc53-1.*G*Gc^2*P3*Tc53-1.*B*CC*DDc*Gc*T33*
Tc53+CC*DD*DDc*Gc*T33*Tc53+B*DDc*G*Gc*T33*Tc53-1.*DD*DDc*G*Gc*T33*Tc53+B*CC*
Gc^2*T33*Tc53-1.*CC*DD*Gc^2*T33*Tc53-1.*B*G*Gc^2*T33*Tc53+DD*G*Gc^2*T33*
Tc53;

f9=-1.*A*Bc*CCc*DD-1.*Ac*B*CC*DDc+Ac*CC*DD*DDc+A*CCc*DD*DDc+A*Bc*CCc*G+
Ac*B*DDc*G-1.*Ac*CC*DDc*G-1.*A*CCc*DDc*G-1.*CCc*DD*DDc*G+CCc*DDc*G^2+Ac*
B*CC*Gc+A*Bc*DD*Gc-1.*Ac*CC*DD*Gc-1.*A*CCc*DD*Gc-1.*CC*DD*DDc*Gc-1.*Ac*
```

```
B*G*Gc-1.*A*Bc*G*Gc+Ac*CC*G*Gc+A*CCc*G*Gc+2.*DD*DDc*G*Gc-1.*DDc*G^2*Gc+
CC*DD*Gc^2-1.*DD*G*Gc^2+Bc*CCc*DD*P4+Ac*CC*DDc*P4-1.*Ac*DD*DDc*P4-1.*
Bc*CCc*G*P4-1.*Ac*CC*Gc*P4+Ac*DD*Gc*P4-1.*Bc*DD*Gc*P4+Bc*G*Gc*P4+A*CCc*
DD*Pc4+B*CC*DDc*Pc4-1.*A*DD*DDc*Pc4-1.*A*CCc*G*Pc4+A*DDc*G*Pc4-1.*B*DDc*
G*Pc4-1.*B*CC*Gc*Pc4+B*G*Gc*Pc4-1.*CCc*DD*P4*Pc4-1.*CC*DDc*P4*Pc4+DD*
DDc*P4*Pc4+CCc*G*P4*Pc4+CC*Gc*P4*Pc4-1.*G*Gc*P4*Pc4+Ac*B*CC*DDc*T34-1.*
Ac*CC*DD*DDc*T34-1.*Ac*B*DDc*G*T34+Ac*DD*DDc*G*T34-1.*Ac*B*CC*Gc*T34+
Ac*CC*DD*Gc*T34+Ac*B*G*Gc*T34-1.*Ac*DD*G*Gc*T34-1.*B*CC*DDc*Pc4*T34+CC*
DD*DDc*Pc4*T34+B*DDc*G*Pc4*T34-1.*DD*DDc*G*Pc4*T34+B*CC*Gc*Pc4*T34-1.*
CC*DD*Gc*Pc4*T34-1.*B*G*Gc*Pc4*T34+DD*G*Gc*Pc4*T34+Bc*CCc*DD*G*T54+Ac*
CC*DDc*G*T54-1.*Ac*DD*DDc*G*T54-1.*Bc*CCc*G^2*T54-1.*Ac*CC*G*Gc*T54+Ac*
DD*G*Gc*T54-1.*Bc*DD*G*Gc*T54+Bc*G^2*Gc*T54-1.*CCc*DD*G*Pc4*T54-1.*CC*
DDc*G*Pc4*T54+DD*DDc*G*Pc4*T54+CCc*G^2*Pc4*T54+CC*G*Gc*Pc4*T54-1.*G^2*
Gc*Pc4*T54+A*Bc*CCc*DD*Tc34-1.*A*CCc*DD*DDc*Tc34-1.*A*Bc*CCc*G*Tc34+A*
CCc*DDc*G*Tc34-1.*A*Bc*DD*Gc*Tc34+A*DD*DDc*Gc*Tc34+A*Bc*G*Gc*Tc34-1.*A*
DDc*G*Gc*Tc34-1.*Bc*CCc*DD*P4*Tc34+CCc*DD*DDc*P4*Tc34+Bc*CCc*G*P4*Tc34-1.*
CCc*DDc*G*P4*Tc34+Bc*DD*Gc*P4*Tc34-1.*DD*DDc*Gc*P4*Tc34-1.*Bc*G*Gc*P4*
Tc34+DDc*G*Gc*P4*Tc34-1.*Bc*CCc*DD*G*T54*Tc34+CCc*DD*DDc*G*T54*Tc34+Bc*
CCc*G^2*T54*Tc34-1.*CCc*DDc*G^2*T54*Tc34+Bc*DD*G*Gc*T54*Tc34-1.*DD*DDc*
G*Gc*T54*Tc34-1.*Bc*G^2*Gc*T54*Tc34+DDc*G^2*Gc*T54*Tc34+A*CCc*DD*Gc*
Tc54+B*CC*DDc*Gc*Tc54-1.*A*DD*DDc*Gc*Tc54-1.*A*CCc*G*Gc*Tc54+A*DDc*G*
Gc*Tc54-1.*B*DDc*G*Gc*Tc54-1.*B*CC*Gc^2*Tc54+B*G*Gc^2*Tc54-1.*CCc*DD*
Gc*P4*Tc54-1.*CC*DDc*Gc*P4*Tc54+DD*DDc*Gc*P4*Tc54+CCc*G*Gc*P4*Tc54+CC*
Gc^2*P4*Tc54-1.*G*Gc^2*P4*Tc54-1.*B*CC*DDc*Gc*T34*Tc54+CC*DD*DDc*Gc*
T34*Tc54+B*DDc*G*Gc*T34*Tc54-1.*DD*DDc*G*Gc*T34*Tc54+B*CC*Gc^2*T34*
Tc54-1.*CC*DD*Gc^2*T34*Tc54-1.*B*G*Gc^2*T34*Tc54+DD*G*Gc^2*T34*Tc54;
```

f10=−1.∗A∗Bc∗CCc∗DD−1.∗Ac∗B∗CC∗DDc+Ac∗CC∗DD∗DDc+A∗CCc∗DD∗DDc+A∗Bc∗CCc∗G+Ac∗B∗DDc∗G−1.∗Ac∗CC∗DDc∗G−1.∗A∗CCc∗DDc∗G−1.∗CCc∗DD∗DDc∗G+CCc∗DDc∗G^2+Ac∗B∗CC∗Gc+A∗Bc∗DD∗Gc−1.∗Ac∗CC∗DD∗Gc−1.∗A∗CCc∗DD∗Gc−1.∗CC∗DD∗DDc∗Gc−1.∗Ac∗B∗G∗Gc−1.∗A∗Bc∗G∗Gc+Ac∗CC∗G∗Gc+A∗CCc∗G∗Gc+2.∗DD∗DDc∗G∗Gc−1.∗DDc∗G^2∗Gc+CC∗DD∗Gc^2−1.∗DD∗G∗Gc^2+Bc∗CCc∗DD∗P5+Ac∗CC∗DDc∗P5−1.∗Ac∗DD∗DDc∗P5−1.∗Bc∗CCc∗G∗P5−1.∗Ac∗CC∗Gc∗P5+Ac∗DD∗Gc∗P5−1.∗Bc∗DD∗Gc∗P5+Bc∗G∗Gc∗P5+A∗CCc∗DD∗Pc5+B∗CC∗DDc∗Pc5−1.∗A∗DD∗DDc∗Pc5−1.∗A∗CCc∗G∗Pc5+A∗DDc∗G∗Pc5−1.∗B∗DDc∗G∗Pc5−1.∗B∗CC∗Gc∗Pc5+B∗G∗Gc∗Pc5−1.∗CCc∗DD∗P5∗Pc5−1.∗CC∗DDc∗P5∗Pc5+DD∗DDc∗P5∗Pc5+CCc∗G∗P5∗Pc5+CC∗Gc∗P5∗Pc5−1.∗G∗Gc∗P5∗Pc5+Ac∗B∗CC∗DDc∗T35−1.∗Ac∗CC∗DD∗DDc∗T35−1.∗Ac∗B∗DDc∗G∗T35+Ac∗DD∗DDc∗G∗T35−1.∗Ac∗B∗CC∗Gc∗T35+Ac∗CC∗DD∗Gc∗T35+Ac∗B∗G∗Gc∗T35−1.∗Ac∗DD∗G∗Gc∗T35−1.∗B∗CC∗DDc∗Pc5∗T35+CC∗DD∗DDc∗Pc5∗T35+B∗DDc∗G∗Pc5∗T35−1.∗DD∗DDc∗G∗Pc5∗T35+B∗CC∗Gc∗Pc5∗T35−1.∗CC∗DD∗Gc∗Pc5∗T35−1.∗B∗G∗Gc∗Pc5∗T35+DD∗G∗Gc∗Pc5∗T35+Bc∗CCc∗DD∗G∗T55+Ac∗CC∗DDc∗G∗T55−1.∗Ac∗DD∗DDc∗G∗T55−1.∗Bc∗CCc∗G^2∗T55−1.∗Ac∗CC∗G∗Gc∗T55+Ac∗DD∗G∗Gc∗T55−1.∗Bc∗DD∗G∗Gc∗T55+Bc∗G^2∗Gc∗T55−1.∗CCc∗DD∗G∗Pc5∗T55−1.∗CC∗DDc∗G∗Pc5∗T55+DD∗DDc∗G∗Pc5∗T55+CCc∗G^2∗Pc5∗T55+CC∗G∗Gc∗Pc5∗T55−1.∗G^2∗Gc∗Pc5∗T55+A∗Bc∗CCc∗DD∗Tc35−1.∗A∗CCc∗DD∗DDc∗Tc35−1.∗A∗Bc∗CCc∗G∗Tc35+A∗CCc∗DDc∗G∗Tc35−1.∗A∗Bc∗DD∗Gc∗Tc35+A∗DD∗DDc∗Gc∗Tc35+A∗Bc∗G∗Gc∗Tc35−1.∗A∗DDc∗G∗Gc∗Tc35−1.∗Bc∗CCc∗DD∗P5∗Tc35+CCc∗DD∗DDc∗P5∗Tc35+Bc∗CCc∗G∗P5∗Tc35−1.∗CCc∗DDc∗G∗P5∗Tc35+Bc∗DD∗Gc∗P5∗Tc35−1.∗DD∗DDc∗Gc∗P5∗Tc35−1.∗Bc∗G∗Gc∗P5∗Tc35+DDc∗G∗Gc∗P5∗Tc35−1.∗Bc∗CCc∗DD∗G∗T55∗Tc35+CCc∗DD∗DDc∗G∗T55∗Tc35+Bc∗CCc∗G^2∗T55∗Tc35−1.∗CCc∗DDc∗G^2∗T55∗Tc35+Bc∗DD∗G∗Gc∗T55∗Tc35−1.∗DD∗DDc∗G∗Gc∗T55∗Tc35−1.∗Bc∗G^2∗Gc∗T55∗Tc35+DDc∗G^2∗Gc∗T55∗Tc35+A∗CCc∗DD∗Gc∗Tc55+B∗CC∗DDc∗Gc∗Tc55−1.∗A∗DD∗DDc∗Gc∗Tc55−1.∗A∗CCc∗G∗Gc∗Tc55+A∗DDc∗G∗Gc∗Tc55−1.∗B∗DDc∗G∗Gc∗Tc55−1.∗B∗CC∗Gc^2∗Tc55+B∗G∗Gc^2∗Tc55−1.∗CCc∗DD∗Gc∗P5∗Tc55−1.∗CC∗DDc∗Gc∗P5∗Tc55+DD∗DDc∗Gc∗P5∗Tc55+CCc∗G∗Gc∗P5∗Tc55+CC∗Gc^2∗P5∗Tc55−1.∗G∗Gc^2∗P5∗Tc55−1.∗B∗CC∗

```
DDc*Gc*T35*Tc55+CC*DD*DDc*Gc*T35*Tc55+B*DDc*G*Gc*T35*Tc55-1.*DD*DDc*
G*Gc*T35*Tc55+B*CC*Gc^2*T35*Tc55-1.*CC*DD*Gc^2*T35*Tc55-1.*B*G*Gc^2*
T35*Tc55+DD*G*Gc^2*T35*Tc55;

f11=2.*B*Bc-1.*Bc*F-1.*B*Fc+Fc*H+F*Hc-1.*Bc*P1-1.*B*Pc1+P1*Pc1-1.*B*Bc*
T31+Bc*F*T31+B*Pc1*T31-1.*F*Pc1*T31-1.*Bc*H*T51+H*Pc1*T51-1.*B*Bc*Tc31+
B*Fc*Tc31+Bc*P1*Tc31-1.*Fc*P1*Tc31+Bc*H*T51*Tc31-1.*Fc*H*T51*Tc31-1.*B*
Hc*Tc51+Hc*P1*Tc51+B*Hc*T31*Tc51-1.*F*Hc*T31*Tc51;

f12=2.*B*Bc-1.*Bc*F-1.*B*Fc+Fc*H+F*Hc-1.*Bc*P2-1.*B*Pc2+P2*Pc2-1.*B*Bc*
T32+Bc*F*T32+B*Pc2*T32-1.*F*Pc2*T32-1.*Bc*H*T52+H*Pc2*T52-1.*B*Bc*Tc32+
B*Fc*Tc32+Bc*P2*Tc32-1.*Fc*P2*Tc32+Bc*H*T52*Tc32-1.*Fc*H*T52*Tc32-1.*B*
Hc*Tc52+Hc*P2*Tc52+B*Hc*T32*Tc52-1.*F*Hc*T32*Tc52;

f13=2.*B*Bc-1.*Bc*F-1.*B*Fc+Fc*H+F*Hc-1.*Bc*P3-1.*B*Pc3+P3*Pc3-1.*B*Bc*
T33+Bc*F*T33+B*Pc3*T33-1.*F*Pc3*T33-1.*Bc*H*T53+H*Pc3*T53-1.*B*Bc*Tc33+
B*Fc*Tc33+Bc*P3*Tc33-1.*Fc*P3*Tc33+Bc*H*T53*Tc33-1.*Fc*H*T53*Tc33-1.*B*
Hc*Tc53+Hc*P3*Tc53+B*Hc*T33*Tc53-1.*F*Hc*T33*Tc53;

f14=2.*B*Bc-1.*Bc*F-1.*B*Fc+Fc*H+F*Hc-1.*Bc*P4-1.*B*Pc4+P4*Pc4-1.*B*Bc*
T34+Bc*F*T34+B*Pc4*T34-1.*F*Pc4*T34-1.*Bc*H*T54+H*Pc4*T54-1.*B*Bc*Tc34+
B*Fc*Tc34+Bc*P4*Tc34-1.*Fc*P4*Tc34+Bc*H*T54*Tc34-1.*Fc*H*T54*Tc34-1.*B*
Hc*Tc54+Hc*P4*Tc54+B*Hc*T34*Tc54-1.*F*Hc*T34*Tc54;

f15=2.*B*Bc-1.*Bc*F-1.*B*Fc+Fc*H+F*Hc-1.*Bc*P5-1.*B*Pc5+P5*Pc5-1.*B*Bc*
T35+Bc*F*T35+B*Pc5*T35-1.*F*Pc5*T35-1.*Bc*H*T55+H*Pc5*T55-1.*B*Bc*Tc35+
B*Fc*Tc35+Bc*P5*Tc35-1.*Fc*P5*Tc35+Bc*H*T55*Tc35-1.*Fc*H*T55*Tc35-1.*B*
```

```
Hc*Tc55+Hc*P5*Tc55+B*Hc*T35*Tc55-1.*F*Hc*T35*Tc55;

f16=-1.+T31*Tc31;
f17=-1.+T32*Tc32;
f18=-1.+T33*Tc33;
f19=-1.+T34*Tc34;
f20=-1.+T35*Tc35;
constraintEquations = {f1, f2, f3, f4, f5, f6,f7, f8, f9, f10, f11, f12,
f13, f14, f15, f16, f17,f18,f19,f20};
unknownVars = {CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31,Tc31,T32, Tc32,
T33, Tc33,T34, Tc34, T35,Tc35};
parameters = {P1, P2, P3, P4, P5, Pc1, Pc2, Pc3, Pc4, Pc5, T51, T52, T53,
T54, T55, A, Ac, B, Bc, Tc51, Tc52, Tc53, Tc54, Tc55};

GenerateParameterRule[var_] := Module[{varString, varStringStart,
varStringFinal, resubRuleString, resubRuleExpr},
varString = ToString[var];
varStringStart = varString <> "Start";
varStringFinal = varString <> "Final";
resubRuleString = varString <> " -> " <> varStringStart <> " * t + "
<> varStringFinal <> " * gamma * (1 - t)";
resubRuleExpr = ToExpression[resubRuleString];

Return[resubRuleExpr];
]

parameterResubRules = Map[GenerateParameterRule, parameters];
```

```
HomotopySystem = Map[# /. parameterResubRules&, constraintEquations];

Calculating the Jacobian of the System (JH) and the partial with
respect to t (Ht).
JH = D[HomotopySystem, {unknownVars}];
Note: Simplifying Ht to put it in a form that works better for C++
to avoid stack overflow
Ht = D[HomotopySystem, t] // Simplify;

Finding the re-substitution rules for JH
BoolMatrix = Table[ Map[Boole[!PossibleZeroQ[#]]&, JH[[i]]],
{i, Length[JH]}]; BoolMatrix // MatrixForm;
AssignSymbol[row_, rowNum_, unknowns_] := Module[{rowVars},
rowVars = Table[
If[PossibleZeroQ[row[[i]]], 0,
ToExpression[ToString[unknowns[[i]]] <> ToString[rowNum]]],
{i, Length[row]}];
rowVars]
SymbolMatrix = Table[AssignSymbol[BoolMatrix[[i]], i, unknownVars],
{i, Length[BoolMatrix]}] ; SymbolMatrix // MatrixForm;
ResubRules[jsrow_, jrow_, rowNum_] := Module[{rules},
rules = Table[
If[PossibleZeroQ[jsrow[[i]]], 0,
Rule[jsrow[[i]], jrow[[i]]]],
{i, Length[jsrow]}];
rules]
```

```
Resubs = Table[Select[ResubRules[SymbolMatrix[[i]], JH[[i]], i],
!PossibleZeroQ[#]&], {i, Length[SymbolMatrix]}];

Exporting Resub Rules
SetDirectory[NotebookDirectory[]];
jacobianCPP = StringDrop["JMat << " <> Map[ToString[#] <> ","&,
 Flatten[SymbolMatrix]], -1] <> ";";
Export["JacobianMatrixExpr.txt", jacobianCPP];
JacobianResubRule[rule_] := Module[{matlabCode},
matlabCode = ToString[rule[[1]]] <> " = " <> ToString[rule[[2]]
// CForm] <> ";";
Return[matlabCode];
]
ResubSymbols = Table[Map[First, Resubs[[i]]],
{i, Length[Resubs]}] // Flatten;
JacobianRules = Table[Map[JacobianResubRule, Resubs[[i]]],
{i, Length[Resubs]}];
JacobianRules // Dimensions;
With[{file=OpenWrite@FileNameJoin@{NotebookDirectory[],
"JacobianResubs.txt"}},
Do[
Do[
WriteLine[file,JacobianRules[[row, col]]],
{col,Length[JacobianRules[[row]]]}
],
{row, Length[JacobianRules]}
];
```

```
Close@file];
SymbolArray = Flatten[SymbolMatrix];
With[{file=OpenWrite@FileNameJoin@{NotebookDirectory[],
"JacobianMatrixAssignments.txt"}},
Table[WriteLine[file, "J[" <> ToString[i-1]  <> "] = " <>
ToString[SymbolArray[[i]]]], {i, Length[SymbolArray]}];
Close@file];
HtMatLab = Table["Ht" <> ToString[i] <> " = " <> ToString[Ht[[i]]
// CForm ]<> ";", {i, Length[Ht]}];
Export["HtResubs.txt", HtMatLab];
HomotopySystemExport = Table["f" <> ToString[i] <> " = " <>
ToString[HomotopySystem[[i]] // CForm ] <> ";",
{i, Length[HomotopySystem]}];
With[{file=OpenWrite@FileNameJoin@{NotebookDirectory[],
"HomotopySystem.txt"}},
Do[
WriteLine[file,HomotopySystemExport[[i]]],{i,Length[HomotopySystemExport]}
];

Close@file];
```

Appendix B

CUDA Code

B.1 Four-Bar Synthesis

B.1.1 Definitions.h

```
#pragma once
#ifndef DEFINITIONS_H
#define DEFINITIONS_H

#define _USE_MATH_DEFINES

#include "thrust\complex.h"
#include <math.h>

const int kNumParameters = 16; // Number of start and final parameters

const unsigned kNumLinesPerPoint = 8; // Number of unknowns (used in file
```

```
reading)

const unsigned n = kNumLinesPerPoint; // Number of unknowns

const int kNumStartPoints = 25; // Number of start points as
calculated by Bertini

const unsigned kNumPointsAnalysis = n / 2; // unknowns converted to points
(x, y)

const int kNumIterations = 200; // Number of iterations for randomization

const double kTolX = 0.05; // x tolerance for randomization

const double kTolY = 0.05; // y tolerance for randomization

const double kTolTheta = 0.5; // theta tolerance for randomization

const int kNumTaskPositions = 5; // Number of task positions in the global
frame.

const int kNumTPParameters = 3 * kNumTaskPositions; // Number of parameters
(angle, x, y) per task position

const int kNumNewtonCorrections = 2; // Number of corrections to run in
the correction step
```

```
const double kResolution = 0.1 * M_PI / 180.0; // Angle resolution for the
analysis code

const double kEndGameBoundary = 0.0; // path tracking from 1 to
endGameBoundary

const double kNumSteps = 100; // number of steps to take in the path tracking

const double kStepSize = (1 - kEndGameBoundary) / kNumSteps; // Step size
for path tracking

const int kNumSamplesEndGame = 15; // Number of sample points to use for
the end game

const double kEndGameLambda = 0.5; // Lambda to use to generate the series
R, lambda*R, lambda^2*R, ...

const int kNumSamplesInterpolation = kNumSamplesEndGame - 1; // Number of
sample points to use to predict the final sample point in the end game.

const double kNumStepsEndGame = 10; // Number of steps to take between
sample times for the end game

const int kMaxC = 4; // Max winding number to test in the endgame

const int kNumDerivatives = 1; // using the 0-th and 1-st derivative in
hermite interpolation, just the 0-th in newton interpolation (just the sample
```

```
typedef thrust::complex<double> complex_t; // Choosing whether to use
single or double precision complex numbers
#endif
```

B.1.2 LinearSolver.cuh

```
#pragma once
#ifndef LINEARSOLVER_CUH
#define LINEARSOLVER_CUH

#include <cublas_v2.h>
#include "Definitions.h"

#include "cuda_runtime.h"
#include "device_launch_parameters.h"

using namespace thrust;

__device__ void solveLinearSystem(complex_t *A, complex_t *L,
complex_t *U, complex_t *P, complex_t *b, int n, double t);

__device__ int matrixToArray(int row, int col, int n);

__device__ void matrixVectorProduct(complex_t *M, complex_t *v,
complex_t *y, int n);
```

```
__device__ void matrixMatrixProduct(complex_t *A, complex_t *B,
complex_t *C, int n);

#endif
```

B.1.3 LinearSolver.cu

```
#include "LinearSolver.cuh"



__device__ int matrixToArray(int row, int col, int n)
{
return (row * n + col);
}

__device__ void matrixVectorProduct(complex_t *M,
complex_t *v, complex_t *y, int n)
{
for (int i = 0; i < n; i++)
{
*(y + i) = 0;
for (int j = 0; j < n; j++)
{
*(y + i) += *(M + matrixToArray(i, j, n)) * *(v + j);
}
}
```

```
}

__device__ void matrixMatrixProduct(complex_t *A,
complex_t *B, complex_t *C, int n)
{
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
complex_t currentSum = complex_t(0, 0);
for (int k = 0; k < n; k++)
{
currentSum += *(A + matrixToArray(i, k, n)) *
*(B + matrixToArray(k, j, n));
}
*(C + matrixToArray(i, j, n)) = currentSum;
}
}
}


// Decomposes Matrix A into an Upper (stored in U)
and a Lower (stored in L) triangular matrix using
// doolittle factorization.
//
// n is the number of rows/columns of A, A is a row-major
 matrix.
```

```
//
// Code implemented from pesudocode found in Cheney/Kincade,
 "Numerical_Mathematics", pg 300.
__device__ void LUDecomp(complex_t *A, complex_t *L,
 complex_t *U, int n)
{

//printf("A(device) = \n");
for (int row = 0; row < n; row++)
{
for (int col = 0; col < n; col++)
{
//printf("%0.2f ", *(A + matrixToArray(row, col, n)));
*(L + matrixToArray(row, col, n)) = complex_t(0, 0);
*(U + matrixToArray(row, col, n)) = complex_t(0, 0);
}
//printf("\n");
}

for (int k = 0; k < n; k++)
{
// L_kk = 1 (set 1 along the diagonal of L)
*(L + matrixToArray(k, k, n)) = 1;

for (int j = k; j < n; j++)
{
complex_t currentSum = 0;
```

```
for (int s = 0; s < k; s++)
{
complex_t L_Val = *(L + matrixToArray(k, s, n));
complex_t U_Val = *(U + matrixToArray(s, j, n));;
currentSum += L_Val * U_Val;
}

// U_kj = A_kj - currentSum
*(U + matrixToArray(k, j, n)) = *(A + matrixToArray(k, j, n)) - currentSum;
}

for (int i = k + 1; i < n; i++)
{
complex_t currentSum = 0;
for (int s = 0; s < k; s++)
{
complex_t L_Val = *(L + matrixToArray(i, s, n));
complex_t U_Val = *(U + matrixToArray(s, k, n));;
currentSum += L_Val * U_Val;
}

// L_ik = (A_ik - currentSum)/U_kk
*(L + matrixToArray(i, k, n)) = (*(A + matrixToArray(i, k, n)) - currentSum)
 / *(U + matrixToArray(k, k, n));
}


}
```

```
}



__device__ void matrixVectorProduct(double *M, double *v,
 double *y, int n)
{
for (int i = 0; i < n; i++)
{
*(y + i) = 0;
for (int j = 0; j < n; j++)
{
*(y + i) += *(M + matrixToArray(i, j, n)) * *(v + j);
}
}
}

// Solves Ax = b for x, where A has been decomposed into upper
 (U) and lower (L) triangular matrices.
//
//       LUx = b -> first, let Ux = z => Lz = b, solve for z.
//
//       Second, solve Ux = z for x.
// Code written from pseudocode in Cheney/Kincaid, pg 301.
__device__ void linSolve(complex_t *A, complex_t *L,
complex_t *U, complex_t *b, int n)
{
complex_t *x = &A[0];
```

```
complex_t *z = &A[n];

for (int i = 0; i < n; i++)
{
*(z + i) = 0;
}
// ******************
// Solve Lz = b for z.
// ******************

// z_0 = b_0.
*(z) = *(b);

// finding the rest of z
for (int i = 1; i < n; i++)
{
complex_t currentSum = 0;
for (int j = 0; j < i; j++)
{
complex_t lVal = *(L + matrixToArray(i, j, n));
complex_t zVal = *(z + j);
currentSum += lVal * zVal;
}

*(z + i) = *(b + i) - currentSum;
}
```

```
// ******************
// Solve Ux = z for x.
// ******************

//x_n = z_n/U_nn
*(x + (n - 1)) = *(z + (n - 1)) / *(U + matrixToArray(n - 1, n - 1, n));

// finding the rest of x
for (int i = (n - 2); i >= 0; i--)
{
complex_t currentSum = 0;
for (int j = i + 1; j < n; j++)
{
complex_t uVal = *(U + matrixToArray(i, j, n));
complex_t xVal = *(x + j);
currentSum += uVal * xVal;
}

*(x + i) = (*(z + i) - currentSum) / *(U + matrixToArray(i, i, n));
}

}

__device__ void LUDecomp_PartialPivot(complex_t *A, complex_t *L,
 complex_t *U, complex_t *P, int n)
{
```

```
//printf("A(device) = \n");
for (int row = 0; row < n; row++)
{
for (int col = 0; col < n; col++)
{
//printf("%0.2f ", *(A + matrixToArray(row, col, n)));
*(L + matrixToArray(row, col, n)) = complex_t(0, 0);
*(U + matrixToArray(row, col, n)) = complex_t(0, 0);
*(P + matrixToArray(row, col, n)) = complex_t(0, 0);
}
}

for (int k = 0; k < n; k++)
{
*(P + matrixToArray(k, k, n)) = complex_t(1, 0);
}

int pivotRow;
complex_t pivotVal;
complex_t pivotCandidate;

complex_t tempP;
complex_t tempA;
for (int k = 0; k < n; k++)
{
// L_kk = 1 (set 1 along the diagonal of L)
*(L + matrixToArray(k, k, n)) = complex_t(1, 0);
```

```
// Pivoting
pivotRow = k;
pivotVal = *(A + matrixToArray(k, k, n));
double pivotNorm = pivotVal.real() * pivotVal.real() + pivotVal.imag()
 * pivotVal.imag();
for (int i = k + 1; i < n; i++)
{
pivotCandidate = *(A + matrixToArray(i, k, n));
double candidateNorm = pivotCandidate.real() * pivotCandidate.real() +
 pivotCandidate.imag() * pivotCandidate.imag();
if (candidateNorm > pivotNorm)
{
pivotVal = pivotCandidate;
pivotRow = i;
pivotNorm = candidateNorm;
}
}

// Swapping rows if necessary.
if (pivotRow != k)
{
for (int j = 0; j < n; j++)
{
// swap rows pivotRow and k of P
tempP = *(P + matrixToArray(pivotRow, j, n));
*(P + matrixToArray(pivotRow, j, n)) = *(P + matrixToArray(k, j, n));
```

```
*(P + matrixToArray(k, j, n)) = tempP;

// swap rows pivotRow and k of A
tempA = *(A + matrixToArray(pivotRow, j, n));
*(A + matrixToArray(pivotRow, j, n)) = *(A + matrixToArray(k, j, n));
*(A + matrixToArray(k, j, n)) = tempA;
}

if (k >= 1)
{
// interchange rows pivotRow and k in cols 1:k-1 of L
for (int j = 0; j < k; j++)
{
tempA = *(L + matrixToArray(pivotRow, j, n));
*(L + matrixToArray(pivotRow, j, n)) = *(L + matrixToArray(k, j, n));
*(L + matrixToArray(k, j, n)) = tempA;
}

}
}

// Peforming gaussian elimination
for (int i = k + 1; i < n; i++)
{
// L[i][k] = A[i][k] / A[k][k]
*(L + matrixToArray(i, k, n)) = *(A + matrixToArray(i, k, n)) /
 *(A + matrixToArray(k, k, n));
```

```
for (int j = k + 1; j < n; j++)
{
// A[i][j] = A[i][j] - L[i][k] * A[k][j]
*(A + matrixToArray(i, j, n)) -= *(L + matrixToArray(i, k, n)) *
 *(A + matrixToArray(k, j, n));
}
}

for (int j = k; j < n; j++)
{
//U[k][j] = A[k][j];
*(U + matrixToArray(k, j, n)) = *(A + matrixToArray(k, j, n));
}
}
}



// Solves the linear system Ax = b.
// Stores the value of x in A from 0:n
__device__ void solveLinearSystem(complex_t *A, complex_t *L,
 complex_t *U, complex_t *P, complex_t *b, int n, double t)
{

LUDecomp_PartialPivot(A, L, U, P, n);
//if (threadIdx.x == 0 && t == 1.0)
//{
```

```
//      printf("A = \n");
//      for (int i = 0; i < n; i++)
//      {
//              for (int j = 0; j < n; j++)
//              {
//                      complex_t currentVal = *(A + (i * n) + j);
//                      printf("(%0.5f,  %0.5f) ", currentVal.real(),
 currentVal.imag());
//              }
//              printf("\n");
//      }
//      printf("\n\n");

//      printf("L = \n");
//      for (int i = 0; i < n; i++)
//      {
//              for (int j = 0; j < n; j++)
//              {
//                      printf("%f ", *(L + (i * n) + j));
//              }
//              printf("\n");
//      }
//      printf("\n\n");
//}

// Permutating b vec by the permutation matrix P
matrixVectorProduct(P, b, &A[0], n);
```

```
for (int i = 0; i < n; i++)
{
*(b + i) = *(A + i);
}

linSolve(A, L, U, b, n);
// A from 0:n holds the value of x



}
```

B.1.4 PredictorCorrector.cuh

```
#pragma once
#ifndef PREDICTORCORRECTOR_CUH
#define PREDICTORCORRECTOR_CUH

#define _USE_MATH_DEFINES

#include <assert.h>
#include <cublas_v2.h>
#include "Definitions.h"

#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include "LinearSolver.cuh"
```

```
using namespace thrust;

__global__ void randomizeTaskPositions(complex_t *startParameters,
 complex_t *finalParameters, complex_t *startPoints, double *taskPositions,
double *matrixWorkspace, complex_t *taskPositionWorkspace, double kTolX,
double kTolY, double kTolTheta, int kNumParametersToCopy,
int kNumTPParametersToCopy);

__global__ void trackPaths(complex_t *xArray, complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspaceArray,
complex_t *LArray, complex_t *UArray, complex_t *PArray,
complex_t *bArray, complex_t *xPredictArray, complex_t *xWorkspaceArray,
int n, int numSteps, const double stepSize, int numParameters,
int *solFlagArray);

__global__ void H_global(double t, complex_t *xArray,
complex_t *startParams, complex_t *finalParams, complex_t *bArray,
complex_t gamma, int n, int *solFlagArray, int numParameters);

__global__ void filterAndTransformSolutions(complex_t *solutions,
double *taskPositionArray, int *solFlagArray, const int n,
const int kNumTPParameters);


#endif
```

B.1.5 PredictorCorrector.cu

```
#include "PredictorCorrector.cuh"

#include <curand_kernel.h>

#define Power pow



const int kNumParamsPerTP = 3;
const int kLDA = 3; // Leading dimension of our homogeneous transform
 matrices

__device__ void isComplexConjugate(complex_t v1, complex_t v2,
 const double tol, bool &flag)
{
if (abs(v1.real()) - abs(v2.real()) < tol)
{
if (v1.imag() + v2.imag() < tol) // if v1.imag = -v2.imag
{
flag = true; // v1 and v2 are complex conjugate pairs
}
else
{
flag = false;
}
```

```
}
else
{
flag = false;
}
}

__global__ void filterAndTransformSolutions(complex_t *solutions,
double *taskPositionArray, int *solFlagArray, const int n,
 const int kNumTPParameters)
{
// Calculating the array offsets for the current thread
int xStart = (blockIdx.x * blockDim.x + threadIdx.x) * n;

complex_t *xPtr = &solutions[xStart];

// checking if solutions are true complex conjugate pairs
complex_t O = xPtr[0];
complex_t Oc = xPtr[1];
complex_t A = xPtr[2];
complex_t Ac = xPtr[3];
complex_t B = xPtr[4];
complex_t Bc = xPtr[5];
complex_t C = xPtr[6];
complex_t Cc = xPtr[7];

const double kTol = 0.001; // Tolerance for comparing doubles
```

```
bool complexConjugateFlag = false;
int checkSum = 0;
isComplexConjugate(O, Oc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(A, Ac, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(B, Bc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(C, Cc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}

if (checkSum == 4)
{
```

```
//complex_t O = xPtr[0];
//complex_t Oc = xPtr[1];
//complex_t A = xPtr[2];
//complex_t Ac = xPtr[3];
//complex_t B = xPtr[4];
//complex_t Bc = xPtr[5];
//complex_t C = xPtr[6];
//complex_t Cc = xPtr[7];

//printf("Effective thread idx: %d, Solution: {%0.2f, %0.2f}
{%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f}
{%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f}\n\n",
//       blockIdx.x * blockDim.x + threadIdx.x,
//       O.real(), O.imag(), Oc.real(), Oc.imag(), A.real(), A.imag(),
 Ac.real(), Ac.imag(), B.real(), B.imag(), Bc.real(), Bc.imag(), C.real(),
  C.imag(), Cc.real(), Cc.imag());



// Transforming the solutions from the first frame to the global frame
double *tpPtr = &taskPositionArray[blockIdx.x * kNumTPParameters];
double theta0 = tpPtr[0];
double x0 = tpPtr[1];
double y0 = tpPtr[2];

double cosTheta = cos(theta0 * M_PI / 180.0);
double sinTheta = sin(theta0 * M_PI / 180.0);
for (int i = 0; i < n; i++)
```

```
{
complex_t currentSolCoord = xPtr[i];
double solXTemp = currentSolCoord.real();
double solYTemp = currentSolCoord.imag();

double solXTransformed = solXTemp * cosTheta - solYTemp * sinTheta + x0;
double solYTransformed = solXTemp * sinTheta + solYTemp * cosTheta + y0;

xPtr[i] = complex_t(solXTransformed, solYTransformed);
}



solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 1;

}
else
{
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = -1;
}


}

__global__ void randomizeTaskPositions(complex_t *startParameters,
 complex_t *finalParameters, complex_t *startPoints, double *taskPositions,
double *matrixWorkspace, complex_t *taskPositionWorkspace, double kTolX,
```

```
 double kTolY, double kTolTheta, int kNumParametersToCopy,
 int kNumTPParametersToCopy)
{
// The first block does not require randomization.
if (blockIdx.x != 0)
{
// Only the first thread in the block needs to do the work
if (threadIdx.x == 0)
{
int tpOffset = blockIdx.x * kNumTPParametersToCopy;
double *tpPtr = &taskPositions[tpOffset];
complex_t *tpWorkspacePtr = &taskPositionWorkspace[tpOffset];

int relTPOffset = blockIdx.x * kNumParametersToCopy;
complex_t *relTPPtr = &finalParameters[relTPOffset];

int matOffset = blockIdx.x * kLDA * kLDA * kNumTaskPositions;
double *matPtr = &matrixWorkspace[matOffset];



// Copy the start Parameters from the first block to the current block.
for (int i = 0; i < kNumParametersToCopy; i++)
{
startParameters[relTPOffset + i] = startParameters[i];
}

// init RNG library
```

```
curandState_t curandState;
curand_init(1234, blockIdx.x, 0, &curandState);

// Randomize X, Y and theta
for (int i = 0; i < kNumTaskPositions; i++)
{

int startTPIdx = (i * kNumParamsPerTP);
int randomTPIdx = startTPIdx + tpOffset;

taskPositions[randomTPIdx] = taskPositions[startTPIdx] + -kTolX +
 curand_uniform_double(&curandState) * (2.0f * kTolX); // Randomizing X
taskPositions[randomTPIdx + 1] = taskPositions[startTPIdx + 1] + -kTolY +
 curand_uniform_double(&curandState) * (2.0f * kTolY); // Randomizing Y
taskPositions[randomTPIdx + 2] = taskPositions[startTPIdx + 2] + -kTolTheta +
curand_uniform_double(&curandState) * (2.0f * kTolTheta);
// Randomizing Theta

}

// create 3x3 homogeneous transformation matrices from the task positions
for (int i = 0; i < kNumTaskPositions; i++)
{
double theta_DEG = *(tpPtr + i * 3);
double currentX = *(tpPtr + i * 3 + 1);
double currentY = *(tpPtr + i * 3 + 2);
```

```
double theta_RAD = theta_DEG * (M_PI / 180.0);

const int matOffset = i * 3 * 3;
matPtr[matOffset] = cos(theta_RAD);
matPtr[matOffset + 1] = -sin(theta_RAD);
matPtr[matOffset + 2] = currentX;
matPtr[matOffset + 3] = sin(theta_RAD);
matPtr[matOffset + 4] = cos(theta_RAD);
matPtr[matOffset + 5] = currentY;
matPtr[matOffset + 6] = 0;
matPtr[matOffset + 7] = 0;
matPtr[matOffset + 8] = 1;
}

//// DEBUGGING: storing the first matrix in the second matrix spot to
 check if H^{-1}H = I.
//if (blockIdx.x == 1)
//{
//      const int matOffset = 9;
//      for (int i = 0; i < 9; i++)
//      {
//              matPtr[matOffset + i] = matPtr[i];
//      }
//}

// Invert first matrix, as this is a 3x3 homogeneous transform, the
// inverse has a very specific structure that is easy to compute.
```

```
// transpose the first 2x2 rotation submatrix of the first homogeneous
 transform
double tempVal;
tempVal = matPtr[matrixToArray(0, 1, 3)];
matPtr[matrixToArray(0, 1, 3)] = matPtr[matrixToArray(1, 0, 3)];
matPtr[matrixToArray(1, 0, 3)] = tempVal;

// set the translational part to be -[rotation submatrix (transposed)].
[(x, y) position vector]
// note: have to store H[[0, 2]] as tempVal as we need the original value
 for
 H[[1, 2]] as well.
tempVal = matPtr[matrixToArray(0, 2, 3)];
matPtr[matrixToArray(0, 2, 3)] = -(matPtr[matrixToArray(0, 0, 3)] *
 matPtr[matrixToArray(0, 2, 3)] + matPtr[matrixToArray(0, 1, 3)] *
  matPtr[matrixToArray(1, 2, 3)]);
matPtr[matrixToArray(1, 2, 3)] = -(matPtr[matrixToArray(1, 0, 3)] *
tempVal + matPtr[matrixToArray(1, 1, 3)] * matPtr[matrixToArray(1, 2, 3)]);

// Multiply the rest of the matrices by the first matrix
double cosThetaRel;
double sinThetaRel;
double relX;
double relY;

const int numRelTaskPositions = kNumTaskPositions - 1;
```

```
for␣(int␣i␣=␣1;␣i␣<␣kNumTaskPositions;␣i++)
{

//␣note:␣this␣part␣is␣doing␣H0^{-1}*H_i␣manually,␣for␣just␣the␣parts
␣we␣need.␣Wherever␣there␣is␣a␣(matrixOffset␣+)␣refers␣to␣the␣H_i␣matrix,
//␣while␣if␣there␣is␣no␣offset␣it␣is␣referencing␣the␣inverse␣of␣the␣first
␣TP␣matrix.
const␣int␣matOffset␣=␣i␣*␣3␣*␣3;
cosThetaRel␣=␣matPtr[matrixToArray(0,␣0,␣3)]␣*␣matPtr[matOffset␣+
␣matrixToArray(0,␣0,␣3)]␣+␣matPtr[matrixToArray(0,␣1,␣3)]␣*
␣␣matPtr[matOffset␣+␣matrixToArray(1,␣0,␣3)];
sinThetaRel␣=␣matPtr[matrixToArray(1,␣0,␣3)]␣*␣matPtr[matOffset␣+
␣matrixToArray(0,␣0,␣3)]␣+␣matPtr[matrixToArray(1,␣1,␣3)]␣*
␣matPtr[matOffset␣+␣matrixToArray(1,␣0,␣3)];

//␣recover␣and␣store␣the␣new␣relative␣(and␣randomized)␣task␣positions
relX␣=␣matPtr[matrixToArray(0,␣0,␣3)]␣*␣matPtr[matOffset␣+
␣matrixToArray(0,␣2,␣3)]␣+␣matPtr[matrixToArray(0,␣1,␣3)]␣*
␣␣matPtr[matOffset␣+␣matrixToArray(1,␣2,␣3)]␣+
␣␣matPtr[matrixToArray(0,␣2,␣3)];
relY␣=␣matPtr[matrixToArray(1,␣0,␣3)]␣*␣matPtr[matOffset␣+
matrixToArray(0,␣2,␣3)]␣+␣matPtr[matrixToArray(1,␣1,␣3)]␣*
matPtr[matOffset␣+␣matrixToArray(1,␣2,␣3)]␣+
␣matPtr[matrixToArray(1,␣2,␣3)];

//␣storing␣T1,␣P1,␣T2,␣P2,␣...,␣in␣taskPositionWorkspace␣before
```

```
 transferring
// them to final parameters in the correct order
tpWorkspacePtr[i * 2 - 2] = complex_t(cosThetaRel, sinThetaRel);
tpWorkspacePtr[i * 2 - 1] = complex_t(relX, relY);

// transferring the new relative task positions to the final parameters
// expected form is T1, T2, ..., T4, Tc1, Tc2, ..., Tc4, P1, P2, ...,
 P4, Pc1, Pc2, ..., Pc4
for (int i = 0; i < numRelTaskPositions; i++)
{
complex_t Ti = tpWorkspacePtr[i * 2];
complex_t Pi = tpWorkspacePtr[i * 2 + 1];
relTPPtr[i] = Ti;
relTPPtr[i + numRelTaskPositions] = complex_t(Ti.real(), -Ti.imag());
 // Tc_i
relTPPtr[i + 2 * numRelTaskPositions] = Pi;
relTPPtr[i + 3 * numRelTaskPositions] = complex_t(Pi.real(), -Pi.imag());
 // Pc_i
}

}
}

}

// Returns the value of the homotopy system at point x at time t.
```

```
// effectively, evaluates H(t, x(t))
__device__ void H_device(double t, complex_t *x, complex_t *startParams,
 complex_t *finalParams, complex_t *b, complex_t gamma, int n)
{

complex_t T1Start, T2Start, T3Start, T4Start, Tc1Start, Tc2Start,
 Tc3Start, Tc4Start;
complex_t P1Start, P2Start, P3Start, P4Start, Pc1Start, Pc2Start,
 Pc3Start, Pc4Start;
T1Start = startParams[0]; T2Start = startParams[1]; T3Start =
startParams[2]; T4Start = startParams[3];
Tc1Start = startParams[4]; Tc2Start = startParams[5]; Tc3Start =
startParams[6]; Tc4Start = startParams[7];
P1Start = startParams[8]; P2Start = startParams[9]; P3Start =
 startParams[10]; P4Start = startParams[11];
Pc1Start = startParams[12]; Pc2Start = startParams[13]; Pc3Start
 = startParams[14]; Pc4Start = startParams[15];

complex_t T1Final, T2Final, T3Final, T4Final, Tc1Final, Tc2Final,
 Tc3Final, Tc4Final;
complex_t P1Final, P2Final, P3Final, P4Final, Pc1Final, Pc2Final,
 Pc3Final, Pc4Final;
T1Final = finalParams[0]; T2Final = finalParams[1]; T3Final =
 finalParams[2]; T4Final = finalParams[3];
Tc1Final = finalParams[4]; Tc2Final = finalParams[5]; Tc3Final =
finalParams[6]; Tc4Final = finalParams[7];
P1Final = finalParams[8]; P2Final = finalParams[9]; P3Final =
```

```
finalParams[10];␣P4Final␣=␣finalParams[11];
Pc1Final␣=␣finalParams[12];␣Pc2Final␣=␣finalParams[13];␣Pc3Final␣=
␣finalParams[14];␣Pc4Final␣=␣finalParams[15];

complex_t␣O,␣Oc,␣A,␣Ac,␣B,␣Bc,␣C,␣Cc;
O␣=␣x[0];␣Oc␣=␣x[1];␣A␣=␣x[2];␣Ac␣=␣x[3];␣B␣=␣x[4];␣Bc␣=␣x[5];
␣C␣=␣x[6];␣Cc␣=␣x[7];

complex_t␣f1,␣f2,␣f3,␣f4,␣f5,␣f6,␣f7,␣f8;
f1␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+
␣A*((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*(-Oc␣+␣Pc1Final*(1␣-␣t)␣+
␣␣gamma*Pc1Start*t␣+␣Ac*((1␣-␣t)*Tc1Final␣+␣gamma*t*Tc1Start));
f2␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+
␣B*((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*(-Cc␣+␣Pc1Final*(1␣-␣t)␣+
␣␣gamma*Pc1Start*t␣+␣Bc*((1␣-␣t)*Tc1Final␣+␣gamma*t*Tc1Start));
f3␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P2Final*(1␣-␣t)␣+␣gamma*P2Start*t␣+
␣A*((1␣-␣t)*T2Final␣+␣gamma*t*T2Start))*(-Oc␣+␣Pc2Final*(1␣-␣t)␣+
␣␣gamma*Pc2Start*t␣+␣Ac*((1␣-␣t)*Tc2Final␣+␣gamma*t*Tc2Start));
f4␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P2Final*(1␣-␣t)␣+␣gamma*P2Start*t␣+
␣B*((1␣-␣t)*T2Final␣+␣gamma*t*T2Start))*(-Cc␣+␣Pc2Final*(1␣-␣t)␣+
␣␣gamma*Pc2Start*t␣+␣Bc*((1␣-␣t)*Tc2Final␣+␣gamma*t*Tc2Start));
f5␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+
␣A*((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*(-Oc␣+␣Pc3Final*(1␣-␣t)␣+
␣␣gamma*Pc3Start*t␣+␣Ac*((1␣-␣t)*Tc3Final␣+␣gamma*t*Tc3Start));
f6␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+
␣B*((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*(-Cc␣+␣Pc3Final*(1␣-␣t)␣+
␣␣gamma*Pc3Start*t␣+␣Bc*((1␣-␣t)*Tc3Final␣+␣gamma*t*Tc3Start));
```

```
f7 = -((A - O)*(Ac - Oc)) + (-O + P4Final*(1 - t) + gamma*P4Start*t +
 A*((1 - t)*T4Final + gamma*t*T4Start))*(-Oc + Pc4Final*(1 - t) +
  gamma*Pc4Start*t + Ac*((1 - t)*Tc4Final + gamma*t*Tc4Start));
f8 = -((B - C)*(Bc - Cc)) + (-C + P4Final*(1 - t) + gamma*P4Start*t +
 B*((1 - t)*T4Final + gamma*t*T4Start))*(-Cc + Pc4Final*(1 - t) +
 gamma*Pc4Start*t + Bc*((1 - t)*Tc4Final + gamma*t*Tc4Start));



//complex_t residual = sqrt(f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 +
f6*f6 + f7*f7 + f8*f8);
//printf("Residual: (%e, %e)\n", residual.real(), residual.imag());
b[0] = f1;
b[1] = f2;
b[2] = f3;
b[3] = f4;
b[4] = f5;
b[5] = f6;
b[6] = f7;
b[7] = f8;

}

// prints out the value of H at time t for all start points.
 Does not populate bArray. Intended for use in debugging.
__device__ void H_DEBUG_DEVICE(double t, complex_t *x,
 complex_t *startParams, complex_t *finalParams, complex_t *bArray,
```

```
  complex_t gamma, int n)
{

complex_t T1Start, T2Start, T3Start, T4Start, Tc1Start, Tc2Start,
 Tc3Start, Tc4Start;
complex_t P1Start, P2Start, P3Start, P4Start, Pc1Start, Pc2Start,
 Pc3Start, Pc4Start;
T1Start = startParams[0]; T2Start = startParams[1]; T3Start =
 startParams[2]; T4Start = startParams[3];
Tc1Start = startParams[4]; Tc2Start = startParams[5]; Tc3Start =
 startParams[6]; Tc4Start = startParams[7];
P1Start = startParams[8]; P2Start = startParams[9]; P3Start =
 startParams[10]; P4Start = startParams[11];
Pc1Start = startParams[12]; Pc2Start = startParams[13]; Pc3Start =
 startParams[14]; Pc4Start = startParams[15];

complex_t T1Final, T2Final, T3Final, T4Final, Tc1Final, Tc2Final,
 Tc3Final, Tc4Final;
complex_t P1Final, P2Final, P3Final, P4Final, Pc1Final, Pc2Final,
 Pc3Final, Pc4Final;
T1Final = finalParams[0]; T2Final = finalParams[1]; T3Final =
finalParams[2]; T4Final = finalParams[3];
Tc1Final = finalParams[4]; Tc2Final = finalParams[5]; Tc3Final =
finalParams[6]; Tc4Final = finalParams[7];
P1Final = finalParams[8]; P2Final = finalParams[9]; P3Final =
finalParams[10]; P4Final = finalParams[11];
Pc1Final = finalParams[12]; Pc2Final = finalParams[13]; Pc3Final =
```

```
finalParams[14]; Pc4Final = finalParams[15];

complex_t O, Oc, A, Ac, B, Bc, C, Cc;
O = x[0]; Oc = x[1]; A = x[2]; Ac = x[3]; B = x[4]; Bc = x[5];
 C = x[6]; Cc = x[7];

complex_t f1, f2, f3, f4, f5, f6, f7, f8;
f1 = -((A - O)*(Ac - Oc)) + (-O + P1Final*(1 - t) + gamma*P1Start*t +
 A*((1 - t)*T1Final + gamma*t*T1Start))*(-Oc + Pc1Final*(1 - t) +
  gamma*Pc1Start*t + Ac*((1 - t)*Tc1Final + gamma*t*Tc1Start));
f2 = -((B - C)*(Bc - Cc)) + (-C + P1Final*(1 - t) + gamma*P1Start*t +
B*((1 - t)*T1Final + gamma*t*T1Start))*(-Cc + Pc1Final*(1 - t) +
gamma*Pc1Start*t + Bc*((1 - t)*Tc1Final + gamma*t*Tc1Start));
f3 = -((A - O)*(Ac - Oc)) + (-O + P2Final*(1 - t) + gamma*P2Start*t +
 A*((1 - t)*T2Final + gamma*t*T2Start))*(-Oc + Pc2Final*(1 - t) +
  gamma*Pc2Start*t + Ac*((1 - t)*Tc2Final + gamma*t*Tc2Start));
f4 = -((B - C)*(Bc - Cc)) + (-C + P2Final*(1 - t) + gamma*P2Start*t +
 B*((1 - t)*T2Final + gamma*t*T2Start))*(-Cc + Pc2Final*(1 - t) +
  gamma*Pc2Start*t + Bc*((1 - t)*Tc2Final + gamma*t*Tc2Start));
f5 = -((A - O)*(Ac - Oc)) + (-O + P3Final*(1 - t) + gamma*P3Start*t +
 A*((1 - t)*T3Final + gamma*t*T3Start))*(-Oc + Pc3Final*(1 - t) +
  gamma*Pc3Start*t + Ac*((1 - t)*Tc3Final + gamma*t*Tc3Start));
f6 = -((B - C)*(Bc - Cc)) + (-C + P3Final*(1 - t) + gamma*P3Start*t +
 B*((1 - t)*T3Final + gamma*t*T3Start))*(-Cc + Pc3Final*(1 - t) +
  gamma*Pc3Start*t + Bc*((1 - t)*Tc3Final + gamma*t*Tc3Start));
f7 = -((A - O)*(Ac - Oc)) + (-O + P4Final*(1 - t) + gamma*P4Start*t +
 A*((1 - t)*T4Final + gamma*t*T4Start))*(-Oc + Pc4Final*(1 - t) +
```

```
  gamma*Pc4Start*t + Ac*((1 - t)*Tc4Final + gamma*t*Tc4Start));
f8 = -((B - C)*(Bc - Cc)) + (-C + P4Final*(1 - t) + gamma*P4Start*t +
 B*((1 - t)*T4Final + gamma*t*T4Start))*(-Cc + Pc4Final*(1 - t) +
  gamma*Pc4Start*t + Bc*((1 - t)*Tc4Final + gamma*t*Tc4Start));




complex_t residual = sqrt(f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 +
 f6*f6 + f7*f7 + f8*f8);
printf("Residual: (%e, %e)\n", residual.real(), residual.imag());

}

// prints out the value of H at time t for all start points. Does not
 populate bArray. Intended for use in debugging.
__global__ void H_global(double t, complex_t *xArray,
 complex_t *startParams, complex_t *finalParams, complex_t *bArray,
  complex_t gamma, int n, int *solFlagArray, int numParameters)
{
int xIndex = (blockIdx.x * blockDim.x + threadIdx.x) * n;

int parameterOffset = blockIdx.x * numParameters;
complex_t *finalParamPtr = &finalParams[parameterOffset];

complex_t *x = &xArray[xIndex];
```

```
complex_t T1Start, T2Start, T3Start, T4Start, Tc1Start, Tc2Start,
 Tc3Start, Tc4Start;
complex_t P1Start, P2Start, P3Start, P4Start, Pc1Start, Pc2Start,
 Pc3Start, Pc4Start;
T1Start = startParams[0]; T2Start = startParams[1]; T3Start =
 startParams[2]; T4Start = startParams[3];
Tc1Start = startParams[4]; Tc2Start = startParams[5]; Tc3Start =
 startParams[6]; Tc4Start = startParams[7];
P1Start = startParams[8]; P2Start = startParams[9]; P3Start =
 startParams[10]; P4Start = startParams[11];
Pc1Start = startParams[12]; Pc2Start = startParams[13]; Pc3Start =
 startParams[14]; Pc4Start = startParams[15];

complex_t T1Final, T2Final, T3Final, T4Final, Tc1Final, Tc2Final,
 Tc3Final, Tc4Final;
complex_t P1Final, P2Final, P3Final, P4Final, Pc1Final, Pc2Final,
 Pc3Final, Pc4Final;
T1Final = finalParamPtr[0]; T2Final = finalParamPtr[1]; T3Final =
 finalParamPtr[2]; T4Final = finalParamPtr[3];
Tc1Final = finalParamPtr[4]; Tc2Final = finalParamPtr[5]; Tc3Final =
 finalParamPtr[6]; Tc4Final = finalParamPtr[7];
P1Final = finalParamPtr[8]; P2Final = finalParamPtr[9]; P3Final =
 finalParamPtr[10]; P4Final = finalParamPtr[11];
Pc1Final = finalParamPtr[12]; Pc2Final = finalParamPtr[13]; Pc3Final =
 finalParamPtr[14]; Pc4Final = finalParamPtr[15];

complex_t O, Oc, A, Ac, B, Bc, C, Cc;
```

```
O␣=␣x[0];␣Oc␣=␣x[1];␣A␣=␣x[2];␣Ac␣=␣x[3];␣B␣=␣x[4];␣Bc␣=␣x[5];
␣C␣=␣x[6];␣Cc␣=␣x[7];

complex_t␣f1,␣f2,␣f3,␣f4,␣f5,␣f6,␣f7,␣f8;
f1␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+
␣A*((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*(-Oc␣+␣Pc1Final*(1␣-␣t)␣+
␣␣gamma*Pc1Start*t␣+␣Ac*((1␣-␣t)*Tc1Final␣+␣gamma*t*Tc1Start));
f2␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+
␣B*((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*(-Cc␣+␣Pc1Final*(1␣-␣t)␣+
␣␣gamma*Pc1Start*t␣+␣Bc*((1␣-␣t)*Tc1Final␣+␣gamma*t*Tc1Start));
f3␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P2Final*(1␣-␣t)␣+␣gamma*P2Start*t␣+
␣A*((1␣-␣t)*T2Final␣+␣gamma*t*T2Start))*(-Oc␣+␣Pc2Final*(1␣-␣t)␣+
␣␣gamma*Pc2Start*t␣+␣Ac*((1␣-␣t)*Tc2Final␣+␣gamma*t*Tc2Start));
f4␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P2Final*(1␣-␣t)␣+␣gamma*P2Start*t␣+
␣B*((1␣-␣t)*T2Final␣+␣gamma*t*T2Start))*(-Cc␣+␣Pc2Final*(1␣-␣t)␣+
␣␣gamma*Pc2Start*t␣+␣Bc*((1␣-␣t)*Tc2Final␣+␣gamma*t*Tc2Start));
f5␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+
␣A*((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*(-Oc␣+␣Pc3Final*(1␣-␣t)␣+
␣␣gamma*Pc3Start*t␣+␣Ac*((1␣-␣t)*Tc3Final␣+␣gamma*t*Tc3Start));
f6␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+
␣B*((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*(-Cc␣+␣Pc3Final*(1␣-␣t)␣+
␣␣gamma*Pc3Start*t␣+␣Bc*((1␣-␣t)*Tc3Final␣+␣gamma*t*Tc3Start));
f7␣=␣-((A␣-␣O)*(Ac␣-␣Oc))␣+␣(-O␣+␣P4Final*(1␣-␣t)␣+␣gamma*P4Start*t␣+
␣A*((1␣-␣t)*T4Final␣+␣gamma*t*T4Start))*(-Oc␣+␣Pc4Final*(1␣-␣t)␣+
␣␣gamma*Pc4Start*t␣+␣Ac*((1␣-␣t)*Tc4Final␣+␣gamma*t*Tc4Start));
f8␣=␣-((B␣-␣C)*(Bc␣-␣Cc))␣+␣(-C␣+␣P4Final*(1␣-␣t)␣+␣gamma*P4Start*t␣+
␣B*((1␣-␣t)*T4Final␣+␣gamma*t*T4Start))*(-Cc␣+␣Pc4Final*(1␣-␣t)␣+
```

```
  gamma*Pc4Start*t + Bc*((1 - t)*Tc4Final + gamma*t*Tc4Start));




complex_t residual = sqrt(f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 +
 f6*f6 + f7*f7 + f8*f8);
printf("Effective Thread index: %d, Residual: (%e, %e)\n",
 blockIdx.x * blockDim.x + threadIdx.x, residual.real(),
  residual.imag());

if (residual.real() < 1e-8 && residual.imag() < 1e-8)
{
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 1;
} else {
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 0;
}




}

// Returns the value of the partial derivative of H with respect to
 t at point x at time t.
// effectively, dH/dt(t, x(t))
//
// *** NOTE: Actually returns -Ht, because we need to solve J(dx) =
```

```
␣−Ht␣for␣dx.
__device__␣void␣Ht(double␣t,␣complex_t␣*x,␣complex_t␣*startParams,
␣complex_t␣*finalParams,␣complex_t␣*b,␣complex_t␣gamma,␣int␣n)
{

complex_t␣T1Start,␣T2Start,␣T3Start,␣T4Start,␣Tc1Start,␣Tc2Start,
␣Tc3Start,␣Tc4Start;
complex_t␣P1Start,␣P2Start,␣P3Start,␣P4Start,␣Pc1Start,␣Pc2Start,
␣Pc3Start,␣Pc4Start;
T1Start␣=␣startParams[0];␣T2Start␣=␣startParams[1];␣T3Start␣=
startParams[2];␣T4Start␣=␣startParams[3];
Tc1Start␣=␣startParams[4];␣Tc2Start␣=␣startParams[5];␣Tc3Start␣=
␣startParams[6];␣Tc4Start␣=␣startParams[7];
P1Start␣=␣startParams[8];␣P2Start␣=␣startParams[9];␣P3Start␣=
startParams[10];␣P4Start␣=␣startParams[11];
Pc1Start␣=␣startParams[12];␣Pc2Start␣=␣startParams[13];␣Pc3Start␣=
␣startParams[14];␣Pc4Start␣=␣startParams[15];

complex_t␣T1Final,␣T2Final,␣T3Final,␣T4Final,␣Tc1Final,␣Tc2Final,
␣Tc3Final,␣Tc4Final;
complex_t␣P1Final,␣P2Final,␣P3Final,␣P4Final,␣Pc1Final,␣Pc2Final,
␣Pc3Final,␣Pc4Final;
T1Final␣=␣finalParams[0];␣T2Final␣=␣finalParams[1];␣T3Final␣=
␣finalParams[2];␣T4Final␣=␣finalParams[3];
Tc1Final␣=␣finalParams[4];␣Tc2Final␣=␣finalParams[5];␣Tc3Final␣=
␣finalParams[6];␣Tc4Final␣=␣finalParams[7];
P1Final␣=␣finalParams[8];␣P2Final␣=␣finalParams[9];␣P3Final␣=
```

```
 finalParams[10]; P4Final = finalParams[11];
Pc1Final = finalParams[12]; Pc2Final = finalParams[13]; Pc3Final =
 finalParams[14]; Pc4Final = finalParams[15];

complex_t O, Oc, A, Ac, B, Bc, C, Cc;
O = x[0]; Oc = x[1]; A = x[2]; Ac = x[3]; B = x[4]; Bc = x[5];
 C = x[6]; Cc = x[7];

complex_t Ht1, Ht2, Ht3, Ht4, Ht5, Ht6, Ht7, Ht8;
Ht1 = (-O + P1Final*(1 - t) + gamma*P1Start*t + A*((1 - t)*T1Final +
 gamma*t*T1Start))*(-Pc1Final + gamma*Pc1Start + Ac*(-Tc1Final +
  gamma*Tc1Start)) + (-P1Final + gamma*P1Start + A*(-T1Final +
   gamma*T1Start))*(-Oc + Pc1Final*(1 - t) + gamma*Pc1Start*t +
    Ac*((1 - t)*Tc1Final + gamma*t*Tc1Start));

Ht2 = (-C + P1Final*(1 - t) + gamma*P1Start*t + B*((1 - t)*T1Final +
 gamma*t*T1Start))*(-Pc1Final + gamma*Pc1Start + Bc*(-Tc1Final +
  gamma*Tc1Start)) + (-P1Final + gamma*P1Start + B*(-T1Final +
  gamma*T1Start))*(-Cc + Pc1Final*(1 - t) + gamma*Pc1Start*t +
  Bc*((1 - t)*Tc1Final + gamma*t*Tc1Start));

Ht3 = (-O + P2Final*(1 - t) + gamma*P2Start*t + A*((1 - t)*T2Final +
 gamma*t*T2Start))*(-Pc2Final + gamma*Pc2Start + Ac*(-Tc2Final +
  gamma*Tc2Start)) + (-P2Final + gamma*P2Start + A*(-T2Final +
   gamma*T2Start))*(-Oc + Pc2Final*(1 - t) + gamma*Pc2Start*t +
    Ac*((1 - t)*Tc2Final + gamma*t*Tc2Start));
```

```
Ht4␣=␣(-C␣+␣P2Final*(1␣-␣t)␣+␣gamma*P2Start*t␣+␣B*((1␣-␣t)*T2Final␣+
␣gamma*t*T2Start))*(-Pc2Final␣+␣gamma*Pc2Start␣+␣Bc*(-Tc2Final␣+
␣␣gamma*Tc2Start))␣+␣(-P2Final␣+␣gamma*P2Start␣+␣B*(-T2Final␣+
␣␣gamma*T2Start))*(-Cc␣+␣Pc2Final*(1␣-␣t)␣+␣gamma*Pc2Start*t␣+␣Bc*
␣␣((1␣-␣t)*Tc2Final␣+␣gamma*t*Tc2Start));

Ht5␣=␣(-O␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+␣A*((1␣-␣t)*T3Final␣+
␣gamma*t*T3Start))*(-Pc3Final␣+␣gamma*Pc3Start␣+␣Ac*(-Tc3Final␣+
␣␣gamma*Tc3Start))␣+␣(-P3Final␣+␣gamma*P3Start␣+␣A*(-T3Final␣+␣gamma*
␣␣T3Start))*(-Oc␣+␣Pc3Final*(1␣-␣t)␣+␣gamma*Pc3Start*t␣+␣Ac*((1␣-␣t)*
␣␣Tc3Final␣+␣gamma*t*Tc3Start));

Ht6␣=␣(-C␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+␣B*((1␣-␣t)*T3Final␣+
␣gamma*t*T3Start))*(-Pc3Final␣+␣gamma*Pc3Start␣+␣Bc*(-Tc3Final␣+
␣␣gamma*Tc3Start))␣+␣(-P3Final␣+␣gamma*P3Start␣+␣B*(-T3Final␣+␣gamma*
␣␣T3Start))*(-Cc␣+␣Pc3Final*(1␣-␣t)␣+␣gamma*Pc3Start*t␣+␣Bc*((1␣-␣t)*
␣␣Tc3Final␣+␣gamma*t*Tc3Start));

Ht7␣=␣(-O␣+␣P4Final*(1␣-␣t)␣+␣gamma*P4Start*t␣+␣A*((1␣-␣t)*T4Final␣+
gamma*t*T4Start))*(-Pc4Final␣+␣gamma*Pc4Start␣+␣Ac*(-Tc4Final␣+␣gamma*
Tc4Start))␣+␣(-P4Final␣+␣gamma*P4Start␣+␣A*(-T4Final␣+␣gamma*T4Start))
*(-Oc␣+␣Pc4Final*(1␣-␣t)␣+␣gamma*Pc4Start*t␣+␣Ac*((1␣-␣t)*Tc4Final␣+
␣gamma*t*Tc4Start));

Ht8␣=␣(-C␣+␣P4Final*(1␣-␣t)␣+␣gamma*P4Start*t␣+␣B*((1␣-␣t)*T4Final␣+
␣gamma*t*T4Start))*(-Pc4Final␣+␣gamma*Pc4Start␣+␣Bc*(-Tc4Final␣+
␣␣gamma*Tc4Start))␣+␣(-P4Final␣+␣gamma*P4Start␣+␣B*(-T4Final␣+␣gamma*
```

```
  T4Start))*(-Cc + Pc4Final*(1 - t) + gamma*Pc4Start*t + Bc*((1 - t)*
  Tc4Final + gamma*t*Tc4Start));


// Populating b with the values of -Ht.
complex_t *HtVec = b;
HtVec[0] = -Ht1;
HtVec[1] = -Ht2;
HtVec[2] = -Ht3;
HtVec[3] = -Ht4;
HtVec[4] = -Ht5;
HtVec[5] = -Ht6;
HtVec[6] = -Ht7;
HtVec[7] = -Ht8;
}

// Returns the Jacobian Matrix of the Homotopy System at point x at
 time t.
// effectively, dH/dx(t, x(t))
__device__ void Hx(double t, complex_t* x, complex_t* startParams,
 complex_t* finalParams, complex_t* currentJMatWorkspace,
  complex_t gamma, int n)
{
complex_t T1Start, T2Start, T3Start, T4Start, Tc1Start, Tc2Start,
 Tc3Start, Tc4Start;
complex_t P1Start, P2Start, P3Start, P4Start, Pc1Start, Pc2Start,
 Pc3Start, Pc4Start;
```

```
T1Start␣=␣startParams[0];␣T2Start␣=␣startParams[1];␣T3Start␣=
␣startParams[2];␣T4Start␣=␣startParams[3];
Tc1Start␣=␣startParams[4];␣Tc2Start␣=␣startParams[5];␣Tc3Start␣=
startParams[6];␣Tc4Start␣=␣startParams[7];
P1Start␣=␣startParams[8];␣P2Start␣=␣startParams[9];␣P3Start␣=
startParams[10];␣P4Start␣=␣startParams[11];
Pc1Start␣=␣startParams[12];␣Pc2Start␣=␣startParams[13];␣Pc3Start␣=
␣startParams[14];␣Pc4Start␣=␣startParams[15];

complex_t␣T1Final,␣T2Final,␣T3Final,␣T4Final,␣Tc1Final,␣Tc2Final,
␣Tc3Final,␣Tc4Final;
complex_t␣P1Final,␣P2Final,␣P3Final,␣P4Final,␣Pc1Final,␣Pc2Final,
␣Pc3Final,␣Pc4Final;
T1Final␣=␣finalParams[0];␣T2Final␣=␣finalParams[1];␣T3Final␣=
␣finalParams[2];␣T4Final␣=␣finalParams[3];
Tc1Final␣=␣finalParams[4];␣Tc2Final␣=␣finalParams[5];␣Tc3Final␣=
␣finalParams[6];␣Tc4Final␣=␣finalParams[7];
P1Final␣=␣finalParams[8];␣P2Final␣=␣finalParams[9];␣P3Final␣=
finalParams[10];␣P4Final␣=␣finalParams[11];
Pc1Final␣=␣finalParams[12];␣Pc2Final␣=␣finalParams[13];␣Pc3Final␣=
␣finalParams[14];␣Pc4Final␣=␣finalParams[15];

complex_t␣O,␣Oc,␣A,␣Ac,␣B,␣Bc,␣C,␣Cc;
O␣=␣x[0];␣Oc␣=␣x[1];␣A␣=␣x[2];␣Ac␣=␣x[3];␣B␣=␣x[4];␣Bc␣=␣x[5];
␣C␣=␣x[6];␣Cc␣=␣x[7];

complex_t␣O1,␣Oc1,␣A1,␣Ac1,␣B2,␣Bc2,␣C2,␣Cc2,␣O3,␣Oc3,␣A3,␣Ac3,
```

```
␣B4,␣Bc4;
complex_t␣C4,␣Cc4,␣O5,␣Oc5,␣A5,␣Ac5,␣B6,␣Bc6,␣C6,␣Cc6,␣O7,␣Oc7,
␣A7,␣Ac7;
complex_t␣B8,␣Bc8,␣C8,␣Cc8;

O1␣=␣Ac␣-␣Pc1Final*(1␣-␣t)␣-␣gamma*Pc1Start*t␣-␣Ac*((1␣-␣t)*
Tc1Final␣+␣gamma*t*Tc1Start);
Oc1␣=␣A␣-␣P1Final*(1␣-␣t)␣-␣gamma*P1Start*t␣-␣A*((1␣-␣t)*T1Final␣+
␣gamma*t*T1Start);

A1␣=␣-Ac␣+␣Oc␣+␣((1␣-␣t)*T1Final␣+␣gamma*t*T1Start)*(-Oc␣+
␣Pc1Final*(1␣-␣t)␣+␣gamma*Pc1Start*t␣+␣Ac*((1␣-␣t)*Tc1Final␣+
␣␣gamma*t*Tc1Start));

Ac1␣=␣-A␣+␣O␣+␣(-O␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+␣A*
((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*((1␣-␣t)*Tc1Final␣+
␣gamma*t*Tc1Start);

B2␣=␣-Bc␣+␣Cc␣+␣((1␣-␣t)*T1Final␣+␣gamma*t*T1Start)*(-Cc␣+
Pc1Final*(1␣-␣t)␣+␣gamma*Pc1Start*t␣+␣Bc*((1␣-␣t)*Tc1Final␣+
␣gamma*t*Tc1Start));

Bc2␣=␣-B␣+␣C␣+␣(-C␣+␣P1Final*(1␣-␣t)␣+␣gamma*P1Start*t␣+␣B*
((1␣-␣t)*T1Final␣+␣gamma*t*T1Start))*((1␣-␣t)*Tc1Final␣+␣gamma*
t*Tc1Start);

C2␣=␣Bc␣-␣Pc1Final*(1␣-␣t)␣-␣gamma*Pc1Start*t␣-␣Bc*((1␣-␣t)*
```

```
Tc1Final + gamma*t*Tc1Start);

Cc2 = B - P1Final*(1 - t) - gamma*P1Start*t - B*((1 - t)*
T1Final + gamma*t*T1Start);

O3 = Ac - Pc2Final*(1 - t) - gamma*Pc2Start*t - Ac*((1 - t)*
Tc2Final + gamma*t*Tc2Start);

Oc3 = A - P2Final*(1 - t) - gamma*P2Start*t - A*((1 - t)*
T2Final + gamma*t*T2Start);

A3 = -Ac + Oc + ((1 - t)*T2Final + gamma*t*T2Start)*(-Oc +
 Pc2Final*(1 - t) + gamma*Pc2Start*t + Ac*((1 - t)*Tc2Final +
  gamma*t*Tc2Start));

Ac3 = -A + O + (-O + P2Final*(1 - t) + gamma*P2Start*t + A*
((1 - t)*T2Final + gamma*t*T2Start))*((1 - t)*Tc2Final +
 gamma*t*Tc2Start);

B4 = -Bc + Cc + ((1 - t)*T2Final + gamma*t*T2Start)*(-Cc +
 Pc2Final*(1 - t) + gamma*Pc2Start*t + Bc*((1 - t)*
 Tc2Final + gamma*t*Tc2Start));

Bc4 = -B + C + (-C + P2Final*(1 - t) + gamma*P2Start*t + B*
((1 - t)*T2Final + gamma*t*T2Start))*((1 - t)*Tc2Final +
 gamma*t*Tc2Start);
```

```
C4␣=␣Bc␣-␣Pc2Final*(1␣-␣t)␣-␣gamma*Pc2Start*t␣-␣Bc*((1␣-␣t)*
Tc2Final␣+␣gamma*t*Tc2Start);

Cc4␣=␣B␣-␣P2Final*(1␣-␣t)␣-␣gamma*P2Start*t␣-␣B*((1␣-␣t)*
T2Final␣+␣gamma*t*T2Start);

O5␣=␣Ac␣-␣Pc3Final*(1␣-␣t)␣-␣gamma*Pc3Start*t␣-␣Ac*((1␣-␣t)*
Tc3Final␣+␣gamma*t*Tc3Start);

Oc5␣=␣A␣-␣P3Final*(1␣-␣t)␣-␣gamma*P3Start*t␣-␣A*((1␣-␣t)*
T3Final␣+␣gamma*t*T3Start);

A5␣=␣-Ac␣+␣Oc␣+␣((1␣-␣t)*T3Final␣+␣gamma*t*T3Start)*(-Oc␣+
␣Pc3Final*(1␣-␣t)␣+␣gamma*Pc3Start*t␣+␣Ac*((1␣-␣t)*Tc3Final␣+
␣␣gamma*t*Tc3Start));

Ac5␣=␣-A␣+␣O␣+␣(-O␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+␣A*
((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*((1␣-␣t)*Tc3Final␣+
␣gamma*t*Tc3Start);

B6␣=␣-Bc␣+␣Cc␣+␣((1␣-␣t)*T3Final␣+␣gamma*t*T3Start)*(-Cc␣+
␣Pc3Final*(1␣-␣t)␣+␣gamma*Pc3Start*t␣+␣Bc*((1␣-␣t)*Tc3Final␣+
␣␣gamma*t*Tc3Start));

Bc6␣=␣-B␣+␣C␣+␣(-C␣+␣P3Final*(1␣-␣t)␣+␣gamma*P3Start*t␣+␣B*
((1␣-␣t)*T3Final␣+␣gamma*t*T3Start))*((1␣-␣t)*Tc3Final␣+
gamma*t*Tc3Start);
```

```
C6 = Bc - Pc3Final*(1 - t) - gamma*Pc3Start*t - Bc*((1 - t)*
Tc3Final + gamma*t*Tc3Start);

Cc6 = B - P3Final*(1 - t) - gamma*P3Start*t - B*((1 - t)*
T3Final + gamma*t*T3Start);

O7 = Ac - Pc4Final*(1 - t) - gamma*Pc4Start*t - Ac*((1 - t)*
Tc4Final + gamma*t*Tc4Start);

Oc7 = A - P4Final*(1 - t) - gamma*P4Start*t - A*((1 - t)*
T4Final + gamma*t*T4Start);

A7 = -Ac + Oc + ((1 - t)*T4Final + gamma*t*T4Start)*(-Oc +
 Pc4Final*(1 - t) + gamma*Pc4Start*t + Ac*((1 - t)*Tc4Final +
  gamma*t*Tc4Start));

Ac7 = -A + O + (-O + P4Final*(1 - t) + gamma*P4Start*t + A*
((1 - t)*T4Final + gamma*t*T4Start))*((1 - t)*Tc4Final +
 gamma*t*Tc4Start);

B8 = -Bc + Cc + ((1 - t)*T4Final + gamma*t*T4Start)*(-Cc +
 Pc4Final*(1 - t) + gamma*Pc4Start*t + Bc*((1 - t)*Tc4Final +
  gamma*t*Tc4Start));

Bc8 = -B + C + (-C + P4Final*(1 - t) + gamma*P4Start*t + B*
((1 - t)*T4Final + gamma*t*T4Start))*((1 - t)*Tc4Final +
```

```
 gamma*t*Tc4Start);

C8 = Bc - Pc4Final*(1 - t) - gamma*Pc4Start*t - Bc*((1 - t)*
Tc4Final + gamma*t*Tc4Start);

Cc8 = B - P4Final*(1 - t) - gamma*P4Start*t - B*((1 - t)*
T4Final + gamma*t*T4Start);



// Populating J Matrix with the jacobian values.
complex_t *J = currentJMatWorkspace;
J[0] = O1;
J[1] = Oc1;
J[2] = A1;
J[3] = Ac1;
J[4] = 0;
J[5] = 0;
J[6] = 0;
J[7] = 0;
J[8] = 0;
J[9] = 0;
J[10] = 0;
J[11] = 0;
J[12] = B2;
J[13] = Bc2;
J[14] = C2;
J[15] = Cc2;
```

```
J[16] = O3;
J[17] = Oc3;
J[18] = A3;
J[19] = Ac3;
J[20] = 0;
J[21] = 0;
J[22] = 0;
J[23] = 0;
J[24] = 0;
J[25] = 0;
J[26] = 0;
J[27] = 0;
J[28] = B4;
J[29] = Bc4;
J[30] = C4;
J[31] = Cc4;
J[32] = O5;
J[33] = Oc5;
J[34] = A5;
J[35] = Ac5;
J[36] = 0;
J[37] = 0;
J[38] = 0;
J[39] = 0;
J[40] = 0;
J[41] = 0;
J[42] = 0;
```

```
J[43] = 0;
J[44] = B6;
J[45] = Bc6;
J[46] = C6;
J[47] = Cc6;
J[48] = O7;
J[49] = Oc7;
J[50] = A7;
J[51] = Ac7;
J[52] = 0;
J[53] = 0;
J[54] = 0;
J[55] = 0;
J[56] = 0;
J[57] = 0;
J[58] = 0;
J[59] = 0;
J[60] = B8;
J[61] = Bc8;
J[62] = C8;
J[63] = Cc8;

}

// Returns the vector xDot from the Davidenko Differential Equation:
// xDot = -(J)^-1*Ht. Where J is the Jacobian Matrix of the Homotopy
 System and
```

```
// Ht is the partial derivative of the homotopy system with respect
to t.
//
// Note: Uses LU Factorization to find xDot instead of finding J^(-1).
 This solves the equation Jdx = -Ht for vector dx.
// This is also getting the equation for point x at time t.
//
__device__ void getDavidenkoEquation(double t, complex_t *x,
 complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
 complex_t *L, complex_t *U,
complex_t *P, complex_t *b, int n)
{

// Populate JMatWorkspace
Hx(t, x, startParams, finalParams, JMatWorkspace, gamma, n);

// Populate bArray with -Ht. Note: the Ht function returns the minus
 of dH/dt.
Ht(t, x, startParams, finalParams, b, gamma, n);

// solving Jdx = -Ht for dx.
solveLinearSystem(JMatWorkspace, L, U, P, b, n, t);

//JMatWorkspace should now hold the values of dx.
```

```
}

// Runge-Kutta 45 predictor step
// Calculates:  xPredict = x + (1 / 6)*(K1 + 2 * K2 + 2 * K3 + K4),
 where:
//       K1 = stepSize * getDavidenkoEquation(t, x, ....);
//       K2 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x +
 0.5 * K1, ....);
//       K3 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x +
 0.5 * K2, ....);
//       K4 = stepSize * getDavidenkoEquation(t + stepSize, x + K3, ....);
//
// NOTE: This method starts with xPredict = x, then adds K1, K2, K3,
 K4 on the fly as they are found.
// Therefore, xPredict = x + (1/6)K1 + (1/3)K2 + (1/3)K3 + (1/6)K4.
__device__ void predict(double t, const double stepSize, complex_t *x,
 complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
 complex_t *L, complex_t *U, complex_t *P,
complex_t *b, complex_t *xPredict, complex_t *predictWorkspace, int n)
{
// xPredict = x;
for (int i = 0; i < n; i++)
{
xPredict[i] = x[i];
}
```

```
//K1 = stepSize * getDavidenkoEquation(t, x, startParams, finalParams,
 gamma);
// Using xPredict and predictWorkspace to solve the linear system.
 NOTE: the result dx from getDavidenkoEquation()
// is stored in predictWorkspace
getDavidenkoEquation(t, x, startParams, finalParams, gamma, JMatWorkspace,
 L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K1 = stepSize * dx.
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/6) * K1;
xPredict[i] += (1 / 6.0) * JMatWorkspace[i];
}

// The input for K2 is x + 0.5 * K1. K1 is currently stored in
 predictWorkspace.
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + 0.5 * JMatWorkspace[i];
}

// K2 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x + 0.5 * K1,
 startParams, finalParams, gamma);
// NOTE: t and x are different than the call to getDavidenkoEquation()
 above.
getDavidenkoEquation(t + 0.5 * stepSize, predictWorkspace, startParams,
```

```
 finalParams, gamma, JMatWorkspace, L, U, P, b, n);

for (int i = 0; i < n; i++)
{
// K2 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/3) * K2
xPredict[i] += (1 / 3.0) * JMatWorkspace[i];
}



// The input for K3 is x + 0.5 * K2
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + 0.5 * JMatWorkspace[i];
}

//K3 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x + 0.5 * K2,
 startParams, finalParams, gamma);
getDavidenkoEquation(t + 0.5 * stepSize, predictWorkspace, startParams,
 finalParams, gamma, JMatWorkspace, L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K3 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/3) * K3
xPredict[i] += (1 / 3.0) * JMatWorkspace[i];
```

```
}

// K4's input is just x + K3.
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + JMatWorkspace[i];
}

//K4 = stepSize * getDavidenkoEquation(t + stepSize, x + K3, startParams,
finalParams, gamma);
// NOTE: t is different from K3 and K2's t.
getDavidenkoEquation(t + stepSize, predictWorkspace, startParams,
finalParams,
gamma, JMatWorkspace, L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K4 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/6) * K4
xPredict[i] += (1 / 6.0) * JMatWorkspace[i];
}



}

// Uses newton's method to correct the predicted point. Executes Newton's
```

```
 Method kNumNewtonCorrections times.
__device__ void correct(double t, complex_t *x, complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
 complex_t *L,
complex_t *U, complex_t *P, complex_t *b, complex_t *xCorrect, int n)
{

// xCorrect = x
for (int i = 0; i < n; i++)
{
xCorrect[i] = x[i];
}

for (int i = 0; i < kNumNewtonCorrections; i++)
{
// Populating JMatWorkspace with J
Hx(t, xCorrect, startParams, finalParams, JMatWorkspace, gamma, n);
// Populating b with HVec
H_device(t, xCorrect, startParams, finalParams, b, gamma, n);

// solve for J.dx = HVec
solveLinearSystem(JMatWorkspace, L, U, P, b, n, t);

// JMatWorkspace 0:n now holds the solution dx.

// xCorrect -= dx;
for (int j = 0; j < n; j++)
```

```
{
xCorrect[j] -= JMatWorkspace[j];
}

}

// x = xCorrect
for (int i = 0; i < n; i++)
{
x[i] = xCorrect[i];
}



}

// *******
//
//       MAIN HOMOTOPY CONTINUATION LOOP
//
// *******
__global__ void trackPaths(complex_t *xArray, complex_t *startParams,
 complex_t *finalParams,
complex_t gamma, complex_t *JMatWorkspaceArray, complex_t *LArray,
 complex_t *UArray, complex_t *PArray,
complex_t *bArray, complex_t *xPredictArray,
complex_t *xWorkspaceArray, int n, int numSteps, const double stepSize,
```

```
 int numParameters, int *solFlagArray)
{

// Calculating the array offsets for the current thread
int xStart = (blockIdx.x * blockDim.x  + threadIdx.x) * n;

int bStart = xStart;

complex_t *xPtr = &xArray[xStart];
complex_t *xWorkspacePtr = &xWorkspaceArray[xStart];
complex_t *xPredictPtr = &xPredictArray[xStart];
complex_t *bPtr = &bArray[bStart];

int JStart = (blockIdx.x * blockDim.x + threadIdx.x) * n * n;
complex_t *JPtr = &JMatWorkspaceArray[JStart];

complex_t *LPtr = &LArray[JStart];
complex_t *UPtr = &UArray[JStart];

complex_t *PPtr = &PArray[JStart];

int parameterOffset = blockIdx.x * numParameters;
complex_t *finalParamPtr = &finalParams[parameterOffset];

// Main tracking loop

double currentT = 1;
```

```
for (int i = 0; i < numSteps; i++)
{
// Take a predict step using Runge-Kutta 4/5 prediction.
predict(currentT, stepSize, xPtr, startParams, finalParamPtr, gamma,
 JPtr, LPtr, UPtr, PPtr, bPtr, xPredictPtr, xWorkspacePtr, n);
// xPredictPtr should point to the predicted point.

// Correcting xPredict using newtons method kNumNewtonCorrections times.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
 UPtr, PPtr, bPtr, xPredictPtr, n);
// xPtr should point to the corrected point.

//if (threadIdx.x == 1 && blockIdx.x == 0)
//{
//      printf("current t: %0.2f", currentT);
//      for (int k = 0; k < n; k++)
//      {
//              complex_t currentVal = xPtr[k];
//              printf("{%0.5f, %0.5f}\n", currentVal.real(),
 currentVal.imag());
//      }
//      printf("\n\n");

//}

currentT -= stepSize;
```

```
}

// Running correct twice (2 * numiterations of newton's method) to finalize
 paths.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr, UPtr,
 PPtr, bPtr, xPredictPtr, n);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr, UPtr,
 PPtr, bPtr, xPredictPtr, n);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr, UPtr,
 PPtr, bPtr, xPredictPtr, n);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr, UPtr,
 PPtr, bPtr, xPredictPtr, n);

//complex_t O = xPtr[0];
//complex_t Oc = xPtr[1];
//complex_t A = xPtr[2];
//complex_t Ac = xPtr[3];
//complex_t B = xPtr[4];
//complex_t Bc = xPtr[5];
//complex_t C = xPtr[6];
//complex_t Cc = xPtr[7];

//printf("Effective thread idx: %d, Solution: {%0.2f, %0.2f} {%0.2f, %0.2f}
 {%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f} {%0.2f, %0.2f} {
```

```
//␣␣␣␣␣␣blockIdx.x␣*␣blockDim.x␣+␣threadIdx.x,
//␣␣␣␣␣␣O.real(),␣O.imag(),␣Oc.real(),␣Oc.imag(),␣A.real(),␣A.imag(),␣Ac.real
␣Ac.imag(),␣B.real(),␣B.imag(),␣Bc.real(),␣Bc.imag(),␣C.real(),␣C.imag(),
␣␣Cc.real(),␣Cc.imag());

H_device(currentT,␣xPtr,␣startParams,␣finalParamPtr,␣bPtr,␣gamma,␣n);
complex_t␣f1␣=␣*(bPtr);
complex_t␣f2␣=␣*(bPtr␣+␣1);
complex_t␣f3␣=␣*(bPtr␣+␣2);
complex_t␣f4␣=␣*(bPtr␣+␣3);
complex_t␣f5␣=␣*(bPtr␣+␣4);
complex_t␣f6␣=␣*(bPtr␣+␣5);
complex_t␣f7␣=␣*(bPtr␣+␣6);
complex_t␣f8␣=␣*(bPtr␣+␣7);

complex_t␣residual␣=␣sqrt(f1*f1␣+␣f2*f2␣+␣f3*f3␣+␣f4*f4␣+␣f5*f5␣+␣f6*f6␣+
␣f7*f7␣+␣f8*f8);
//printf("Effective Thread index: %d, Residual: (%e, %e)\n",␣blockIdx.x␣*
blockDim.x␣+␣threadIdx.x,␣residual.real(),␣residual.imag());
if␣(residual.real()␣<␣1e-12␣&&␣residual.imag()␣<␣1e-12)
{
solFlagArray[blockIdx.x␣*␣blockDim.x␣+␣threadIdx.x]␣=␣1;
}
else␣{
solFlagArray[blockIdx.x␣*␣blockDim.x␣+␣threadIdx.x]␣=␣0;
}
```

```
}



// Old Davidenko Method using cublas.
// __device__ void getDavidenkoEquation(double t, complex_t *x,
 complex_t *startParams,
//       complex_t *finalParams, complex_t *JMatWorkspace, complex_t *b,
 complex_t gamma, cublasHandle_t handle, int n, int *pivot, int *info)
//{
//
//       complex_t **JPtr = &JMatWorkspace;
//       complex_t **bPtr = &b;
//
//       cublasStatus_t status;
//
//       // Populate JMatWorkspace
//
//       Hx(t, x, startParams, finalParams, JMatWorkspace, gamma, n);
//
//       // Populate bArray with -Ht. Note: the Ht function returns the minus
 of dH/dt.
//       Ht(t, x, startParams, finalParams, b, gamma, n);
//
//       cudoubleComplex **bPtr_cublas = reinterpret_cast<cudoubleComplex **>
(bPtr);
//       cudoubleComplex **JPtr_cublas = reinterpret_cast<cudoubleComplex **>
(JPtr);
```

```
//
//      // Computing the LU factorization of all Jacobian Matrices with
 getrfbatched()
//      if (threadIdx.x == 0)
//      {
//              printf("Calling getRF.\n");
//      }
//      status = cublasZgetrfBatched(handle, n, JPtr_cublas, n, pivot, info,
 1);
//      assert(status == CUBLAS_STATUS_SUCCESS);
//      if (threadIdx.x == 0)
//      {
//              printf("getRF Called.\n");
//      }
//
//      // Computing dx for J(dx) = -Ht using getrsbatched()
//      status = cublasZgetrsBatched(handle, CUBLAS_OP_N, n, 1, JPtr_cublas,
 n, pivot, bPtr_cublas, n, info, 1);
//      assert(status == CUBLAS_STATUS_SUCCESS);
//
//      //bArray should now hold dx.
//
//}
```

B.1.6 Kernel.cu

```
// v3 - Re-wrote LU Decomp and got everything working for initial set
```

```
of task positions.
// v4 - Added randomization.
// v5 - Dropped precision from double to single (float), added solution
stat tracking, and the writing of the solutions/tps to txt files
// so Mathematica could draw them.

#define _USE_MATH_DEFINES

#include "cuda_runtime.h"
#include "cublas_v2.h"
#include "device_launch_parameters.h"
#include "Definitions.h"
#include "FourBarAnalysis.h"
#include "PredictorCorrector.cuh"

#include <assert.h>
#include <chrono>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <math.h>
#include <random>
#include <stdio.h>
#include <vector>

struct solution_t
{
```

```
double *solutionCoordinates; // 16 floats of the form (x, y), (x, y), ...
double *taskPositions;
};



int main()
{
auto t0 = std::chrono::high_resolution_clock::now();
cudaError_t cudaStatus;
cudaStatus = cudaSetDevice(0);
assert(cudaStatus == cudaSuccess);

const int numParameters = 16;
const unsigned numLinesPerPoint = 8;
const unsigned n = numLinesPerPoint;
const unsigned numPointsAnalysis = n / 2;
const int kNumIterations = 5000;
const int kNumTaskPositions = 5;
const int kNumTPParameters = 3 * kNumTaskPositions;
const double kResolution = 0.1 * M_PI / 180.0; // Angle resolution for the
 analysis code

std::vector<complex_t> startParameters(numParameters);
std::vector<complex_t> finalParameters(numParameters);
std::vector<std::vector<complex_t>> startPoints;
std::vector<std::string> inputBuffer;
```

```
std::vector<double> taskPositions(kNumTPParameters); // task positions are
 of the form: (angle in degrees, x, y)

std::ifstream inputStream("start_parameters", std::ios::in);
if (inputStream.is_open())
{
std::cout << "Start File is open" << std::endl;
std::string line;
while (getline(inputStream, line))
{
inputBuffer.push_back(line);
}
inputStream.close();
}
else
{
std::cout << "Failed to open start parameters file." << std::endl;
exit(-1);
}

for (unsigned i = 1; i < inputBuffer.size(); i++)
{
std::istringstream inputSS(inputBuffer[i]);
std::vector<std::string> stringSplit(std::istream_iterator<std::string>
{inputSS}, std::istream_iterator<std::string>());
std::string realStr = stringSplit[0];
std::string imgStr = stringSplit[1];
```

```
std::istringstream realStrStream(realStr);
std::istringstream imgStrStream(imgStr);

double realPart, imgPart;
realStrStream >> realPart;
imgStrStream >> imgPart;

complex_t currentParam(realPart, imgPart);
startParameters[i - 1] = currentParam;
}

inputBuffer.clear();
inputStream.open("final_parameters");
if (inputStream.is_open())
{
std::cout << "Final Parameters File is open" << std::endl;
std::string line;
while (getline(inputStream, line))
{
inputBuffer.push_back(line);
}
inputStream.close();
}
else
{
std::cout << "Failed to open final parameters file." << std::endl;
```

```
exit(-1);
}

for (unsigned i = 1; i < inputBuffer.size(); i++)
{
std::istringstream inputSS(inputBuffer[i]);
std::vector<std::string> stringSplit(std::istream_iterator<std::string>
{inputSS}, std::istream_iterator<std::string>());
std::string realStr = stringSplit[0];
std::string imgStr = stringSplit[1];

std::istringstream realStrStream(realStr);
std::istringstream imgStrStream(imgStr);

double realPart, imgPart;
realStrStream >> realPart;
imgStrStream >> imgPart;

complex_t currentParam(realPart, imgPart);
finalParameters[i - 1] = currentParam;
}

inputBuffer.clear();
inputStream.open("start");
if (inputStream.is_open())
{
std::cout << "Start Points File is open" << std::endl;
```

```
std::string line;
while (getline(inputStream, line))
{
if (!line.empty())
{
inputBuffer.push_back(line);
}

}
inputStream.close();
}
else
{
std::cout << "Failed to open start points file." << std::endl;
exit(-1);
}


unsigned numStartPoints = (inputBuffer.size() - 1) / numLinesPerPoint;
for (unsigned pointIdx = 0; pointIdx < numStartPoints; pointIdx++)
{
std::vector<complex_t> currentStartPoint(numLinesPerPoint);
for (unsigned lineIdx = 0; lineIdx < numLinesPerPoint; lineIdx++)
{
// the +1 at the end there is to skip the first line of the start file
 (which just says the number of start points in the file).
std::istringstream inputSS(inputBuffer[pointIdx * numLinesPerPoint +
```

```
 lineIdx + 1]);
std::vector<std::string> stringSplit(std::istream_iterator<std::string>
{inputSS}, std::istream_iterator<std::string>());
std::string realStr = stringSplit[0];
std::string imgStr = stringSplit[1];

std::istringstream realStrStream(realStr);
std::istringstream imgStrStream(imgStr);

double realPart, imgPart;
realStrStream >> realPart;
imgStrStream >> imgPart;

complex_t currentParam(realPart, imgPart);
currentStartPoint[lineIdx] = currentParam;
}
startPoints.push_back(currentStartPoint);
}

inputBuffer.clear();
inputStream.open("task_positions");
if (inputStream.is_open())
{
std::cout << "Task positions file is open." << std::endl;
std::string line;
while (getline(inputStream, line))
{
```

```
inputBuffer.push_back(line);
}
inputStream.close();
}
else
{
std::cout << "Failed to open task positions file." << std::endl;
exit(-1);
}

for (unsigned i = 0; i < inputBuffer.size(); i++)
{
std::istringstream inputSS(inputBuffer[i]);

double currentVal;
inputSS >> currentVal;

taskPositions[i] = currentVal;
}

// Seeding the new random value of gamma on the unit circle.
std::chrono::system_clock::rep seed = std::chrono::system_clock::now().
time_since_epoch().count();
std::default_random_engine generator(seed);
std::uniform_real_distribution<double> distribution(0, 2 * M_PI);
double thetaRand = distribution(generator);
```

```
complex_t gamma(cos(thetaRand), sin(thetaRand));
//complex<float>gamma(1, 0);

std::cout << "Seeded gamma: " << gamma << std::endl;

// **********************************
//                         MEMORY ALLOCATION
// **********************************

// Transfer the start parameters, final parameters, and start points to the
  device.
const int kBytesPerStartPoint = numStartPoints * numLinesPerPoint *
sizeof(complex_t);
const int kBytesPerParameterSet = numParameters * sizeof(complex_t);

const int kStartBytesTotal = kBytesPerStartPoint * kNumIterations;
const int kParameterBytesTotal = kBytesPerParameterSet * kNumIterations *
sizeof(complex_t);

const int kTPSetBytesTotal = kNumTPParameters * kNumIterations *
sizeof(double);

complex_t *host_start = (complex_t *)malloc(kStartBytesTotal);
complex_t *host_start_parameters = (complex_t *)malloc(
kParameterBytesTotal);
complex_t *host_final_parameters = (complex_t *)malloc(
kParameterBytesTotal);
```

```
double *host_task_positions = (double *)malloc(kTPSetBytesTotal);

for (unsigned i = 0; i < startParameters.size(); i++)
{
*(host_start_parameters + i) = startParameters[i];
*(host_final_parameters + i) = finalParameters[i];
}

for (unsigned i = 0; i < startPoints.size(); i++)
{
std::vector<complex_t> currentPoint = startPoints[i];
for (unsigned j = 0; j < numLinesPerPoint; j++)
{
*(host_start + i * numLinesPerPoint + j) = currentPoint[j];
}

}

for (unsigned i = 0; i < taskPositions.size(); i++)
{
*(host_task_positions + i) = taskPositions[i];
}

//size_t freeMem = 0;
//size_t totMem = 0;
//cudaStatus = cudaMemGetInfo(&freeMem, &totMem);
//std::cout << cudaGetErrorString(cudaStatus) << std::endl;
```

```
//assert(cudaStatus == cudaSuccess);
//std::cout << "Memory (free, tot) = " << freeMem << " " << totMem <<
 std::endl;

complex_t *device_start = NULL;
cudaStatus = cudaMalloc((void **)&device_start, kStartBytesTotal);
assert(cudaStatus == cudaSuccess);

complex_t *device_start_parameters = NULL;
cudaStatus = cudaMalloc((void **)&device_start_parameters,
 kParameterBytesTotal);
assert(cudaStatus == cudaSuccess);

complex_t *device_final_parameters = NULL;
cudaStatus = cudaMalloc((void **)&device_final_parameters,
 kParameterBytesTotal);
assert(cudaStatus == cudaSuccess);

double *device_task_positions = NULL;
cudaStatus = cudaMalloc((void **)&device_task_positions,
 kTPSetBytesTotal);
assert(cudaStatus == cudaSuccess);

// Transferring start/final parameter data from host to device.

// For the start points, copy the start point values to each block.
for (int i = 0; i < kNumIterations; i++)
```

```
{
int copySize = numStartPoints * numLinesPerPoint * sizeof(complex_t);
int copyOffset = i * numStartPoints * numLinesPerPoint;
cudaStatus = cudaMemcpy(&device_start[copyOffset], host_start, copySize,
 cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
}

cudaStatus = cudaMemcpy(device_start_parameters, host_start_parameters,
 numParameters * sizeof(complex_t), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(device_final_parameters, host_final_parameters,
 numParameters * sizeof(complex_t), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
// Transfering original set of task positions to beginning of
 device_task_positions
cudaStatus = cudaMemcpy(device_task_positions, host_task_positions,
 kNumTPParameters * sizeof(double), cudaMemcpyHostToDevice);

// Calculating the workspace for needed
// For each start point, there will be an n x n jacobian matrix
unsigned batchSize = startPoints.size();

const int kBytesPerMatrix = batchSize * n * n * sizeof(complex_t);
const int kMatrixBytesTotal = kBytesPerMatrix * kNumIterations;

// next, allocate an array of pointers on the device
```

```
complex_t *device_JArray = NULL;
cudaStatus = cudaMalloc((void **)&device_JArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);

// bArray is an n x batchSize array to be filled in by Ht/H during the
 predict/correct stages
// and are to be used in cublasgetrs.
const int kBytesPerVector = batchSize * n * sizeof(complex_t);
const int kVectorBytesTotal = kBytesPerVector * kNumIterations;

complex_t *device_bArray = NULL;
cudaStatus = cudaMalloc((void **)&device_bArray, kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);

// allocating the xPredictArray and xWorkspaceArray.
complex_t *device_xPredictArray = NULL;
cudaStatus = cudaMalloc((void **)&device_xPredictArray,
 kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);

complex_t *device_xWorkspaceArray = NULL;
cudaStatus = cudaMalloc((void **)&device_xWorkspaceArray,
 kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);

complex_t *device_LArray = NULL;
cudaStatus = cudaMalloc((void **)&device_LArray, kMatrixBytesTotal);
```

```
assert(cudaStatus == cudaSuccess);

complex_t *device_UArray = NULL;
cudaStatus = cudaMalloc((void **)&device_UArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);

complex_t *device_PArray = NULL;
cudaStatus = cudaMalloc((void **)&device_PArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);

int *device_solFlag_array = NULL;
cudaStatus = cudaMalloc((void **)&device_solFlag_array, kNumIterations *
 numStartPoints * sizeof(int));
assert(cudaStatus == cudaSuccess);

int *host_solFlag_array = NULL;
host_solFlag_array = (int *)malloc(kNumIterations * numStartPoints *
 sizeof(int));

// **********************************
//                MAIN KERNEL CALLS
// **********************************
//float t = 1.0;
//H_global<<<1, numStartPoints>>> (t, device_start, device_start_parameters,
 device_final_parameters, device_bArray, gamma, n, device_solFlag_array,
  numParameters);
```

```
const double kTolX = 0.05;
const double kTolY = 0.05;
const double kTolTheta = 5;


double endGameBoundary = 1.0;
double numSteps = 100;
double stepSize = endGameBoundary / numSteps;

dim3 numBlocks = kNumIterations;
dim3 numThreadsPerBlock = numStartPoints;

double *device_LArray_floatPtr = reinterpret_cast<double*>(device_JArray);

printf("Starting Randomization. Num Iterations: %i, TolX: %0.2f, TolY: %0.2f,
 TolTheta: %0.2f\n", kNumIterations, kTolX, kTolY, kTolTheta);
// TP RANDOMIZATION KERNEL CALL. Note casting LArray as a pointer to a float,
 which is fine because it is half the size (in bytes) as the original
  complex_t array.
randomizeTaskPositions<<<numBlocks, numThreadsPerBlock>>>(
device_start_parameters, device_final_parameters, device_start,
 device_task_positions,
device_LArray_floatPtr, device_bArray, kTolX, kTolY, kTolTheta,
 numParameters, kNumTPParameters);

printf("Randomization complete. Starting path tracking...\n");
```

```
trackPaths << <numBlocks, numThreadsPerBlock >> >(
device_start, device_start_parameters, device_final_parameters, gamma,
 device_JArray,
device_LArray, device_UArray, device_PArray, device_bArray,
device_xPredictArray, device_xWorkspaceArray,
n, numSteps, stepSize, numParameters, device_solFlag_array);

cudaStatus = cudaThreadSynchronize();
assert(cudaStatus == cudaSuccess);

cudaStatus = cudaMemcpy(host_solFlag_array, device_solFlag_array,
 kNumIterations * numStartPoints * sizeof(int), cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
int numSolutionsPolynomialSystem = 0;
for (int i = 0; i < kNumIterations * numStartPoints; i++)
{
numSolutionsPolynomialSystem += host_solFlag_array[i];
}
printf("Path tracking complete. Number of unfiltered solutions: %d\n",
 numSolutionsPolynomialSystem);

filterAndTransformSolutions<<<numBlocks, numThreadsPerBlock>>>(
device_start, device_task_positions, device_solFlag_array, n,
 kNumTPParameters);

cudaStatus = cudaThreadSynchronize();
assert(cudaStatus == cudaSuccess);
```

```
printf("Filtering complete. Transferring successful solutions to CPU.\n");
cudaStatus = cudaMemcpy(host_task_positions, device_task_positions,
 kTPSetBytesTotal, cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(host_solFlag_array, device_solFlag_array,
 kNumIterations * numStartPoints * sizeof(int), cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);

complex_t *solBuf = NULL;
solBuf = (complex_t *)malloc(n * sizeof(complex_t));

std::vector<solution_t> solVec;
for (unsigned i = 0; i < kNumIterations * numStartPoints; i++)
{
if (host_solFlag_array[i] == 1)
{
// Recover block index from Effective Thread Index (ETI)
float ETI = (float)i;
float threadIndex = fmod(ETI, numStartPoints);
float blockIndexD = floor((ETI - threadIndex) / numStartPoints);
int blockIndex = (int)blockIndexD;

// CudaMemCpy deviceTaskPositions[blockIdx] back to host
solution_t currentSol;
cudaStatus = cudaMemcpy(solBuf, &device_start[i*n], n * sizeof(complex_t),
 cudaMemcpyDeviceToHost);
```

```
assert(cudaStatus == cudaSuccess);
currentSol.solutionCoordinates = (double *)malloc(n * sizeof(double));
for (unsigned j = 0; j < numPointsAnalysis; j++)
{
complex_t currentVal = solBuf[j*2];
currentSol.solutionCoordinates[j * 2] = (double)currentVal.real();
currentSol.solutionCoordinates[j * 2 + 1] = (double)currentVal.imag();
}
currentSol.taskPositions = (double *)malloc(kNumTPParameters *
 sizeof(double));
for (int j = 0; j < kNumTPParameters; j++)
{
currentSol.taskPositions[j] = (double)host_task_positions[kNumTPParameters *
 blockIndex + j];

}

solVec.push_back(currentSol);

}
}
printf("%d solutions passed filtering.\n", solVec.size());



//for (int solIdx = 0; solIdx < 2; solIdx++)
//{
//        printf("Solution %d:\n", solIdx);
```

```
//         solution_t  currSol = solVec[solIdx];
//         for (int i = 0; i < n; i++)
//         {
//                   printf("%0.5f ", currSol.solutionCoordinates[i]);
//         }
//         printf("\n\n");
//}



// ANALYSIS OF THE FILTERED SOLUTIONS
double rangeTheta = 0;
double sortedCrankAngles[5] = { 0 };
std::vector<solution_t>designsVec;
for (unsigned i = 0; i < solVec.size(); i++)
{
solution_t currentSol = solVec[i];
bool goodSol = analysisWithDefectCheck(currentSol.solutionCoordinates,
 currentSol.taskPositions, kResolution, rangeTheta, sortedCrankAngles);
if (goodSol)
{
designsVec.push_back(currentSol);
}
}

printf("%d design candidates passed analysis.\n", designsVec.size());
```

```
// Writing the solutions to a text file so they can be drawn by Mathematica
std::ofstream solFile("solutions.txt");
std::ofstream tpsFile("taskpositions.txt");
for (unsigned i = 0; i < designsVec.size(); i++)
{
solution_t currentSol = designsVec[i];
for (int j = 0; j < n; j++)
{
solFile << currentSol.solutionCoordinates[j] << " ";
}
for (int k = 0; k < kNumTPParameters; k++)
{
tpsFile << currentSol.taskPositions[k] << " ";
}
solFile << std::endl;
tpsFile << std::endl;
}
solFile.close();
tpsFile.close();

cudaFree(device_start);
cudaFree(device_start_parameters);
cudaFree(device_final_parameters);
cudaFree(device_JArray);
cudaFree(device_bArray);
cudaFree(device_xPredictArray);
cudaFree(device_xWorkspaceArray);
```

```
cudaFree(device_PArray);
cudaFree(device_LArray);
cudaFree(device_UArray);
cudaFree(device_task_positions);
cudaFree(device_solFlag_array);

free(host_start);
free(host_start_parameters);
free(host_final_parameters);
free(host_task_positions);
free(solBuf);
free(host_solFlag_array);

// cudaDeviceReset must be called before exiting in order for profiling and
// tracing tools such as Nsight and Visual Profiler to show complete traces.
cudaStatus = cudaDeviceReset();
if (cudaStatus != cudaSuccess) {
fprintf(stderr, "cudaDeviceReset failed!");
return 1;
}

auto tf = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> dt = tf - t0;
std::chrono::seconds dtSeconds = std::chrono::duration_cast<
std::chrono::seconds>(dt);
std::cout << "runtime: " << dtSeconds.count() << " seconds." << std::endl;
```

```
return 0;
}
```

B.2 Six-Bar Synthesis

This is the CUDA for for Watt 1 Six-Bar synthesis. The files are somewhat similar to the four-bar synthesis code but with some significant differences. The functions for H, Hx (or J_x, the jacobian matrix) and H_t have been moved to their external files. Most of the values originally in Definitions.h have been moved to kernel.cu. All of this was done to keep the compiler from recompiling H.cu, Hx.cu and Ht.cu every time which took hours to complete.

B.2.1 Definitions.h

```
#pragma once
#ifndef DEFINITIONS_H
#define DEFINITIONS_H

#define _USE_MATH_DEFINES

#include "thrust\complex.h"
#include <math.h>


typedef thrust::complex<double> complex_t; // Choosing whether to use single
 or double precision complex numbers
```

```
//typedef thrust::complex<double> complex_t_EG; // Choosing whether to use
 single or double precision in the endgame

#endif
```

B.2.2 H.h

```
#ifndef H_H
#define H_H

#include "Definitions.h"

#include "cuda_runtime.h"
#include "device_launch_parameters.h"

__global__ void H_global(double t, complex_t* xArray, complex_t* startParams,
 complex_t* finalParams, complex_t* bArray, complex_t gamma, int n,
  int* solFlagArray, int numParameters, int kNumStartPoints);

__device__ void H_device(double t, complex_t *x, complex_t *startParams,
 complex_t *finalParams, complex_t *b, complex_t gamma, int n);

__device__ void H_DEBUG_DEVICE(double t, complex_t* x, complex_t* startParams
 complex_t* finalParams, complex_t* bArray, complex_t gamma, int n);

#endif
```

B.2.3 H.cu

```
#include "H.h"

#define Power pow

// Returns the value of the homotopy system at point x at time t.
// effectively, evaluates H(t, x(t))
__device__ void H_device(double t, complex_t *x, complex_t *startParams,
 complex_t *finalParams, complex_t *b, complex_t gamma, int n)
{

complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start, Pc2Start,
 Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start, Tc51Start,
Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1]; P3Start = startParams[2];
 P4Start = startParams[3]; P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6];
Pc3Start = startParams[7];
 Pc4Start = startParams[8]; Pc5Start = startParams[9];
T51Start = startParams[10]; T52Start = startParams[11];
 T53Start = startParams[12]; T54Start = startParams[13];
 T55Start = startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start;
Tc53Start = 1.0 / T53Start; Tc54Start = 1.0 / T54Start;
Tc55Start = 1.0 / T55Start;
```

```
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start);
 Tc53Start = conj(T53Start); Tc54Start = conj(T54Start);
  Tc55Start = conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16];
 BStart = startParams[17]; BcStart = startParams[18];

complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final, Pc2Final,
 Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final, Tc51Final,
 Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;
P1Final = finalParams[0]; P2Final = finalParams[1]; P3Final = finalParams[2];
 P4Final = finalParams[3]; P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6];
Pc3Final = finalParams[7]; Pc4Final = finalParams[8];
Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11];
 T53Final = finalParams[12]; T54Final = finalParams[13];
  T55Final = finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final;
 Tc53Final = 1.0 / T53Final; Tc54Final = 1.0 / T54Final;
  Tc55Final = 1.0 / T55Final;
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final);
 Tc53Final = conj(T53Final); Tc54Final = conj(T54Final);
  Tc55Final = conj(T55Final);
AFinal = finalParams[15]; AcFinal = finalParams[16]; BFinal = finalParams[17]
 BcFinal = finalParams[18];
```

```
complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31, T32, Tc32, T33,
 Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4], Fc = x[5], G = x[6],
 Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13], T33 = x[14],
 Tc33 = x[15], T34 = x[16], Tc34 = x[17], T35 = x[18], Tc35 = x[19];

complex_t f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15,
 f16, f17, f18, f19, f20;

f1 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma *
 t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) +
gamma * P1Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) + (P1Final * (1 - t) +
gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) +
DD * (BcFinal * (1 - t) + BcStart * gamma * t) * T31 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * T31 - 1. * DD * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + G * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) +
DDc * (BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. *
```

```
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * Tc31 - 1. * DDc * (P1Final *
(1 - t) + gamma * P1Start * t) * Tc31 + (BcFinal * (1 - t) +
BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start *
t) * Tc31 - 1. * DDc * G * ((1 - t) * T51Final + gamma * t *
T51Start) * Tc31 + G * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) *
Tc31 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc51Final + gamma * t * Tc51Start) + Gc *
(P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) *
Tc51Final + gamma * t * Tc51Start) - 1. * DD * Gc *
T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) +
Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 *
((1 - t) * Tc51Final + gamma * t * Tc51Start);


f2 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) - 1. * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) +
(P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) + DD * (BcFinal * (1 - t) + BcStart * gamma *
t) * T32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * T32 - 1. * DD * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 + (BFinal * (1 - t) + BStart * gamma * t) *
```

(Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) + G * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) + DDc * (BFinal * (1 - t) + BStart * gamma * t) * Tc32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc32 - 1. * DDc * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 - 1. * DDc * G * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + Gc * (P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * DD * Gc * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + Gc * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start);

f3 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + (P3Final * (1 - t) + gamma * P3Start * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + DD * (BcFinal * (1 - t) + BcStart * gamma * t) * T33 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T33 - 1. * DD * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + (BFinal * (1 - t) +

BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 −
1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) + G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗
((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DDc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ (P3Final ∗
(1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗
G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + G ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) ∗ Tc33 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗ (P3Final ∗ (1 − t) +
gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗
DD ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final +
gamma ∗ t ∗ Tc53Start);

f4 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) −
1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗
P4Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) + (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗
t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗

t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t * T54Start) + DDc * (BFinal * (1 − t) + BStart * gamma * t) * Tc34 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc34 − 1. * DDc * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 − 1. * DDc * G * ((1 − t) * T54Final + gamma * t * T54Start) * Tc34 + G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) * Tc34 − 1. * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + Gc * (P4Final * (1 − t) + gamma * P4Start * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) − 1. * DD * Gc * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + Gc * (BFinal * (1 − t) + BStart * gamma * t) * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start);

f5 = DDc * G + DD * Gc − 1. * DD * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * T35 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * DD * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + (BFinal * (1 − t) +

```
BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 -
1. * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) *
((1 - t) * T55Final + gamma * t * T55Start) + DDc * (BFinal * (1 - t) +
BStart * gamma * t) * Tc35 - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. * DDc * (P5Final *
(1 - t) + gamma * P5Start * t) * Tc35 + (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * DDc *
G * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 + G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) * Tc35 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) -
1. * DD * Gc * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T35 * ((1 - t) * Tc55Final +
gamma * t * Tc55Start);

f6 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD *
DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC *
DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 -
t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) -
1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G *
Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal *
```

(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗

```
Pc1Start * t) + CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * DDc * G * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start *
t) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final *
(1 - t) + gamma * Pc1Start * t) + G * Gc * (BFinal * (1 - t) + BStart *
gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CCc * DD *
(P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) - 1. * CC * DDc * (P1Final * (1 - t) + gamma * P1Start * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) + DD * DDc * (P1Final *
(1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start *
t) + CCc * G * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final *
(1 - t) + gamma * Pc1Start * t) + CC * Gc * (P1Final * (1 - t) + gamma *
P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * G * Gc *
(P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) - 1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma *
t) * T31 + DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 +
CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 - 1. * DD *
G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 + CC * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T31 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma *
t) * (BFinal * (1 - t) + BStart * gamma * t) * T31 - 1. * CC * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T31 + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * T31 + CC * DD * DDc *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * DD * DDc * G *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * CC * DD * Gc *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 + DD * G * Gc *
```

(Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 − 1. * CC * DDc *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma *
Pc1Start * t) * T31 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + CC * Gc * (BFinal *
(1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) * T31 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + CC * DDc * G *
(AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T51Final +
gamma * t * T51Start) − 1. * DD * DDc * G * (AcFinal * (1 − t) +
AcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) −
1. * CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) *
((1 − t) * T51Final + gamma * t * T51Start) + DD * G * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) * ((1 − t) * T51Final + gamma * t *
T51Start) + CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma *
t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * CCc *
Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) *
T51Final + gamma * t * T51Start) − 1. * DD * G * Gc * (BcFinal *
(1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t *
T51Start) + Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma *
t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * CCc * DD * G *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final +
gamma * t * T51Start) − 1. * CC * DDc * G * (Pc1Final * (1 − t) +
gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t *
T51Start) + DD * DDc * G * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) * ((1 − t) * T51Final + gamma * t * T51Start) + CCc * Power(G, 2) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final +
gamma * t * T51Start) + CC * G * Gc * (Pc1Final * (1 − t) + gamma *

```
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. *
Power(G, 2) * Gc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) *
((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc * DD * DDc *
AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * DDc * G *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 + DD * DDc * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 - 1. * DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * DD *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc31 - 1. * CCc * G * (AFinal * (1 - t) +
AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) *
Tc31 - 1. * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc31 + G * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) * Tc31 + CCc * DD * DDc * (P1Final * (1 - t) + gamma * P1Start *
t) * Tc31 - 1. * CCc * DDc * G * (P1Final * (1 - t) + gamma *
P1Start * t) * Tc31 - 1. * DD * DDc * Gc * (P1Final * (1 - t) +
gamma * P1Start * t) * Tc31 + DDc * G * Gc * (P1Final * (1 - t) +
gamma * P1Start * t) * Tc31 - 1. * CCc * DD * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) *
Tc31 + CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final *
(1 - t) + gamma * P1Start * t) * Tc31 + DD * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 -
1. * G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final *
(1 - t) + gamma * P1Start * t) * Tc31 + CCc * DD * DDc * G * ((1 - t) *
T51Final + gamma * t * T51Start) * Tc31 - 1. * CCc * DDc * Power(G, 2) *
((1 - t) * T51Final + gamma * t * T51Start) * Tc31 - 1. * DD * DDc * G *
Gc * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 + DDc *
```

Power(G, 2) * Gc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 −
1. * CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + CCc * Power(G, 2) *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final +
gamma * t * T51Start) * Tc31 + DD * G * Gc * (BcFinal * (1 − t) +
BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) *
Tc31 − 1. * Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma *
t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + CCc * DD *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final +
gamma * t * Tc51Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) +
AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) −
1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * G * Gc *
(BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma *
t * Tc51Start) − 1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + G *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * CCc * DD * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) − 1. * CC * DDc * Gc * (P1Final * (1 − t) + gamma *
P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DD *
DDc * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) + CCc * G * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *

Tc51Start) + CC * Power(Gc, 2) * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * G * Power(Gc, 2) * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + CC * DD * DDc * Gc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * DD * DDc * G * Gc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * CC * DD * Power(Gc, 2) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + DD * G * Power(Gc, 2) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * G * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start);

f7 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) +

CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗
CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) +
DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗
G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗
P2Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗
CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + DD ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) +
gamma ∗ P2Start ∗ t) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗
CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ DD ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) +
gamma ∗ P2Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + CCc ∗

DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) − 1. * DD * DDc * (AFinal *
(1 − t) + AStart * gamma * t) * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) − 1. * CCc * G * (AFinal * (1 − t) + AStart *
gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DDc *
G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) + CC * DDc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) −
1. * DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) − 1. * CC * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) +
G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) − 1. * CCc * DD * (P2Final * (1 − t) +
gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) −
1. * CC * DDc * (P2Final * (1 − t) + gamma * P2Start * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) + DD * DDc * (P2Final *
(1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) + CCc * G * (P2Final * (1 − t) + gamma * P2Start * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) + CC * Gc * (P2Final *
(1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) − 1. * G * Gc * (P2Final * (1 − t) + gamma * P2Start *
t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * DD *
DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + DD * DDc *
G * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + CC * DD *
Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 − 1. * DD *
G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + CC *
DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *

(1 − t) + BStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T32 − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 + G ∗
Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T32 + CC ∗ DD ∗ DDc ∗ (Pc2Final ∗
(1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ DD ∗ DDc ∗ G ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ CC ∗
DD ∗ Gc ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 +
DD ∗ G ∗ Gc ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 −
1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + DDc ∗ G ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) +
gamma ∗ Pc2Start ∗ t) ∗ T32 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗
t) ∗ T32 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + CC ∗
DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ DD ∗ DDc ∗
G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CC ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T52Final + gamma ∗ t ∗ T52Start) + DD ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) −
1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗

((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ DD ∗ G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) + Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) −
1. ∗ CCc ∗ DD ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗
((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CC ∗ DDc ∗ G ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) + DD ∗ DDc ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗
Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + CCc ∗
Power(G, 2) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗
((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + CC ∗ G ∗ Gc ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc2Final ∗ (1 − t) +
gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) −
1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ G ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc32 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc32 + CCc ∗ DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗

Tc32 + DDc * G * Gc * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 −
1. * CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final *
(1 − t) + gamma * P2Start * t) * Tc32 + CCc * G * (BcFinal * (1 − t) +
BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) *
Tc32 + DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
(P2Final * (1 − t) + gamma * P2Start * t) * Tc32 − 1. * G * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) +
gamma * P2Start * t) * Tc32 + CCc * DD * DDc * G * ((1 − t) * T52Final +
gamma * t * T52Start) * Tc32 − 1. * CCc * DDc * Power(G, 2) * ((1 − t) *
T52Final + gamma * t * T52Start) * Tc32 − 1. * DD * DDc * G * Gc *
((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + DDc * Power(G, 2) *
Gc * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 1. * CCc *
DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) *
T52Final + gamma * t * T52Start) * Tc32 + CCc * Power(G, 2) * (BcFinal *
(1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t *
T52Start) * Tc32 + DD * G * Gc * (BcFinal * (1 − t) + BcStart * gamma *
t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 1. *
Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + CCc * DD * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc52Final + gamma *
t * Tc52Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * CCc *
G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc52Final +
gamma * t * Tc52Start) + DDc * G * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + CC * DDc *
Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final +
gamma * t * Tc52Start) − 1. * DDc * G * Gc * (BFinal * (1 − t) +

```
BStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + G * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * CCc * DD * Gc * (P2Final * (1 - t) +
gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * CC * DDc * Gc * (P2Final * (1 - t) + gamma * P2Start * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + DD * DDc * Gc *
(P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + CCc * G * Gc * (P2Final * (1 - t) + gamma *
P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC *
Power(Gc, 2) * (P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) *
Tc52Final + gamma * t * Tc52Start) - 1. * G * Power(Gc, 2) *
(P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + CC * DD * DDc * Gc * T32 * ((1 - t) *
Tc52Final + gamma * t * Tc52Start) - 1. * DD * DDc * G * Gc * T32 *
((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * CC * DD *
Power(Gc, 2) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) +
DD * G * Power(Gc, 2) * T32 * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) - 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart *
gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) +
DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T32 *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) *
Tc52Final + gamma * t * Tc52Start) - 1. * G * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) *
Tc52Final + gamma * t * Tc52Start);
```

```
f8 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC *
DD * DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) *
Gc + CC * DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD *
DDc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma *
t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) - 1. * CC * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) + DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P3Final * (1 - t) + gamma * P3Start * t) - 1. * DD * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start *
t) - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P3Final * (1 - t) + gamma * P3Start * t) + DD * Gc * (AcFinal *
```

(1 − t) + AcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ CCc ∗ G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) +
gamma ∗ P3Start ∗ t) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) +
gamma ∗ P3Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗
DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc3Final ∗
(1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗
Pc3Start ∗ t) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗
t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + CC ∗ DDc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) −
1. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗
(1 − t) + gamma ∗ Pc3Start ∗ t) + G ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) −
1. ∗ CCc ∗ DD ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ CC ∗ DDc ∗
(P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) + DD ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + CCc ∗
G ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) + CC ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗

P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗
G ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗
(1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + DD ∗ DDc ∗ G ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DD ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 −
1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ T33 + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DD ∗ DDc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ DD ∗ DDc ∗ G ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ CC ∗ DD ∗ Gc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + DD ∗ G ∗ Gc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ CC ∗ DDc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗
Pc3Start ∗ t) ∗ T33 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ Gc ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗
t) ∗ T33 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ DDc ∗ G ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗

((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗
Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) +
Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗
CC ∗ DDc ∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) + DD ∗ DDc ∗ G ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + CCc ∗
Power(G, 2) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) + CC ∗ G ∗ Gc ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗
Power(G, 2) ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ Tc33 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc33 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 +
CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DD ∗
Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ Tc33 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DD ∗ DDc ∗ (P3Final

(1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P3Final ∗
(1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P3Final ∗
(1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + DDc ∗ G ∗ Gc ∗ (P3Final ∗ (1 − t) +
gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + CCc ∗ G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ Tc33 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗
Tc33 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗
Tc33 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) ∗ Tc33 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗
t ∗ T53Start) ∗ Tc33 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗
Tc33 + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + DD ∗ G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) ∗ Tc33 + CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗
t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗
t ∗ Tc53Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗
t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DDc ∗ G ∗ Gc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗
t ∗ Tc53Start) + CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

```
((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DDc * G * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) - 1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + G *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * CCc * DD * Gc * (P3Final *
(1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 1. * CC * DDc * Gc * (P3Final * (1 - t) + gamma * P3Start *
t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + DD * DDc * Gc *
(P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) + CCc * G * Gc * (P3Final * (1 - t) + gamma * P3Start * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc, 2) *
(P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) - 1. * G * Power(Gc, 2) * (P3Final * (1 - t) + gamma *
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * DD *
DDc * Gc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DD *
DDc * G * Gc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. *
CC * DD * Power(Gc, 2) * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) + DD * G * Power(Gc, 2) * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start) - 1. * CC * DDc * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) + DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start) - 1. * G * Power(Gc, 2) * (BFinal * (1 - t) +
BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start);
```

```
f9 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD *
DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC *
DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma *
t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) +
CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc *
DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart * gamma *
t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) - 1. * CC * DDc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + DDc *
G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart * gamma *
t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final *
(1 - t) + gamma * P4Start * t) - 1. * DD * DDc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. *
CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final *
(1 - t) + gamma * P4Start * t) + DD * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) +
```

```
CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final *
(1 - t) + gamma * P4Start * t) - 1. * CCc * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. *
DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final *
(1 - t) + gamma * P4Start * t) + G * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) +
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) - 1. * DD * DDc * (AFinal *
(1 - t) + AStart * gamma * t) * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) - 1. * CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + DDc *
G * (AFinal * (1 - t) + AStart * gamma * t) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) + CC * DDc * (BFinal * (1 - t) + BStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. *
DDc * G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) - 1. * CC * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) +
G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) - 1. * CCc * DD * (P4Final *
(1 - t) + gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) - 1. * CC * DDc * (P4Final * (1 - t) + gamma *
P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) +
DD * DDc * (P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + CCc * G * (P4Final * (1 - t) +
gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start *
t) + CC * Gc * (P4Final * (1 - t) + gamma * P4Start * t) *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. * G * Gc *
```

(P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 − 1. * DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + CC * DD * DDc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * DD * DDc * G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DD * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + DD * G * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. *

```
CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T54Final + gamma * t * T54Start) + DD * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) *
T54Final + gamma * t * T54Start) + CCc * DD * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma *
t * T54Start) - 1. * CCc * Power(G, 2) * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) - 1. * DD * G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) +
Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T54Final + gamma * t * T54Start) - 1. * CCc * DD *
G * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) *
T54Final + gamma * t * T54Start) - 1. * CC * DDc * G *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) *
T54Final + gamma * t * T54Start) + DD * DDc * G * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) * ((1 - t) * T54Final + gamma *
t * T54Start) + CCc * Power(G, 2) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) * ((1 - t) * T54Final + gamma * t *
T54Start) + CC * G * Gc * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * ((1 - t) * T54Final + gamma * t * T54Start) -
1. * Power(G, 2) * Gc * (Pc4Final * (1 - t) + gamma * Pc4Start *
t) * ((1 - t) * T54Final + gamma * t * T54Start) - 1. * CCc *
DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * Tc34 +
CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) *
Tc34 + DD * DDc * Gc * (AFinal * (1 - t) + AStart * gamma *
t) * Tc34 - 1. * DDc * G * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * Tc34 + CCc * DD * (AFinal * (1 - t) + AStart *
```

gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 −
1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 − 1. * DD *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *
(1 − t) + BcStart * gamma * t) * Tc34 + G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart *
gamma * t) * Tc34 + CCc * DD * DDc * (P4Final * (1 − t) +
gamma * P4Start * t) * Tc34 − 1. * CCc * DDc * G * (P4Final *
(1 − t) + gamma * P4Start * t) * Tc34 − 1. * DD * DDc * Gc *
(P4Final * (1 − t) + gamma * P4Start * t) * Tc34 + DDc * G *
Gc * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 − 1. *
CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) *
(P4Final * (1 − t) + gamma * P4Start * t) * Tc34 + CCc * G *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P4Final *
(1 − t) + gamma * P4Start * t) * Tc34 + DD * Gc * (BcFinal *
(1 − t) + BcStart * gamma * t) * (P4Final * (1 − t) +
gamma * P4Start * t) * Tc34 − 1. * G * Gc * (BcFinal *
(1 − t) + BcStart * gamma * t) * (P4Final * (1 − t) + gamma *
P4Start * t) * Tc34 + CCc * DD * DDc * G * ((1 − t) *
T54Final + gamma * t * T54Start) * Tc34 − 1. * CCc * DDc *
Power(G, 2) * ((1 − t) * T54Final + gamma * t * T54Start) *
Tc34 − 1. * DD * DDc * G * Gc * ((1 − t) * T54Final + gamma *
t * T54Start) * Tc34 + DDc * Power(G, 2) * Gc * ((1 − t) *
T54Final + gamma * t * T54Start) * Tc34 − 1. * CCc * DD * G *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) *
T54Final + gamma * t * T54Start) * Tc34 + CCc * Power(G, 2) *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) *

T54Final + gamma * t * T54Start) * Tc34 + DD * G * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) * Tc34 - 1. * Power(G, 2) * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) * Tc34 + CCc * DD * Gc * (AFinal * (1 - t) +
AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) - 1. * DD * DDc * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) -
1. * CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final +
gamma * t * Tc54Start) + CC * DDc * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) - 1. * DDc * G * Gc * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) -
1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma *
t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + G *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CCc *
DD * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) - 1. * CC * DDc * Gc *
(P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) + DD * DDc * Gc * (P4Final *
(1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma *
t * Tc54Start) + CCc * G * Gc * (P4Final * (1 - t) + gamma *
P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) +
CC * Power(Gc, 2) * (P4Final * (1 - t) + gamma * P4Start * t) *

((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ G ∗
Power(Gc, 2) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ DD ∗ DDc ∗
Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗
DD ∗ DDc ∗ G ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗
Tc54Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T34 ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) + DD ∗ G ∗ Power(Gc, 2) ∗
T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗
DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ G ∗ Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗
t ∗ Tc54Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

f10 = −1. ∗ CCc ∗ DD ∗ DDc ∗ G + CCc ∗ DDc ∗ Power(G, 2) − 1. ∗ CC ∗ DD ∗
DDc ∗ Gc + 2. ∗ DD ∗ DDc ∗ G ∗ Gc − 1. ∗ DDc ∗ Power(G, 2) ∗ Gc + CC ∗
DD ∗ Power(Gc, 2) − 1. ∗ DD ∗ G ∗ Power(Gc, 2) + CC ∗ DD ∗ DDc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ G ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) + CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗
t) + CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗
DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ Gc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗

G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) −
1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) −
1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ DDc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + CCc ∗
DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) +
gamma ∗ P5Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗
t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗
(1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + DDc ∗
G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) +
gamma ∗ Pc5Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗

gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * DDc *
G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) +
gamma * Pc5Start * t) − 1. * CC * Gc * (BFinal * (1 − t) + BStart *
gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + G * Gc *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma *
Pc5Start * t) − 1. * CCc * DD * (P5Final * (1 − t) + gamma * P5Start *
t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CC * DDc *
(P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma *
Pc5Start * t) + DD * DDc * (P5Final * (1 − t) + gamma * P5Start * t) *
(Pc5Final * (1 − t) + gamma * Pc5Start * t) + CCc * G * (P5Final *
(1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start *
t) + CC * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final *
(1 − t) + gamma * Pc5Start * t) − 1. * G * Gc * (P5Final * (1 − t) +
gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) −
1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 +
DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC *
DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 − 1. * DD *
G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * DDc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart *
gamma * t) * T35 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma *
t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * CC * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) +
BStart * gamma * t) * T35 + G * Gc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 + CC *
DD * DDc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DD *
DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC *
DD * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + DD * G *

Gc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ DDc ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CCc ∗ Power(G, 2) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CC ∗ G ∗ Gc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc5Final ∗ (1 − t) +

gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) −
1. * CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 +
CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + DD * DDc *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 − 1. * DDc * G * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * Tc35 + CCc * DD * (AFinal *
(1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma *
t) * Tc35 − 1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc35 − 1. * DD * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart *
gamma * t) * Tc35 + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc35 + CCc * DD * DDc *
(P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * CCc * DDc * G *
(P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * DD * DDc * Gc *
(P5Final * (1 − t) + gamma * P5Start * t) * Tc35 + DDc * G * Gc *
(P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * CCc * DD *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma *
P5Start * t) * Tc35 + CCc * G * (BcFinal * (1 − t) + BcStart * gamma *
t) * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 + DD * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma *
P5Start * t) * Tc35 − 1. * G * Gc * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 + CCc *
DD * DDc * G * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 −
1. * CCc * DDc * Power(G, 2) * ((1 − t) * T55Final + gamma * t *
T55Start) * Tc35 − 1. * DD * DDc * G * Gc * ((1 − t) * T55Final +
gamma * t * T55Start) * Tc35 + DDc * Power(G, 2) * Gc * ((1 − t) *
T55Final + gamma * t * T55Start) * Tc35 − 1. * CCc * DD * G * (BcFinal *
(1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma * t *

T55Start) * Tc35 + CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 + DD *
G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T55Final + gamma * t * T55Start) * Tc35 - 1. * Power(G, 2) * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma *
t * T55Start) * Tc35 + CCc * DD * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * DD *
DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) - 1. * CCc * G * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) + DDc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc55Final + gamma * t * Tc55Start) + CC * DDc * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc55Final + gamma *
t * Tc55Start) - 1. * DDc * G * Gc * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * CC *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + G * Power(Gc, 2) * (BFinal *
(1 - t) + BStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) - 1. * CCc * DD * Gc * (P5Final * (1 - t) + gamma * P5Start *
t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * CC * DDc *
Gc * (P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) * Tc55Final +
gamma * t * Tc55Start) + DD * DDc * Gc * (P5Final * (1 - t) + gamma *
P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + CCc *
G * Gc * (P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + CC * Power(Gc, 2) * (P5Final *
(1 - t) + gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) - 1. * G * Power(Gc, 2) * (P5Final * (1 - t) + gamma *

P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ DD ∗ DDc ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DD ∗ G ∗ Power(Gc, 2) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);

f11 = Fc ∗ H + F ∗ Hc − 1. ∗ F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ F ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗

```
T51Final + gamma * t * T51Start) + H * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) + Fc *
(BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) * Tc31 - 1. * Fc * (P1Final * (1 - t) + gamma * P1Start * t) *
Tc31 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) +
gamma * P1Start * t) * Tc31 - 1. * Fc * H * ((1 - t) * T51Final +
gamma * t * T51Start) * Tc31 + H * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 -
1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc51Final + gamma * t * Tc51Start) + Hc * (P1Final * (1 - t) + gamma *
P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * F *
Hc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + Hc *
(BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final +
gamma * t * Tc51Start);

f12 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) + 2. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) + (P2Final * (1 - t) + gamma * P2Start * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) + F * (BcFinal *
(1 - t) + BcStart * gamma * t) * T32 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
T32 - 1. * F * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 +
```

(BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) + H * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) + Fc * (BFinal * (1 - t) + BStart * gamma * t) * Tc32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc32 - 1. * Fc * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 - 1. * Fc * H * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + Hc * (P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * F * Hc * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start);

f13 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + (P3Final * (1 - t) + gamma * P3Start * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + F * (BcFinal * (1 - t) + BcStart * gamma * t) * T33 - 1. * (BcFinal * (1 - t) +

```
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
T33 - 1. * F * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 +
(BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * T33 - 1. * H * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) + H * (Pc3Final * (1 - t) + gamma * Pc3Start * t) *
((1 - t) * T53Final + gamma * t * T53Start) + Fc * (BFinal *
(1 - t) + BStart * gamma * t) * Tc33 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
Tc33 - 1. * Fc * (P3Final * (1 - t) + gamma * P3Start * t) *
Tc33 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * Fc * H * ((1 - t) *
T53Final + gamma * t * T53Start) * Tc33 + H * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma *
t * T53Start) * Tc33 - 1. * Hc * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) +
Hc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * F * Hc * T33 *
((1 - t) * Tc53Final + gamma * t * Tc53Start) + Hc *
(BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) *
Tc53Final + gamma * t * Tc53Start);

f14 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start *
```

```
t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + (P4Final * (1 - t) + gamma *
P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + F *
(BcFinal * (1 - t) + BcStart * gamma * t) * T34 - 1. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T34 - 1. * F * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * T34 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 - 1. *
H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T54Final + gamma * t * T54Start) + H * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) * ((1 - t) * T54Final + gamma * t *
T54Start) + Fc * (BFinal * (1 - t) + BStart * gamma * t) *
Tc34 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * Tc34 - 1. * Fc *
(P4Final * (1 - t) + gamma * P4Start * t) * Tc34 + (BcFinal *
(1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma *
P4Start * t) * Tc34 - 1. * Fc * H * ((1 - t) * T54Final +
gamma * t * T54Start) * Tc34 + H * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) * Tc34 - 1. * Hc * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) +
Hc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) - 1. * F * Hc * T34 *
((1 - t) * Tc54Final + gamma * t * Tc54Start) + Hc *
(BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) *
Tc54Final + gamma * t * Tc54Start);
```

f15 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start *
t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final *
(1 - t) + gamma * Pc5Start * t) + (P5Final * (1 - t) + gamma *
P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + F *
(BcFinal * (1 - t) + BcStart * gamma * t) * T35 - 1. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T35 - 1. * F * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * T35 + (BFinal * (1 - t) + BStart * gamma * t) *
(Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 - 1. * H *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + H * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) +
Fc * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * Tc35 - 1. * Fc * (P5Final * (1 - t) +
gamma * P5Start * t) * Tc35 + (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 -
1. * Fc * H * ((1 - t) * T55Final + gamma * t * T55Start) *
Tc35 + H * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T55Final + gamma * t * T55Start) * Tc35 - 1. *
Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + Hc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t *

```
Tc55Start) - 1. * F * Hc * T35 * ((1 - t) * Tc55Final + gamma *
t * Tc55Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) *
T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start);

f16 = -1. + T31 * Tc31;

f17 = -1. + T32 * Tc32;

f18 = -1. + T33 * Tc33;

f19 = -1. + T34 * Tc34;

f20 = -1. + T35 * Tc35;




//complex_t residual = sqrt(
//        f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 + f6*f6 + f7*f7 +
 f8*f8 + f9*f9 + f10*f10 +
//        f11*f11 + f12*f12 + f13*f13 + f14*f14 + f15*f15 + f16*f16 +
 f17*f17 + f18*f18 +
//        f19*f19 + f20*f20
//        );
//printf("Residual: (%e, %e)\n", residual.real(),
 residual.imag());
b[0] = f1;
```

```
b[1] = f2;
b[2] = f3;
b[3] = f4;
b[4] = f5;
b[5] = f6;
b[6] = f7;
b[7] = f8;
b[8] = f9;
b[9] = f10;
b[10] = f11;
b[11] = f12;
b[12] = f13;
b[13] = f14;
b[14] = f15;
b[15] = f16;
b[16] = f17;
b[17] = f18;
b[18] = f19;
b[19] = f20;

}

// prints out the value of H at time t for all start points.
 Does not populate bArray. Intended for use in debugging.
__device__ void H_DEBUG_DEVICE(double t, complex_t *x,
 complex_t *startParams, complex_t *finalParams, complex_t *bArray,
  complex_t gamma, int n)
```

```
{

complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start, Pc2Start,
Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start, Tc51Start,
Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1]; P3Start = startParams[2];
P4Start = startParams[3]; P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6];
Pc3Start = startParams[7];
Pc4Start = startParams[8]; Pc5Start = startParams[9];
T51Start = startParams[10]; T52Start = startParams[11];
T53Start = startParams[12]; T54Start = startParams[13];
T55Start = startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start;
Tc53Start = 1.0 / T53Start; Tc54Start = 1.0 / T54Start;
Tc55Start = 1.0 / T55Start;
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start);
Tc53Start = conj(T53Start); Tc54Start = conj(T54Start);
Tc55Start = conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16];
BStart = startParams[17]; BcStart = startParams[18];

complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final, Pc2Final,
Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final, Tc51Final,
```

```
Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;
P1Final = finalParams[0]; P2Final = finalParams[1]; P3Final = finalParams[2];
P4Final = finalParams[3]; P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6];
Pc3Final = finalParams[7]; Pc4Final = finalParams[8];
Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11];
T53Final = finalParams[12]; T54Final = finalParams[13];
T55Final = finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final;
Tc53Final = 1.0 / T53Final; Tc54Final = 1.0 / T54Final;
Tc55Final = 1.0 / T55Final;
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final);
Tc53Final = conj(T53Final); Tc54Final = conj(T54Final);
Tc55Final = conj(T55Final);
AFinal = finalParams[15]; AcFinal = finalParams[16]; BFinal = finalParams[17]
BcFinal = finalParams[18];

complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31, T32, Tc32, T33,
Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4], Fc = x[5], G = x[6],
Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13], T33 = x[14],
Tc33 = x[15], T34 = x[16], Tc34 = x[17], T35 = x[18], Tc35 = x[19];

complex_t f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15,
```

```
f16, f17, f18, f19, f20;

f1 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) +
gamma * P1Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) + (P1Final * (1 - t) +
gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) +
DD * (BcFinal * (1 - t) + BcStart * gamma * t) * T31 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * T31 - 1. * DD * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + G * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) +
DDc * (BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * Tc31 - 1. * DDc * (P1Final *
(1 - t) + gamma * P1Start * t) * Tc31 + (BcFinal * (1 - t) +
BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start *
t) * Tc31 - 1. * DDc * G * ((1 - t) * T51Final + gamma * t *
T51Start) * Tc31 + G * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) *
Tc31 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc51Final + gamma * t * Tc51Start) + Gc *
```

```
(P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) *
Tc51Final + gamma * t * Tc51Start) - 1. * DD * Gc *
T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) +
Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 *
((1 - t) * Tc51Final + gamma * t * Tc51Start);


f2 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) - 1. * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) +
(P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) + DD * (BcFinal * (1 - t) + BcStart * gamma *
t) * T32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * T32 - 1. * DD * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 + (BFinal * (1 - t) + BStart * gamma * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) + G * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) *
T52Final + gamma * t * T52Start) + DDc * (BFinal * (1 - t) + BStart *
gamma * t) * Tc32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * Tc32 - 1. * DDc * (P2Final *
(1 - t) + gamma * P2Start * t) * Tc32 + (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 - 1. * DDc *
G * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + G * (BcFinal *
```

(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DD ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

f3 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + G ∗ (BcFinal ∗

```
(1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) + Gc * (P3Final * (1 - t) +
gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. *
DD * Gc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start);

f4 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma *
P4Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + (P4Final * (1 - t) + gamma * P4Start *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + DD * (BcFinal * (1 - t) +
BcStart * gamma * t) * T34 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * T34 - 1. * DD * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) * T34 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 - 1. * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start)
G * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) * T54Final +
gamma * t * T54Start) + DDc * (BFinal * (1 - t) + BStart * gamma * t) *
Tc34 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * Tc34 - 1. * DDc * (P4Final * (1 - t) + gamma *
P4Start * t) * Tc34 + (BcFinal * (1 - t) + BcStart * gamma * t) *
(P4Final * (1 - t) + gamma * P4Start * t) * Tc34 - 1. * DDc * G *
((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + G * (BcFinal *
```

```
(1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t *
T54Start) * Tc34 − 1. * Gc * (BFinal * (1 − t) + BStart * gamma *
t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + Gc * (P4Final *
(1 − t) + gamma * P4Start * t) * ((1 − t) * Tc54Final + gamma * t *
Tc54Start) − 1. * DD * Gc * T34 * ((1 − t) * Tc54Final + gamma * t *
Tc54Start) + Gc * (BFinal * (1 − t) + BStart * gamma * t) * T34 * ((1 − t) *
Tc54Final + gamma * t * Tc54Start);

f5 = DDc * G + DD * Gc − 1. * DD * (BcFinal * (1 − t) + BcStart * gamma *
t) − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal *
(1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) −
1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) +
gamma * P5Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc5Final * (1 − t) + gamma * Pc5Start * t) + (P5Final * (1 − t) + gamma *
P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DD *
(BcFinal * (1 − t) + BcStart * gamma * t) * T35 − 1. * (BcFinal * (1 − t) +
BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. *
DD * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + (BFinal * (1 − t) +
BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 −
1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final +
gamma * t * T55Start) + G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) *
((1 − t) * T55Final + gamma * t * T55Start) + DDc * (BFinal * (1 − t) +
BStart * gamma * t) * Tc35 − 1. * (BcFinal * (1 − t) + BcStart * gamma *
t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc35 − 1. * DDc * (P5Final *
(1 − t) + gamma * P5Start * t) * Tc35 + (BcFinal * (1 − t) + BcStart *
gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * DDc *
G * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 + G * (BcFinal *
```

```
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) * Tc35 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) -
1. * DD * Gc * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T35 * ((1 - t) * Tc55Final +
gamma * t * Tc55Start);

f6 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD *
DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC *
DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 -
t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) -
1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G *
Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) +
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) - 1. * G * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) -
1. * CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) + DDc * G * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
```

1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) + CC * DDc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * DD * DDc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (P1Final * (1 − t) + gamma *
P1Start * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P1Final * (1 − t) + gamma * P1Start * t) + DD * Gc * (AcFinal * (1 − t) +
AcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + CCc *
DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) +
gamma * P1Start * t) − 1. * CCc * G * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * DD * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma *
P1Start * t) + G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
(P1Final * (1 − t) + gamma * P1Start * t) + CCc * DD * (AFinal * (1 − t) +
AStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. *
DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) +
gamma * Pc1Start * t) − 1. * CCc * G * (AFinal * (1 − t) + AStart *
gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + DDc * G *
(AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) + gamma *
Pc1Start * t) + CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * DDc * G * (BFinal *
(1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) − 1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final *
(1 − t) + gamma * Pc1Start * t) + G * Gc * (BFinal * (1 − t) + BStart *
gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * CCc * DD *
(P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma *
Pc1Start * t) − 1. * CC * DDc * (P1Final * (1 − t) + gamma * P1Start * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) + DD * DDc * (P1Final *

(1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + CCc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + CC ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ G ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DD ∗ DDc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ DD ∗ DDc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ DD ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DD ∗ G ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +

```
AcStart * gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) -
1. * CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T51Final + gamma * t * T51Start) + DD * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + CCc * DD * G * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc *
Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T51Final + gamma * t * T51Start) - 1. * DD * G * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc * DD * G *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * ((1 - t) * T51Final +
gamma * t * T51Start) - 1. * CC * DDc * G * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + DD * DDc * G * (Pc1Final * (1 - t) + gamma * Pc1Start *
t) * ((1 - t) * T51Final + gamma * t * T51Start) + CCc * Power(G, 2) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + CC * G * Gc * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. *
Power(G, 2) * Gc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) *
((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc * DD * DDc *
AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * DDc * G *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 + DD * DDc * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 - 1. * DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * DD *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc31 - 1. * CCc * G * (AFinal * (1 - t) +
```

AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
Tc31 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc31 + G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc31 + CCc ∗ DD ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗
t) ∗ Tc31 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗
P1Start ∗ t) ∗ Tc31 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P1Final ∗ (1 − t) +
gamma ∗ P1Start ∗ t) ∗ Tc31 + DDc ∗ G ∗ Gc ∗ (P1Final ∗ (1 − t) +
gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗
Tc31 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗
(1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 −
1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗
(1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗
T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗
((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ DD ∗ DDc ∗ G ∗
Gc ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + DDc ∗
Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 −
1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + CCc ∗ Power(G, 2) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final +
gamma ∗ t ∗ T51Start) ∗ Tc31 + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗
Tc31 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + CCc ∗ DD ∗
Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final +

gamma * t * Tc51Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) +
AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) −
1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * G * Gc *
(BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma *
t * Tc51Start) − 1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + G *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * CCc * DD * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) − 1. * CC * DDc * Gc * (P1Final * (1 − t) + gamma *
P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DD *
DDc * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) + CCc * G * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + CC * Power(Gc, 2) * (P1Final * (1 − t) + gamma * P1Start *
t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * G *
Power(Gc, 2) * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) + CC * DD * DDc * Gc * T31 *
((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DD * DDc * G *
Gc * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * CC *
DD * Power(Gc, 2) * T31 * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + DD * G * Power(Gc, 2) * T31 * ((1 − t) * Tc51Final +
gamma * t * Tc51Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) +

BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

f7 = −1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) − 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc − 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) − 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + DDc *

```
G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) - 1. * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) + CC * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P2Final * (1 - t) + gamma *
P2Start * t) - 1. * DD * DDc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. *
CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) + DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) + CCc * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. *
CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) - 1. * DD * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) + G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) + CCc *
DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) - 1. * DD * DDc * (AFinal *
(1 - t) + AStart * gamma * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) - 1. * CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + DDc *
G * (AFinal * (1 - t) + AStart * gamma * t) * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) + CC * DDc * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) -
1. * DDc * G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final *
```

```
(1 - t) + gamma * Pc2Start * t) - 1. * CC * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) +
G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) - 1. * CCc * DD * (P2Final * (1 - t) +
gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) -
1. * CC * DDc * (P2Final * (1 - t) + gamma * P2Start * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) + DD * DDc * (P2Final *
(1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) + CCc * G * (P2Final * (1 - t) + gamma * P2Start * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) + CC * Gc * (P2Final *
(1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) - 1. * G * Gc * (P2Final * (1 - t) + gamma * P2Start *
t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * CC * DD *
DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + DD * DDc *
G * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + CC * DD *
Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 - 1. * DD *
G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + CC *
DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * T32 - 1. * DDc * G * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T32 - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 + G *
Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * T32 + CC * DD * DDc * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) * T32 - 1. * DD * DDc * G *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * CC *
DD * Gc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 +
```

```
DD * G * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 −
1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + DDc * G *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) +
gamma * Pc2Start * t) * T32 + CC * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start *
t) * T32 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma *
t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + CC *
DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) *
((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * DDc *
G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) *
T52Final + gamma * t * T52Start) − 1. * CC * G * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) *
T52Final + gamma * t * T52Start) + DD * G * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final +
gamma * t * T52Start) + CCc * DD * G * (BcFinal * (1 − t) +
BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) −
1. * CCc * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * G * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final +
gamma * t * T52Start) + Power(G, 2) * Gc * (BcFinal * (1 − t) +
BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) −
1. * CCc * DD * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) *
((1 − t) * T52Final + gamma * t * T52Start) − 1. * CC * DDc * G *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final +
gamma * t * T52Start) + DD * DDc * G * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CCc *
```

Power(G, 2) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗
((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + CC ∗ G ∗ Gc ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc2Final ∗ (1 − t) +
gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) −
1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
Tc32 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ G ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc32 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc32 + CCc ∗ DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 + DDc ∗ G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 −
1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗
(1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗
Tc32 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) +
gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗

```
T52Final + gamma * t * T52Start) * Tc32 - 1. * DD * DDc * G * Gc *
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + DDc * Power(G, 2) *
Gc * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. * CCc *
DD * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start) * Tc32 + CCc * Power(G, 2) * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32 + DD * G * Gc * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. *
Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + CCc * DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc52Final + gamma *
t * Tc52Start) - 1. * DD * DDc * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * CCc *
G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + DDc * G * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC * DDc *
Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * DDc * G * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + G * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * CCc * DD * Gc * (P2Final * (1 - t) +
gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * CC * DDc * Gc * (P2Final * (1 - t) + gamma * P2Start * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + DD * DDc * Gc *
(P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final +
```

gamma * t * Tc52Start) + CCc * G * Gc * (P2Final * (1 − t) + gamma * P2Start * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + CC * Power(Gc, 2) * (P2Final * (1 − t) + gamma * P2Start * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * G * Power(Gc, 2) * (P2Final * (1 − t) + gamma * P2Start * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + CC * DD * DDc * Gc * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * DD * DDc * G * Gc * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * CC * DD * Power(Gc, 2) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + DD * G * Power(Gc, 2) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start);

f8 = −1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) − 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc − 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) − 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DDc * G * (AFinal * (1 − t) + AStart *

gamma * t) − 1. * CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma *
t) + CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. *
CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *
(1 − t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 − t) + AStart *
gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart *
gamma * t) − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) − 1. * CC * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma *
t) + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * G * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart *
gamma * t) + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P3Final * (1 − t) + gamma * P3Start * t) − 1. * DD * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start *
t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P3Final * (1 − t) + gamma * P3Start * t) + DD * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma *
P3Start * t) + CCc * DD * (BcFinal * (1 − t) + BcStart * gamma *
t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * CCc * G *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) +
gamma * P3Start * t) − 1. * DD * Gc * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + G * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) +
gamma * P3Start * t) + CCc * DD * (AFinal * (1 − t) + AStart *
gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. *

DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) − 1. * CCc * G * (AFinal *
(1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma *
Pc3Start * t) + DDc * G * (AFinal * (1 − t) + AStart * gamma *
t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CC * DDc *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) − 1. * DDc * G * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) −
1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) + G * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) −
1. * CCc * DD * (P3Final * (1 − t) + gamma * P3Start * t) *
(Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DDc *
(P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) + DD * DDc * (P3Final * (1 − t) + gamma *
P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CCc *
G * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) + CC * Gc * (P3Final * (1 − t) + gamma *
P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. *
G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) − 1. * CC * DD * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * T33 + DD * DDc * G * (AcFinal *
(1 − t) + AcStart * gamma * t) * T33 + CC * DD * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) * T33 − 1. * DD * G * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) * T33 + CC * DDc * (AcFinal * (1 − t) +
AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 −
1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *

```
(1 - t) + BStart * gamma * t) * T33 - 1. * CC * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) * T33 + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * T33 + CC * DD * DDc *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 - 1. * DD * DDc * G *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 - 1. * CC * DD * Gc *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + DD * G * Gc *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 - 1. * CC * DDc *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) * T33 + DDc * G * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + CC * Gc * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start *
t) * T33 - 1. * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T53Final +
gamma * t * T53Start) - 1. * DD * DDc * G * (AcFinal * (1 - t) +
AcStart * gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) -
1. * CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T53Final + gamma * t * T53Start) + DD * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T53Final +
gamma * t * T53Start) + CCc * DD * G * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. * CCc *
Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * DD * G * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) +
Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * CCc * DD * G * (Pc3Final * (1 - t) +
```

gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. *
CC * DDc * G * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + DD * DDc * G * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) + CCc *
Power(G, 2) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + CC * G * Gc * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. *
Power(G, 2) * Gc * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) * Tc33 + CCc * DDc * G * (AFinal * (1 - t) + AStart *
gamma * t) * Tc33 + DD * DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc33 - 1. * DDc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * Tc33 +
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc33 - 1. * CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) * Tc33 - 1. * DD *
Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) * Tc33 + G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc33 + CCc * DD * DDc * (P3Final
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * CCc * DDc * G * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * DD * DDc * Gc * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 + DDc * G * Gc * (P3Final * (1 - t) +
gamma * P3Start * t) * Tc33 - 1. * CCc * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) * Tc33 + CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma *
P3Start * t) * Tc33 + DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P3Final * (1 - t) + gamma * P3Start * t) * Tc33 - 1. * G * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) *

```
Tc33 + CCc * DD * DDc * G * ((1 - t) * T53Final + gamma * t * T53Start) *
Tc33 - 1. * CCc * DDc * Power(G, 2) * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33 - 1. * DD * DDc * G * Gc * ((1 - t) * T53Final + gamma *
t * T53Start) * Tc33 + DDc * Power(G, 2) * Gc * ((1 - t) * T53Final +
gamma * t * T53Start) * Tc33 - 1. * CCc * DD * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) *
Tc33 + CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T53Final + gamma * t * T53Start) * Tc33 + DD * G * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final +
gamma * t * T53Start) * Tc33 - 1. * Power(G, 2) * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33 + CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma *
t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DD * DDc * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) - 1. * CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma *
t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) + CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DDc * G * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) - 1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + G *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * CCc * DD * Gc * (P3Final *
(1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 1. * CC * DDc * Gc * (P3Final * (1 - t) + gamma * P3Start *
t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + DD * DDc * Gc *
```

(P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CCc * G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc, 2) * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * G * Power(Gc, 2) * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * DD * DDc * Gc * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * DDc * G * Gc * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * DD * Power(Gc, 2) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DD * G * Power(Gc, 2) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start);

f9 = −1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) − 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc − 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) − 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) +

CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗

```
Pc4Start * t) - 1. * CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + DDc *
G * (AFinal * (1 - t) + AStart * gamma * t) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) + CC * DDc * (BFinal * (1 - t) + BStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. *
DDc * G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) - 1. * CC * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) +
G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) - 1. * CCc * DD * (P4Final *
(1 - t) + gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) - 1. * CC * DDc * (P4Final * (1 - t) + gamma *
P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) +
DD * DDc * (P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + CCc * G * (P4Final * (1 - t) +
gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start *
t) + CC * Gc * (P4Final * (1 - t) + gamma * P4Start * t) *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. * G * Gc *
(P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) - 1. * CC * DD * DDc * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 + DD * DDc * G * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 + CC * DD * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 - 1. * DD * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * T34 + CC * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T34 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T34 - 1. *
```

CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + CC * DD * DDc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * DD * DDc * G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DD * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + DD * G * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) + DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) + CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * CCc * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * DD * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) +

Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T54Final + gamma * t * T54Start) − 1. * CCc * DD *
G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) *
T54Final + gamma * t * T54Start) − 1. * CC * DDc * G *
(Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) *
T54Final + gamma * t * T54Start) + DD * DDc * G * (Pc4Final *
(1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma *
t * T54Start) + CCc * Power(G, 2) * (Pc4Final * (1 − t) +
gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t *
T54Start) + CC * G * Gc * (Pc4Final * (1 − t) + gamma *
Pc4Start * t) * ((1 − t) * T54Final + gamma * t * T54Start) −
1. * Power(G, 2) * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start *
t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * CCc *
DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc34 +
CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) *
Tc34 + DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma *
t) * Tc34 − 1. * DDc * G * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * Tc34 + CCc * DD * (AFinal * (1 − t) + AStart *
gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 −
1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 − 1. * DD *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *
(1 − t) + BcStart * gamma * t) * Tc34 + G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart *
gamma * t) * Tc34 + CCc * DD * DDc * (P4Final * (1 − t) +
gamma * P4Start * t) * Tc34 − 1. * CCc * DDc * G * (P4Final *
(1 − t) + gamma * P4Start * t) * Tc34 − 1. * DD * DDc * Gc *

```
(P4Final * (1 - t) + gamma * P4Start * t) * Tc34 + DDc * G *
Gc * (P4Final * (1 - t) + gamma * P4Start * t) * Tc34 - 1. *
CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P4Final * (1 - t) + gamma * P4Start * t) * Tc34 + CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final *
(1 - t) + gamma * P4Start * t) * Tc34 + DD * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) +
gamma * P4Start * t) * Tc34 - 1. * G * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma *
P4Start * t) * Tc34 + CCc * DD * DDc * G * ((1 - t) *
T54Final + gamma * t * T54Start) * Tc34 - 1. * CCc * DDc *
Power(G, 2) * ((1 - t) * T54Final + gamma * t * T54Start) *
Tc34 - 1. * DD * DDc * G * Gc * ((1 - t) * T54Final + gamma *
t * T54Start) * Tc34 + DDc * Power(G, 2) * Gc * ((1 - t) *
T54Final + gamma * t * T54Start) * Tc34 - 1. * CCc * DD * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T54Final + gamma * t * T54Start) * Tc34 + CCc * Power(G, 2) *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T54Final + gamma * t * T54Start) * Tc34 + DD * G * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) * Tc34 - 1. * Power(G, 2) * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) * Tc34 + CCc * DD * Gc * (AFinal * (1 - t) +
AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) - 1. * DD * DDc * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) -
1. * CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
```

((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ G ∗ Gc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final +
gamma ∗ t ∗ Tc54Start) + CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗
Tc54Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) −
1. ∗ CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + G ∗
Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CCc ∗
DD ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗
(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) + DD ∗ DDc ∗ Gc ∗ (P4Final ∗
(1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗
t ∗ Tc54Start) + CCc ∗ G ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗
P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) +
CC ∗ Power(Gc, 2) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ G ∗
Power(Gc, 2) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ DD ∗ DDc ∗
Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗
DD ∗ DDc ∗ G ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗
Tc54Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T34 ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) + DD ∗ G ∗ Power(Gc, 2) ∗
T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗
DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗

((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

f10 = −1. ∗ CCc ∗ DD ∗ DDc ∗ G + CCc ∗ DDc ∗ Power(G, 2) − 1. ∗ CC ∗ DD ∗ DDc ∗ Gc + 2. ∗ DD ∗ DDc ∗ G ∗ Gc − 1. ∗ DDc ∗ Power(G, 2) ∗ Gc + CC ∗ DD ∗ Power(Gc, 2) − 1. ∗ DD ∗ G ∗ Power(Gc, 2) + CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗

(1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ DDc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + CCc ∗
DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) +
gamma ∗ P5Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗
t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗
(1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + DDc ∗
G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) +
gamma ∗ Pc5Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ DDc ∗
G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) +
gamma ∗ Pc5Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + G ∗ Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) − 1. ∗ CCc ∗ DD ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗
t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CC ∗ DDc ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) + DD ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗

(Pc5Final * (1 − t) + gamma * Pc5Start * t) + CCc * G * (P5Final *
(1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start *
t) + CC * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final *
(1 − t) + gamma * Pc5Start * t) − 1. * G * Gc * (P5Final * (1 − t) +
gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) −
1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 +
DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC *
DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 − 1. * DD *
G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * DDc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart *
gamma * t) * T35 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma *
t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * CC * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) +
BStart * gamma * t) * T35 + G * Gc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 + CC *
DD * DDc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DD *
DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC *
DD * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + DD * G *
Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC * DDc *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma *
Pc5Start * t) * T35 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + CC * Gc * (BFinal *
(1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start *
t) * T35 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
(Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + CC * DDc * G *
(AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T55Final + gamma *
t * T55Start) − 1. * DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma *

```
t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * CC * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T55Final + gamma *
t * T55Start) + DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T55Final + gamma * t * T55Start) + CCc * DD * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) - 1. * CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * DD * G *
Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * CCc * DD *
G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) - 1. * CC * DDc * G * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) + DD * DDc *
G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + CCc * Power(G, 2) * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) + CC * G *
Gc * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) - 1. * Power(G, 2) * Gc * (Pc5Final * (1 - t) +
gamma * Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) -
1. * CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 +
CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 + DD * DDc *
Gc * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 - 1. * DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * Tc35 + CCc * DD * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) * Tc35 - 1. * CCc * G * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc35 - 1. * DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
```

gamma ∗ t) ∗ Tc35 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 + CCc ∗ DD ∗ DDc ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ DDc ∗ G ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ Gc ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DDc ∗ G ∗ Gc ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ DD ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ Tc35 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DD ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ Tc35 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + CCc ∗
DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 −
1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T55Final +
gamma ∗ t ∗ T55Start) ∗ Tc35 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗
T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) ∗ Tc35 + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DD ∗
G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ Power(G, 2) ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗
t ∗ T55Start) ∗ Tc35 + CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗
DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal ∗

(1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start) + DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ DDc ∗ Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗
t ∗ Tc55Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗
Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start) + G ∗ Power(Gc, 2) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗
t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗
Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final +
gamma ∗ t ∗ Tc55Start) + DD ∗ DDc ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CCc ∗
G ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ Power(Gc, 2) ∗ (P5Final ∗
(1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗
DD ∗ DDc ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) −
1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T35 ∗ ((1 − t) ∗ Tc55Final +
gamma ∗ t ∗ Tc55Start) + DD ∗ G ∗ Power(Gc, 2) ∗ T35 ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗
t ∗ Tc55Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗

Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start);

f11 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + (P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + F * (BcFinal * (1 − t) + BcStart * gamma * t) * T31 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 − 1. * F * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 − 1. * H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) + H * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) + Fc * (BFinal * (1 − t) + BStart * gamma * t) * Tc31 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc31 − 1. * Fc * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 − 1. * Fc * H * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 −

1. * Hc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Hc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * F * Hc * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Hc * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

f12 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + (P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + F * (BcFinal * (1 − t) + BcStart * gamma * t) * T32 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 − 1. * F * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 − 1. * H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + H * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + Fc * (BFinal * (1 − t) + BStart * gamma * t) * Tc32 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc32 − 1. * Fc * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 − 1. * Fc * H *

```
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + H * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32 - 1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + Hc * (P2Final *
(1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) - 1. * F * Hc * T32 * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * T32 *
((1 - t) * Tc52Final + gamma * t * Tc52Start);

f13 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) + (P3Final * (1 - t) + gamma * P3Start * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) + F * (BcFinal *
(1 - t) + BcStart * gamma * t) * T33 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
T33 - 1. * F * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 +
(BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * T33 - 1. * H * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) + H * (Pc3Final * (1 - t) + gamma * Pc3Start * t) *
((1 - t) * T53Final + gamma * t * T53Start) + Fc * (BFinal *
(1 - t) + BStart * gamma * t) * Tc33 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
```

```
Tc33 - 1. * Fc * (P3Final * (1 - t) + gamma * P3Start * t) *
Tc33 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * Fc * H * ((1 - t) *
T53Final + gamma * t * T53Start) * Tc33 + H * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma *
t * T53Start) * Tc33 - 1. * Hc * (BFinal * (1 - t) + BStart *
gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) +
Hc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * F * Hc * T33 *
((1 - t) * Tc53Final + gamma * t * Tc53Start) + Hc *
(BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) *
Tc53Final + gamma * t * Tc53Start);

f14 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start *
t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + (P4Final * (1 - t) + gamma *
P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + F *
(BcFinal * (1 - t) + BcStart * gamma * t) * T34 - 1. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T34 - 1. * F * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * T34 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 - 1. *
H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
```

T54Final + gamma * t * T54Start) + H * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t * T54Start) + Fc * (BFinal * (1 − t) + BStart * gamma * t) * Tc34 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc34 − 1. * Fc * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 − 1. * Fc * H * ((1 − t) * T54Final + gamma * t * T54Start) * Tc34 + H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) * Tc34 − 1. * Hc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + Hc * (P4Final * (1 − t) + gamma * P4Start * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) − 1. * F * Hc * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + Hc * (BFinal * (1 − t) + BStart * gamma * t) * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start);

f15 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + F * (BcFinal * (1 − t) + BcStart * gamma * t) * T35 − 1. * (BcFinal *

(1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ F ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + H ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ Fc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ Fc ∗ H ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + Hc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ F ∗ Hc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);

f16 = −1. + T31 ∗ Tc31;

f17 = −1. + T32 ∗ Tc32;

f18 = −1. + T33 ∗ Tc33;

```
f19 = -1. + T34 * Tc34;

f20 = -1. + T35 * Tc35;

complex_t residual = sqrt(
f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 + f6*f6 + f7*f7 +
 f8*f8 + f9*f9 + f10*f10 +
f11*f11 + f12*f12 + f13*f13 + f14*f14 + f15*f15 + f16*f16 +
 f17*f17 + f18*f18 +
f19*f19 + f20*f20
);
printf("Residual: (%e, %e)\n", residual.real(),
 residual.imag());

}

// prints out the value of H at time t for all start points.
 Does not populate bArray. Intended for use in debugging.
__global__ void H_global(double t, complex_t *xArray,
 complex_t *startParams, complex_t *finalParams, complex_t *bArray,
  complex_t gamma, int n, int *solFlagArray, int numParameters,
   int kNumStartPoints)
{
int ETI = blockIdx.x * blockDim.x + threadIdx.x;
int EBI = ETI / kNumStartPoints;
```

```
int xIndex = (blockIdx.x * blockDim.x + threadIdx.x) * n;

int parameterOffset = EBI * numParameters;
complex_t *finalParamPtr = &finalParams[parameterOffset];

complex_t *x = &xArray[xIndex];

complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start, Pc2Start,
Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start, Tc51Start,
Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1]; P3Start = startParams[2];
P4Start = startParams[3]; P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6];
Pc3Start = startParams[7];
Pc4Start = startParams[8]; Pc5Start = startParams[9];
T51Start = startParams[10]; T52Start = startParams[11];
T53Start = startParams[12]; T54Start = startParams[13];
T55Start = startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start;
Tc53Start = 1.0 / T53Start; Tc54Start = 1.0 / T54Start;
Tc55Start = 1.0 / T55Start;
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start);
Tc53Start = conj(T53Start); Tc54Start = conj(T54Start);
Tc55Start = conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16];
```

```
BStart = startParams[17]; BcStart = startParams[18];

complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final, Pc2Final,
Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final, Tc51Final,
Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;
P1Final = finalParams[0]; P2Final = finalParams[1]; P3Final = finalParams[2];
P4Final = finalParams[3]; P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6];
Pc3Final = finalParams[7]; Pc4Final = finalParams[8];
Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11];
T53Final = finalParams[12]; T54Final = finalParams[13];
T55Final = finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final;
Tc53Final = 1.0 / T53Final; Tc54Final = 1.0 / T54Final;
Tc55Final = 1.0 / T55Final;
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final);
Tc53Final = conj(T53Final); Tc54Final = conj(T54Final);
Tc55Final = conj(T55Final);
AFinal = finalParams[15]; AcFinal = finalParams[16]; BFinal = finalParams[17]
BcFinal = finalParams[18];

complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31, T32, Tc32, T33,
Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4], Fc = x[5], G = x[6],
```

```
Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13], T33 = x[14],
Tc33 = x[15], T34 = x[16], Tc34 = x[17], T35 = x[18], Tc35 = x[19];

complex_t f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15,
f16, f17, f18, f19, f20;

f1 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) +
gamma * P1Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) + (P1Final * (1 - t) +
gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) +
DD * (BcFinal * (1 - t) + BcStart * gamma * t) * T31 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * T31 - 1. * DD * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 + (BFinal * (1 - t) + BStart * gamma *
t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + G * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) +
DDc * (BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * Tc31 - 1. * DDc * (P1Final *
(1 - t) + gamma * P1Start * t) * Tc31 + (BcFinal * (1 - t) +
BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start *
```

t) * Tc31 − 1. * DDc * G * ((1 − t) * T51Final + gamma * t *
T51Start) * Tc31 + G * (BcFinal * (1 − t) + BcStart *
gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) *
Tc31 − 1. * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) + Gc *
(P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * DD * Gc *
T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) +
Gc * (BFinal * (1 − t) + BStart * gamma * t) * T31 *
((1 − t) * Tc51Final + gamma * t * Tc51Start);

f2 = DDc * G + DD * Gc − 1. * DD * (BcFinal * (1 − t) + BcStart *
gamma * t) − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) +
2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) +
BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) *
(P2Final * (1 − t) + gamma * P2Start * t) − 1. * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) +
(P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) +
gamma * Pc2Start * t) + DD * (BcFinal * (1 − t) + BcStart * gamma *
t) * T32 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) * T32 − 1. * DD * (Pc2Final * (1 − t) +
gamma * Pc2Start * t) * T32 + (BFinal * (1 − t) + BStart * gamma * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 − 1. * G * (BcFinal *
(1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t *
T52Start) + G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) *
T52Final + gamma * t * T52Start) + DDc * (BFinal * (1 − t) + BStart *

gamma * t) * Tc32 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * Tc32 − 1. * DDc * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 − 1. * DDc * G * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 1. * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + Gc * (P2Final * (1 − t) + gamma * P2Start * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * DD * Gc * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + Gc * (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start);

f3 = DDc * G + DD * Gc − 1. * DD * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * T33 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 − 1. * DD * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 + (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) + G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) + DDc * (BFinal * (1 − t) +

BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);

f4 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) G ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

Tc34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

f5 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DDc ∗ (BFinal ∗ (1 − t) +

```
BStart * gamma * t) * Tc35 - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. * DDc * (P5Final *
(1 - t) + gamma * P5Start * t) * Tc35 + (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * DDc *
G * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 + G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) * Tc35 - 1. * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) -
1. * DD * Gc * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T35 * ((1 - t) * Tc55Final +
gamma * t * Tc55Start);

f6 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD *
DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC *
DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 -
t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) -
1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G *
Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) +
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) - 1. * G * Gc * (AFinal *
```

(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) - 1. * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) + DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) + CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) - 1. * CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) - 1. * DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) + G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) + CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + DDc * G * (AFinal * (1 - t) + AStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * DDc * G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final *

(1 − t) + gamma ∗ Pc1Start ∗ t) + G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CCc ∗ DD ∗
(P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗
Pc1Start ∗ t) − 1. ∗ CC ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + DD ∗ DDc ∗ (P1Final ∗
(1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗
t) + CCc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗
(1 − t) + gamma ∗ Pc1Start ∗ t) + CC ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗
P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ G ∗ Gc ∗
(P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗
Pc1Start ∗ t) − 1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗
t) ∗ T31 + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 +
CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ DD ∗
G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DDc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T31 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗
t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ CC ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T31 + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DD ∗ DDc ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ DD ∗ DDc ∗ G ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ DD ∗ Gc ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DD ∗ G ∗ Gc ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ DDc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗
Pc1Start ∗ t) ∗ T31 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ Gc ∗ (BFinal ∗

(1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start *
t) * T31 - 1. * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 + CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) - 1. * DD * DDc * G * (AcFinal * (1 - t) +
AcStart * gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) -
1. * CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T51Final + gamma * t * T51Start) + DD * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + CCc * DD * G * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc *
Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T51Final + gamma * t * T51Start) - 1. * DD * G * Gc * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc * DD * G *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * ((1 - t) * T51Final +
gamma * t * T51Start) - 1. * CC * DDc * G * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * ((1 - t) * T51Final + gamma * t *
T51Start) + DD * DDc * G * (Pc1Final * (1 - t) + gamma * Pc1Start *
t) * ((1 - t) * T51Final + gamma * t * T51Start) + CCc * Power(G, 2) *
(Pc1Final * (1 - t) + gamma * Pc1Start * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + CC * G * Gc * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. *
Power(G, 2) * Gc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) *
((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc * DD * DDc *
AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * DDc * G *

```
(AFinal * (1 − t) + AStart * gamma * t) * Tc31 + DD * DDc * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * Tc31 − 1. * DDc * G * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * Tc31 + CCc * DD *
(AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) +
BcStart * gamma * t) * Tc31 − 1. * CCc * G * (AFinal * (1 − t) +
AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) *
Tc31 − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc31 + G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma *
t) * Tc31 + CCc * DD * DDc * (P1Final * (1 − t) + gamma * P1Start *
t) * Tc31 − 1. * CCc * DDc * G * (P1Final * (1 − t) + gamma *
P1Start * t) * Tc31 − 1. * DD * DDc * Gc * (P1Final * (1 − t) +
gamma * P1Start * t) * Tc31 + DDc * G * Gc * (P1Final * (1 − t) +
gamma * P1Start * t) * Tc31 − 1. * CCc * DD * (BcFinal * (1 − t) +
BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) *
Tc31 + CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final *
(1 − t) + gamma * P1Start * t) * Tc31 + DD * Gc * (BcFinal * (1 − t) +
BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 −
1. * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final *
(1 − t) + gamma * P1Start * t) * Tc31 + CCc * DD * DDc * G * ((1 − t) *
T51Final + gamma * t * T51Start) * Tc31 − 1. * CCc * DDc * Power(G, 2) *
((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * DD * DDc * G *
Gc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + DDc *
Power(G, 2) * Gc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 −
1. * CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + CCc * Power(G, 2) *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final +
```

gamma * t * T51Start) * Tc31 + DD * G * Gc * (BcFinal * (1 − t) +
BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) *
Tc31 − 1. * Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma *
t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + CCc * DD *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final +
gamma * t * Tc51Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) +
AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) −
1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * G * Gc *
(BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma *
t * Tc51Start) − 1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + G *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * CCc * DD * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) − 1. * CC * DDc * Gc * (P1Final * (1 − t) + gamma *
P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DD *
DDc * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) + CCc * G * Gc * (P1Final *
(1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + CC * Power(Gc, 2) * (P1Final * (1 − t) + gamma * P1Start *
t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * G *
Power(Gc, 2) * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) + CC * DD * DDc * Gc * T31 *

((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DD ∗ G ∗ Power(Gc, 2) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);

f7 = −1. ∗ CCc ∗ DD ∗ DDc ∗ G + CCc ∗ DDc ∗ Power(G, 2) − 1. ∗ CC ∗ DD ∗ DDc ∗ Gc + 2. ∗ DD ∗ DDc ∗ G ∗ Gc − 1. ∗ DDc ∗ Power(G, 2) ∗ Gc + CC ∗ DD ∗ Power(Gc, 2) − 1. ∗ DD ∗ G ∗ Power(Gc, 2) + CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) +

```
DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) - 1. * G * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * CC * DDc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + DDc *
G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) - 1. * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) + CC * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P2Final * (1 - t) + gamma *
P2Start * t) - 1. * DD * DDc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. *
CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) + DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) + CCc * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. *
CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) - 1. * DD * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) + G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) + CCc *
DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) - 1. * DD * DDc * (AFinal *
(1 - t) + AStart * gamma * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) - 1. * CCc * G * (AFinal * (1 - t) + AStart *
```

gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DDc *
G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) + CC * DDc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) −
1. * DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) − 1. * CC * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) +
G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final *
(1 − t) + gamma * Pc2Start * t) − 1. * CCc * DD * (P2Final * (1 − t) +
gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) −
1. * CC * DDc * (P2Final * (1 − t) + gamma * P2Start * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) + DD * DDc * (P2Final *
(1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) + CCc * G * (P2Final * (1 − t) + gamma * P2Start * t) *
(Pc2Final * (1 − t) + gamma * Pc2Start * t) + CC * Gc * (P2Final *
(1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma *
Pc2Start * t) − 1. * G * Gc * (P2Final * (1 − t) + gamma * P2Start *
t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * DD *
DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + DD * DDc *
G * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + CC * DD *
Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 − 1. * DD *
G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + CC *
DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) * T32 − 1. * DDc * G * (AcFinal *
(1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart *
gamma * t) * T32 − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 + G *

```
Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * T32 + CC * DD * DDc * (Pc2Final *
(1 - t) + gamma * Pc2Start * t) * T32 - 1. * DD * DDc * G *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * CC *
DD * Gc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 +
DD * G * Gc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 -
1. * CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 + DDc * G *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 + CC * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start *
t) * T32 - 1. * G * Gc * (BFinal * (1 - t) + BStart * gamma *
t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 + CC *
DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T52Final + gamma * t * T52Start) - 1. * DD * DDc *
G * (AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start) - 1. * CC * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start) + DD * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T52Final +
gamma * t * T52Start) + CCc * DD * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) -
1. * CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T52Final + gamma * t * T52Start) - 1. * DD * G * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final +
gamma * t * T52Start) + Power(G, 2) * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) -
```

1. * CCc * DD * G * (Pc2Final * (1 - t) + gamma * Pc2Start * t) *
((1 - t) * T52Final + gamma * t * T52Start) - 1. * CC * DDc * G *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final +
gamma * t * T52Start) + DD * DDc * G * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) + CCc *
Power(G, 2) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) *
((1 - t) * T52Final + gamma * t * T52Start) + CC * G * Gc *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final +
gamma * t * T52Start) - 1. * Power(G, 2) * Gc * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) -
1. * CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc32 + CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) *
Tc32 + DD * DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc32 - 1. * DDc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc32 + CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) * Tc32 - 1. * CCc * G * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) * Tc32 - 1. * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc32 + G * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) * Tc32 + CCc * DD * DDc * (P2Final * (1 - t) + gamma * P2Start * t) *
Tc32 - 1. * CCc * DDc * G * (P2Final * (1 - t) + gamma * P2Start * t) *
Tc32 - 1. * DD * DDc * Gc * (P2Final * (1 - t) + gamma * P2Start * t) *
Tc32 + DDc * G * Gc * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 -
1. * CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final *
(1 - t) + gamma * P2Start * t) * Tc32 + CCc * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) *

Tc32 + DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P2Final * (1 - t) + gamma * P2Start * t) * Tc32 - 1. * G * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) * Tc32 + CCc * DD * DDc * G * ((1 - t) * T52Final +
gamma * t * T52Start) * Tc32 - 1. * CCc * DDc * Power(G, 2) * ((1 - t) *
T52Final + gamma * t * T52Start) * Tc32 - 1. * DD * DDc * G * Gc *
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + DDc * Power(G, 2) *
Gc * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. * CCc *
DD * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start) * Tc32 + CCc * Power(G, 2) * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32 + DD * G * Gc * (BcFinal * (1 - t) + BcStart * gamma *
t) * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. *
Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + CCc * DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc52Final + gamma *
t * Tc52Start) - 1. * DD * DDc * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * CCc *
G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + DDc * G * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC * DDc *
Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * DDc * G * Gc * (BFinal * (1 - t) +
BStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + G * Power(Gc, 2) *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +

gamma ∗ t ∗ Tc52Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CCc ∗ G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ Power(Gc, 2) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ DD ∗ DDc ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DD ∗ G ∗ Power(Gc, 2) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

f8 = −1. ∗ CCc ∗ DD ∗ DDc ∗ G + CCc ∗ DDc ∗ Power(G, 2) − 1. ∗ CC ∗ DD ∗ DDc ∗ Gc + 2. ∗ DD ∗ DDc ∗ G ∗ Gc − 1. ∗ DDc ∗ Power(G, 2) ∗ Gc + CC ∗ DD ∗ Power(Gc, 2) − 1. ∗ DD ∗ G ∗ Power(Gc, 2) + CC ∗ DD ∗

DDc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart *
gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma *
t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) - 1. * CC * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) + DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P3Final * (1 - t) + gamma * P3Start * t) - 1. * DD * DDc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start *
t) - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P3Final * (1 - t) + gamma * P3Start * t) + DD * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma *
P3Start * t) + CCc * DD * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) +

gamma * P3Start * t) − 1. * DD * Gc * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + G * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) +
gamma * P3Start * t) + CCc * DD * (AFinal * (1 − t) + AStart *
gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. *
DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) − 1. * CCc * G * (AFinal *
(1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma *
Pc3Start * t) + DDc * G * (AFinal * (1 − t) + AStart * gamma *
t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CC * DDc *
(BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) − 1. * DDc * G * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) −
1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) + G * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) −
1. * CCc * DD * (P3Final * (1 − t) + gamma * P3Start * t) *
(Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DDc *
(P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) + DD * DDc * (P3Final * (1 − t) + gamma *
P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CCc *
G * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) +
gamma * Pc3Start * t) + CC * Gc * (P3Final * (1 − t) + gamma *
P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. *
G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final *
(1 − t) + gamma * Pc3Start * t) − 1. * CC * DD * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * T33 + DD * DDc * G * (AcFinal *

(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DD ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 −
1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ T33 + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 + CC ∗ DD ∗ DDc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ DD ∗ DDc ∗ G ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ CC ∗ DD ∗ Gc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + DD ∗ G ∗ Gc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ CC ∗ DDc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗
Pc3Start ∗ t) ∗ T33 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ Gc ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗
t) ∗ T33 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ DDc ∗ G ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗

```
Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * DD * G * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) +
Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * CCc * DD * G * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. *
CC * DDc * G * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + DD * DDc * G * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) + CCc *
Power(G, 2) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + CC * G * Gc * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. *
Power(G, 2) * Gc * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * ((1 - t) *
T53Final + gamma * t * T53Start) - 1. * CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) * Tc33 + CCc * DDc * G * (AFinal * (1 - t) + AStart *
gamma * t) * Tc33 + DD * DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc33 - 1. * DDc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * Tc33 +
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc33 - 1. * CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) * Tc33 - 1. * DD *
Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) * Tc33 + G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc33 + CCc * DD * DDc * (P3Final
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * CCc * DDc * G * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 - 1. * DD * DDc * Gc * (P3Final *
(1 - t) + gamma * P3Start * t) * Tc33 + DDc * G * Gc * (P3Final * (1 - t) +
gamma * P3Start * t) * Tc33 - 1. * CCc * DD * (BcFinal * (1 - t) + BcStart *
```

gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 − 1. * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + CCc * DD * DDc * G * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * CCc * DDc * Power(G, 2) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * DD * DDc * G * Gc * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + DDc * Power(G, 2) * Gc * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + CCc * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + DD * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DDc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + G *

Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DD ∗ DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CCc ∗ G ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CC ∗ Power(Gc, 2) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CC ∗ DD ∗ DDc ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DD ∗ G ∗ Power(Gc, 2) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);

f9 = −1. ∗ CCc ∗ DD ∗ DDc ∗ G + CCc ∗ DDc ∗ Power(G, 2) − 1. ∗ CC ∗ DD ∗ DDc ∗ Gc + 2. ∗ DD ∗ DDc ∗ G ∗ Gc − 1. ∗ DDc ∗ Power(G, 2) ∗ Gc + CC ∗ DD ∗ Power(Gc, 2) − 1. ∗ DD ∗ G ∗ Power(Gc, 2) + CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ G ∗

$(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) + DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) + CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final *$

(1 − t) + gamma ∗ P4Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) +
CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DDc ∗
G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) +
gamma ∗ Pc4Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗
DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) +
G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CCc ∗ DD ∗ (P4Final ∗
(1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) − 1. ∗ CC ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗
P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) +
DD ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) + CCc ∗ G ∗ (P4Final ∗ (1 − t) +
gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗
t) + CC ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗
(Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ G ∗ Gc ∗
(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) +
gamma ∗ Pc4Start ∗ t) − 1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ T34 + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ T34 + CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) +

AcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 + CC ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T34 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗
CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ T34 + G ∗ Gc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T34 + CC ∗ DD ∗ DDc ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) ∗ T34 − 1. ∗ DD ∗ DDc ∗ G ∗ (Pc4Final ∗ (1 − t) +
gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ CC ∗ DD ∗ Gc ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + DD ∗ G ∗ Gc ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ CC ∗ DDc ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) ∗ T34 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 +
CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ G ∗ Gc ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) ∗ T34 + CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗
CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗

(1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma *
t * T54Start) − 1. * CCc * Power(G, 2) * (BcFinal * (1 − t) +
BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t *
T54Start) − 1. * DD * G * Gc * (BcFinal * (1 − t) + BcStart *
gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) +
Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T54Final + gamma * t * T54Start) − 1. * CCc * DD *
G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) *
T54Final + gamma * t * T54Start) − 1. * CC * DDc * G *
(Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) *
T54Final + gamma * t * T54Start) + DD * DDc * G * (Pc4Final *
(1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma *
t * T54Start) + CCc * Power(G, 2) * (Pc4Final * (1 − t) +
gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t *
T54Start) + CC * G * Gc * (Pc4Final * (1 − t) + gamma *
Pc4Start * t) * ((1 − t) * T54Final + gamma * t * T54Start) −
1. * Power(G, 2) * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start *
t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * CCc *
DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc34 +
CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) *
Tc34 + DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma *
t) * Tc34 − 1. * DDc * G * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * Tc34 + CCc * DD * (AFinal * (1 − t) + AStart *
gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 −
1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) * Tc34 − 1. * DD *
Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *

(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc34 + G ∗ Gc ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ Tc34 + CCc ∗ DD ∗ DDc ∗ (P4Final ∗ (1 − t) +
gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P4Final ∗
(1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ DD ∗ DDc ∗ Gc ∗
(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + DDc ∗ G ∗
Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗
CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + CCc ∗ G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗
(1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + DD ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) +
gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗
P4Start ∗ t) ∗ Tc34 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ DDc ∗
Power(G, 2) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗
Tc34 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗
t ∗ T54Start) ∗ Tc34 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ DD ∗ G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + CCc ∗ Power(G, 2) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + DD ∗ G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final +
gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗

T54Start) * Tc34 + CCc * DD * Gc * (AFinal * (1 − t) +
AStart * gamma * t) * ((1 − t) * Tc54Final + gamma * t *
Tc54Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) −
1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
((1 − t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc54Final +
gamma * t * Tc54Start) + CC * DDc * Gc * (BFinal * (1 − t) +
BStart * gamma * t) * ((1 − t) * Tc54Final + gamma * t *
Tc54Start) − 1. * DDc * G * Gc * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) −
1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma *
t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) + G *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) *
((1 − t) * Tc54Final + gamma * t * Tc54Start) − 1. * CCc *
DD * Gc * (P4Final * (1 − t) + gamma * P4Start * t) * ((1 − t) *
Tc54Final + gamma * t * Tc54Start) − 1. * CC * DDc * Gc *
(P4Final * (1 − t) + gamma * P4Start * t) * ((1 − t) *
Tc54Final + gamma * t * Tc54Start) + DD * DDc * Gc * (P4Final *
(1 − t) + gamma * P4Start * t) * ((1 − t) * Tc54Final + gamma *
t * Tc54Start) + CCc * G * Gc * (P4Final * (1 − t) + gamma *
P4Start * t) * ((1 − t) * Tc54Final + gamma * t * Tc54Start) +
CC * Power(Gc, 2) * (P4Final * (1 − t) + gamma * P4Start * t) *
((1 − t) * Tc54Final + gamma * t * Tc54Start) − 1. * G *
Power(Gc, 2) * (P4Final * (1 − t) + gamma * P4Start * t) *
((1 − t) * Tc54Final + gamma * t * Tc54Start) + CC * DD * DDc *
Gc * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start) − 1. *

```
DD * DDc * G * Gc * T34 * ((1 - t) * Tc54Final + gamma * t *
 Tc54Start) - 1. * CC * DD * Power(Gc, 2) * T34 * ((1 - t) *
 Tc54Final + gamma * t * Tc54Start) + DD * G * Power(Gc, 2) *
 T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC *
 DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T34 *
 ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc *
 (BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) *
 Tc54Final + gamma * t * Tc54Start) + CC * Power(Gc, 2) * (BFinal *
 (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma *
 t * Tc54Start) - 1. * G * Power(Gc, 2) * (BFinal * (1 - t) +
 BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start);

f10 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD *
 DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC *
 DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
 (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G * (AcFinal *
 (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal * (1 - t) +
 AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma *
 t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc *
 DDc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * Gc *
 (AFinal * (1 - t) + AStart * gamma * t) + CCc * G * Gc * (AFinal *
 (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) +
 AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + CCc *
 G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
 BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
 (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * G * Gc * (AFinal *
 (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) -
```

1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) + DDc * G * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + CC * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) −
1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) + CC * DDc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * DD * DDc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (P5Final * (1 − t) + gamma *
P5Start * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P5Final * (1 − t) + gamma * P5Start * t) + DD * Gc * (AcFinal * (1 − t) +
AcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) + CCc *
DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) +
gamma * P5Start * t) − 1. * CCc * G * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * DD * Gc *
(BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma *
P5Start * t) + G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) *
(P5Final * (1 − t) + gamma * P5Start * t) + CCc * DD * (AFinal *
(1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start *
t) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final *
(1 − t) + gamma * Pc5Start * t) − 1. * CCc * G * (AFinal * (1 − t) +
AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DDc *
G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) +
gamma * Pc5Start * t) + CC * DDc * (BFinal * (1 − t) + BStart *
gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * DDc *
G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) +
gamma * Pc5Start * t) − 1. * CC * Gc * (BFinal * (1 − t) + BStart *
gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + G * Gc *

(BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CCc * DD * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CC * DDc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DD * DDc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + CCc * G * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + CC * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * G * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 − 1. * DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 + CC * DD * DDc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DD * DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC * DD * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + DD * G * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + CC * Gc * (BFinal *

(1 - t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start *
t) * T35 - 1. * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 + CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T55Final + gamma *
t * T55Start) - 1. * DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma *
t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * CC * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T55Final + gamma *
t * T55Start) + DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) *
((1 - t) * T55Final + gamma * t * T55Start) + CCc * DD * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) - 1. * CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * DD * G *
Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) - 1. * CCc * DD *
G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) - 1. * CC * DDc * G * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) + DD * DDc *
G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + CCc * Power(G, 2) * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) + CC * G *
Gc * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * ((1 - t) * T55Final +
gamma * t * T55Start) - 1. * Power(G, 2) * Gc * (Pc5Final * (1 - t) +
gamma * Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) -
1. * CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 +
CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 + DD * DDc *
Gc * (AFinal * (1 - t) + AStart * gamma * t) * Tc35 - 1. * DDc * G * Gc *

(AFinal * (1 - t) + AStart * gamma * t) * Tc35 + CCc * DD * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) * Tc35 - 1. * CCc * G * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc35 - 1. * DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart *
gamma * t) * Tc35 + G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) * Tc35 + CCc * DD * DDc *
(P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * CCc * DDc * G *
(P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * DD * DDc * Gc *
(P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + DDc * G * Gc *
(P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * CCc * DD *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma *
P5Start * t) * Tc35 + CCc * G * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + DD * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma *
P5Start * t) * Tc35 - 1. * G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + CCc *
DD * DDc * G * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 -
1. * CCc * DDc * Power(G, 2) * ((1 - t) * T55Final + gamma * t *
T55Start) * Tc35 - 1. * DD * DDc * G * Gc * ((1 - t) * T55Final +
gamma * t * T55Start) * Tc35 + DDc * Power(G, 2) * Gc * ((1 - t) *
T55Final + gamma * t * T55Start) * Tc35 - 1. * CCc * DD * G * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) * Tc35 + CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 + DD *
G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T55Final + gamma * t * T55Start) * Tc35 - 1. * Power(G, 2) * Gc *

(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma *
t * T55Start) * Tc35 + CCc * DD * Gc * (AFinal * (1 − t) + AStart *
gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DD *
DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) *
Tc55Final + gamma * t * Tc55Start) − 1. * CCc * G * Gc * (AFinal *
(1 − t) + AStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t *
Tc55Start) + DDc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC * DDc * Gc *
(BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma *
t * Tc55Start) − 1. * DDc * G * Gc * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) *
Tc55Final + gamma * t * Tc55Start) + G * Power(Gc, 2) * (BFinal *
(1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t *
Tc55Start) − 1. * CCc * DD * Gc * (P5Final * (1 − t) + gamma * P5Start *
t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC * DDc *
Gc * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final +
gamma * t * Tc55Start) + DD * DDc * Gc * (P5Final * (1 − t) + gamma *
P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CCc *
G * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) *
Tc55Final + gamma * t * Tc55Start) + CC * Power(Gc, 2) * (P5Final *
(1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final + gamma * t *
Tc55Start) − 1. * G * Power(Gc, 2) * (P5Final * (1 − t) + gamma *
P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC *
DD * DDc * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) −
1. * DD * DDc * G * Gc * T35 * ((1 − t) * Tc55Final + gamma * t *
Tc55Start) − 1. * CC * DD * Power(Gc, 2) * T35 * ((1 − t) * Tc55Final +

```
gamma * t * Tc55Start) + DD * G * Power(Gc, 2) * T35 * ((1 − t) *
Tc55Final + gamma * t * Tc55Start) − 1. * CC * DDc * Gc * (BFinal *
(1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma *
t * Tc55Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma *
t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC *
Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 −
t) * Tc55Final + gamma * t * Tc55Start) − 1. * G * Power(Gc, 2) *
(BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final +
gamma * t * Tc55Start);

f11 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart * gamma *
t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) + 2. *
(BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) +
BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma *
t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * (BFinal *
(1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma *
Pc1Start * t) + (P1Final * (1 − t) + gamma * P1Start * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) + F * (BcFinal * (1 − t) +
BcStart * gamma * t) * T31 − 1. * (BcFinal * (1 − t) + BcStart * gamma *
t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 − 1. * F *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + (BFinal * (1 − t) +
BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) *
T31 − 1. * H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) *
T51Final + gamma * t * T51Start) + H * (Pc1Final * (1 − t) + gamma *
Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) + Fc *
(BFinal * (1 − t) + BStart * gamma * t) * Tc31 − 1. * (BcFinal *
(1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma *
```

```
t) * Tc31 - 1. * Fc * (P1Final * (1 - t) + gamma * P1Start * t) *
Tc31 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) +
gamma * P1Start * t) * Tc31 - 1. * Fc * H * ((1 - t) * T51Final +
gamma * t * T51Start) * Tc31 + H * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 -
1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc51Final + gamma * t * Tc51Start) + Hc * (P1Final * (1 - t) + gamma *
P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * F *
Hc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + Hc *
(BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final +
gamma * t * Tc51Start);

f12 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) + 2. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) + (P2Final * (1 - t) + gamma * P2Start * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) + F * (BcFinal *
(1 - t) + BcStart * gamma * t) * T32 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
T32 - 1. * F * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 +
(BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 - 1. * H * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) + H *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final +
```

```
gamma * t * T52Start) + Fc * (BFinal * (1 - t) + BStart * gamma * t) *
Tc32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) * Tc32 - 1. * Fc * (P2Final * (1 - t) +
gamma * P2Start * t) * Tc32 + (BcFinal * (1 - t) + BcStart * gamma *
t) * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 - 1. * Fc * H *
((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + H * (BcFinal *
(1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32 - 1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) + Hc * (P2Final *
(1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) - 1. * F * Hc * T32 * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * T32 *
((1 - t) * Tc52Final + gamma * t * Tc52Start);

f13 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) +
2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma *
t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * (BFinal *
(1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) + (P3Final * (1 - t) + gamma * P3Start * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) + F * (BcFinal *
(1 - t) + BcStart * gamma * t) * T33 - 1. * (BcFinal * (1 - t) +
BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) *
T33 - 1. * F * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 +
(BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * T33 - 1. * H * (BcFinal * (1 - t) +
```

BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t *
T53Start) + H * (Pc3Final * (1 − t) + gamma * Pc3Start * t) *
((1 − t) * T53Final + gamma * t * T53Start) + Fc * (BFinal *
(1 − t) + BStart * gamma * t) * Tc33 − 1. * (BcFinal * (1 − t) +
BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) *
Tc33 − 1. * Fc * (P3Final * (1 − t) + gamma * P3Start * t) *
Tc33 + (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final *
(1 − t) + gamma * P3Start * t) * Tc33 − 1. * Fc * H * ((1 − t) *
T53Final + gamma * t * T53Start) * Tc33 + H * (BcFinal *
(1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma *
t * T53Start) * Tc33 − 1. * Hc * (BFinal * (1 − t) + BStart *
gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) +
Hc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) *
Tc53Final + gamma * t * Tc53Start) − 1. * F * Hc * T33 *
((1 − t) * Tc53Final + gamma * t * Tc53Start) + Hc *
(BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) *
Tc53Final + gamma * t * Tc53Start);

f14 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart *
gamma * t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) +
2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) +
BcStart * gamma * t) * (P4Final * (1 − t) + gamma * P4Start *
t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final *
(1 − t) + gamma * Pc4Start * t) + (P4Final * (1 − t) + gamma *
P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) + F *
(BcFinal * (1 − t) + BcStart * gamma * t) * T34 − 1. * (BcFinal *

```
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ T34 − 1. ∗ F ∗ (Pc4Final ∗ (1 − t) + gamma ∗
Pc4Start ∗ t) ∗ T34 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗
H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) + H ∗ (Pc4Final ∗ (1 − t) +
gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) + Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
Tc34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ Fc ∗
(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗
P4Start ∗ t) ∗ Tc34 − 1. ∗ Fc ∗ H ∗ ((1 − t) ∗ T54Final +
gamma ∗ t ∗ T54Start) ∗ Tc34 + H ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) ∗ Tc34 − 1. ∗ Hc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) +
Hc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ F ∗ Hc ∗ T34 ∗
((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Hc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗
Tc54Final + gamma ∗ t ∗ Tc54Start);

f15 = Fc ∗ H + F ∗ Hc − 1. ∗ F ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) − 1. ∗ Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) +
2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) +
```

```
BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start *
t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final *
(1 - t) + gamma * Pc5Start * t) + (P5Final * (1 - t) + gamma *
P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + F *
(BcFinal * (1 - t) + BcStart * gamma * t) * T35 - 1. * (BcFinal *
(1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart *
gamma * t) * T35 - 1. * F * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * T35 + (BFinal * (1 - t) + BStart * gamma * t) *
(Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 - 1. * H *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) + H * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) * ((1 - t) * T55Final + gamma * t * T55Start) +
Fc * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) * Tc35 - 1. * Fc * (P5Final * (1 - t) +
gamma * P5Start * t) * Tc35 + (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 -
1. * Fc * H * ((1 - t) * T55Final + gamma * t * T55Start) *
Tc35 + H * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T55Final + gamma * t * T55Start) * Tc35 - 1. *
Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + Hc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) - 1. * F * Hc * T35 * ((1 - t) * Tc55Final + gamma *
t * Tc55Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) *
T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start);
```

```
f16 = -1. + T31 * Tc31;

f17 = -1. + T32 * Tc32;

f18 = -1. + T33 * Tc33;

f19 = -1. + T34 * Tc34;

f20 = -1. + T35 * Tc35;

complex_t residual = sqrt(
f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 + f6*f6 + f7*f7 +
 f8*f8 + f9*f9 + f10*f10 +
f11*f11 + f12*f12 + f13*f13 + f14*f14 + f15*f15 + f16*f16 +
 f17*f17 + f18*f18 +
f19*f19 + f20*f20
);
printf("Effective Thread index: %d, Residual: (%e, %e)\n",
 blockIdx.x * blockDim.x + threadIdx.x, residual.real(), residual.imag());


if (residual.real() < 1e-8 && residual.imag() < 1e-8)
{
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 1;
}
else {
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 0;
```

```
}
```

B.2.4 Hx.h

```
#ifndef HX_H
#define HX_H
#include "Definitions.h"

#include "cuda_runtime.h"
#include "device_launch_parameters.h"

// Returns the Jacobian Matrix of the Homotopy System at point
 x at time t.
// effectively, dH/dx(t, x(t))
__device__ void Hx(double t, complex_t* x, complex_t* startParams,
 complex_t* finalParams, complex_t* currentJMatWorkspace,
  complex_t gamma, int n);

#endif
```

B.2.5 Hx.cu

```
// Returns the Jacobian Matrix of the Homotopy System at point x
 at time t.
// effectively, dH/dx(t, x(t))
__device__ void Hx(double t, complex_t* x, complex_t* startParams,
complex_t* finalParams, complex_t* currentJMatWorkspace,
complex_t gamma, int n)
```

```
{
complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start,
Pc2Start, Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start,
Tc51Start, Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1];
P3Start = startParams[2]; P4Start = startParams[3];
P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6];
Pc3Start = startParams[7]; Pc4Start = startParams[8];
Pc5Start = startParams[9];
T51Start = startParams[10]; T52Start = startParams[11];
T53Start = startParams[12]; T54Start = startParams[13];
T55Start = startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start;
Tc53Start = 1.0 / T53Start; Tc54Start = 1.0 / T54Start;
Tc55Start = 1.0 / T55Start;
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start);
Tc53Start = conj(T53Start); Tc54Start = conj(T54Start);
Tc55Start = conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16];
BStart = startParams[17]; BcStart = startParams[18];

complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final,
Pc2Final, Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final,
```

```
Tc51Final, Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;

P1Final = finalParams[0]; P2Final = finalParams[1];
P3Final = finalParams[2]; P4Final = finalParams[3];
P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6];
Pc3Final = finalParams[7]; Pc4Final = finalParams[8];
Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11];
T53Final = finalParams[12]; T54Final = finalParams[13];
T55Final = finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final;
Tc53Final = 1.0 / T53Final; Tc54Final = 1.0 / T54Final;
Tc55Final = 1.0 / T55Final;
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final);
Tc53Final = conj(T53Final); Tc54Final = conj(T54Final);
Tc55Final = conj(T55Final);

AFinal = finalParams[15]; AcFinal = finalParams[16];
BFinal = finalParams[17]; BcFinal = finalParams[18];

complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31,
T32, Tc32, T33, Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4],
Fc = x[5], G = x[6], Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13],
```

```
T33 = x[14], Tc33 = x[15], T34 = x[16], Tc34 = x[17],
T35 = x[18], Tc35 = x[19];

complex_t DD1, DDc1, G1, Gc1, T311, Tc311, DD2, DDc2, G2,
Gc2, T322, Tc322, DD3, DDc3, G3, Gc3;
complex_t T333, Tc333, DD4, DDc4, G4, Gc4, T344, Tc344, DD5,
DDc5, G5, Gc5, T355, Tc355, CC6;
complex_t CCc6, DD6, DDc6, G6, Gc6, T316, Tc316, CC7, CCc7,
DD7, DDc7, G7, Gc7, T327, Tc327;
complex_t CC8, CCc8, DD8, DDc8, G8, Gc8, T338, Tc338, CC9,
CCc9, DD9, DDc9, G9, Gc9, T349, Tc349;
complex_t CC10, CCc10, DD10, DDc10, G10, Gc10, T3510, Tc3510,
F11, Fc11, H11, Hc11, T3111, Tc3111;
complex_t F12, Fc12, H12, Hc12, T3212, Tc3212, F13, Fc13, H13,
Hc13, T3313, Tc3313, F14, Fc14, H14, Hc14;
complex_t T3414, Tc3414, F15, Fc15, H15, Hc15, T3515, Tc3515,
T3116, Tc3116, T3217, Tc3217, T3318, Tc3318;
complex_t T3419, Tc3419, T3520, Tc3520;

DD1 = Gc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
(1 - t) + BcStart * gamma * t) * T31 - 1. * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 - 1. * Gc * T31 * ((1 - t) * Tc51Final +
gamma * t * Tc51Start);

DDc1 = G - 1. * (BFinal * (1 - t) + BStart * gamma * t) + (BFinal * (1
- t) + BStart * gamma * t) * Tc31 - 1. * (P1Final * (1 - t) + gamma *
P1Start * t) * Tc31 - 1. * G * ((1 - t) * T51Final + gamma * t *
```

T51Start) * Tc31;

G1 = DDc − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) + (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * DDc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31;

Gc1 = DD − 1. * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DD * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

T311 = DD * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * DD * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * DD * Gc * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

Tc311 = DDc * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * DDc * (P1Final * (1 − t) + gamma * P1Start * t) + (BcFinal

```
* (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma *
P1Start * t) - 1. * DDc * G * ((1 - t) * T51Final + gamma * t *
T51Start) + G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T51Final + gamma * t * T51Start);

DD2 = Gc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
(1 - t) + BcStart * gamma * t) * T32 - 1. * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 - 1. * Gc * T32 * ((1 - t) * Tc52Final +
gamma * t * Tc52Start);

DDc2 = G - 1. * (BFinal * (1 - t) + BStart * gamma * t) + (BFinal * (1
- t) + BStart * gamma * t) * Tc32 - 1. * (P2Final * (1 - t) + gamma *
P2Start * t) * Tc32 - 1. * G * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32;

G2 = DDc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start) + (Pc2Final * (1 - t) + gamma *
Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) - 1. * DDc
* ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 + (BcFinal * (1 -
t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t *
T52Start) * Tc32;

Gc2 = DD - 1. * (BFinal * (1 - t) + BStart * gamma * t) * ((1
- t) * Tc52Final + gamma * t * Tc52Start) + (P2Final * (1 - t) + gamma
* P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. *
DD * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + (BFinal *
(1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t
```

∗ Tc52Start);

T322 = DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DD ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ DD ∗ Gc ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

Tc322 = DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start);

DD3 = Gc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);

DDc3 = G − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33;

```
G3 = DDc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + (Pc3Final * (1 - t) + gamma *
Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. * DDc
* ((1 - t) * T53Final + gamma * t * T53Start) * Tc33 + (BcFinal * (1 -
t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33;
Gc3 = DD - 1. * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) + (P3Final * (1 - t) + gamma *
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DD
* T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + (BFinal * (1 -
t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start);
T333 = DD * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * (BcFinal
* (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) - 1. * DD * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + (BFinal
* (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) - 1. * DD * Gc * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) + Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start);
Tc333 = DDc * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal
* (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) - 1. * DDc * (P3Final * (1 - t) + gamma * P3Start * t) + (BcFinal
* (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma *
P3Start * t) - 1. * DDc * G * ((1 - t) * T53Final + gamma * t *
T53Start) + G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start);
DD4 = Gc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
```

(1 − t) + BcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

DDc4 = G − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34;

G4 = DDc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ DDc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34;

Gc4 = DD − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

T344 = DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DD ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ Gc ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

Tc344 = DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal

∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start);
DD5 = Gc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);
DDc5 = G − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35;
G5 = DDc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DDc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35;
Gc5 = DD − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);
T355 = DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ (BcFinal

∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ DD ∗ Gc ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);
Tc355 = DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start);
CC6 = −1. ∗ DD ∗ DDc ∗ Gc + DD ∗ Power(Gc, 2) + DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + Gc ∗ (P1Final ∗ (1 − t) + gamma ∗

P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T31 + DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T31 + DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 − 1. * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 + DD * DDc * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 − 1. * DD * Gc * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * DDc * G * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) + G * Gc * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) + DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Power(Gc, 2) * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DD * DDc * Gc * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DD * Power(Gc, 2) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ...

gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start);
CCc6 = −1. * DD * DDc * G + DDc * Power(G, 2) + DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * DD * (P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + G * (P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * DD * G * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) + Power(G, 2) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t * T51Start) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc31 + DDc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc31 + DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc31 − 1. * G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc31 + DD * DDc * (P1Final * (1 − t)

+ gamma * P1Start * t) * Tc31 − 1. * DDc * G * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 − 1. * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + DD * DDc * G * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * DDc * Power(G, 2) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DD * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + G * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

DD6 = −1. * CCc * DDc * G − 1. * CC * DDc * Gc + 2. * DDc * G * Gc + CC * Power(Gc, 2) − 1. * G * Power(Gc, 2) + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + CCc * (BcFinal * (1 − t) + BcStart * gamma * t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CCc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DDc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ DDc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + G ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DDc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) +

```
BcStart * gamma * t) * Tc31 - 1. * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) * Tc31 + CCc *
DDc * (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 - 1. * DDc * Gc
* (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 - 1. * CCc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma
* P1Start * t) * Tc31 + Gc * (BcFinal * (1 - t) + BcStart * gamma * t)
* (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 + CCc * DDc * G *
((1 - t) * T51Final + gamma * t * T51Start) * Tc31 - 1. * DDc * G * Gc
* ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 - 1. * CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) * Tc31 + G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 + CCc
* Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc51Final
+ gamma * t * Tc51Start) - 1. * DDc * Gc * (AFinal * (1 - t) + AStart
* gamma * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. *
CCc * Gc * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) *
Tc51Final + gamma * t * Tc51Start) + DDc * Gc * (P1Final * (1 - t) +
gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) +
CC * DDc * Gc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) -
1. * DDc * G * Gc * T31 * ((1 - t) * Tc51Final + gamma * t *
Tc51Start) - 1. * CC * Power(Gc, 2) * T31 * ((1 - t) * Tc51Final +
gamma * t * Tc51Start) + G * Power(Gc, 2) * T31 * ((1 - t) * Tc51Final
+ gamma * t * Tc51Start);
DDc6 = -1. * CCc * DD * G + CCc * Power(G, 2) - 1. * CC * DD * Gc + 2.
* DD * G * Gc - 1. * Power(G, 2) * Gc + CC * DD * (AcFinal * (1 - t) +
AcStart * gamma * t) - 1. * CC * G * (AcFinal * (1 - t) + AcStart *
gamma * t) + CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
```

CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CC ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + DD ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T31 + CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 + CC ∗ DD ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ DD ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CC ∗ G

* (Pc1Final * (1 - t) + gamma * Pc1Start * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + DD * G * (Pc1Final * (1 - t) + gamma *
Pc1Start * t) * ((1 - t) * T51Final + gamma * t * T51Start) - 1. * CCc
* DD * (AFinal * (1 - t) + AStart * gamma * t) * Tc31 + CCc * G *
(AFinal * (1 - t) + AStart * gamma * t) * Tc31 + DD * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * Tc31 - 1. * G * Gc * (AFinal * (1 - t)
+ AStart * gamma * t) * Tc31 + CCc * DD * (P1Final * (1 - t) + gamma *
P1Start * t) * Tc31 - 1. * CCc * G * (P1Final * (1 - t) + gamma *
P1Start * t) * Tc31 - 1. * DD * Gc * (P1Final * (1 - t) + gamma *
P1Start * t) * Tc31 + G * Gc * (P1Final * (1 - t) + gamma * P1Start *
t) * Tc31 + CCc * DD * G * ((1 - t) * T51Final + gamma * t * T51Start)
* Tc31 - 1. * CCc * Power(G, 2) * ((1 - t) * T51Final + gamma * t *
T51Start) * Tc31 - 1. * DD * G * Gc * ((1 - t) * T51Final + gamma * t
* T51Start) * Tc31 + Power(G, 2) * Gc * ((1 - t) * T51Final + gamma *
t * T51Start) * Tc31 - 1. * DD * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc51Final + gamma
* t * Tc51Start) + CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * G * Gc * (BFinal
* (1 - t) + BStart * gamma * t) * ((1 - t) * Tc51Final + gamma * t *
Tc51Start) - 1. * CC * Gc * (P1Final * (1 - t) + gamma * P1Start * t)
* ((1 - t) * Tc51Final + gamma * t * Tc51Start) + DD * Gc * (P1Final *
(1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t *
Tc51Start) + CC * DD * Gc * T31 * ((1 - t) * Tc51Final + gamma * t *
Tc51Start) - 1. * DD * G * Gc * T31 * ((1 - t) * Tc51Final + gamma * t
* Tc51Start) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t)
* T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + G * Gc *

```
(BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final +
gamma * t * Tc51Start);
G6 = −1. * CCc * DD * DDc + 2 * CCc * DDc * G + 2. * DD * DDc * Gc −
2. * DDc * G * Gc − 1. * DD * Power(Gc, 2) − 1. * CC * DDc * (AcFinal
* (1 − t) + AcStart * gamma * t) + CC * Gc * (AcFinal * (1 − t) +
AcStart * gamma * t) − 1. * CCc * DDc * (AFinal * (1 − t) + AStart *
gamma * t) + CCc * Gc * (AFinal * (1 − t) + AStart * gamma * t) + CCc
* (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) +
BcStart * gamma * t) − 1. * Gc * (AFinal * (1 − t) + AStart * gamma *
t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DDc * (AcFinal * (1 −
t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) −
1. * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 −
t) + BStart * gamma * t) − 1. * CCc * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + Gc * (BcFinal
* (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma *
P1Start * t) − 1. * CCc * (AFinal * (1 − t) + AStart * gamma * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) + DDc * (AFinal * (1 − t)
+ AStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) −
1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 −
t) + gamma * Pc1Start * t) + Gc * (BFinal * (1 − t) + BStart * gamma *
t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + CCc * (P1Final * (1
− t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) − 1. * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final *
(1 − t) + gamma * Pc1Start * t) + DD * DDc * (AcFinal * (1 − t) +
AcStart * gamma * t) * T31 − 1. * DD * Gc * (AcFinal * (1 − t) +
AcStart * gamma * t) * T31 − 1. * DDc * (AcFinal * (1 − t) + AcStart *
gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 + Gc *
```

(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 − 1. ∗ DD ∗ DDc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DD ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 2. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + 2 ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DD ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CC ∗ DDc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DD ∗ DDc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + 2 ∗ CCc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CC ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 2. ∗ G ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗

T51Final + gamma * t * T51Start) + CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc31 − 1. * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc31 − 1. * CCc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc31 + Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc31 − 1. * CCc * DDc * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + DDc * Gc * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + CCc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 − 1. * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) * Tc31 + CCc * DD * DDc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 2. * CCc * DDc * G * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * DD * DDc * Gc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + 2 * DDc * G * Gc * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + 2 * CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 + DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 2. * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) * Tc31 − 1. * CCc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + CCc * Gc *

```
(P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final +
gamma * t * Tc51Start) - 1. * Power(Gc, 2) * (P1Final * (1 - t) +
gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) -
1. * DD * DDc * Gc * T31 * ((1 - t) * Tc51Final + gamma * t *
Tc51Start) + DD * Power(Gc, 2) * T31 * ((1 - t) * Tc51Final + gamma *
t * Tc51Start) + DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) *
Tc51Final + gamma * t * Tc51Start);
Gc6 = -1. * CC * DD * DDc + 2. * DD * DDc * G - 1. * DDc * Power(G, 2)
+ 2 * CC * DD * Gc - 2. * DD * G * Gc - 1. * CC * DD * (AcFinal * (1 -
t) + AcStart * gamma * t) + CC * G * (AcFinal * (1 - t) + AcStart *
gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) +
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD * (AFinal * (1
- t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t)
- 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 -
t) + BcStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart * gamma
* t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) - 1. * CC * (AcFinal * (1 - t) + AcStart * gamma * t) * (P1Final *
(1 - t) + gamma * P1Start * t) + DD * (AcFinal * (1 - t) + AcStart *
gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) - 1. * DD *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma
* P1Start * t) + G * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P1Final * (1 - t) + gamma * P1Start * t) - 1. * CC * (BFinal * (1 -
t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t)
+ G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) +
```

gamma * Pc1Start * t) + CC * (P1Final * (1 − t) + gamma * P1Start * t)
* (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * G * (P1Final * (1
− t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) + CC * DD * (AcFinal * (1 − t) + AcStart * gamma * t) * T31 − 1. *
DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T31 − 1. * CC *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart
* gamma * t) * T31 + G * (AcFinal * (1 − t) + AcStart * gamma * t) *
(BFinal * (1 − t) + BStart * gamma * t) * T31 − 1. * CC * DD *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + DD * G * (Pc1Final
* (1 − t) + gamma * Pc1Start * t) * T31 + CC * (BFinal * (1 − t) +
BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) *
T31 − 1. * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final *
(1 − t) + gamma * Pc1Start * t) * T31 − 1. * CC * G * (AcFinal * (1 −
t) + AcStart * gamma * t) * ((1 − t) * T51Final + gamma * t *
T51Start) + DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 −
t) * T51Final + gamma * t * T51Start) − 1. * DD * G * (BcFinal * (1 −
t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t *
T51Start) + Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) *
((1 − t) * T51Final + gamma * t * T51Start) + CC * G * (Pc1Final * (1
− t) + gamma * Pc1Start * t) * ((1 − t) * T51Final + gamma * t *
T51Start) − 1. * Power(G, 2) * (Pc1Final * (1 − t) + gamma * Pc1Start
* t) * ((1 − t) * T51Final + gamma * t * T51Start) + DD * DDc *
(AFinal * (1 − t) + AStart * gamma * t) * Tc31 − 1. * DDc * G *
(AFinal * (1 − t) + AStart * gamma * t) * Tc31 − 1. * DD * (AFinal *
(1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma *
t) * Tc31 + G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *
(1 − t) + BcStart * gamma * t) * Tc31 − 1. * DD * DDc * (P1Final * (1

− t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + DDc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 2. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + 2 ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CCc ∗ DD ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CC ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DD ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CCc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗

P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + 2 * CC * Gc * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 2. * G * Gc * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + CC * DD * DDc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * DD * DDc * G * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 2. * CC * DD * Gc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + 2 * DD * G * Gc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + 2 * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 2. * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start);
T316 = -1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) + DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + CC * DD * DDc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * DD * DDc * G * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CC * DD * Gc * (Pc1Final * (1 - t) + gamma *

Pc1Start * t) + DD * G * Gc * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + CC * DD * DDc * Gc * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * DD * DDc * G * Gc * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * CC * DD * Power(Gc, 2) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DD * G * Power(Gc, 2) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start);

Tc316 = −1. * CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) + DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DDc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart

∗ gamma ∗ t) + CCc ∗ DD ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗
t) − 1. ∗ CCc ∗ DDc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) −
1. ∗ DD ∗ DDc ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + DDc ∗
G ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ CCc ∗ DD ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma
∗ P1Start ∗ t) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + DD ∗ Gc ∗ (BcFinal ∗ (1 −
t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t)
− 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗
(1 − t) + gamma ∗ P1Start ∗ t) + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗
T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1
− t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1
− t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1
− t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal
∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗
T51Start) + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DD ∗ G ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final +
gamma ∗ t ∗ T51Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start);
CC7 = −1. ∗ DD ∗ DDc ∗ Gc + DD ∗ Power(Gc, 2) + DD ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗
DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) +

AcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1.
* Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P2Final * (1 - t)
+ gamma * P2Start * t) + DDc * (BFinal * (1 - t) + BStart * gamma * t)
* (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * Gc * (BFinal * (1
- t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start *
t) - 1. * DDc * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final
* (1 - t) + gamma * Pc2Start * t) + Gc * (P2Final * (1 - t) + gamma *
P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * DD *
DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * T32 + DDc * (AcFinal * (1
- t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t)
* T32 - 1. * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal
* (1 - t) + BStart * gamma * t) * T32 + DD * DDc * (Pc2Final * (1 - t)
+ gamma * Pc2Start * t) * T32 - 1. * DD * Gc * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * T32 - 1. * DDc * (BFinal * (1 - t) + BStart *
gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma
* Pc2Start * t) * T32 + DDc * G * (AcFinal * (1 - t) + AcStart * gamma
* t) * ((1 - t) * T52Final + gamma * t * T52Start) - 1. * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T52Final +
gamma * t * T52Start) - 1. * DDc * G * (Pc2Final * (1 - t) + gamma *
Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) + G * Gc *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) * T52Final +
gamma * t * T52Start) + DDc * Gc * (BFinal * (1 - t) + BStart * gamma
* t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * DDc * Gc * (P2Final * (1 - t) + gamma *

```
P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) +
Power(Gc, 2) * (P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) *
Tc52Final + gamma * t * Tc52Start) + DD * DDc * Gc * T32 * ((1 - t) *
Tc52Final + gamma * t * Tc52Start) - 1. * DD * Power(Gc, 2) * T32 *
((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * DDc * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + Power(Gc, 2) * (BFinal * (1 - t) + BStart *
gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start);
CCc7 = -1. * DD * DDc * G + DDc * Power(G, 2) + DD * DDc * (AFinal *
(1 - t) + AStart * gamma * t) - 1. * DDc * G * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * DD * Gc * (AFinal * (1 - t) + AStart *
gamma * t) + G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + G * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) + DD * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1.
* G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) +
gamma * P2Start * t) + DD * (AFinal * (1 - t) + AStart * gamma * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * G * (AFinal * (1 -
t) + AStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t)
- 1. * DD * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1
- t) + gamma * Pc2Start * t) + G * (P2Final * (1 - t) + gamma *
P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + DD * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final +
gamma * t * T52Start) - 1. * Power(G, 2) * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) -
1. * DD * G * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * ((1 - t) *
```

T52Final + gamma * t * T52Start) + Power(G, 2) * (Pc2Final * (1 - t) +
gamma * Pc2Start * t) * ((1 - t) * T52Final + gamma * t * T52Start) -
1. * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * Tc32 + DDc *
G * (AFinal * (1 - t) + AStart * gamma * t) * Tc32 + DD * (AFinal * (1
- t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t)
* Tc32 - 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) * Tc32 + DD * DDc * (P2Final * (1 - t)
+ gamma * P2Start * t) * Tc32 - 1. * DDc * G * (P2Final * (1 - t) +
gamma * P2Start * t) * Tc32 - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) * Tc32 + G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma
* P2Start * t) * Tc32 + DD * DDc * G * ((1 - t) * T52Final + gamma * t
* T52Start) * Tc32 - 1. * DDc * Power(G, 2) * ((1 - t) * T52Final +
gamma * t * T52Start) * Tc32 - 1. * DD * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) *
Tc32 + Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 -
t) * T52Final + gamma * t * T52Start) * Tc32 + DD * Gc * (AFinal * (1
- t) + AStart * gamma * t) * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * DD * Gc *
(P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + G * Gc * (P2Final * (1 - t) + gamma * P2Start
* t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start);
DD7 = -1. * CCc * DDc * G - 1. * CC * DDc * Gc + 2. * DDc * G * Gc +
CC * Power(Gc, 2) - 1. * G * Power(Gc, 2) + CC * DDc * (AcFinal * (1 -
t) + AcStart * gamma * t) - 1. * CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) + CCc * DDc * (AFinal * (1 - t) + AStart * gamma

∗ t) − 1. ∗ CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗
CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) + Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t)
+ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t)
+ gamma ∗ P2Start ∗ t) + CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start
∗ t) + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1
− t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ CCc ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ (Pc2Final ∗ (1 − t) +
gamma ∗ Pc2Start ∗ t) + DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗
t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ CC ∗ DDc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T32 + DDc ∗ G ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ T32 + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ T32 + CC ∗ DDc ∗ (Pc2Final ∗ (1 − t) + gamma ∗
Pc2Start ∗ t) ∗ T32 − 1. ∗ DDc ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗
Pc2Start ∗ t) ∗ T32 − 1. ∗ CC ∗ Gc ∗ (Pc2Final ∗ (1 − t) + gamma ∗
Pc2Start ∗ t) ∗ T32 + G ∗ Gc ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start
∗ t) ∗ T32 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t)
∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + G ∗ Gc ∗ (AcFinal ∗ (1
− t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗
T52Start) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1
− t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1

− t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DDc ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CCc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ DDc ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗

Tc52Start) − 1. * CC * Power(Gc, 2) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + G * Power(Gc, 2) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start);

DDc7 = −1. * CCc * DD * G + CCc * Power(G, 2) − 1. * CC * DD * Gc + 2. * DD * G * Gc − 1. * Power(G, 2) * Gc + CC * DD * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * G * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CC * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + CC * (AcFinal * (1 − t) + AcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) − 1. * DD * (AcFinal * (1 − t) + AcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) − 1. * DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + CC * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * (P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DD * (P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * DD * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 + CC * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 − 1. * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 + CC * DD * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 −

1. ∗ DD ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CC ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DD ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ DD ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ DD ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ DD ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ CCc ∗ Power(G, 2) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ DD ∗ G ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗

Tc52Start) - 1. * CC * Gc * (P2Final * (1 - t) + gamma * P2Start * t)
* ((1 - t) * Tc52Final + gamma * t * Tc52Start) + DD * Gc * (P2Final *
(1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) + CC * DD * Gc * T32 * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) - 1. * DD * G * Gc * T32 * ((1 - t) * Tc52Final + gamma * t
* Tc52Start) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t)
* T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + G * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final +
gamma * t * Tc52Start);
G7 = -1. * CCc * DD * DDc + 2 * CCc * DDc * G + 2. * DD * DDc * Gc -
2. * DDc * G * Gc - 1. * DD * Power(Gc, 2) - 1. * CC * DDc * (AcFinal
* (1 - t) + AcStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) - 1. * CCc * DDc * (AFinal * (1 - t) + AStart *
gamma * t) + CCc * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc
* (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) - 1. * Gc * (AFinal * (1 - t) + AStart * gamma *
t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DDc * (AcFinal * (1 -
t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 -
t) + BStart * gamma * t) - 1. * CCc * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) + Gc * (BcFinal
* (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma *
P2Start * t) - 1. * CCc * (AFinal * (1 - t) + AStart * gamma * t) *
(Pc2Final * (1 - t) + gamma * Pc2Start * t) + DDc * (AFinal * (1 - t)
+ AStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) -
1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 -
t) + gamma * Pc2Start * t) + Gc * (BFinal * (1 - t) + BStart * gamma *

t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + CCc * (P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * Gc * (P2Final * (1 − t) + gamma * P2Start * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 − 1. * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T32 − 1. * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 + Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T32 − 1. * DD * DDc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + DD * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 − 1. * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 2. * CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + 2 * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CCc * DD * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CC * DDc *

(Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + DD * DDc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + 2 * CCc * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CC * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 2. * G * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc32 − 1. * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc32 − 1. * CCc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc32 + Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc32 − 1. * CCc * DDc * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 + DDc * Gc * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 + CCc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 − 1. * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 + CCc * DD * DDc * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 2. * CCc * DDc * G * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 1. * DD * DDc * Gc * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + 2 * DDc * G * Gc * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 1. * CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + 2 * CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 − 2. * G * Gc * (BcFinal * (1 − t) + BcStart * gamma

```
* t) * ((1 - t) * T52Final + gamma * t * T52Start) * Tc32 - 1. * CCc *
Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) + DDc * Gc * (AFinal * (1 - t) + AStart * gamma
* t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * DDc * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc52Final + gamma
* t * Tc52Start) + Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma *
t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CCc * Gc *
(P2Final * (1 - t) + gamma * P2Start * t) * ((1 - t) * Tc52Final +
gamma * t * Tc52Start) - 1. * Power(Gc, 2) * (P2Final * (1 - t) +
gamma * P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) -
1. * DD * DDc * Gc * T32 * ((1 - t) * Tc52Final + gamma * t *
Tc52Start) + DD * Power(Gc, 2) * T32 * ((1 - t) * Tc52Final + gamma *
t * Tc52Start) + DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) *
Tc52Final + gamma * t * Tc52Start);
Gc7 = -1. * CC * DD * DDc + 2. * DD * DDc * G - 1. * DDc * Power(G, 2)
+ 2 * CC * DD * Gc - 2. * DD * G * Gc - 1. * CC * DD * (AcFinal * (1 -
t) + AcStart * gamma * t) + CC * G * (AcFinal * (1 - t) + AcStart *
gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) +
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD * (AFinal * (1
- t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t)
- 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 -
t) + BcStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart * gamma
* t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * (AcFinal *
(1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma *
t) - 1. * CC * (AcFinal * (1 - t) + AcStart * gamma * t) * (P2Final *
```

(1 − t) + gamma ∗ P2Start ∗ t) + DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + CC ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 + G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ CC ∗ DD ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + DD ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + CC ∗ G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗

T52Start) − 1. ∗ Power(G, 2) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + DDc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 2. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗

Tc52Start) + 2 ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CCc ∗ DD ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CCc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + 2 ∗ CC ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 2. ∗ G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ DD ∗ DDc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DD ∗ DDc ∗ G ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 2. ∗ CC ∗ DD ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + 2 ∗ DD ∗ G ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + 2 ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 2. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

T327 = −1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) −

1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + CC * DD * DDc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * DD * DDc * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * DD * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DD * G * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + CC * DD * DDc * Gc * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * DD * DDc * G * Gc * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * CC * DD * Power(Gc, 2) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + DD * G * Power(Gc, 2) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start);
Tc327 = −1. * CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) + DD * DDc *

Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ CCc ∗ DDc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + DDc ∗ G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start);

```
CC8 = −1. ∗ DD ∗ DDc ∗ Gc + DD ∗ Power(Gc, 2) + DD ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗
gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗
DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1.
∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t)
+ gamma ∗ P3Start ∗ t) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t)
∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ Gc ∗ (BFinal ∗ (1
− t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗
t) − 1. ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final
∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + Gc ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ DD ∗
DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + DD ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 + DDc ∗ (AcFinal ∗ (1
− t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t)
∗ T33 − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal
∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 + DD ∗ DDc ∗ (Pc3Final ∗ (1 − t)
+ gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ DD ∗ Gc ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma
∗ Pc3Start ∗ t) ∗ T33 + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma
∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
```

gamma * t * T53Start) − 1. * DDc * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) + G * Gc * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) + DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DDc * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + Power(Gc, 2) * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DD * DDc * Gc * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * Power(Gc, 2) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start);

CCc8 = −1. * DD * DDc * G + DDc * Power(G, 2) + DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t)

− 1. ∗ DD ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1
− t) + gamma ∗ Pc3Start ∗ t) + G ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + DD ∗ G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) − 1. ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
1. ∗ DD ∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) + Power(G, 2) ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + DDc ∗
G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + DD ∗ (AFinal ∗ (1
− t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t)
∗ Tc33 − 1. ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + DD ∗ DDc ∗ (P3Final ∗ (1 − t)
+ gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ (P3Final ∗ (1 − t) +
gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + G ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma
∗ P3Start ∗ t) ∗ Tc33 + DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t
∗ T53Start) ∗ Tc33 − 1. ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗
Tc33 + Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + DD ∗ Gc ∗ (AFinal ∗ (1
− t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗
Tc53Start) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ Gc ∗

(P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start);
DD8 = −1. * CCc * DDc * G − 1. * CC * DDc * Gc + 2. * DDc * G * Gc + CC * Power(Gc, 2) − 1. * G * Power(Gc, 2) + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) + CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * CCc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + CCc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + CCc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CCc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + DDc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 − 1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + CC * DDc * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * DDc * G * (Pc3Final * (1 − t) + gamma *

Pc3Start ∗ t) ∗ T33 − 1. ∗ CC ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + G ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DDc ∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + CCc ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final

```
+ gamma * t * Tc53Start) - 1. * DDc * Gc * (AFinal * (1 - t) + AStart
* gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. *
CCc * Gc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) + DDc * Gc * (P3Final * (1 - t) +
gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) +
CC * DDc * Gc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) -
1. * DDc * G * Gc * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 1. * CC * Power(Gc, 2) * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start) + G * Power(Gc, 2) * T33 * ((1 - t) * Tc53Final
+ gamma * t * Tc53Start);
DDc8 = -1. * CCc * DD * G + CCc * Power(G, 2) - 1. * CC * DD * Gc + 2.
* DD * G * Gc - 1. * Power(G, 2) * Gc + CC * DD * (AcFinal * (1 - t) +
AcStart * gamma * t) - 1. * CC * G * (AcFinal * (1 - t) + AcStart *
gamma * t) + CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CC * (AcFinal
* (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) + G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 -
t) + BStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart * gamma *
t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * DD * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start
* t) - 1. * DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc3Final *
(1 - t) + gamma * Pc3Start * t) + G * (AFinal * (1 - t) + AStart *
gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + CC *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma
* Pc3Start * t) - 1. * G * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) - 1. * CC * (P3Final * (1
- t) + gamma * P3Start * t) * (Pc3Final * (1 - t) + gamma * Pc3Start *
```

t) + DD * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DD * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + CC * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 − 1. * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 + CC * DD * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * DD * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * CC * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 + G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 + CC * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) − 1. * DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) − 1. * CC * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) + DD * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) − 1. * CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 + CCc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 + DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 + CCc * DD * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 − 1. * CCc * G * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 − 1. * DD * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + CCc * DD * G * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * CCc * Power(G, 2) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * DD * G * Gc * ((1 − t) * T53Final + gamma * t

* T53Start) * Tc33 + Power(G, 2) * Gc * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DD * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * DD * Gc * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * G * Gc * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start);

G8 = −1. * CCc * DD * DDc + 2 * CCc * DDc * G + 2. * DD * DDc * Gc − 2. * DDc * G * Gc − 1. * DD * Power(Gc, 2) − 1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) + CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * Gc * (AFinal * (1 − t) + AStart * gamma * t) + CCc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * CCc * (BcFinal * (1 − t) + BcStart * gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + CCc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) − 1. ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ DDc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + DD ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1

− t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 2. ∗ CCc ∗ G ∗ (BcFinal ∗ (1
− t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + 2 ∗ G ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) − 1. ∗ CCc ∗ DD ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗
t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CC ∗ DDc ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) + DD ∗ DDc ∗ (Pc3Final ∗ (1 − t) + gamma ∗
Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + 2 ∗ CCc
∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) + CC ∗ Gc ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
2. ∗ G ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) + CCc ∗ DDc ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗
gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + Gc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart
∗ gamma ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ Tc33 + DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start
∗ t) ∗ Tc33 + CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
(P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ Gc ∗ (BcFinal
∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗
P3Start ∗ t) ∗ Tc33 + CCc ∗ DD ∗ DDc ∗ ((1 − t) ∗ T53Final + gamma ∗ t
∗ T53Start) ∗ Tc33 − 2. ∗ CCc ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma
∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ DD ∗ DDc ∗ Gc ∗ ((1 − t) ∗ T53Final +

gamma ∗ t ∗ T53Start) ∗ Tc33 + 2 ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + 2 ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 2. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CCc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ Power(Gc, 2) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DD ∗ Power(Gc, 2) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);
Gc8 = −1. ∗ CC ∗ DD ∗ DDc + 2. ∗ DD ∗ DDc ∗ G − 1. ∗ DDc ∗ Power(G, 2) + 2 ∗ CC ∗ DD ∗ Gc − 2. ∗ DD ∗ G ∗ Gc − 1. ∗ CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) +

CCc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * CC * (AcFinal * (1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) + DD * (AcFinal * (1 - t) + AcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) + G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) - 1. * CC * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + CC * (P3Final * (1 - t) + gamma * P3Start * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) - 1. * G * (P3Final * (1 - t) + gamma * P3Start * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + CC * DD * (AcFinal * (1 - t) + AcStart * gamma * t) * T33 - 1. * DD * G * (AcFinal * (1 - t) + AcStart * gamma * t) * T33 - 1. * CC * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T33 + G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T33 - 1. * CC * DD * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + DD * G * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 + CC * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 - 1. * G * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) * T33 - 1. * CC * G * (AcFinal * (1 -

t) + AcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) + DD * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) − 1. * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) + Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) + CC * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) − 1. * Power(G, 2) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * ((1 − t) * T53Final + gamma * t * T53Start) + DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 − 1. * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc33 − 1. * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc33 + G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc33 − 1. * DD * DDc * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + DDc * G * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) * Tc33 − 1. * DD * DDc * G * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + DDc * Power(G, 2) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CCc * G *

(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc53Final + gamma
* t * Tc53Start) + DDc * G * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * DDc * (BFinal *
(1 - t) + BStart * gamma * t) * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 1. * DDc * G * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) - 2. * CC * Gc * (BFinal
* (1 - t) + BStart * gamma * t) * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) + 2 * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * CCc * DD *
(P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final +
gamma * t * Tc53Start) - 1. * CC * DDc * (P3Final * (1 - t) + gamma *
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + DD *
DDc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final
+ gamma * t * Tc53Start) + CCc * G * (P3Final * (1 - t) + gamma *
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + 2 * CC
* Gc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 2. * G * Gc * (P3Final * (1 - t)
+ gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start)
+ CC * DD * DDc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start)
- 1. * DD * DDc * G * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 2. * CC * DD * Gc * T33 * ((1 - t) * Tc53Final + gamma *
t * Tc53Start) + 2 * DD * G * Gc * T33 * ((1 - t) * Tc53Final + gamma
* t * Tc53Start) - 1. * CC * DDc * (BFinal * (1 - t) + BStart * gamma
* t) * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + DDc * G *
(BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start) + 2 * CC * Gc * (BFinal * (1 - t) + BStart *
gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 2.

```
* G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) *
Tc53Final + gamma * t * Tc53Start);
T338 = -1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t)
+ DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * DD *
Gc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * DD * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * DDc * (AcFinal * (1 -
t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) -
1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal *
(1 - t) + BStart * gamma * t) - 1. * CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) + G *
Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) +
BStart * gamma * t) + CC * DD * DDc * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) - 1. * DD * DDc * G * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) - 1. * CC * DD * Gc * (Pc3Final * (1 - t) + gamma *
Pc3Start * t) + DD * G * Gc * (Pc3Final * (1 - t) + gamma * Pc3Start *
t) - 1. * CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc3Final * (1 - t) + gamma * Pc3Start * t) + DDc * G * (BFinal * (1 -
t) + BStart * gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t)
+ CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc3Final * (1 -
t) + gamma * Pc3Start * t) - 1. * G * Gc * (BFinal * (1 - t) + BStart
* gamma * t) * (Pc3Final * (1 - t) + gamma * Pc3Start * t) + CC * DD *
DDc * Gc * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * DD *
DDc * G * Gc * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * CC
* DD * Power(Gc, 2) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) +
DD * G * Power(Gc, 2) * ((1 - t) * Tc53Final + gamma * t * Tc53Start)
- 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 -
t) * Tc53Final + gamma * t * Tc53Start) + DDc * G * Gc * (BFinal * (1
```

− t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);

Tc338 = −1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CCc ∗ DD ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ CCc ∗ DDc ∗ G ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + DDc ∗ G ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1

− t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start);

CC9 = −1. ∗ DD ∗ DDc ∗ Gc + DD ∗ Power(Gc, 2) + DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 + DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 + DD ∗ DDc ∗ (Pc4Final ∗ (1 − t)

+ gamma * Pc4Start * t) * T34 - 1. * DD * Gc * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) * T34 - 1. * DDc * (BFinal * (1 - t) + BStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 + Gc *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) + gamma
* Pc4Start * t) * T34 + DDc * G * (AcFinal * (1 - t) + AcStart * gamma
* t) * ((1 - t) * T54Final + gamma * t * T54Start) - 1. * G * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) - 1. * DDc * G * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * ((1 - t) * T54Final + gamma * t * T54Start) + G * Gc *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) * T54Final +
gamma * t * T54Start) + DDc * Gc * (BFinal * (1 - t) + BStart * gamma
* t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final +
gamma * t * Tc54Start) - 1. * DDc * Gc * (P4Final * (1 - t) + gamma *
P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) +
Power(Gc, 2) * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) + DD * DDc * Gc * T34 * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) - 1. * DD * Power(Gc, 2) * T34 *
((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * DDc * Gc *
(BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final +
gamma * t * Tc54Start) + Power(Gc, 2) * (BFinal * (1 - t) + BStart *
gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start);
CCc9 = -1. * DD * DDc * G + DDc * Power(G, 2) + DD * DDc * (AFinal *
(1 - t) + AStart * gamma * t) - 1. * DDc * G * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * DD * Gc * (AFinal * (1 - t) + AStart *
gamma * t) + G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +

```
BcStart * gamma * t) + G * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) + DD * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1.
* G * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) +
gamma * P4Start * t) + DD * (AFinal * (1 - t) + AStart * gamma * t) *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. * G * (AFinal * (1 -
t) + AStart * gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t)
- 1. * DD * (P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final * (1
- t) + gamma * Pc4Start * t) + G * (P4Final * (1 - t) + gamma *
P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + DD * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) - 1. * Power(G, 2) * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) -
1. * DD * G * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) *
T54Final + gamma * t * T54Start) + Power(G, 2) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) * ((1 - t) * T54Final + gamma * t * T54Start) -
1. * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) * Tc34 + DDc *
G * (AFinal * (1 - t) + AStart * gamma * t) * Tc34 + DD * (AFinal * (1
- t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t)
* Tc34 - 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal *
(1 - t) + BcStart * gamma * t) * Tc34 + DD * DDc * (P4Final * (1 - t)
+ gamma * P4Start * t) * Tc34 - 1. * DDc * G * (P4Final * (1 - t) +
gamma * P4Start * t) * Tc34 - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) * Tc34 + G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma
* P4Start * t) * Tc34 + DD * DDc * G * ((1 - t) * T54Final + gamma * t
* T54Start) * Tc34 - 1. * DDc * Power(G, 2) * ((1 - t) * T54Final +
```

gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + G ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

DD9 = −1. ∗ CCc ∗ DDc ∗ G − 1. ∗ CC ∗ DDc ∗ Gc + 2. ∗ DDc ∗ G ∗ Gc + CC ∗ Power(Gc, 2) − 1. ∗ G ∗ Power(Gc, 2) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CCc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) +

gamma * Pc4Start * t) + DDc * (P4Final * (1 - t) + gamma * P4Start *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. * CC * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) * T34 + DDc * G * (AcFinal *
(1 - t) + AcStart * gamma * t) * T34 + CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 - 1. * G * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 + CC * DDc * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * T34 - 1. * DDc * G * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * T34 - 1. * CC * Gc * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * T34 + G * Gc * (Pc4Final * (1 - t) + gamma * Pc4Start
* t) * T34 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t)
* ((1 - t) * T54Final + gamma * t * T54Start) + G * Gc * (AcFinal * (1
- t) + AcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) + CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1
- t) * T54Final + gamma * t * T54Start) - 1. * G * Gc * (BcFinal * (1
- t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) - 1. * CCc * G * (Pc4Final * (1 - t) + gamma * Pc4Start * t)
* ((1 - t) * T54Final + gamma * t * T54Start) + DDc * G * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) * ((1 - t) * T54Final + gamma * t *
T54Start) - 1. * CCc * DDc * (AFinal * (1 - t) + AStart * gamma * t) *
Tc34 + DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) * Tc34 + CCc
* (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc34 - 1. * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) * Tc34 + CCc *
DDc * (P4Final * (1 - t) + gamma * P4Start * t) * Tc34 - 1. * DDc * Gc
* (P4Final * (1 - t) + gamma * P4Start * t) * Tc34 - 1. * CCc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma
* P4Start * t) * Tc34 + Gc * (BcFinal * (1 - t) + BcStart * gamma * t)

```
* (P4Final * (1 - t) + gamma * P4Start * t) * Tc34 + CCc * DDc * G *
((1 - t) * T54Final + gamma * t * T54Start) * Tc34 - 1. * DDc * G * Gc
* ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 - 1. * CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) * Tc34 + G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + CCc
* Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final
+ gamma * t * Tc54Start) - 1. * DDc * Gc * (AFinal * (1 - t) + AStart
* gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. *
CCc * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) + DDc * Gc * (P4Final * (1 - t) +
gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) +
CC * DDc * Gc * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) -
1. * DDc * G * Gc * T34 * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) - 1. * CC * Power(Gc, 2) * T34 * ((1 - t) * Tc54Final +
gamma * t * Tc54Start) + G * Power(Gc, 2) * T34 * ((1 - t) * Tc54Final
+ gamma * t * Tc54Start);
DDc9 = -1. * CCc * DD * G + CCc * Power(G, 2) - 1. * CC * DD * Gc + 2.
* DD * G * Gc - 1. * Power(G, 2) * Gc + CC * DD * (AcFinal * (1 - t) +
AcStart * gamma * t) - 1. * CC * G * (AcFinal * (1 - t) + AcStart *
gamma * t) + CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CC * (AcFinal
* (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) + G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 -
t) + BStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart * gamma *
t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * DD * (AcFinal *
(1 - t) + AcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start
```

* t) - 1. * DD * (AFinal * (1 - t) + AStart * gamma * t) * (Pc4Final *
(1 - t) + gamma * Pc4Start * t) + G * (AFinal * (1 - t) + AStart *
gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + CC *
(BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) + gamma
* Pc4Start * t) - 1. * G * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc4Final * (1 - t) + gamma * Pc4Start * t) - 1. * CC * (P4Final * (1
- t) + gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start *
t) + DD * (P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final * (1 -
t) + gamma * Pc4Start * t) - 1. * CC * DD * (AcFinal * (1 - t) +
AcStart * gamma * t) * T34 + DD * G * (AcFinal * (1 - t) + AcStart *
gamma * t) * T34 + CC * (AcFinal * (1 - t) + AcStart * gamma * t) *
(BFinal * (1 - t) + BStart * gamma * t) * T34 - 1. * G * (AcFinal * (1
- t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t)
* T34 + CC * DD * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 -
1. * DD * G * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 - 1. *
CC * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) +
gamma * Pc4Start * t) * T34 + G * (BFinal * (1 - t) + BStart * gamma *
t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 + CC * G *
(AcFinal * (1 - t) + AcStart * gamma * t) * ((1 - t) * T54Final +
gamma * t * T54Start) - 1. * DD * G * (AcFinal * (1 - t) + AcStart *
gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) - 1. * CC * G
* (Pc4Final * (1 - t) + gamma * Pc4Start * t) * ((1 - t) * T54Final +
gamma * t * T54Start) + DD * G * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) * ((1 - t) * T54Final + gamma * t * T54Start) - 1. * CCc
* DD * (AFinal * (1 - t) + AStart * gamma * t) * Tc34 + CCc * G *
(AFinal * (1 - t) + AStart * gamma * t) * Tc34 + DD * Gc * (AFinal *
(1 - t) + AStart * gamma * t) * Tc34 - 1. * G * Gc * (AFinal * (1 - t)

+ AStart ∗ gamma ∗ t) ∗ Tc34 + CCc ∗ DD ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ CCc ∗ G ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ DD ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + G ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + CCc ∗ DD ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ Power(G, 2) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ DD ∗ G ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DD ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CC ∗ DD ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ G ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

G9 = −1. ∗ CCc ∗ DD ∗ DDc + 2 ∗ CCc ∗ DDc ∗ G + 2. ∗ DD ∗ DDc ∗ Gc − 2. ∗ DDc ∗ G ∗ Gc − 1. ∗ DD ∗ Power(Gc, 2) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗

gamma ∗ t) + CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + CCc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ DDc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + DD ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗

T34 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 2. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + 2 ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CCc ∗ DD ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CC ∗ DDc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ DDc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + 2 ∗ CCc ∗ G ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + CC ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 2. ∗ G ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc34 + Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ CCc ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗

P4Start ∗ t) ∗ Tc34 + DDc ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + CCc ∗ DD ∗ DDc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 2. ∗ CCc ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ DD ∗ DDc ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + 2 ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + 2 ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 2. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + CCc ∗ Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ Power(Gc, 2) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DD ∗ Power(Gc, 2) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

Gc9 = −1. ∗ CC ∗ DD ∗ DDc + 2. ∗ DD ∗ DDc ∗ G − 1. ∗ DDc ∗ Power(G, 2) + 2 ∗ CC ∗ DD ∗ Gc − 2. ∗ DD ∗ G ∗ Gc − 1. ∗ CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) + DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) + CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + CC ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ G ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart

∗ gamma ∗ t) ∗ T34 + G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ CC ∗ DD ∗
(Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + DD ∗ G ∗ (Pc4Final
∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + CC ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗
T34 − 1. ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ CC ∗ G ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) + DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ G ∗ (BcFinal ∗ (1 −
t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) + Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + CC ∗ G ∗ (Pc4Final ∗ (1
− t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) − 1. ∗ Power(G, 2) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start
∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ DDc ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ G ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DD ∗ (AFinal ∗
(1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗
t) ∗ Tc34 + G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DD ∗ DDc ∗ (P4Final ∗ (1
− t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + DDc ∗ G ∗ (P4Final ∗ (1 − t) +
gamma ∗ P4Start ∗ t) ∗ Tc34 + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ G
∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) +
gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final
+ gamma ∗ t ∗ T54Start) ∗ Tc34 + DDc ∗ Power(G, 2) ∗ ((1 − t) ∗

T54Final + gamma * t * T54Start) * Tc34 + DD * G * (BcFinal * (1 - t)
+ BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) *
Tc34 - 1. * Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + CCc * DD *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma
* t * Tc54Start) - 1. * DD * DDc * (AFinal * (1 - t) + AStart * gamma
* t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CCc * G *
(AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma
* t * Tc54Start) + DDc * G * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc54Final + gamma * t * Tc54Start) + CC * DDc * (BFinal *
(1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) - 1. * DDc * G * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc54Final + gamma * t * Tc54Start) - 2. * CC * Gc * (BFinal
* (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) + 2 * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CCc * DD *
(P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final +
gamma * t * Tc54Start) - 1. * CC * DDc * (P4Final * (1 - t) + gamma *
P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DD *
DDc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final
+ gamma * t * Tc54Start) + CCc * G * (P4Final * (1 - t) + gamma *
P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + 2 * CC
* Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) - 2. * G * Gc * (P4Final * (1 - t)
+ gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start)
+ CC * DD * DDc * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start)
- 1. * DD * DDc * G * T34 * ((1 - t) * Tc54Final + gamma * t *

Tc54Start) − 2. ∗ CC ∗ DD ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + 2 ∗ DD ∗ G ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + 2 ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 2. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);

T349 = −1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DD ∗ DDc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ G ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CC ∗ DD ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DD ∗ G ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart

```
* gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + CC * DD *
DDc * Gc * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * DD *
DDc * G * Gc * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC
* DD * Power(Gc, 2) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) +
DD * G * Power(Gc, 2) * ((1 - t) * Tc54Final + gamma * t * Tc54Start)
- 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 -
t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc * (BFinal * (1
- t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) + CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma *
t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * G *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start);
Tc349 = -1. * CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t)
+ CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD * DDc *
Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * DDc * G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) + CCc * DD * (AFinal * (1 - t)
+ AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1.
* CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 -
t) + BcStart * gamma * t) - 1. * DD * Gc * (AFinal * (1 - t) + AStart
* gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + G * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart
* gamma * t) + CCc * DD * DDc * (P4Final * (1 - t) + gamma * P4Start *
t) - 1. * CCc * DDc * G * (P4Final * (1 - t) + gamma * P4Start * t) -
1. * DD * DDc * Gc * (P4Final * (1 - t) + gamma * P4Start * t) + DDc *
G * Gc * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * CCc * DD *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma
* P4Start * t) + CCc * G * (BcFinal * (1 - t) + BcStart * gamma * t) *
```

(P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start);
CC10 = −1. ∗ DD ∗ DDc ∗ Gc + DD ∗ Power(Gc, 2) + DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final

* (1 − t) + gamma * Pc5Start * t) + Gc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 − 1. * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T35 + DD * DDc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DD * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) + G * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) + DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DDc * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + Power(Gc, 2) * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + DD * DDc * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DD * Power(Gc, 2) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final +

gamma * t * Tc55Start) + Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start);
CCc10 = −1. * DD * DDc * G + DDc * Power(G, 2) + DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) + DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * DD * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + G * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * DD * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) + Power(G, 2) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + DDc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc35 − 1. * G * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *

```
(1 - t) + BcStart * gamma * t) * Tc35 + DD * DDc * (P5Final * (1 - t)
+ gamma * P5Start * t) * Tc35 - 1. * DDc * G * (P5Final * (1 - t) +
gamma * P5Start * t) * Tc35 - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + G *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma
* P5Start * t) * Tc35 + DD * DDc * G * ((1 - t) * T55Final + gamma * t
* T55Start) * Tc35 - 1. * DDc * Power(G, 2) * ((1 - t) * T55Final +
gamma * t * T55Start) * Tc35 - 1. * DD * G * (BcFinal * (1 - t) +
BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) *
Tc35 + Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 -
t) * T55Final + gamma * t * T55Start) * Tc35 + DD * Gc * (AFinal * (1
- t) + AStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) - 1. * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) *
((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * DD * Gc *
(P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) * Tc55Final +
gamma * t * Tc55Start) + G * Gc * (P5Final * (1 - t) + gamma * P5Start
* t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start);
DD10 = -1. * CCc * DDc * G - 1. * CC * DDc * Gc + 2. * DDc * G * Gc +
CC * Power(Gc, 2) - 1. * G * Power(Gc, 2) + CC * DDc * (AcFinal * (1 -
t) + AcStart * gamma * t) - 1. * CC * Gc * (AcFinal * (1 - t) +
AcStart * gamma * t) + CCc * DDc * (AFinal * (1 - t) + AStart * gamma
* t) - 1. * CCc * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + Gc * (AFinal * (1 - t) + AStart * gamma * t) *
(BcFinal * (1 - t) + BcStart * gamma * t) - 1. * DDc * (AcFinal * (1 -
t) + AcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t)
+ Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (P5Final * (1 - t)
```

+ gamma * P5Start * t) + CCc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) − 1. * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P5Final * (1 − t) + gamma * P5Start * t) + CCc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CCc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) + DDc * (P5Final * (1 − t) + gamma * P5Start * t) * (Pc5Final * (1 − t) + gamma * Pc5Start * t) − 1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 − 1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T35 + CC * DDc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * CC * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 + G * Gc * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * T35 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) + CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * CCc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) + DDc * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * CCc * DDc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + CCc

```
* (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) * Tc35 - 1. * Gc * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) * Tc35 + CCc *
DDc * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * DDc * Gc
* (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 - 1. * CCc *
(BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma
* P5Start * t) * Tc35 + Gc * (BcFinal * (1 - t) + BcStart * gamma * t)
* (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + CCc * DDc * G *
((1 - t) * T55Final + gamma * t * T55Start) * Tc35 - 1. * DDc * G * Gc
* ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 - 1. * CCc * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start) * Tc35 + G * Gc * (BcFinal * (1 - t) + BcStart *
gamma * t) * ((1 - t) * T55Final + gamma * t * T55Start) * Tc35 + CCc
* Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc55Final
+ gamma * t * Tc55Start) - 1. * DDc * Gc * (AFinal * (1 - t) + AStart
* gamma * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. *
CCc * Gc * (P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + DDc * Gc * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) +
CC * DDc * Gc * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) -
1. * DDc * G * Gc * T35 * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) - 1. * CC * Power(Gc, 2) * T35 * ((1 - t) * Tc55Final +
gamma * t * Tc55Start) + G * Power(Gc, 2) * T35 * ((1 - t) * Tc55Final
+ gamma * t * Tc55Start);
DDc10 = -1. * CCc * DD * G + CCc * Power(G, 2) - 1. * CC * DD * Gc +
2. * DD * G * Gc - 1. * Power(G, 2) * Gc + CC * DD * (AcFinal * (1 -
t) + AcStart * gamma * t) - 1. * CC * G * (AcFinal * (1 - t) + AcStart
```

∗ gamma ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CC ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + DD ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T35 + DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T35 + CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 + CC ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ DD ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) +

AcStart * gamma * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * CC * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) + DD * G * (Pc5Final * (1 − t) + gamma * Pc5Start * t) * ((1 − t) * T55Final + gamma * t * T55Start) − 1. * CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + CCc * G * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc35 + CCc * DD * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * CCc * G * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 − 1. * DD * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 + G * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * Tc35 + CCc * DD * G * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 − 1. * CCc * Power(G, 2) * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 − 1. * DD * G * Gc * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 + Power(G, 2) * Gc * ((1 − t) * T55Final + gamma * t * T55Start) * Tc35 − 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + DD * Gc * (P5Final * (1 − t) + gamma * P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC * DD * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DD * G * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC * Gc * (BFinal * (1 −

t) + BStart * gamma * t) * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start);

G10 = -1. * CCc * DD * DDc + 2 * CCc * DDc * G + 2. * DD * DDc * Gc - 2. * DDc * G * Gc - 1. * DD * Power(Gc, 2) - 1. * CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CCc * DDc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * CCc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) + Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) - 1. * CCc * (AFinal * (1 - t) + AStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + DDc * (AFinal * (1 - t) + AStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + CCc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * Gc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 - 1. * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 - 1. * DDc * (AcFinal * (1 - t) + AcStart *

gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 + Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart
∗ gamma ∗ t) ∗ T35 − 1. ∗ DD ∗ DDc ∗ (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) ∗ T35 + DD ∗ Gc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start
∗ t) ∗ T35 + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final
∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t)
+ BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗
T35 + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t)
∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ Gc ∗ (AcFinal ∗ (1
− t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1
− t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 2. ∗ CCc ∗ G ∗ (BcFinal ∗ (1
− t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + 2 ∗ G ∗ Gc ∗ (BcFinal ∗
(1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) − 1. ∗ CCc ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗
t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CC ∗ DDc ∗
(Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final +
gamma ∗ t ∗ T55Start) + DD ∗ DDc ∗ (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + 2 ∗ CCc
∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗
T55Final + gamma ∗ t ∗ T55Start) + CC ∗ Gc ∗ (Pc5Final ∗ (1 − t) +
gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) −

2. ∗ G ∗ Gc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CCc ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 + Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DDc ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + CCc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + CCc ∗ DD ∗ DDc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 2. ∗ CCc ∗ DDc ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ Gc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + 2 ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + 2 ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 2. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ CCc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗

```
t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) + CCc * Gc *
(P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) * Tc55Final +
gamma * t * Tc55Start) - 1. * Power(Gc, 2) * (P5Final * (1 - t) +
gamma * P5Start * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) -
1. * DD * DDc * Gc * T35 * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) + DD * Power(Gc, 2) * T35 * ((1 - t) * Tc55Final + gamma *
t * Tc55Start) + DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * T35 * ((1 - t) *
Tc55Final + gamma * t * Tc55Start);
Gc10 = -1. * CC * DD * DDc + 2. * DD * DDc * G - 1. * DDc * Power(G,
2) + 2 * CC * DD * Gc - 2. * DD * G * Gc - 1. * CC * DD * (AcFinal *
(1 - t) + AcStart * gamma * t) + CC * G * (AcFinal * (1 - t) + AcStart
* gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t)
+ CCc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD * (AFinal *
(1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma *
t) - 1. * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1
- t) + BcStart * gamma * t) + CC * (AcFinal * (1 - t) + AcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * G *
(AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart
* gamma * t) - 1. * CC * (AcFinal * (1 - t) + AcStart * gamma * t) *
(P5Final * (1 - t) + gamma * P5Start * t) + DD * (AcFinal * (1 - t) +
AcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) - 1.
* DD * (BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1 - t)
+ gamma * P5Start * t) + G * (BcFinal * (1 - t) + BcStart * gamma * t)
* (P5Final * (1 - t) + gamma * P5Start * t) - 1. * CC * (BFinal * (1 -
t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t)
```

+ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + CC ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ G ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + CC ∗ DD ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ CC ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 + G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ CC ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + DD ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ CC ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CC ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ Power(G, 2) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 + G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗

(1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DDc ∗ G ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 2. ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 2 ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CCc ∗ DD ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DD ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final

+ gamma ∗ t ∗ Tc55Start) + CCc ∗ G ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 2 ∗ CC ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 2. ∗ G ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ DD ∗ DDc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗ DDc ∗ G ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 2. ∗ CC ∗ DD ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 2 ∗ DD ∗ G ∗ Gc ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 2 ∗ CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 2. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);

T3510 = −1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) − 1. ∗ DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DD ∗ DDc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ DD ∗ DDc ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗

```
Pc5Start * t) - 1. * CC * DD * Gc * (Pc5Final * (1 - t) + gamma *
Pc5Start * t) + DD * G * Gc * (Pc5Final * (1 - t) + gamma * Pc5Start *
t) - 1. * CC * DDc * (BFinal * (1 - t) + BStart * gamma * t) *
(Pc5Final * (1 - t) + gamma * Pc5Start * t) + DDc * G * (BFinal * (1 -
t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t)
+ CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final * (1 -
t) + gamma * Pc5Start * t) - 1. * G * Gc * (BFinal * (1 - t) + BStart
* gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + CC * DD *
DDc * Gc * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * DD *
DDc * G * Gc * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * CC
* DD * Power(Gc, 2) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) +
DD * G * Power(Gc, 2) * ((1 - t) * Tc55Final + gamma * t * Tc55Start)
- 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 -
t) * Tc55Final + gamma * t * Tc55Start) + DDc * G * Gc * (BFinal * (1
- t) + BStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t *
Tc55Start) + CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma *
t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start) - 1. * G *
Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start);
Tc3510 = -1. * CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma *
t) + CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) + DD *
DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * DDc * G * Gc
* (AFinal * (1 - t) + AStart * gamma * t) + CCc * DD * (AFinal * (1 -
t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) -
1. * CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1
- t) + BcStart * gamma * t) - 1. * DD * Gc * (AFinal * (1 - t) +
AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + G *
```

```
Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) +
BcStart * gamma * t) + CCc * DD * DDc * (P5Final * (1 - t) + gamma *
P5Start * t) - 1. * CCc * DDc * G * (P5Final * (1 - t) + gamma *
P5Start * t) - 1. * DD * DDc * Gc * (P5Final * (1 - t) + gamma *
P5Start * t) + DDc * G * Gc * (P5Final * (1 - t) + gamma * P5Start *
t) - 1. * CCc * DD * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P5Final * (1 - t) + gamma * P5Start * t) + CCc * G * (BcFinal * (1 -
t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t)
+ DD * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * (P5Final * (1
- t) + gamma * P5Start * t) - 1. * G * Gc * (BcFinal * (1 - t) +
BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t) + CCc
* DD * DDc * G * ((1 - t) * T55Final + gamma * t * T55Start) - 1. *
CCc * DDc * Power(G, 2) * ((1 - t) * T55Final + gamma * t * T55Start)
- 1. * DD * DDc * G * Gc * ((1 - t) * T55Final + gamma * t * T55Start)
+ DDc * Power(G, 2) * Gc * ((1 - t) * T55Final + gamma * t * T55Start)
- 1. * CCc * DD * G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1
- t) * T55Final + gamma * t * T55Start) + CCc * Power(G, 2) * (BcFinal
* (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final + gamma * t *
T55Start) + DD * G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) *
((1 - t) * T55Final + gamma * t * T55Start) - 1. * Power(G, 2) * Gc *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T55Final +
gamma * t * T55Start);
F11 = Hc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
(1 - t) + BcStart * gamma * t) * T31 - 1. * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 - 1. * Hc * T31 * ((1 - t) * Tc51Final +
gamma * t * Tc51Start);
Fc11 = H - 1. * (BFinal * (1 - t) + BStart * gamma * t) + (BFinal * (1
```

− t) + BStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ H ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31;
H11 = Fc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ Fc ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31;
Hc11 = F − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ F ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);
T3111 = F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ F ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) − 1. ∗ F ∗ Hc ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);
Tc3111 = Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ Fc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) − 1. ∗ Fc ∗ H ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗

T51Start) + H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start);

F12 = Hc − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) + (BcFinal * (1 − t) + BcStart * gamma * t) * T32 − 1. * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 − 1. * Hc * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start);

Fc12 = H − 1. * (BFinal * (1 − t) + BStart * gamma * t) + (BFinal * (1 − t) + BStart * gamma * t) * Tc32 − 1. * (P2Final * (1 − t) + gamma * P2Start * t) * Tc32 − 1. * H * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32;

H12 = Fc − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * Fc * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32 + (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) * Tc32;

Hc12 = F − 1. * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + (P2Final * (1 − t) + gamma * P2Start * t) * ((1 − t) * Tc52Final + gamma * t * Tc52Start) − 1. * F * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start) + (BFinal * (1 − t) + BStart * gamma * t) * T32 * ((1 − t) * Tc52Final + gamma * t * Tc52Start);

T3212 = F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * F * (Pc2Final * (1 − t) + gamma * Pc2Start * t) + (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) − 1. * F * Hc * ((1 − t) * Tc52Final + gamma * t *

```
Tc52Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc52Final + gamma * t * Tc52Start);
Tc3212 = Fc * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal
* (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) - 1. * Fc * (P2Final * (1 - t) + gamma * P2Start * t) + (BcFinal
* (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma *
P2Start * t) - 1. * Fc * H * ((1 - t) * T52Final + gamma * t *
T52Start) + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T52Final + gamma * t * T52Start);
F13 = Hc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
(1 - t) + BcStart * gamma * t) * T33 - 1. * (Pc3Final * (1 - t) +
gamma * Pc3Start * t) * T33 - 1. * Hc * T33 * ((1 - t) * Tc53Final +
gamma * t * Tc53Start);
Fc13 = H - 1. * (BFinal * (1 - t) + BStart * gamma * t) + (BFinal * (1
- t) + BStart * gamma * t) * Tc33 - 1. * (P3Final * (1 - t) + gamma *
P3Start * t) * Tc33 - 1. * H * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33;
H13 = Fc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T53Final + gamma * t * T53Start) + (Pc3Final * (1 - t) + gamma *
Pc3Start * t) * ((1 - t) * T53Final + gamma * t * T53Start) - 1. * Fc
* ((1 - t) * T53Final + gamma * t * T53Start) * Tc33 + (BcFinal * (1 -
t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t *
T53Start) * Tc33;
Hc13 = F - 1. * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) + (P3Final * (1 - t) + gamma *
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * F
* T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + (BFinal * (1 -
```

t) + BStart * gamma * t) * T33 * ((1 − t) * Tc53Final + gamma * t * Tc53Start);

T3313 = F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * F * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * F * Hc * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + Hc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start);

Tc3313 = Fc * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * Fc * (P3Final * (1 − t) + gamma * P3Start * t) + (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * Fc * H * ((1 − t) * T53Final + gamma * t * T53Start) + H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start);

F14 = Hc − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) + (BcFinal * (1 − t) + BcStart * gamma * t) * T34 − 1. * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * Hc * T34 * ((1 − t) * Tc54Final + gamma * t * Tc54Start);

Fc14 = H − 1. * (BFinal * (1 − t) + BStart * gamma * t) + (BFinal * (1 − t) + BStart * gamma * t) * Tc34 − 1. * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 − 1. * H * ((1 − t) * T54Final + gamma * t * T54Start) * Tc34;

H14 = Fc − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final + gamma * t * T54Start) + (Pc4Final * (1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t * T54Start) − 1. * Fc

```
* ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + (BcFinal * (1 -
t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t *
T54Start) * Tc34;
Hc14 = F - 1. * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start) + (P4Final * (1 - t) + gamma *
P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * F
* T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + (BFinal * (1 -
t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t *
Tc54Start);
T3414 = F * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * (BcFinal
* (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) - 1. * F * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + (BFinal
* (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) + gamma *
Pc4Start * t) - 1. * F * Hc * ((1 - t) * Tc54Final + gamma * t *
Tc54Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) *
Tc54Final + gamma * t * Tc54Start);
Tc3414 = Fc * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal
* (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma
* t) - 1. * Fc * (P4Final * (1 - t) + gamma * P4Start * t) + (BcFinal
* (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma *
P4Start * t) - 1. * Fc * H * ((1 - t) * T54Final + gamma * t *
T54Start) + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T54Final + gamma * t * T54Start);
F15 = Hc - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) + (BcFinal *
(1 - t) + BcStart * gamma * t) * T35 - 1. * (Pc5Final * (1 - t) +
gamma * Pc5Start * t) * T35 - 1. * Hc * T35 * ((1 - t) * Tc55Final +
gamma * t * Tc55Start);
```

```
Fc15 = H − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + (BFinal ∗ (1
− t) + BStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ Tc35 − 1. ∗ H ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) ∗ Tc35;
H15 = Fc − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
T55Final + gamma ∗ t ∗ T55Start) + (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ Fc
∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + (BcFinal ∗ (1 −
t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗
T55Start) ∗ Tc35;
Hc15 = F − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start) + (P5Final ∗ (1 − t) + gamma ∗
P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ F
∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + (BFinal ∗ (1 −
t) + BStart ∗ gamma ∗ t) ∗ T35 ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start);
T3515 = F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ (BcFinal
∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma
∗ t) − 1. ∗ F ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + (BFinal
∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗
Pc5Start ∗ t) − 1. ∗ F ∗ Hc ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗
Tc55Start) + Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
Tc55Final + gamma ∗ t ∗ Tc55Start);
Tc3515 = Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal
∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma
∗ t) − 1. ∗ Fc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + (BcFinal
∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗
```

```
P5Start * t) - 1. * Fc * H * ((1 - t) * T55Final + gamma * t *
T55Start) + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T55Final + gamma * t * T55Start);
T3116 = Tc31;
Tc3116 = T31;
T3217 = Tc32;
Tc3217 = T32;
T3318 = Tc33;
Tc3318 = T33;
T3419 = Tc34;
Tc3419 = T34;
T3520 = Tc35;
Tc3520 = T35;

// Populating J Matrix with the jacobian values.
complex_t *J = currentJMatWorkspace;
J[0] = 0;
J[1] = 0;
J[2] = DD1;
J[3] = DDc1;
J[4] = 0;
J[5] = 0;
J[6] = G1;
J[7] = Gc1;
J[8] = 0;
J[9] = 0;
J[10] = T311;
```

```
J[11] = Tc311;
J[12] = 0;
J[13] = 0;
J[14] = 0;
J[15] = 0;
J[16] = 0;
J[17] = 0;
J[18] = 0;
J[19] = 0;
J[20] = 0;
J[21] = 0;
J[22] = DD2;
J[23] = DDc2;
J[24] = 0;
J[25] = 0;
J[26] = G2;
J[27] = Gc2;
J[28] = 0;
J[29] = 0;
J[30] = 0;
J[31] = 0;
J[32] = T322;
J[33] = Tc322;
J[34] = 0;
J[35] = 0;
J[36] = 0;
J[37] = 0;
```

```
J[38] = 0;
J[39] = 0;
J[40] = 0;
J[41] = 0;
J[42] = DD3;
J[43] = DDc3;
J[44] = 0;
J[45] = 0;
J[46] = G3;
J[47] = Gc3;
J[48] = 0;
J[49] = 0;
J[50] = 0;
J[51] = 0;
J[52] = 0;
J[53] = 0;
J[54] = T333;
J[55] = Tc333;
J[56] = 0;
J[57] = 0;
J[58] = 0;
J[59] = 0;
J[60] = 0;
J[61] = 0;
J[62] = DD4;
J[63] = DDc4;
J[64] = 0;
```

```
J[65] = 0;
J[66] = G4;
J[67] = Gc4;
J[68] = 0;
J[69] = 0;
J[70] = 0;
J[71] = 0;
J[72] = 0;
J[73] = 0;
J[74] = 0;
J[75] = 0;
J[76] = T344;
J[77] = Tc344;
J[78] = 0;
J[79] = 0;
J[80] = 0;
J[81] = 0;
J[82] = DD5;
J[83] = DDc5;
J[84] = 0;
J[85] = 0;
J[86] = G5;
J[87] = Gc5;
J[88] = 0;
J[89] = 0;
J[90] = 0;
J[91] = 0;
```

```
J[92] = 0;
J[93] = 0;
J[94] = 0;
J[95] = 0;
J[96] = 0;
J[97] = 0;
J[98] = T355;
J[99] = Tc355;
J[100] = CC6;
J[101] = CCc6;
J[102] = DD6;
J[103] = DDc6;
J[104] = 0;
J[105] = 0;
J[106] = G6;
J[107] = Gc6;
J[108] = 0;
J[109] = 0;
J[110] = T316;
J[111] = Tc316;
J[112] = 0;
J[113] = 0;
J[114] = 0;
J[115] = 0;
J[116] = 0;
J[117] = 0;
J[118] = 0;
```

```
J[119] = 0;
J[120] = CC7;
J[121] = CCc7;
J[122] = DD7;
J[123] = DDc7;
J[124] = 0;
J[125] = 0;
J[126] = G7;
J[127] = Gc7;
J[128] = 0;
J[129] = 0;
J[130] = 0;
J[131] = 0;
J[132] = T327;
J[133] = Tc327;
J[134] = 0;
J[135] = 0;
J[136] = 0;
J[137] = 0;
J[138] = 0;
J[139] = 0;
J[140] = CC8;
J[141] = CCc8;
J[142] = DD8;
J[143] = DDc8;
J[144] = 0;
J[145] = 0;
```

```
J[146] = G8;
J[147] = Gc8;
J[148] = 0;
J[149] = 0;
J[150] = 0;
J[151] = 0;
J[152] = 0;
J[153] = 0;
J[154] = T338;
J[155] = Tc338;
J[156] = 0;
J[157] = 0;
J[158] = 0;
J[159] = 0;
J[160] = CC9;
J[161] = CCc9;
J[162] = DD9;
J[163] = DDc9;
J[164] = 0;
J[165] = 0;
J[166] = G9;
J[167] = Gc9;
J[168] = 0;
J[169] = 0;
J[170] = 0;
J[171] = 0;
J[172] = 0;
```

```
J[173] = 0;
J[174] = 0;
J[175] = 0;
J[176] = T349;
J[177] = Tc349;
J[178] = 0;
J[179] = 0;
J[180] = CC10;
J[181] = CCc10;
J[182] = DD10;
J[183] = DDc10;
J[184] = 0;
J[185] = 0;
J[186] = G10;
J[187] = Gc10;
J[188] = 0;
J[189] = 0;
J[190] = 0;
J[191] = 0;
J[192] = 0;
J[193] = 0;
J[194] = 0;
J[195] = 0;
J[196] = 0;
J[197] = 0;
J[198] = T3510;
J[199] = Tc3510;
```

```
J[200] = 0;
J[201] = 0;
J[202] = 0;
J[203] = 0;
J[204] = F11;
J[205] = Fc11;
J[206] = 0;
J[207] = 0;
J[208] = H11;
J[209] = Hc11;
J[210] = T3111;
J[211] = Tc3111;
J[212] = 0;
J[213] = 0;
J[214] = 0;
J[215] = 0;
J[216] = 0;
J[217] = 0;
J[218] = 0;
J[219] = 0;
J[220] = 0;
J[221] = 0;
J[222] = 0;
J[223] = 0;
J[224] = F12;
J[225] = Fc12;
J[226] = 0;
```

```
J[227] = 0;
J[228] = H12;
J[229] = Hc12;
J[230] = 0;
J[231] = 0;
J[232] = T3212;
J[233] = Tc3212;
J[234] = 0;
J[235] = 0;
J[236] = 0;
J[237] = 0;
J[238] = 0;
J[239] = 0;
J[240] = 0;
J[241] = 0;
J[242] = 0;
J[243] = 0;
J[244] = F13;
J[245] = Fc13;
J[246] = 0;
J[247] = 0;
J[248] = H13;
J[249] = Hc13;
J[250] = 0;
J[251] = 0;
J[252] = 0;
J[253] = 0;
```

```
J[254] = T3313;
J[255] = Tc3313;
J[256] = 0;
J[257] = 0;
J[258] = 0;
J[259] = 0;
J[260] = 0;
J[261] = 0;
J[262] = 0;
J[263] = 0;
J[264] = F14;
J[265] = Fc14;
J[266] = 0;
J[267] = 0;
J[268] = H14;
J[269] = Hc14;
J[270] = 0;
J[271] = 0;
J[272] = 0;
J[273] = 0;
J[274] = 0;
J[275] = 0;
J[276] = T3414;
J[277] = Tc3414;
J[278] = 0;
J[279] = 0;
J[280] = 0;
```

```
J[281] = 0;
J[282] = 0;
J[283] = 0;
J[284] = F15;
J[285] = Fc15;
J[286] = 0;
J[287] = 0;
J[288] = H15;
J[289] = Hc15;
J[290] = 0;
J[291] = 0;
J[292] = 0;
J[293] = 0;
J[294] = 0;
J[295] = 0;
J[296] = 0;
J[297] = 0;
J[298] = T3515;
J[299] = Tc3515;
J[300] = 0;
J[301] = 0;
J[302] = 0;
J[303] = 0;
J[304] = 0;
J[305] = 0;
J[306] = 0;
J[307] = 0;
```

```
J[308] = 0;
J[309] = 0;
J[310] = T3116;
J[311] = Tc3116;
J[312] = 0;
J[313] = 0;
J[314] = 0;
J[315] = 0;
J[316] = 0;
J[317] = 0;
J[318] = 0;
J[319] = 0;
J[320] = 0;
J[321] = 0;
J[322] = 0;
J[323] = 0;
J[324] = 0;
J[325] = 0;
J[326] = 0;
J[327] = 0;
J[328] = 0;
J[329] = 0;
J[330] = 0;
J[331] = 0;
J[332] = T3217;
J[333] = Tc3217;
J[334] = 0;
```

```
J[335] = 0;
J[336] = 0;
J[337] = 0;
J[338] = 0;
J[339] = 0;
J[340] = 0;
J[341] = 0;
J[342] = 0;
J[343] = 0;
J[344] = 0;
J[345] = 0;
J[346] = 0;
J[347] = 0;
J[348] = 0;
J[349] = 0;
J[350] = 0;
J[351] = 0;
J[352] = 0;
J[353] = 0;
J[354] = T3318;
J[355] = Tc3318;
J[356] = 0;
J[357] = 0;
J[358] = 0;
J[359] = 0;
J[360] = 0;
J[361] = 0;
```

```
J[362] = 0;
J[363] = 0;
J[364] = 0;
J[365] = 0;
J[366] = 0;
J[367] = 0;
J[368] = 0;
J[369] = 0;
J[370] = 0;
J[371] = 0;
J[372] = 0;
J[373] = 0;
J[374] = 0;
J[375] = 0;
J[376] = T3419;
J[377] = Tc3419;
J[378] = 0;
J[379] = 0;
J[380] = 0;
J[381] = 0;
J[382] = 0;
J[383] = 0;
J[384] = 0;
J[385] = 0;
J[386] = 0;
J[387] = 0;
J[388] = 0;
```

```
J[389] = 0;
J[390] = 0;
J[391] = 0;
J[392] = 0;
J[393] = 0;
J[394] = 0;
J[395] = 0;
J[396] = 0;
J[397] = 0;
J[398] = T3520;
J[399] = Tc3520;
```

B.2.6 Ht.cuh

```
#ifndef HT_CUH
#define HT_CUH
#include "Definitions.h"

#include "cuda_runtime.h"
#include "device_launch_parameters.h"

// Returns the value of the partial derivative of H with respect to t
 at point x at time t.
// effectively, dH/dt(t, x(t))
//
// *** NOTE: Actually returns -Ht, because we need to solve J(dx) =
-Ht for dx.
```

```
__device__ void Ht(double t, complex_t *x, complex_t *startParams,
complex_t *finalParams, complex_t *b, complex_t gamma, int n);

#endif
```

B.2.7 Ht.cu

```
#include "Ht.cuh"
#define Power pow
// Returns the value of the partial derivative of H with respect to t
at point x at time t.
// effectively, dH/dt(t, x(t))
//
// *** NOTE: Actually returns -Ht, because we need to solve J(dx) =
-Ht for dx.
__device__ void Ht(double t, complex_t *x, complex_t *startParams,
complex_t *finalParams, complex_t *b, complex_t gamma, int n)
{
complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start,
Pc2Start, Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start, Tc51Start,
Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1]; P3Start =
startParams[2]; P4Start = startParams[3]; P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6]; Pc3Start =
startParams[7]; Pc4Start = startParams[8]; Pc5Start = startParams[9];
```

```
T51Start = startParams[10]; T52Start = startParams[11]; T53Start =
startParams[12]; T54Start = startParams[13]; T55Start =
startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start; Tc53Start =
1.0 / T53Start; Tc54Start = 1.0 / T54Start; Tc55Start = 1.0 /
T55Start;
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start); Tc53Start =
conj(T53Start); Tc54Start = conj(T54Start); Tc55Start =
conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16]; BStart =
startParams[17]; BcStart = startParams[18];
complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final,
Pc2Final, Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final, Tc51Final,
Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;
P1Final = finalParams[0]; P2Final = finalParams[1]; P3Final =
finalParams[2]; P4Final = finalParams[3]; P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6]; Pc3Final =
finalParams[7]; Pc4Final = finalParams[8]; Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11]; T53Final =
finalParams[12]; T54Final = finalParams[13]; T55Final =
finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final; Tc53Final =
1.0 / T53Final; Tc54Final = 1.0 / T54Final; Tc55Final = 1.0 /
T55Final;
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final); Tc53Final =
```

```
conj(T53Final); Tc54Final = conj(T54Final); Tc55Final =
conj(T55Final);
AFinal = finalParams[15]; AcFinal = finalParams[16]; BFinal =
finalParams[17]; BcFinal = finalParams[18];
complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31, T32, Tc32,
T33, Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4], Fc = x[5], G =
x[6], Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13], T33 = x[14],
Tc33 = x[15], T34 = x[16], Tc34 = x[17], T35 = x[18], Tc35 = x[19];
complex_t Ht1, Ht2, Ht3, Ht4, Ht5, Ht6, Ht7, Ht8, Ht9, Ht10, Ht11,
Ht12, Ht13, Ht14, Ht15, Ht16, Ht17, Ht18, Ht19, Ht20;
Ht1 = 1. * DD * (BcFinal - 1. * BcStart * gamma) + 1. * DDc * (BFinal
- 1. * BStart * gamma) - 2. * (BFinal - 1. * BStart * gamma) *
(BcFinal - 1. * BcFinal * t + BcStart * gamma * t) + 1. * (P1Final -
1. * gamma * P1Start) * (BcFinal - 1. * BcFinal * t + BcStart * gamma
* t) - 2. * (BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal *
t + BStart * gamma * t) + 1. * (Pc1Final - 1. * gamma * Pc1Start) *
(BFinal - 1. * BFinal * t + BStart * gamma * t) - (Pc1Final - gamma *
Pc1Start) * (P1Final - P1Final * t + gamma * P1Start * t) + 1. *
(BcFinal - 1. * BcStart * gamma) * (P1Final - 1. * P1Final * t + gamma
* P1Start * t) - (P1Final - gamma * P1Start) * (Pc1Final - Pc1Final *
t + gamma * Pc1Start * t) + 1. * (BFinal - 1. * BStart * gamma) *
(Pc1Final - 1. * Pc1Final * t + gamma * Pc1Start * t) + DD * (-BcFinal
+ BcStart * gamma) * T31 + 1. * DD * (Pc1Final - 1. * gamma *
Pc1Start) * T31 + 1. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1.
* BcFinal * t + BcStart * gamma * t) * T31 - (Pc1Final - gamma *
```

Pc1Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − (BFinal − BStart ∗ gamma) ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + G ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − G ∗ (Pc1Final − gamma ∗ Pc1Start) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T51Final − 1. ∗ t ∗ T51Final + gamma ∗ t ∗ T51Start) + DDc ∗ (−BFinal + BStart ∗ gamma) ∗ Tc31 + 1. ∗ DDc ∗ (P1Final − 1. ∗ gamma ∗ P1Start) ∗ Tc31 − (P1Final − gamma ∗ P1Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ Tc31 − (BcFinal − BcStart ∗ gamma) ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 + 1. ∗ DDc ∗ G ∗ (T51Final − 1. ∗ gamma ∗ T51Start) ∗ Tc31 + G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 − G ∗ (BcFinal − BcStart ∗ gamma) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + 1. ∗ DD ∗ Gc ∗ T31 ∗ (Tc51Final − 1. ∗ gamma ∗ Tc51Start) − 1. ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + Gc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 ∗ (−Tc51Final + gamma ∗ Tc51Start) − Gc ∗ (P1Final − gamma ∗ P1Start) ∗ (Tc51Final − t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − (BFinal − BStart ∗ gamma) ∗ Gc ∗ T31 ∗ (Tc51Final − t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + 1. ∗ (BFinal − 1. ∗ BStart ∗

gamma) ∗ Gc ∗ (Tc51Final − 1. ∗ t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);

Ht2 = 1. ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) + 1. ∗ DDc ∗ (BFinal − 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc2Final − gamma ∗ Pc2Start) ∗ (P2Final − P2Final ∗ t + gamma ∗ P2Start ∗ t) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P2Final − 1. ∗ P2Final ∗ t + gamma ∗ P2Start ∗ t) − (P2Final − gamma ∗ P2Start) ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc2Final − 1. ∗ Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) + DD ∗ (−BcFinal + BcStart ∗ gamma) ∗ T32 + 1. ∗ DD ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ T32 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T32 − (Pc2Final − gamma ∗ Pc2Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T32 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T32 − (BFinal − BStart ∗ gamma) ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T52Final + gamma ∗ T52Start) + G ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) ∗ (−T52Final + gamma ∗ T52Start) − G ∗ (Pc2Final − gamma ∗ Pc2Start) ∗ (T52Final − t ∗ T52Final + gamma ∗ t ∗ T52Start) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T52Final − 1. ∗ t ∗ T52Final + gamma ∗ t ∗ T52Start) + DDc ∗ (−BFinal + BStart ∗ gamma) ∗ Tc32 + 1. ∗ DDc ∗ (P2Final − 1. ∗ gamma ∗

P2Start) * Tc32 − (P2Final − gamma * P2Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * (BFinal − 1. * BStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * (BcFinal − 1. * BcStart * gamma) * (BFinal − 1. * BFinal * t + BStart * gamma * t) * Tc32 − (BcFinal − BcStart * gamma) * (P2Final − P2Final * t + gamma * P2Start * t) * Tc32 + 1. * DDc * G * (T52Final − 1. * gamma * T52Start) * Tc32 + G * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T52Final + gamma * T52Start) * Tc32 − G * (BcFinal − BcStart * gamma) * (T52Final − t * T52Final + gamma * t * T52Start) * Tc32 + 1. * DD * Gc * T32 * (Tc52Final − 1. * gamma * Tc52Start) − 1. * Gc * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) + Gc * (P2Final − P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start) + Gc * (BFinal − BFinal * t + BStart * gamma * t) * T32 * (−Tc52Final + gamma * Tc52Start) − Gc * (P2Final − gamma * P2Start) * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) − (BFinal − BStart * gamma) * Gc * T32 * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) + 1. * (BFinal − 1. * BStart * gamma) * Gc * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start);
Ht3 = 1. * DD * (BcFinal − 1. * BcStart * gamma) + 1. * DDc * (BFinal − 1. * BStart * gamma) − 2. * (BFinal − 1. * BStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) + 1. * (P3Final − 1. * gamma * P3Start) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) − 2. * (BcFinal − 1. * BcStart * gamma) * (BFinal − 1. * BFinal * t + BStart * gamma * t) + 1. * (Pc3Final − 1. * gamma * Pc3Start) * (BFinal − 1. * BFinal * t + BStart * gamma * t) − (Pc3Final − gamma * Pc3Start) * (P3Final − P3Final * t + gamma * P3Start * t) + 1. *

(BcFinal - 1. * BcStart * gamma) * (P3Final - 1. * P3Final * t + gamma
* P3Start * t) - (P3Final - gamma * P3Start) * (Pc3Final - Pc3Final *
t + gamma * Pc3Start * t) + 1. * (BFinal - 1. * BStart * gamma) *
(Pc3Final - 1. * Pc3Final * t + gamma * Pc3Start * t) + DD * (-BcFinal
+ BcStart * gamma) * T33 + 1. * DD * (Pc3Final - 1. * gamma *
Pc3Start) * T33 + 1. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1.
* BcFinal * t + BcStart * gamma * t) * T33 - (Pc3Final - gamma *
Pc3Start) * (BFinal - BFinal * t + BStart * gamma * t) * T33 + 1. *
(BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart
* gamma * t) * T33 - (BFinal - BStart * gamma) * (Pc3Final - Pc3Final
* t + gamma * Pc3Start * t) * T33 - 1. * G * (BcFinal - BcFinal * t +
BcStart * gamma * t) * (-T53Final + gamma * T53Start) + G * (Pc3Final
- Pc3Final * t + gamma * Pc3Start * t) * (-T53Final + gamma *
T53Start) - G * (Pc3Final - gamma * Pc3Start) * (T53Final - t *
T53Final + gamma * t * T53Start) + 1. * G * (BcFinal - 1. * BcStart *
gamma) * (T53Final - 1. * t * T53Final + gamma * t * T53Start) + DDc *
(-BFinal + BStart * gamma) * Tc33 + 1. * DDc * (P3Final - 1. * gamma *
P3Start) * Tc33 - (P3Final - gamma * P3Start) * (BcFinal - BcFinal * t
+ BcStart * gamma * t) * Tc33 + 1. * (BFinal - 1. * BStart * gamma) *
(BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * Tc33 + 1. *
(BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart
* gamma * t) * Tc33 - (BcFinal - BcStart * gamma) * (P3Final - P3Final
* t + gamma * P3Start * t) * Tc33 + 1. * DDc * G * (T53Final - 1. *
gamma * T53Start) * Tc33 + G * (BcFinal - BcFinal * t + BcStart *
gamma * t) * (-T53Final + gamma * T53Start) * Tc33 - G * (BcFinal -
BcStart * gamma) * (T53Final - t * T53Final + gamma * t * T53Start) *
Tc33 + 1. * DD * Gc * T33 * (Tc53Final - 1. * gamma * Tc53Start) - 1.

∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc53Final +
gamma ∗ Tc53Start) + Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗
t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + Gc ∗ (BFinal − BFinal ∗ t +
BStart ∗ gamma ∗ t) ∗ T33 ∗ (−Tc53Final + gamma ∗ Tc53Start) − Gc ∗
(P3Final − gamma ∗ P3Start) ∗ (Tc53Final − t ∗ Tc53Final + gamma ∗ t ∗
Tc53Start) − (BFinal − BStart ∗ gamma) ∗ Gc ∗ T33 ∗ (Tc53Final − t ∗
Tc53Final + gamma ∗ t ∗ Tc53Start) + 1. ∗ (BFinal − 1. ∗ BStart ∗
gamma) ∗ Gc ∗ (Tc53Final − 1. ∗ t ∗ Tc53Final + gamma ∗ t ∗
Tc53Start);
Ht4 = 1. ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) + 1. ∗ DDc ∗ (BFinal
− 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗
(BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P4Final −
1. ∗ gamma ∗ P4Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma
∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗
t + BStart ∗ gamma ∗ t) + 1. ∗ (Pc4Final − 1. ∗ gamma ∗ Pc4Start) ∗
(BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc4Final − gamma ∗
Pc4Start) ∗ (P4Final − P4Final ∗ t + gamma ∗ P4Start ∗ t) + 1. ∗
(BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P4Final − 1. ∗ P4Final ∗ t + gamma
∗ P4Start ∗ t) − (P4Final − gamma ∗ P4Start) ∗ (Pc4Final − Pc4Final ∗
t + gamma ∗ Pc4Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗
(Pc4Final − 1. ∗ Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + DD ∗ (−BcFinal
+ BcStart ∗ gamma) ∗ T34 + 1. ∗ DD ∗ (Pc4Final − 1. ∗ gamma ∗
Pc4Start) ∗ T34 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1.
∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T34 − (Pc4Final − gamma ∗
Pc4Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T34 + 1. ∗
(BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart
∗ gamma ∗ t) ∗ T34 − (BFinal − BStart ∗ gamma) ∗ (Pc4Final − Pc4Final

* t + gamma * Pc4Start * t) * T34 − 1. * G * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T54Final + gamma * T54Start) + G * (Pc4Final − Pc4Final * t + gamma * Pc4Start * t) * (−T54Final + gamma * T54Start) − G * (Pc4Final − gamma * Pc4Start) * (T54Final − t * T54Final + gamma * t * T54Start) + 1. * G * (BcFinal − 1. * BcStart * gamma) * (T54Final − 1. * t * T54Final + gamma * t * T54Start) + DDc * (−BFinal + BStart * gamma) * Tc34 + 1. * DDc * (P4Final − 1. * gamma * P4Start) * Tc34 − (P4Final − gamma * P4Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc34 + 1. * (BFinal − 1. * BStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc34 + 1. * (BcFinal − 1. * BcStart * gamma) * (BFinal − 1. * BFinal * t + BStart * gamma * t) * Tc34 − (BcFinal − BcStart * gamma) * (P4Final − P4Final * t + gamma * P4Start * t) * Tc34 + 1. * DDc * G * (T54Final − 1. * gamma * T54Start) * Tc34 + G * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T54Final + gamma * T54Start) * Tc34 − G * (BcFinal − BcStart * gamma) * (T54Final − t * T54Final + gamma * t * T54Start) * Tc34 + 1. * DD * Gc * T34 * (Tc54Final − 1. * gamma * Tc54Start) − 1. * Gc * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc54Final + gamma * Tc54Start) + Gc * (P4Final − P4Final * t + gamma * P4Start * t) * (−Tc54Final + gamma * Tc54Start) + Gc * (BFinal − BFinal * t + BStart * gamma * t) * T34 * (−Tc54Final + gamma * Tc54Start) − Gc * (P4Final − gamma * P4Start) * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − (BFinal − BStart * gamma) * Gc * T34 * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) + 1. * (BFinal − 1. * BStart * gamma) * Gc * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start);

Ht5 = 1. * DD * (BcFinal − 1. * BcStart * gamma) + 1. * DDc * (BFinal

− 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P5Final − 1. ∗ gamma ∗ P5Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc5Final − gamma ∗ Pc5Start) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) − (P5Final − gamma ∗ P5Start) ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc5Final − 1. ∗ Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) + DD ∗ (−BcFinal + BcStart ∗ gamma) ∗ T35 + 1. ∗ DD ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ T35 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T35 − (Pc5Final − gamma ∗ Pc5Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T35 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T35 − (BFinal − BStart ∗ gamma) ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T55Final + gamma ∗ T55Start) + G ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − G ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T55Final − 1. ∗ t ∗ T55Final + gamma ∗ t ∗ T55Start) + DDc ∗ (−BFinal + BStart ∗ gamma) ∗ Tc35 + 1. ∗ DDc ∗ (P5Final − 1. ∗ gamma ∗ P5Start) ∗ Tc35 − (P5Final − gamma ∗ P5Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc35 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc35 + 1. ∗

```
(BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart
* gamma * t) * Tc35 - (BcFinal - BcStart * gamma) * (P5Final - P5Final
* t + gamma * P5Start * t) * Tc35 + 1. * DDc * G * (T55Final - 1. *
gamma * T55Start) * Tc35 + G * (BcFinal - BcFinal * t + BcStart *
gamma * t) * (-T55Final + gamma * T55Start) * Tc35 - G * (BcFinal -
BcStart * gamma) * (T55Final - t * T55Final + gamma * t * T55Start) *
Tc35 + 1. * DD * Gc * T35 * (Tc55Final - 1. * gamma * Tc55Start) - 1.
* Gc * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc55Final +
gamma * Tc55Start) + Gc * (P5Final - P5Final * t + gamma * P5Start *
t) * (-Tc55Final + gamma * Tc55Start) + Gc * (BFinal - BFinal * t +
BStart * gamma * t) * T35 * (-Tc55Final + gamma * Tc55Start) - Gc *
(P5Final - gamma * P5Start) * (Tc55Final - t * Tc55Final + gamma * t *
Tc55Start) - (BFinal - BStart * gamma) * Gc * T35 * (Tc55Final - t *
Tc55Final + gamma * t * Tc55Start) + 1. * (BFinal - 1. * BStart *
gamma) * Gc * (Tc55Final - 1. * t * Tc55Final + gamma * t *
Tc55Start);
Ht6 = 1. * CC * DDc * G * (AcFinal - 1. * AcStart * gamma) + CC * DD *
DDc * (-AcFinal + AcStart * gamma) + 1. * CCc * DDc * G * (AFinal - 1.
* AStart * gamma) + CCc * DD * DDc * (-AFinal + AStart * gamma) + 1. *
CC * DD * (AcFinal - 1. * AcStart * gamma) * Gc + CC * G * (-AcFinal +
AcStart * gamma) * Gc + 1. * CCc * DD * (AFinal - 1. * AStart * gamma)
* Gc + CCc * G * (-AFinal + AStart * gamma) * Gc - DDc * G * (BFinal -
BStart * gamma) * (AcFinal - AcFinal * t + AcStart * gamma * t) - CC *
(BFinal - BStart * gamma) * Gc * (AcFinal - AcFinal * t + AcStart *
gamma * t) - CC * DDc * (P1Final - gamma * P1Start) * (AcFinal -
AcFinal * t + AcStart * gamma * t) - DD * Gc * (P1Final - gamma *
P1Start) * (AcFinal - AcFinal * t + AcStart * gamma * t) + 1. * CC *
```

DDc * (BFinal - 1. * BStart * gamma) * (AcFinal - 1. * AcFinal * t +
AcStart * gamma * t) + 1. * G * (BFinal - 1. * BStart * gamma) * Gc *
(AcFinal - 1. * AcFinal * t + AcStart * gamma * t) + 1. * DD * DDc *
(P1Final - 1. * gamma * P1Start) * (AcFinal - 1. * AcFinal * t +
AcStart * gamma * t) + 1. * CC * Gc * (P1Final - 1. * gamma * P1Start)
* (AcFinal - 1. * AcFinal * t + AcStart * gamma * t) - CCc * G *
(BcFinal - BcStart * gamma) * (AFinal - AFinal * t + AStart * gamma *
t) - DD * (BcFinal - BcStart * gamma) * Gc * (AFinal - AFinal * t +
AStart * gamma * t) - CCc * DD * (Pc1Final - gamma * Pc1Start) *
(AFinal - AFinal * t + AStart * gamma * t) - DDc * G * (Pc1Final -
gamma * Pc1Start) * (AFinal - AFinal * t + AStart * gamma * t) + 1. *
CCc * DD * (BcFinal - 1. * BcStart * gamma) * (AFinal - 1. * AFinal *
t + AStart * gamma * t) + 1. * G * (BcFinal - 1. * BcStart * gamma) *
Gc * (AFinal - 1. * AFinal * t + AStart * gamma * t) + 1. * DD * DDc *
(Pc1Final - 1. * gamma * Pc1Start) * (AFinal - 1. * AFinal * t +
AStart * gamma * t) + 1. * CCc * G * (Pc1Final - 1. * gamma *
Pc1Start) * (AFinal - 1. * AFinal * t + AStart * gamma * t) - CCc * G
* (AFinal - AStart * gamma) * (BcFinal - BcFinal * t + BcStart * gamma
* t) - DD * (AFinal - AStart * gamma) * Gc * (BcFinal - BcFinal * t +
BcStart * gamma * t) - CCc * DD * (P1Final - gamma * P1Start) *
(BcFinal - BcFinal * t + BcStart * gamma * t) - G * Gc * (P1Final -
gamma * P1Start) * (BcFinal - BcFinal * t + BcStart * gamma * t) + 1.
* CCc * DD * (AFinal - 1. * AStart * gamma) * (BcFinal - 1. * BcFinal
* t + BcStart * gamma * t) + 1. * G * (AFinal - 1. * AStart * gamma) *
Gc * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) + 1. * CCc * G
* (P1Final - 1. * gamma * P1Start) * (BcFinal - 1. * BcFinal * t +
BcStart * gamma * t) + 1. * DD * Gc * (P1Final - 1. * gamma * P1Start)

* (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) - DDc * G *
(AcFinal - AcStart * gamma) * (BFinal - BFinal * t + BStart * gamma *
t) - CC * (AcFinal - AcStart * gamma) * Gc * (BFinal - BFinal * t +
BStart * gamma * t) - CC * DDc * (Pc1Final - gamma * Pc1Start) *
(BFinal - BFinal * t + BStart * gamma * t) - G * Gc * (Pc1Final -
gamma * Pc1Start) * (BFinal - BFinal * t + BStart * gamma * t) + 1. *
CC * DDc * (AcFinal - 1. * AcStart * gamma) * (BFinal - 1. * BFinal *
t + BStart * gamma * t) + 1. * G * (AcFinal - 1. * AcStart * gamma) *
Gc * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * DDc * G *
(Pc1Final - 1. * gamma * Pc1Start) * (BFinal - 1. * BFinal * t +
BStart * gamma * t) + 1. * CC * Gc * (Pc1Final - 1. * gamma *
Pc1Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) - CC * DDc
* (AcFinal - AcStart * gamma) * (P1Final - P1Final * t + gamma *
P1Start * t) - CCc * DD * (BcFinal - BcStart * gamma) * (P1Final -
P1Final * t + gamma * P1Start * t) - DD * (AcFinal - AcStart * gamma)
* Gc * (P1Final - P1Final * t + gamma * P1Start * t) - G * (BcFinal -
BcStart * gamma) * Gc * (P1Final - P1Final * t + gamma * P1Start * t)
- DD * DDc * (Pc1Final - gamma * Pc1Start) * (P1Final - P1Final * t +
gamma * P1Start * t) - CCc * G * (Pc1Final - gamma * Pc1Start) *
(P1Final - P1Final * t + gamma * P1Start * t) - CC * Gc * (Pc1Final -
gamma * Pc1Start) * (P1Final - P1Final * t + gamma * P1Start * t) + 1.
* DD * DDc * (AcFinal - 1. * AcStart * gamma) * (P1Final - 1. *
P1Final * t + gamma * P1Start * t) + 1. * CCc * G * (BcFinal - 1. *
BcStart * gamma) * (P1Final - 1. * P1Final * t + gamma * P1Start * t)
+ 1. * CC * (AcFinal - 1. * AcStart * gamma) * Gc * (P1Final - 1. *
P1Final * t + gamma * P1Start * t) + 1. * DD * (BcFinal - 1. * BcStart
* gamma) * Gc * (P1Final - 1. * P1Final * t + gamma * P1Start * t) +

1. * CCc * DD * (Pc1Final − 1. * gamma * Pc1Start) * (P1Final − 1. * P1Final * t + gamma * P1Start * t) + 1. * CC * DDc * (Pc1Final − 1. * gamma * Pc1Start) * (P1Final − 1. * P1Final * t + gamma * P1Start * t) + 1. * G * Gc * (Pc1Final − 1. * gamma * Pc1Start) * (P1Final − 1. * P1Final * t + gamma * P1Start * t) − CCc * DD * (AFinal − AStart * gamma) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − DDc * G * (AFinal − AStart * gamma) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − CC * DDc * (BFinal − BStart * gamma) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − G * (BFinal − BStart * gamma) * Gc * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − DD * DDc * (P1Final − gamma * P1Start) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − CCc * G * (P1Final − gamma * P1Start) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) − CC * Gc * (P1Final − gamma * P1Start) * (Pc1Final − Pc1Final * t + gamma * Pc1Start * t) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * CC * (BFinal − 1. * BStart * gamma) * Gc * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * CCc * DD * (P1Final − 1. * gamma * P1Start) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * CC * DDc * (P1Final − 1. * gamma * P1Start) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * G * Gc * (P1Final − 1. * gamma * P1Start) * (Pc1Final − 1. * Pc1Final * t + gamma * Pc1Start * t) + 1. * CC * DD * DDc * (AcFinal − 1. * AcStart * gamma) * T31 + DD * DDc * G * (−AcFinal + AcStart * gamma) * T31 + 1. * DD * G * (AcFinal − 1. * AcStart * gamma) * Gc *

T31 + CC ∗ DD ∗ (−AcFinal + AcStart ∗ gamma) ∗ Gc ∗ T31 + 1. ∗ DD ∗ DDc ∗ G ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ T31 + 1. ∗ CC ∗ DD ∗ Gc ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ T31 + CC ∗ DD ∗ DDc ∗ (−Pc1Final + gamma ∗ Pc1Start) ∗ T31 + DD ∗ G ∗ Gc ∗ (−Pc1Final + gamma ∗ Pc1Start) ∗ T31 − CC ∗ DDc ∗ (BFinal − BStart ∗ gamma) ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T31 − G ∗ (BFinal − BStart ∗ gamma) ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ DDc ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ CC ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T31 − CC ∗ DDc ∗ (AcFinal − AcStart ∗ gamma) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − G ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − DDc ∗ G ∗ (Pc1Final − gamma ∗ Pc1Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − CC ∗ Gc ∗ (Pc1Final − gamma ∗ Pc1Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ DDc ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ CC ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ CC ∗ DDc ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ G ∗ Gc ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − DDc ∗ G ∗ (BFinal − BStart ∗ gamma) ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ T31 − CC ∗ (BFinal − BStart ∗ gamma) ∗ Gc ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ T31 + 1. ∗ CC ∗ DDc ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc1Final − 1. ∗ Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ T31 + 1. ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗ (Pc1Final − 1. ∗ Pc1Final ∗ t +

gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ DDc ∗ G ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + DD ∗ G ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + CCc ∗ DD ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + DD ∗ DDc ∗ G ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + CCc ∗ Power(G, 2) ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + CC ∗ G ∗ Gc ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − CC ∗ DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) − CCc ∗ DD ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) − DD ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) − Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) − DD

* DDc * G * (Pc1Final − gamma * Pc1Start) * (T51Final − t * T51Final +
gamma * t * T51Start) − CCc * Power(G, 2) * (Pc1Final − gamma *
Pc1Start) * (T51Final − t * T51Final + gamma * t * T51Start) − CC * G
* Gc * (Pc1Final − gamma * Pc1Start) * (T51Final − t * T51Final +
gamma * t * T51Start) + 1. * DD * DDc * G * (AcFinal − 1. * AcStart *
gamma) * (T51Final − 1. * t * T51Final + gamma * t * T51Start) + 1. *
CCc * Power(G, 2) * (BcFinal − 1. * BcStart * gamma) * (T51Final − 1.
* t * T51Final + gamma * t * T51Start) + 1. * CC * G * (AcFinal − 1. *
AcStart * gamma) * Gc * (T51Final − 1. * t * T51Final + gamma * t *
T51Start) + 1. * DD * G * (BcFinal − 1. * BcStart * gamma) * Gc *
(T51Final − 1. * t * T51Final + gamma * t * T51Start) + 1. * CCc * DD
* G * (Pc1Final − 1. * gamma * Pc1Start) * (T51Final − 1. * t *
T51Final + gamma * t * T51Start) + 1. * CC * DDc * G * (Pc1Final − 1.
* gamma * Pc1Start) * (T51Final − 1. * t * T51Final + gamma * t *
T51Start) + 1. * Power(G, 2) * Gc * (Pc1Final − 1. * gamma * Pc1Start)
* (T51Final − 1. * t * T51Final + gamma * t * T51Start) + 1. * CCc *
DD * DDc * (AFinal − 1. * AStart * gamma) * Tc31 + CCc * DDc * G *
(−AFinal + AStart * gamma) * Tc31 + 1. * DDc * G * (AFinal − 1. *
AStart * gamma) * Gc * Tc31 + DD * DDc * (−AFinal + AStart * gamma) *
Gc * Tc31 + 1. * CCc * DDc * G * (P1Final − 1. * gamma * P1Start) *
Tc31 + 1. * DD * DDc * Gc * (P1Final − 1. * gamma * P1Start) * Tc31 +
CCc * DD * DDc * (−P1Final + gamma * P1Start) * Tc31 + DDc * G * Gc *
(−P1Final + gamma * P1Start) * Tc31 − CCc * DD * (BcFinal − BcStart *
gamma) * (AFinal − AFinal * t + AStart * gamma * t) * Tc31 − G *
(BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t + AStart *
gamma * t) * Tc31 + 1. * CCc * G * (BcFinal − 1. * BcStart * gamma) *
(AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc31 + 1. * DD *

(BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ Tc31 − CCc ∗ DD ∗ (AFinal − AStart ∗ gamma) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 − G ∗ (AFinal − AStart ∗ gamma) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 − CCc ∗ G ∗ (P1Final − gamma ∗ P1Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 − DD ∗ Gc ∗ (P1Final − gamma ∗ P1Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ CCc ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ CCc ∗ DD ∗ (P1Final − 1. ∗ gamma ∗ P1Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ G ∗ Gc ∗ (P1Final − 1. ∗ gamma ∗ P1Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 − CCc ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 − DD ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P1Final − 1. ∗ P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (P1Final − 1. ∗ P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 + 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ (T51Final − 1. ∗ gamma ∗ T51Start) ∗ Tc31 + 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ (T51Final − 1. ∗ gamma ∗ T51Start) ∗ Tc31 + CCc ∗ DD ∗ DDc ∗ G ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 + DDc ∗ Power(G, 2) ∗ Gc ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 + CCc ∗ Power(G, 2) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 + DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗

gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 − CCc ∗ Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − DD ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T51Final − 1. ∗ t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + 1. ∗ Power(G, 2) ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (T51Final − 1. ∗ t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T31 ∗ (Tc51Final − 1. ∗ gamma ∗ Tc51Start) + 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T31 ∗ (Tc51Final − 1. ∗ gamma ∗ Tc51Start) + CCc ∗ DD ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + DDc ∗ G ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + CC ∗ DDc ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ CC ∗ Power(Gc, 2) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + G ∗ Power(Gc, 2) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + DD ∗ DDc ∗ Gc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ (−Tc51Final + gamma ∗

Tc51Start) + CCc * G * Gc * (P1Final − P1Final * t + gamma * P1Start * t) * (−Tc51Final + gamma * Tc51Start) + CC * Power(Gc, 2) * (P1Final − P1Final * t + gamma * P1Start * t) * (−Tc51Final + gamma * Tc51Start) − 1. * G * Power(Gc, 2) * (P1Final − P1Final * t + gamma * P1Start * t) * (−Tc51Final + gamma * Tc51Start) + CC * DD * DDc * Gc * T31 * (−Tc51Final + gamma * Tc51Start) + DD * G * Power(Gc, 2) * T31 * (−Tc51Final + gamma * Tc51Start) − 1. * CC * DDc * Gc * (BFinal − BFinal * t + BStart * gamma * t) * T31 * (−Tc51Final + gamma * Tc51Start) + DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t) * T31 * (−Tc51Final + gamma * Tc51Start) + CC * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T31 * (−Tc51Final + gamma * Tc51Start) − 1. * G * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T31 * (−Tc51Final + gamma * Tc51Start) − CCc * DD * (AFinal − AStart * gamma) * Gc * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − DDc * G * (AFinal − AStart * gamma) * Gc * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − CC * DDc * (BFinal − BStart * gamma) * Gc * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − G * (BFinal − BStart * gamma) * Power(Gc, 2) * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − DD * DDc * Gc * (P1Final − gamma * P1Start) * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − CCc * G * Gc * (P1Final − gamma * P1Start) * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − CC * Power(Gc, 2) * (P1Final − gamma * P1Start) * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − DDc * G * (BFinal − BStart * gamma) * Gc * T31 * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) − CC * (BFinal − BStart * gamma) * Power(Gc, 2) * T31 * (Tc51Final − t * Tc51Final + gamma * t * Tc51Start) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * Gc * (Tc51Final − 1. * t *

Tc51Final + gamma * t * Tc51Start) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * Gc * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * Gc * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * CC * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * CCc * DD * Gc * (P1Final − 1. * gamma * P1Start) * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * CC * DDc * Gc * (P1Final − 1. * gamma * P1Start) * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * G * Power(Gc, 2) * (P1Final − 1. * gamma * P1Start) * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * Gc * T31 * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start) + 1. * G * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * T31 * (Tc51Final − 1. * t * Tc51Final + gamma * t * Tc51Start);
Ht7 = 1. * CC * DDc * G * (AcFinal − 1. * AcStart * gamma) + CC * DD * DDc * (−AcFinal + AcStart * gamma) + 1. * CCc * DDc * G * (AFinal − 1. * AStart * gamma) + CCc * DD * DDc * (−AFinal + AStart * gamma) + 1. * CC * DD * (AcFinal − 1. * AcStart * gamma) * Gc + CC * G * (−AcFinal + AcStart * gamma) * Gc + 1. * CCc * DD * (AFinal − 1. * AStart * gamma) * Gc + CCc * G * (−AFinal + AStart * gamma) * Gc − DDc * G * (BFinal − BStart * gamma) * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * (BFinal − BStart * gamma) * Gc * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * DDc * (P2Final − gamma * P2Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) − DD * Gc * (P2Final − gamma * P2Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * G * (BFinal − 1. * BStart * gamma) * Gc *

(AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) + 1. ∗ DD ∗ DDc ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) + 1. ∗ CC ∗ Gc ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) − CCc ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − DD ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (Pc2Final − gamma ∗ Pc2Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − DDc ∗ G ∗ (Pc2Final − gamma ∗ Pc2Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ DD ∗ DDc ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) − CCc ∗ G ∗ (AFinal − AStart ∗ gamma) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DD ∗ (AFinal − AStart ∗ gamma) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (P2Final − gamma ∗ P2Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − G ∗ Gc ∗ (P2Final − gamma ∗ P2Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ DD ∗ Gc ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗

t) - CC * (AcFinal - AcStart * gamma) * Gc * (BFinal - BFinal * t +
BStart * gamma * t) - CC * DDc * (Pc2Final - gamma * Pc2Start) *
(BFinal - BFinal * t + BStart * gamma * t) - G * Gc * (Pc2Final -
gamma * Pc2Start) * (BFinal - BFinal * t + BStart * gamma * t) + 1. *
CC * DDc * (AcFinal - 1. * AcStart * gamma) * (BFinal - 1. * BFinal *
t + BStart * gamma * t) + 1. * G * (AcFinal - 1. * AcStart * gamma) *
Gc * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * DDc * G *
(Pc2Final - 1. * gamma * Pc2Start) * (BFinal - 1. * BFinal * t +
BStart * gamma * t) + 1. * CC * Gc * (Pc2Final - 1. * gamma *
Pc2Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) - CC * DDc
* (AcFinal - AcStart * gamma) * (P2Final - P2Final * t + gamma *
P2Start * t) - CCc * DD * (BcFinal - BcStart * gamma) * (P2Final -
P2Final * t + gamma * P2Start * t) - DD * (AcFinal - AcStart * gamma)
* Gc * (P2Final - P2Final * t + gamma * P2Start * t) - G * (BcFinal -
BcStart * gamma) * Gc * (P2Final - P2Final * t + gamma * P2Start * t)
- DD * DDc * (Pc2Final - gamma * Pc2Start) * (P2Final - P2Final * t +
gamma * P2Start * t) - CCc * G * (Pc2Final - gamma * Pc2Start) *
(P2Final - P2Final * t + gamma * P2Start * t) - CC * Gc * (Pc2Final -
gamma * Pc2Start) * (P2Final - P2Final * t + gamma * P2Start * t) + 1.
* DD * DDc * (AcFinal - 1. * AcStart * gamma) * (P2Final - 1. *
P2Final * t + gamma * P2Start * t) + 1. * CCc * G * (BcFinal - 1. *
BcStart * gamma) * (P2Final - 1. * P2Final * t + gamma * P2Start * t)
+ 1. * CC * (AcFinal - 1. * AcStart * gamma) * Gc * (P2Final - 1. *
P2Final * t + gamma * P2Start * t) + 1. * DD * (BcFinal - 1. * BcStart
* gamma) * Gc * (P2Final - 1. * P2Final * t + gamma * P2Start * t) +
1. * CCc * DD * (Pc2Final - 1. * gamma * Pc2Start) * (P2Final - 1. *
P2Final * t + gamma * P2Start * t) + 1. * CC * DDc * (Pc2Final - 1. *

gamma * Pc2Start) * (P2Final − 1. * P2Final * t + gamma * P2Start * t) + 1. * G * Gc * (Pc2Final − 1. * gamma * Pc2Start) * (P2Final − 1. * P2Final * t + gamma * P2Start * t) − CCc * DD * (AFinal − AStart * gamma) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − DDc * G * (AFinal − AStart * gamma) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − CC * DDc * (BFinal − BStart * gamma) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − G * (BFinal − BStart * gamma) * Gc * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − DD * DDc * (P2Final − gamma * P2Start) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − CCc * G * (P2Final − gamma * P2Start) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) − CC * Gc * (P2Final − gamma * P2Start) * (Pc2Final − Pc2Final * t + gamma * Pc2Start * t) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * CC * (BFinal − 1. * BStart * gamma) * Gc * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * CCc * DD * (P2Final − 1. * gamma * P2Start) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * CC * DDc * (P2Final − 1. * gamma * P2Start) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * G * Gc * (P2Final − 1. * gamma * P2Start) * (Pc2Final − 1. * Pc2Final * t + gamma * Pc2Start * t) + 1. * CC * DD * DDc * (AcFinal − 1. * AcStart * gamma) * T32 + DD * DDc * G * (−AcFinal + AcStart * gamma) * T32 + 1. * DD * G * (AcFinal − 1. * AcStart * gamma) * Gc * T32 + CC * DD * (−AcFinal + AcStart * gamma) * Gc * T32 + 1. * DD * DDc * G * (Pc2Final − 1. * gamma * Pc2Start) * T32 + 1. * CC * DD * Gc

* (Pc2Final - 1. * gamma * Pc2Start) * T32 + CC * DD * DDc *
(-Pc2Final + gamma * Pc2Start) * T32 + DD * G * Gc * (-Pc2Final +
gamma * Pc2Start) * T32 - CC * DDc * (BFinal - BStart * gamma) *
(AcFinal - AcFinal * t + AcStart * gamma * t) * T32 - G * (BFinal -
BStart * gamma) * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) *
T32 + 1. * DDc * G * (BFinal - 1. * BStart * gamma) * (AcFinal - 1. *
AcFinal * t + AcStart * gamma * t) * T32 + 1. * CC * (BFinal - 1. *
BStart * gamma) * Gc * (AcFinal - 1. * AcFinal * t + AcStart * gamma *
t) * T32 - CC * DDc * (AcFinal - AcStart * gamma) * (BFinal - BFinal *
t + BStart * gamma * t) * T32 - G * (AcFinal - AcStart * gamma) * Gc *
(BFinal - BFinal * t + BStart * gamma * t) * T32 - DDc * G * (Pc2Final
- gamma * Pc2Start) * (BFinal - BFinal * t + BStart * gamma * t) * T32
- CC * Gc * (Pc2Final - gamma * Pc2Start) * (BFinal - BFinal * t +
BStart * gamma * t) * T32 + 1. * DDc * G * (AcFinal - 1. * AcStart *
gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T32 + 1. *
CC * (AcFinal - 1. * AcStart * gamma) * Gc * (BFinal - 1. * BFinal * t
+ BStart * gamma * t) * T32 + 1. * CC * DDc * (Pc2Final - 1. * gamma *
Pc2Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T32 + 1.
* G * Gc * (Pc2Final - 1. * gamma * Pc2Start) * (BFinal - 1. * BFinal
* t + BStart * gamma * t) * T32 - DDc * G * (BFinal - BStart * gamma)
* (Pc2Final - Pc2Final * t + gamma * Pc2Start * t) * T32 - CC *
(BFinal - BStart * gamma) * Gc * (Pc2Final - Pc2Final * t + gamma *
Pc2Start * t) * T32 + 1. * CC * DDc * (BFinal - 1. * BStart * gamma) *
(Pc2Final - 1. * Pc2Final * t + gamma * Pc2Start * t) * T32 + 1. * G *
(BFinal - 1. * BStart * gamma) * Gc * (Pc2Final - 1. * Pc2Final * t +
gamma * Pc2Start * t) * T32 + CC * DDc * G * (AcFinal - AcFinal * t +
AcStart * gamma * t) * (-T52Final + gamma * T52Start) - 1. * DD * DDc

```
* G * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T52Final +
gamma * T52Start) - 1. * CC * G * Gc * (AcFinal - AcFinal * t +
AcStart * gamma * t) * (-T52Final + gamma * T52Start) + DD * G * Gc *
(AcFinal - AcFinal * t + AcStart * gamma * t) * (-T52Final + gamma *
T52Start) + CCc * DD * G * (BcFinal - BcFinal * t + BcStart * gamma *
t) * (-T52Final + gamma * T52Start) - 1. * CCc * Power(G, 2) *
(BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final + gamma *
T52Start) - 1. * DD * G * Gc * (BcFinal - BcFinal * t + BcStart *
gamma * t) * (-T52Final + gamma * T52Start) + Power(G, 2) * Gc *
(BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final + gamma *
T52Start) - 1. * CCc * DD * G * (Pc2Final - Pc2Final * t + gamma *
Pc2Start * t) * (-T52Final + gamma * T52Start) - 1. * CC * DDc * G *
(Pc2Final - Pc2Final * t + gamma * Pc2Start * t) * (-T52Final + gamma
* T52Start) + DD * DDc * G * (Pc2Final - Pc2Final * t + gamma *
Pc2Start * t) * (-T52Final + gamma * T52Start) + CCc * Power(G, 2) *
(Pc2Final - Pc2Final * t + gamma * Pc2Start * t) * (-T52Final + gamma
* T52Start) + CC * G * Gc * (Pc2Final - Pc2Final * t + gamma *
Pc2Start * t) * (-T52Final + gamma * T52Start) - 1. * Power(G, 2) * Gc
* (Pc2Final - Pc2Final * t + gamma * Pc2Start * t) * (-T52Final +
gamma * T52Start) - CC * DDc * G * (AcFinal - AcStart * gamma) *
(T52Final - t * T52Final + gamma * t * T52Start) - CCc * DD * G *
(BcFinal - BcStart * gamma) * (T52Final - t * T52Final + gamma * t *
T52Start) - DD * G * (AcFinal - AcStart * gamma) * Gc * (T52Final - t
* T52Final + gamma * t * T52Start) - Power(G, 2) * (BcFinal - BcStart
* gamma) * Gc * (T52Final - t * T52Final + gamma * t * T52Start) - DD
* DDc * G * (Pc2Final - gamma * Pc2Start) * (T52Final - t * T52Final +
gamma * t * T52Start) - CCc * Power(G, 2) * (Pc2Final - gamma *
```

Pc2Start) * (T52Final - t * T52Final + gamma * t * T52Start) - CC * G * Gc * (Pc2Final - gamma * Pc2Start) * (T52Final - t * T52Final + gamma * t * T52Start) + 1. * DD * DDc * G * (AcFinal - 1. * AcStart * gamma) * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * CCc * Power(G, 2) * (BcFinal - 1. * BcStart * gamma) * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * CC * G * (AcFinal - 1. * AcStart * gamma) * Gc * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * DD * G * (BcFinal - 1. * BcStart * gamma) * Gc * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * CCc * DD * G * (Pc2Final - 1. * gamma * Pc2Start) * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * CC * DDc * G * (Pc2Final - 1. * gamma * Pc2Start) * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * Power(G, 2) * Gc * (Pc2Final - 1. * gamma * Pc2Start) * (T52Final - 1. * t * T52Final + gamma * t * T52Start) + 1. * CCc * DD * DDc * (AFinal - 1. * AStart * gamma) * Tc32 + CCc * DDc * G * (-AFinal + AStart * gamma) * Tc32 + 1. * DDc * G * (AFinal - 1. * AStart * gamma) * Gc * Tc32 + DD * DDc * (-AFinal + AStart * gamma) * Gc * Tc32 + 1. * CCc * DDc * G * (P2Final - 1. * gamma * P2Start) * Tc32 + 1. * DD * DDc * Gc * (P2Final - 1. * gamma * P2Start) * Tc32 + CCc * DD * DDc * (-P2Final + gamma * P2Start) * Tc32 + DDc * G * Gc * (-P2Final + gamma * P2Start) * Tc32 - CCc * DD * (BcFinal - BcStart * gamma) * (AFinal - AFinal * t + AStart * gamma * t) * Tc32 - G * (BcFinal - BcStart * gamma) * Gc * (AFinal - AFinal * t + AStart * gamma * t) * Tc32 + 1. * CCc * G * (BcFinal - 1. * BcStart * gamma) * (AFinal - 1. * AFinal * t + AStart * gamma * t) * Tc32 + 1. * DD * (BcFinal - 1. * BcStart * gamma) * Gc * (AFinal - 1. * AFinal * t + AStart * gamma * t) * Tc32 - CCc * DD * (AFinal - AStart * gamma) *

$(BcFinal - BcFinal * t + BcStart * gamma * t) * Tc32 - G * (AFinal - AStart * gamma) * Gc * (BcFinal - BcFinal * t + BcStart * gamma * t) * Tc32 - CCc * G * (P2Final - gamma * P2Start) * (BcFinal - BcFinal * t + BcStart * gamma * t) * Tc32 - DD * Gc * (P2Final - gamma * P2Start) * (BcFinal - BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * CCc * G * (AFinal - 1. * AStart * gamma) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * DD * (AFinal - 1. * AStart * gamma) * Gc * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * CCc * DD * (P2Final - 1. * gamma * P2Start) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * Tc32 + 1. * G * Gc * (P2Final - 1. * gamma * P2Start) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * Tc32 - CCc * G * (BcFinal - BcStart * gamma) * (P2Final - P2Final * t + gamma * P2Start * t) * Tc32 - DD * (BcFinal - BcStart * gamma) * Gc * (P2Final - P2Final * t + gamma * P2Start * t) * Tc32 + 1. * CCc * DD * (BcFinal - 1. * BcStart * gamma) * (P2Final - 1. * P2Final * t + gamma * P2Start * t) * Tc32 + 1. * G * (BcFinal - 1. * BcStart * gamma) * Gc * (P2Final - 1. * P2Final * t + gamma * P2Start * t) * Tc32 + 1. * CCc * DDc * Power(G, 2) * (T52Final - 1. * gamma * T52Start) * Tc32 + 1. * DD * DDc * G * Gc * (T52Final - 1. * gamma * T52Start) * Tc32 + CCc * DD * DDc * G * (-T52Final + gamma * T52Start) * Tc32 + DDc * Power(G, 2) * Gc * (-T52Final + gamma * T52Start) * Tc32 - 1. * CCc * DD * G * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final + gamma * T52Start) * Tc32 + CCc * Power(G, 2) * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final + gamma * T52Start) * Tc32 + DD * G * Gc * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final + gamma * T52Start) * Tc32 - 1. * Power(G, 2) * Gc * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T52Final +$

gamma * T52Start) * Tc32 − CCc * Power(G, 2) * (BcFinal − BcStart * gamma) * (T52Final − t * T52Final + gamma * t * T52Start) * Tc32 − DD * G * (BcFinal − BcStart * gamma) * Gc * (T52Final − t * T52Final + gamma * t * T52Start) * Tc32 + 1. * CCc * DD * G * (BcFinal − 1. * BcStart * gamma) * (T52Final − 1. * t * T52Final + gamma * t * T52Start) * Tc32 + 1. * Power(G, 2) * (BcFinal − 1. * BcStart * gamma) * Gc * (T52Final − 1. * t * T52Final + gamma * t * T52Start) * Tc32 + 1. * DD * DDc * G * Gc * T32 * (Tc52Final − 1. * gamma * Tc52Start) + 1. * CC * DD * Power(Gc, 2) * T32 * (Tc52Final − 1. * gamma * Tc52Start) + CCc * DD * Gc * (AFinal − AFinal * t + AStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) − 1. * DD * DDc * Gc * (AFinal − AFinal * t + AStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) − 1. * CCc * G * Gc * (AFinal − AFinal * t + AStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) + DDc * G * Gc * (AFinal − AFinal * t + AStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) + CC * DDc * Gc * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) − 1. * DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) − 1. * CC * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) + G * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc52Final + gamma * Tc52Start) − 1. * CCc * DD * Gc * (P2Final − P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start) − 1. * CC * DDc * Gc * (P2Final − P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start) + DD * DDc * Gc * (P2Final − P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start) + CCc * G * Gc * (P2Final − P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start) + CC * Power(Gc, 2) * (P2Final −

P2Final * t + gamma * P2Start * t) * (−Tc52Final + gamma * Tc52Start)
− 1. * G * Power(Gc, 2) * (P2Final − P2Final * t + gamma * P2Start *
t) * (−Tc52Final + gamma * Tc52Start) + CC * DD * DDc * Gc * T32 *
(−Tc52Final + gamma * Tc52Start) + DD * G * Power(Gc, 2) * T32 *
(−Tc52Final + gamma * Tc52Start) − 1. * CC * DDc * Gc * (BFinal −
BFinal * t + BStart * gamma * t) * T32 * (−Tc52Final + gamma *
Tc52Start) + DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t)
* T32 * (−Tc52Final + gamma * Tc52Start) + CC * Power(Gc, 2) * (BFinal
− BFinal * t + BStart * gamma * t) * T32 * (−Tc52Final + gamma *
Tc52Start) − 1. * G * Power(Gc, 2) * (BFinal − BFinal * t + BStart *
gamma * t) * T32 * (−Tc52Final + gamma * Tc52Start) − CCc * DD *
(AFinal − AStart * gamma) * Gc * (Tc52Final − t * Tc52Final + gamma *
t * Tc52Start) − DDc * G * (AFinal − AStart * gamma) * Gc * (Tc52Final
− t * Tc52Final + gamma * t * Tc52Start) − CC * DDc * (BFinal − BStart
* gamma) * Gc * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) −
G * (BFinal − BStart * gamma) * Power(Gc, 2) * (Tc52Final − t *
Tc52Final + gamma * t * Tc52Start) − DD * DDc * Gc * (P2Final − gamma
* P2Start) * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) − CCc
* G * Gc * (P2Final − gamma * P2Start) * (Tc52Final − t * Tc52Final +
gamma * t * Tc52Start) − CC * Power(Gc, 2) * (P2Final − gamma *
P2Start) * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) − DDc *
G * (BFinal − BStart * gamma) * Gc * T32 * (Tc52Final − t * Tc52Final
+ gamma * t * Tc52Start) − CC * (BFinal − BStart * gamma) * Power(Gc,
2) * T32 * (Tc52Final − t * Tc52Final + gamma * t * Tc52Start) + 1. *
DD * DDc * (AFinal − 1. * AStart * gamma) * Gc * (Tc52Final − 1. * t *
Tc52Final + gamma * t * Tc52Start) + 1. * CCc * G * (AFinal − 1. *
AStart * gamma) * Gc * (Tc52Final − 1. * t * Tc52Final + gamma * t *

Tc52Start) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * Gc * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * CC * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * CCc * DD * Gc * (P2Final − 1. * gamma * P2Start) * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * CC * DDc * Gc * (P2Final − 1. * gamma * P2Start) * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * G * Power(Gc, 2) * (P2Final − 1. * gamma * P2Start) * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * Gc * T32 * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start) + 1. * G * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * T32 * (Tc52Final − 1. * t * Tc52Final + gamma * t * Tc52Start);
Ht8 = 1. * CC * DDc * G * (AcFinal − 1. * AcStart * gamma) + CC * DD * DDc * (−AcFinal + AcStart * gamma) + 1. * CCc * DDc * G * (AFinal − 1. * AStart * gamma) + CCc * DD * DDc * (−AFinal + AStart * gamma) + 1. * CC * DD * (AcFinal − 1. * AcStart * gamma) * Gc + CC * G * (−AcFinal + AcStart * gamma) * Gc + 1. * CCc * DD * (AFinal − 1. * AStart * gamma) * Gc + CCc * G * (−AFinal + AStart * gamma) * Gc − DDc * G * (BFinal − BStart * gamma) * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * (BFinal − BStart * gamma) * Gc * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * DDc * (P3Final − gamma * P3Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) − DD * Gc * (P3Final − gamma * P3Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * G * (BFinal − 1. * BStart * gamma) * Gc * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * DD * DDc * (P3Final − 1. * gamma * P3Start) * (AcFinal − 1. * AcFinal * t +

AcStart * gamma * t) + 1. * CC * Gc * (P3Final − 1. * gamma * P3Start) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) − CCc * G * (BcFinal − BcStart * gamma) * (AFinal − AFinal * t + AStart * gamma * t) − DD * (BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t + AStart * gamma * t) − CCc * DD * (Pc3Final − gamma * Pc3Start) * (AFinal − AFinal * t + AStart * gamma * t) − DDc * G * (Pc3Final − gamma * Pc3Start) * (AFinal − AFinal * t + AStart * gamma * t) + 1. * CCc * DD * (BcFinal − 1. * BcStart * gamma) * (AFinal − 1. * AFinal * t + AStart * gamma * t) + 1. * G * (BcFinal − 1. * BcStart * gamma) * Gc * (AFinal − 1. * AFinal * t + AStart * gamma * t) + 1. * DD * DDc * (Pc3Final − 1. * gamma * Pc3Start) * (AFinal − 1. * AFinal * t + AStart * gamma * t) + 1. * CCc * G * (Pc3Final − 1. * gamma * Pc3Start) * (AFinal − 1. * AFinal * t + AStart * gamma * t) − CCc * G * (AFinal − AStart * gamma) * (BcFinal − BcFinal * t + BcStart * gamma * t) − DD * (AFinal − AStart * gamma) * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) − CCc * DD * (P3Final − gamma * P3Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) − G * Gc * (P3Final − gamma * P3Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) + 1. * CCc * DD * (AFinal − 1. * AStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) + 1. * G * (AFinal − 1. * AStart * gamma) * Gc * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) + 1. * CCc * G * (P3Final − 1. * gamma * P3Start) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) + 1. * DD * Gc * (P3Final − 1. * gamma * P3Start) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) − DDc * G * (AcFinal − AcStart * gamma) * (BFinal − BFinal * t + BStart * gamma * t) − CC * (AcFinal − AcStart * gamma) * Gc * (BFinal − BFinal * t + BStart * gamma * t) − CC * DDc * (Pc3Final − gamma * Pc3Start) *

(BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − G ∗ Gc ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ CC ∗ DDc ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ DDc ∗ G ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ CC ∗ Gc ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ DDc ∗ (AcFinal − AcStart ∗ gamma) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − CCc ∗ DD ∗ (BcFinal − BcStart ∗ gamma) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − DD ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − G ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − DD ∗ DDc ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − CCc ∗ G ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) − CC ∗ Gc ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ DD ∗ DDc ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ CCc ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ CC ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ CCc ∗ DD ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ CC ∗ DDc ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ G ∗ Gc ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (P3Final − 1. ∗

P3Final * t + gamma * P3Start * t) − CCc * DD * (AFinal − AStart * gamma) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − DDc * G * (AFinal − AStart * gamma) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − CC * DDc * (BFinal − BStart * gamma) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − G * (BFinal − BStart * gamma) * Gc * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − DD * DDc * (P3Final − gamma * P3Start) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − CCc * G * (P3Final − gamma * P3Start) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) − CC * Gc * (P3Final − gamma * P3Start) * (Pc3Final − Pc3Final * t + gamma * Pc3Start * t) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * CC * (BFinal − 1. * BStart * gamma) * Gc * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * CCc * DD * (P3Final − 1. * gamma * P3Start) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * CC * DDc * (P3Final − 1. * gamma * P3Start) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * G * Gc * (P3Final − 1. * gamma * P3Start) * (Pc3Final − 1. * Pc3Final * t + gamma * Pc3Start * t) + 1. * CC * DD * DDc * (AcFinal − 1. * AcStart * gamma) * T33 + DD * DDc * G * (−AcFinal + AcStart * gamma) * T33 + 1. * DD * G * (AcFinal − 1. * AcStart * gamma) * Gc * T33 + CC * DD * (−AcFinal + AcStart * gamma) * Gc * T33 + 1. * DD * DDc * G * (Pc3Final − 1. * gamma * Pc3Start) * T33 + 1. * CC * DD * Gc * (Pc3Final − 1. * gamma * Pc3Start) * T33 + CC * DD * DDc * (−Pc3Final + gamma * Pc3Start) * T33 + DD * G * Gc * (−Pc3Final +

gamma ∗ Pc3Start) ∗ T33 − CC ∗ DDc ∗ (BFinal − BStart ∗ gamma) ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T33 − G ∗ (BFinal − BStart ∗ gamma) ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ DDc ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ CC ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T33 − CC ∗ DDc ∗ (AcFinal − AcStart ∗ gamma) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 − G ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 − DDc ∗ G ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 − CC ∗ Gc ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ DDc ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ CC ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ CC ∗ DDc ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ G ∗ Gc ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 − DDc ∗ G ∗ (BFinal − BStart ∗ gamma) ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ T33 − CC ∗ (BFinal − BStart ∗ gamma) ∗ Gc ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ T33 + 1. ∗ CC ∗ DDc ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc3Final − 1. ∗ Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ T33 + 1. ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗ (Pc3Final − 1. ∗ Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ DDc ∗ G ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal − AcFinal ∗ t +

AcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + DD ∗ G ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + CCc ∗ DD ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + DD ∗ DDc ∗ G ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + CCc ∗ Power(G, 2) ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) + CC ∗ G ∗ Gc ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) ∗ (−T53Final + gamma ∗ T53Start) − CC ∗ DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − CCc ∗ DD ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − DD ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − DD ∗ DDc ∗ G ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − CCc ∗ Power(G, 2) ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) − CC ∗ G ∗ Gc ∗ (Pc3Final − gamma ∗ Pc3Start) ∗ (T53Final − t ∗ T53Final +

gamma * t * T53Start) + 1. * DD * DDc * G * (AcFinal − 1. * AcStart * gamma) * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * CCc * Power(G, 2) * (BcFinal − 1. * BcStart * gamma) * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * CC * G * (AcFinal − 1. * AcStart * gamma) * Gc * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * DD * G * (BcFinal − 1. * BcStart * gamma) * Gc * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * CCc * DD * G * (Pc3Final − 1. * gamma * Pc3Start) * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * CC * DDc * G * (Pc3Final − 1. * gamma * Pc3Start) * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * Power(G, 2) * Gc * (Pc3Final − 1. * gamma * Pc3Start) * (T53Final − 1. * t * T53Final + gamma * t * T53Start) + 1. * CCc * DD * DDc * (AFinal − 1. * AStart * gamma) * Tc33 + CCc * DDc * G * (−AFinal + AStart * gamma) * Tc33 + 1. * DDc * G * (AFinal − 1. * AStart * gamma) * Gc * Tc33 + DD * DDc * (−AFinal + AStart * gamma) * Gc * Tc33 + 1. * CCc * DDc * G * (P3Final − 1. * gamma * P3Start) * Tc33 + 1. * DD * DDc * Gc * (P3Final − 1. * gamma * P3Start) * Tc33 + CCc * DD * DDc * (−P3Final + gamma * P3Start) * Tc33 + DDc * G * Gc * (−P3Final + gamma * P3Start) * Tc33 − CCc * DD * (BcFinal − BcStart * gamma) * (AFinal − AFinal * t + AStart * gamma * t) * Tc33 − G * (BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t + AStart * gamma * t) * Tc33 + 1. * CCc * G * (BcFinal − 1. * BcStart * gamma) * (AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc33 + 1. * DD * (BcFinal − 1. * BcStart * gamma) * Gc * (AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc33 − CCc * DD * (AFinal − AStart * gamma) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc33 − G * (AFinal − AStart * gamma) * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) *

Tc33 − CCc ∗ G ∗ (P3Final − gamma ∗ P3Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 − DD ∗ Gc ∗ (P3Final − gamma ∗ P3Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 + 1. ∗ CCc ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 + 1. ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 + 1. ∗ CCc ∗ DD ∗ (P3Final − 1. ∗ gamma ∗ P3Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 + 1. ∗ G ∗ Gc ∗ (P3Final − 1. ∗ gamma ∗ P3Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc33 − CCc ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ Tc33 − DD ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ Tc33 + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ Tc33 + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ Tc33 + 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ (T53Final − 1. ∗ gamma ∗ T53Start) ∗ Tc33 + 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ (T53Final − 1. ∗ gamma ∗ T53Start) ∗ Tc33 + CCc ∗ DD ∗ DDc ∗ G ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 + DDc ∗ Power(G, 2) ∗ Gc ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 + CCc ∗ Power(G, 2) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 + DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T53Final + gamma ∗ T53Start) ∗ Tc33 − CCc ∗ Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − DD

∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T53Final − t ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T53Final − 1. ∗ t ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + 1. ∗ Power(G, 2) ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (T53Final − 1. ∗ t ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ T33 ∗ (Tc53Final − 1. ∗ gamma ∗ Tc53Start) + 1. ∗ CC ∗ DD ∗ Power(Gc, 2) ∗ T33 ∗ (Tc53Final − 1. ∗ gamma ∗ Tc53Start) + CCc ∗ DD ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + DDc ∗ G ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + CC ∗ DDc ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ CC ∗ Power(Gc, 2) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + G ∗ Power(Gc, 2) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + DD ∗ DDc ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + CCc ∗ G ∗ Gc ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) + CC ∗ Power(Gc, 2) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) ∗ (−Tc53Final + gamma ∗ Tc53Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗

t) * (−Tc53Final + gamma * Tc53Start) + CC * DD * DDc * Gc * T33 *
(−Tc53Final + gamma * Tc53Start) + DD * G * Power(Gc, 2) * T33 *
(−Tc53Final + gamma * Tc53Start) − 1. * CC * DDc * Gc * (BFinal −
BFinal * t + BStart * gamma * t) * T33 * (−Tc53Final + gamma *
Tc53Start) + DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t)
* T33 * (−Tc53Final + gamma * Tc53Start) + CC * Power(Gc, 2) * (BFinal
− BFinal * t + BStart * gamma * t) * T33 * (−Tc53Final + gamma *
Tc53Start) − 1. * G * Power(Gc, 2) * (BFinal − BFinal * t + BStart *
gamma * t) * T33 * (−Tc53Final + gamma * Tc53Start) − CCc * DD *
(AFinal − AStart * gamma) * Gc * (Tc53Final − t * Tc53Final + gamma *
t * Tc53Start) − DDc * G * (AFinal − AStart * gamma) * Gc * (Tc53Final
− t * Tc53Final + gamma * t * Tc53Start) − CC * DDc * (BFinal − BStart
* gamma) * Gc * (Tc53Final − t * Tc53Final + gamma * t * Tc53Start) −
G * (BFinal − BStart * gamma) * Power(Gc, 2) * (Tc53Final − t *
Tc53Final + gamma * t * Tc53Start) − DD * DDc * Gc * (P3Final − gamma
* P3Start) * (Tc53Final − t * Tc53Final + gamma * t * Tc53Start) − CCc
* G * Gc * (P3Final − gamma * P3Start) * (Tc53Final − t * Tc53Final +
gamma * t * Tc53Start) − CC * Power(Gc, 2) * (P3Final − gamma *
P3Start) * (Tc53Final − t * Tc53Final + gamma * t * Tc53Start) − DDc *
G * (BFinal − BStart * gamma) * Gc * T33 * (Tc53Final − t * Tc53Final
+ gamma * t * Tc53Start) − CC * (BFinal − BStart * gamma) * Power(Gc,
2) * T33 * (Tc53Final − t * Tc53Final + gamma * t * Tc53Start) + 1. *
DD * DDc * (AFinal − 1. * AStart * gamma) * Gc * (Tc53Final − 1. * t *
Tc53Final + gamma * t * Tc53Start) + 1. * CCc * G * (AFinal − 1. *
AStart * gamma) * Gc * (Tc53Final − 1. * t * Tc53Final + gamma * t *
Tc53Start) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * Gc *
(Tc53Final − 1. * t * Tc53Final + gamma * t * Tc53Start) + 1. * CC *

(BFinal − 1. ∗ BStart ∗ gamma) ∗ Power(Gc, 2) ∗ (Tc53Final − 1. ∗ t ∗
Tc53Final + gamma ∗ t ∗ Tc53Start) + 1. ∗ CCc ∗ DD ∗ Gc ∗ (P3Final −
1. ∗ gamma ∗ P3Start) ∗ (Tc53Final − 1. ∗ t ∗ Tc53Final + gamma ∗ t ∗
Tc53Start) + 1. ∗ CC ∗ DDc ∗ Gc ∗ (P3Final − 1. ∗ gamma ∗ P3Start) ∗
(Tc53Final − 1. ∗ t ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + 1. ∗ G ∗
Power(Gc, 2) ∗ (P3Final − 1. ∗ gamma ∗ P3Start) ∗ (Tc53Final − 1. ∗ t
∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + 1. ∗ CC ∗ DDc ∗ (BFinal − 1. ∗
BStart ∗ gamma) ∗ Gc ∗ T33 ∗ (Tc53Final − 1. ∗ t ∗ Tc53Final + gamma ∗
t ∗ Tc53Start) + 1. ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Power(Gc,
2) ∗ T33 ∗ (Tc53Final − 1. ∗ t ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);
Ht9 = 1. ∗ CC ∗ DDc ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) + CC ∗ DD ∗
DDc ∗ (−AcFinal + AcStart ∗ gamma) + 1. ∗ CCc ∗ DDc ∗ G ∗ (AFinal − 1.
∗ AStart ∗ gamma) + CCc ∗ DD ∗ DDc ∗ (−AFinal + AStart ∗ gamma) + 1. ∗
CC ∗ DD ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc + CC ∗ G ∗ (−AcFinal +
AcStart ∗ gamma) ∗ Gc + 1. ∗ CCc ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma)
∗ Gc + CCc ∗ G ∗ (−AFinal + AStart ∗ gamma) ∗ Gc − DDc ∗ G ∗ (BFinal −
BStart ∗ gamma) ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) − CC ∗
(BFinal − BStart ∗ gamma) ∗ Gc ∗ (AcFinal − AcFinal ∗ t + AcStart ∗
gamma ∗ t) − CC ∗ DDc ∗ (P4Final − gamma ∗ P4Start) ∗ (AcFinal −
AcFinal ∗ t + AcStart ∗ gamma ∗ t) − DD ∗ Gc ∗ (P4Final − gamma ∗
P4Start) ∗ (AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) + 1. ∗ CC ∗
DDc ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (AcFinal − 1. ∗ AcFinal ∗ t +
AcStart ∗ gamma ∗ t) + 1. ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗
(AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) + 1. ∗ DD ∗ DDc ∗
(P4Final − 1. ∗ gamma ∗ P4Start) ∗ (AcFinal − 1. ∗ AcFinal ∗ t +
AcStart ∗ gamma ∗ t) + 1. ∗ CC ∗ Gc ∗ (P4Final − 1. ∗ gamma ∗ P4Start)
∗ (AcFinal − 1. ∗ AcFinal ∗ t + AcStart ∗ gamma ∗ t) − CCc ∗ G ∗

(BcFinal − BcStart ∗ gamma) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − DD ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (Pc4Final − gamma ∗ Pc4Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − DDc ∗ G ∗ (Pc4Final − gamma ∗ Pc4Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ DD ∗ DDc ∗ (Pc4Final − 1. ∗ gamma ∗ Pc4Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (Pc4Final − 1. ∗ gamma ∗ Pc4Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) − CCc ∗ G ∗ (AFinal − AStart ∗ gamma) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DD ∗ (AFinal − AStart ∗ gamma) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (P4Final − gamma ∗ P4Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − G ∗ Gc ∗ (P4Final − gamma ∗ P4Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ DD ∗ Gc ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ DDc ∗ (Pc4Final − gamma ∗ Pc4Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − G ∗ Gc ∗ (Pc4Final − gamma ∗ Pc4Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗

CC * DDc * (AcFinal - 1. * AcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * G * (AcFinal - 1. * AcStart * gamma) * Gc * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * DDc * G * (Pc4Final - 1. * gamma * Pc4Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * CC * Gc * (Pc4Final - 1. * gamma * Pc4Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) - CC * DDc * (AcFinal - AcStart * gamma) * (P4Final - P4Final * t + gamma * P4Start * t) - CCc * DD * (BcFinal - BcStart * gamma) * (P4Final - P4Final * t + gamma * P4Start * t) - DD * (AcFinal - AcStart * gamma) * Gc * (P4Final - P4Final * t + gamma * P4Start * t) - G * (BcFinal - BcStart * gamma) * Gc * (P4Final - P4Final * t + gamma * P4Start * t) - DD * DDc * (Pc4Final - gamma * Pc4Start) * (P4Final - P4Final * t + gamma * P4Start * t) - CCc * G * (Pc4Final - gamma * Pc4Start) * (P4Final - P4Final * t + gamma * P4Start * t) - CC * Gc * (Pc4Final - gamma * Pc4Start) * (P4Final - P4Final * t + gamma * P4Start * t) + 1. * DD * DDc * (AcFinal - 1. * AcStart * gamma) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * CCc * G * (BcFinal - 1. * BcStart * gamma) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * CC * (AcFinal - 1. * AcStart * gamma) * Gc * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * DD * (BcFinal - 1. * BcStart * gamma) * Gc * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * CCc * DD * (Pc4Final - 1. * gamma * Pc4Start) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * CC * DDc * (Pc4Final - 1. * gamma * Pc4Start) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) + 1. * G * Gc * (Pc4Final - 1. * gamma * Pc4Start) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) - CCc * DD * (AFinal - AStart * gamma) * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) - DDc * G *

(AFinal − AStart ∗ gamma) ∗ (Pc4Final − Pc4Final ∗ t + gamma ∗
Pc4Start ∗ t) − CC ∗ DDc ∗ (BFinal − BStart ∗ gamma) ∗ (Pc4Final −
Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) − G ∗ (BFinal − BStart ∗ gamma) ∗
Gc ∗ (Pc4Final − Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) − DD ∗ DDc ∗
(P4Final − gamma ∗ P4Start) ∗ (Pc4Final − Pc4Final ∗ t + gamma ∗
Pc4Start ∗ t) − CCc ∗ G ∗ (P4Final − gamma ∗ P4Start) ∗ (Pc4Final −
Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) − CC ∗ Gc ∗ (P4Final − gamma ∗
P4Start) ∗ (Pc4Final − Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + 1. ∗ DD
∗ DDc ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (Pc4Final − 1. ∗ Pc4Final ∗ t
+ gamma ∗ Pc4Start ∗ t) + 1. ∗ CCc ∗ G ∗ (AFinal − 1. ∗ AStart ∗
gamma) ∗ (Pc4Final − 1. ∗ Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + 1. ∗
DDc ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc4Final − 1. ∗ Pc4Final ∗
t + gamma ∗ Pc4Start ∗ t) + 1. ∗ CC ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗
Gc ∗ (Pc4Final − 1. ∗ Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + 1. ∗ CCc
∗ DD ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (Pc4Final − 1. ∗ Pc4Final ∗
t + gamma ∗ Pc4Start ∗ t) + 1. ∗ CC ∗ DDc ∗ (P4Final − 1. ∗ gamma ∗
P4Start) ∗ (Pc4Final − 1. ∗ Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + 1.
∗ G ∗ Gc ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (Pc4Final − 1. ∗
Pc4Final ∗ t + gamma ∗ Pc4Start ∗ t) + 1. ∗ CC ∗ DD ∗ DDc ∗ (AcFinal −
1. ∗ AcStart ∗ gamma) ∗ T34 + DD ∗ DDc ∗ G ∗ (−AcFinal + AcStart ∗
gamma) ∗ T34 + 1. ∗ DD ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗
T34 + CC ∗ DD ∗ (−AcFinal + AcStart ∗ gamma) ∗ Gc ∗ T34 + 1. ∗ DD ∗
DDc ∗ G ∗ (Pc4Final − 1. ∗ gamma ∗ Pc4Start) ∗ T34 + 1. ∗ CC ∗ DD ∗ Gc
∗ (Pc4Final − 1. ∗ gamma ∗ Pc4Start) ∗ T34 + CC ∗ DD ∗ DDc ∗
(−Pc4Final + gamma ∗ Pc4Start) ∗ T34 + DD ∗ G ∗ Gc ∗ (−Pc4Final +
gamma ∗ Pc4Start) ∗ T34 − CC ∗ DDc ∗ (BFinal − BStart ∗ gamma) ∗
(AcFinal − AcFinal ∗ t + AcStart ∗ gamma ∗ t) ∗ T34 − G ∗ (BFinal −

BStart * gamma) * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) * T34 + 1. * DDc * G * (BFinal - 1. * BStart * gamma) * (AcFinal - 1. * AcFinal * t + AcStart * gamma * t) * T34 + 1. * CC * (BFinal - 1. * BStart * gamma) * Gc * (AcFinal - 1. * AcFinal * t + AcStart * gamma * t) * T34 - CC * DDc * (AcFinal - AcStart * gamma) * (BFinal - BFinal * t + BStart * gamma * t) * T34 - G * (AcFinal - AcStart * gamma) * Gc * (BFinal - BFinal * t + BStart * gamma * t) * T34 - DDc * G * (Pc4Final - gamma * Pc4Start) * (BFinal - BFinal * t + BStart * gamma * t) * T34 - CC * Gc * (Pc4Final - gamma * Pc4Start) * (BFinal - BFinal * t + BStart * gamma * t) * T34 + 1. * DDc * G * (AcFinal - 1. * AcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T34 + 1. * CC * (AcFinal - 1. * AcStart * gamma) * Gc * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T34 + 1. * CC * DDc * (Pc4Final - 1. * gamma * Pc4Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T34 + 1. * G * Gc * (Pc4Final - 1. * gamma * Pc4Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T34 - DDc * G * (BFinal - BStart * gamma) * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * T34 - CC * (BFinal - BStart * gamma) * Gc * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * T34 + 1. * CC * DDc * (BFinal - 1. * BStart * gamma) * (Pc4Final - 1. * Pc4Final * t + gamma * Pc4Start * t) * T34 + 1. * G * (BFinal - 1. * BStart * gamma) * Gc * (Pc4Final - 1. * Pc4Final * t + gamma * Pc4Start * t) * T34 + CC * DDc * G * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T54Final + gamma * T54Start) - 1. * DD * DDc * G * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T54Final + gamma * T54Start) - 1. * CC * G * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T54Final + gamma * T54Start) + DD * G * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T54Final + gamma *

T54Start) + CCc * DD * G * (BcFinal - BcFinal * t + BcStart * gamma *
t) * (-T54Final + gamma * T54Start) - 1. * CCc * Power(G, 2) *
(BcFinal - BcFinal * t + BcStart * gamma * t) * (-T54Final + gamma *
T54Start) - 1. * DD * G * Gc * (BcFinal - BcFinal * t + BcStart *
gamma * t) * (-T54Final + gamma * T54Start) + Power(G, 2) * Gc *
(BcFinal - BcFinal * t + BcStart * gamma * t) * (-T54Final + gamma *
T54Start) - 1. * CCc * DD * G * (Pc4Final - Pc4Final * t + gamma *
Pc4Start * t) * (-T54Final + gamma * T54Start) - 1. * CC * DDc * G *
(Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * (-T54Final + gamma
* T54Start) + DD * DDc * G * (Pc4Final - Pc4Final * t + gamma *
Pc4Start * t) * (-T54Final + gamma * T54Start) + CCc * Power(G, 2) *
(Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * (-T54Final + gamma
* T54Start) + CC * G * Gc * (Pc4Final - Pc4Final * t + gamma *
Pc4Start * t) * (-T54Final + gamma * T54Start) - 1. * Power(G, 2) * Gc
* (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * (-T54Final +
gamma * T54Start) - CC * DDc * G * (AcFinal - AcStart * gamma) *
(T54Final - t * T54Final + gamma * t * T54Start) - CCc * DD * G *
(BcFinal - BcStart * gamma) * (T54Final - t * T54Final + gamma * t *
T54Start) - DD * G * (AcFinal - AcStart * gamma) * Gc * (T54Final - t
* T54Final + gamma * t * T54Start) - Power(G, 2) * (BcFinal - BcStart
* gamma) * Gc * (T54Final - t * T54Final + gamma * t * T54Start) - DD
* DDc * G * (Pc4Final - gamma * Pc4Start) * (T54Final - t * T54Final +
gamma * t * T54Start) - CCc * Power(G, 2) * (Pc4Final - gamma *
Pc4Start) * (T54Final - t * T54Final + gamma * t * T54Start) - CC * G
* Gc * (Pc4Final - gamma * Pc4Start) * (T54Final - t * T54Final +
gamma * t * T54Start) + 1. * DD * DDc * G * (AcFinal - 1. * AcStart *
gamma) * (T54Final - 1. * t * T54Final + gamma * t * T54Start) + 1. *

CCc * Power(G, 2) * (BcFinal − 1. * BcStart * gamma) * (T54Final − 1.
* t * T54Final + gamma * t * T54Start) + 1. * CC * G * (AcFinal − 1. *
AcStart * gamma) * Gc * (T54Final − 1. * t * T54Final + gamma * t *
T54Start) + 1. * DD * G * (BcFinal − 1. * BcStart * gamma) * Gc *
(T54Final − 1. * t * T54Final + gamma * t * T54Start) + 1. * CCc * DD
* G * (Pc4Final − 1. * gamma * Pc4Start) * (T54Final − 1. * t *
T54Final + gamma * t * T54Start) + 1. * CC * DDc * G * (Pc4Final − 1.
* gamma * Pc4Start) * (T54Final − 1. * t * T54Final + gamma * t *
T54Start) + 1. * Power(G, 2) * Gc * (Pc4Final − 1. * gamma * Pc4Start)
* (T54Final − 1. * t * T54Final + gamma * t * T54Start) + 1. * CCc *
DD * DDc * (AFinal − 1. * AStart * gamma) * Tc34 + CCc * DDc * G *
(−AFinal + AStart * gamma) * Tc34 + 1. * DDc * G * (AFinal − 1. *
AStart * gamma) * Gc * Tc34 + DD * DDc * (−AFinal + AStart * gamma) *
Gc * Tc34 + 1. * CCc * DDc * G * (P4Final − 1. * gamma * P4Start) *
Tc34 + 1. * DD * DDc * Gc * (P4Final − 1. * gamma * P4Start) * Tc34 +
CCc * DD * DDc * (−P4Final + gamma * P4Start) * Tc34 + DDc * G * Gc *
(−P4Final + gamma * P4Start) * Tc34 − CCc * DD * (BcFinal − BcStart *
gamma) * (AFinal − AFinal * t + AStart * gamma * t) * Tc34 − G *
(BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t + AStart *
gamma * t) * Tc34 + 1. * CCc * G * (BcFinal − 1. * BcStart * gamma) *
(AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc34 + 1. * DD *
(BcFinal − 1. * BcStart * gamma) * Gc * (AFinal − 1. * AFinal * t +
AStart * gamma * t) * Tc34 − CCc * DD * (AFinal − AStart * gamma) *
(BcFinal − BcFinal * t + BcStart * gamma * t) * Tc34 − G * (AFinal −
AStart * gamma) * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) *
Tc34 − CCc * G * (P4Final − gamma * P4Start) * (BcFinal − BcFinal * t
+ BcStart * gamma * t) * Tc34 − DD * Gc * (P4Final − gamma * P4Start)

∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc34 + 1. ∗ CCc ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc34 + 1. ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc34 + 1. ∗ CCc ∗ DD ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc34 + 1. ∗ G ∗ Gc ∗ (P4Final − 1. ∗ gamma ∗ P4Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc34 − CCc ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (P4Final − P4Final ∗ t + gamma ∗ P4Start ∗ t) ∗ Tc34 − DD ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (P4Final − P4Final ∗ t + gamma ∗ P4Start ∗ t) ∗ Tc34 + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P4Final − 1. ∗ P4Final ∗ t + gamma ∗ P4Start ∗ t) ∗ Tc34 + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (P4Final − 1. ∗ P4Final ∗ t + gamma ∗ P4Start ∗ t) ∗ Tc34 + 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ (T54Final − 1. ∗ gamma ∗ T54Start) ∗ Tc34 + 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ (T54Final − 1. ∗ gamma ∗ T54Start) ∗ Tc34 + CCc ∗ DD ∗ DDc ∗ G ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 + DDc ∗ Power(G, 2) ∗ Gc ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 + CCc ∗ Power(G, 2) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 + DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T54Final + gamma ∗ T54Start) ∗ Tc34 − CCc ∗ Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ (T54Final − t ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − DD ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T54Final − t ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal − 1. ∗

BcStart * gamma) * (T54Final - 1. * t * T54Final + gamma * t *
T54Start) * Tc34 + 1. * Power(G, 2) * (BcFinal - 1. * BcStart * gamma)
* Gc * (T54Final - 1. * t * T54Final + gamma * t * T54Start) * Tc34 +
1. * DD * DDc * G * Gc * T34 * (Tc54Final - 1. * gamma * Tc54Start) +
1. * CC * DD * Power(Gc, 2) * T34 * (Tc54Final - 1. * gamma *
Tc54Start) + CCc * DD * Gc * (AFinal - AFinal * t + AStart * gamma *
t) * (-Tc54Final + gamma * Tc54Start) - 1. * DD * DDc * Gc * (AFinal -
AFinal * t + AStart * gamma * t) * (-Tc54Final + gamma * Tc54Start) -
1. * CCc * G * Gc * (AFinal - AFinal * t + AStart * gamma * t) *
(-Tc54Final + gamma * Tc54Start) + DDc * G * Gc * (AFinal - AFinal * t
+ AStart * gamma * t) * (-Tc54Final + gamma * Tc54Start) + CC * DDc *
Gc * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc54Final + gamma
* Tc54Start) - 1. * DDc * G * Gc * (BFinal - BFinal * t + BStart *
gamma * t) * (-Tc54Final + gamma * Tc54Start) - 1. * CC * Power(Gc, 2)
* (BFinal - BFinal * t + BStart * gamma * t) * (-Tc54Final + gamma *
Tc54Start) + G * Power(Gc, 2) * (BFinal - BFinal * t + BStart * gamma
* t) * (-Tc54Final + gamma * Tc54Start) - 1. * CCc * DD * Gc *
(P4Final - P4Final * t + gamma * P4Start * t) * (-Tc54Final + gamma *
Tc54Start) - 1. * CC * DDc * Gc * (P4Final - P4Final * t + gamma *
P4Start * t) * (-Tc54Final + gamma * Tc54Start) + DD * DDc * Gc *
(P4Final - P4Final * t + gamma * P4Start * t) * (-Tc54Final + gamma *
Tc54Start) + CCc * G * Gc * (P4Final - P4Final * t + gamma * P4Start *
t) * (-Tc54Final + gamma * Tc54Start) + CC * Power(Gc, 2) * (P4Final -
P4Final * t + gamma * P4Start * t) * (-Tc54Final + gamma * Tc54Start)
- 1. * G * Power(Gc, 2) * (P4Final - P4Final * t + gamma * P4Start *
t) * (-Tc54Final + gamma * Tc54Start) + CC * DD * DDc * Gc * T34 *
(-Tc54Final + gamma * Tc54Start) + DD * G * Power(Gc, 2) * T34 *

(−Tc54Final + gamma * Tc54Start) − 1. * CC * DDc * Gc * (BFinal − BFinal * t + BStart * gamma * t) * T34 * (−Tc54Final + gamma * Tc54Start) + DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t) * T34 * (−Tc54Final + gamma * Tc54Start) + CC * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T34 * (−Tc54Final + gamma * Tc54Start) − 1. * G * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T34 * (−Tc54Final + gamma * Tc54Start) − CCc * DD * (AFinal − AStart * gamma) * Gc * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − DDc * G * (AFinal − AStart * gamma) * Gc * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − CC * DDc * (BFinal − BStart * gamma) * Gc * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − G * (BFinal − BStart * gamma) * Power(Gc, 2) * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − DD * DDc * Gc * (P4Final − gamma * P4Start) * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − CCc * G * Gc * (P4Final − gamma * P4Start) * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − CC * Power(Gc, 2) * (P4Final − gamma * P4Start) * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − DDc * G * (BFinal − BStart * gamma) * Gc * T34 * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) − CC * (BFinal − BStart * gamma) * Power(Gc, 2) * T34 * (Tc54Final − t * Tc54Final + gamma * t * Tc54Start) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * Gc * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * Gc * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * Gc * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * CC * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * CCc * DD * Gc * (P4Final −

1. * gamma * P4Start) * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * CC * DDc * Gc * (P4Final − 1. * gamma * P4Start) * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * G * Power(Gc, 2) * (P4Final − 1. * gamma * P4Start) * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * Gc * T34 * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start) + 1. * G * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * T34 * (Tc54Final − 1. * t * Tc54Final + gamma * t * Tc54Start);
Ht10 = 1. * CC * DDc * G * (AcFinal − 1. * AcStart * gamma) + CC * DD * DDc * (−AcFinal + AcStart * gamma) + 1. * CCc * DDc * G * (AFinal − 1. * AStart * gamma) + CCc * DD * DDc * (−AFinal + AStart * gamma) + 1. * CC * DD * (AcFinal − 1. * AcStart * gamma) * Gc + CC * G * (−AcFinal + AcStart * gamma) * Gc + 1. * CCc * DD * (AFinal − 1. * AStart * gamma) * Gc + CCc * G * (−AFinal + AStart * gamma) * Gc − DDc * G * (BFinal − BStart * gamma) * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * (BFinal − BStart * gamma) * Gc * (AcFinal − AcFinal * t + AcStart * gamma * t) − CC * DDc * (P5Final − gamma * P5Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) − DD * Gc * (P5Final − gamma * P5Start) * (AcFinal − AcFinal * t + AcStart * gamma * t) + 1. * CC * DDc * (BFinal − 1. * BStart * gamma) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * G * (BFinal − 1. * BStart * gamma) * Gc * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * DD * DDc * (P5Final − 1. * gamma * P5Start) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) + 1. * CC * Gc * (P5Final − 1. * gamma * P5Start) * (AcFinal − 1. * AcFinal * t + AcStart * gamma * t) − CCc * G * (BcFinal − BcStart * gamma) * (AFinal − AFinal * t + AStart * gamma * t) − DD * (BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t +

AStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) − DDc ∗ G ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (AFinal − AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ DD ∗ DDc ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (AFinal − 1. ∗ AFinal ∗ t + AStart ∗ gamma ∗ t) − CCc ∗ G ∗ (AFinal − AStart ∗ gamma) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DD ∗ (AFinal − AStart ∗ gamma) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − CCc ∗ DD ∗ (P5Final − gamma ∗ P5Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) − G ∗ Gc ∗ (P5Final − gamma ∗ P5Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ DD ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ G ∗ (AFinal − 1. ∗ AStart ∗ gamma) ∗ Gc ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ CCc ∗ G ∗ (P5Final − 1. ∗ gamma ∗ P5Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ DD ∗ Gc ∗ (P5Final − 1. ∗ gamma ∗ P5Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ DDc ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) − G ∗ Gc ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ CC ∗ DDc ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗

Gc ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ DDc ∗ G ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ CC ∗ Gc ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − CC ∗ DDc ∗ (AcFinal − AcStart ∗ gamma) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − CCc ∗ DD ∗ (BcFinal − BcStart ∗ gamma) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − DD ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − G ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − DD ∗ DDc ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − CCc ∗ G ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) − CC ∗ Gc ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (P5Final − P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ DD ∗ DDc ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ CCc ∗ G ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ CC ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ Gc ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ DD ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ Gc ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ CCc ∗ DD ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ CC ∗ DDc ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) + 1. ∗ G ∗ Gc ∗ (Pc5Final − 1. ∗ gamma ∗ Pc5Start) ∗ (P5Final − 1. ∗ P5Final ∗ t + gamma ∗ P5Start ∗ t) − CCc ∗ DD ∗ (AFinal − AStart ∗ gamma) ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) − DDc ∗ G ∗ (AFinal − AStart ∗ gamma) ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) − CC ∗ DDc ∗ (BFinal − BStart ∗ gamma) ∗ (Pc5Final −

Pc5Final * t + gamma * Pc5Start * t) − G * (BFinal − BStart * gamma) *
Gc * (Pc5Final − Pc5Final * t + gamma * Pc5Start * t) − DD * DDc *
(P5Final − gamma * P5Start) * (Pc5Final − Pc5Final * t + gamma *
Pc5Start * t) − CCc * G * (P5Final − gamma * P5Start) * (Pc5Final −
Pc5Final * t + gamma * Pc5Start * t) − CC * Gc * (P5Final − gamma *
P5Start) * (Pc5Final − Pc5Final * t + gamma * Pc5Start * t) + 1. * DD
* DDc * (AFinal − 1. * AStart * gamma) * (Pc5Final − 1. * Pc5Final * t
+ gamma * Pc5Start * t) + 1. * CCc * G * (AFinal − 1. * AStart *
gamma) * (Pc5Final − 1. * Pc5Final * t + gamma * Pc5Start * t) + 1. *
DDc * G * (BFinal − 1. * BStart * gamma) * (Pc5Final − 1. * Pc5Final *
t + gamma * Pc5Start * t) + 1. * CC * (BFinal − 1. * BStart * gamma) *
Gc * (Pc5Final − 1. * Pc5Final * t + gamma * Pc5Start * t) + 1. * CCc
* DD * (P5Final − 1. * gamma * P5Start) * (Pc5Final − 1. * Pc5Final *
t + gamma * Pc5Start * t) + 1. * CC * DDc * (P5Final − 1. * gamma *
P5Start) * (Pc5Final − 1. * Pc5Final * t + gamma * Pc5Start * t) + 1.
* G * Gc * (P5Final − 1. * gamma * P5Start) * (Pc5Final − 1. *
Pc5Final * t + gamma * Pc5Start * t) + 1. * CC * DD * DDc * (AcFinal −
1. * AcStart * gamma) * T35 + DD * DDc * G * (−AcFinal + AcStart *
gamma) * T35 + 1. * DD * G * (AcFinal − 1. * AcStart * gamma) * Gc *
T35 + CC * DD * (−AcFinal + AcStart * gamma) * Gc * T35 + 1. * DD *
DDc * G * (Pc5Final − 1. * gamma * Pc5Start) * T35 + 1. * CC * DD * Gc
* (Pc5Final − 1. * gamma * Pc5Start) * T35 + CC * DD * DDc *
(−Pc5Final + gamma * Pc5Start) * T35 + DD * G * Gc * (−Pc5Final +
gamma * Pc5Start) * T35 − CC * DDc * (BFinal − BStart * gamma) *
(AcFinal − AcFinal * t + AcStart * gamma * t) * T35 − G * (BFinal −
BStart * gamma) * Gc * (AcFinal − AcFinal * t + AcStart * gamma * t) *
T35 + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * (AcFinal − 1. *

AcFinal * t + AcStart * gamma * t) * T35 + 1. * CC * (BFinal - 1. * BStart * gamma) * Gc * (AcFinal - 1. * AcFinal * t + AcStart * gamma * t) * T35 - CC * DDc * (AcFinal - AcStart * gamma) * (BFinal - BFinal * t + BStart * gamma * t) * T35 - G * (AcFinal - AcStart * gamma) * Gc * (BFinal - BFinal * t + BStart * gamma * t) * T35 - DDc * G * (Pc5Final - gamma * Pc5Start) * (BFinal - BFinal * t + BStart * gamma * t) * T35 - CC * Gc * (Pc5Final - gamma * Pc5Start) * (BFinal - BFinal * t + BStart * gamma * t) * T35 + 1. * DDc * G * (AcFinal - 1. * AcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T35 + 1. * CC * (AcFinal - 1. * AcStart * gamma) * Gc * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T35 + 1. * CC * DDc * (Pc5Final - 1. * gamma * Pc5Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T35 + 1. * G * Gc * (Pc5Final - 1. * gamma * Pc5Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T35 - DDc * G * (BFinal - BStart * gamma) * (Pc5Final - Pc5Final * t + gamma * Pc5Start * t) * T35 - CC * (BFinal - BStart * gamma) * Gc * (Pc5Final - Pc5Final * t + gamma * Pc5Start * t) * T35 + 1. * CC * DDc * (BFinal - 1. * BStart * gamma) * (Pc5Final - 1. * Pc5Final * t + gamma * Pc5Start * t) * T35 + 1. * G * (BFinal - 1. * BStart * gamma) * Gc * (Pc5Final - 1. * Pc5Final * t + gamma * Pc5Start * t) * T35 + CC * DDc * G * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T55Final + gamma * T55Start) - 1. * DD * DDc * G * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T55Final + gamma * T55Start) - 1. * CC * G * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T55Final + gamma * T55Start) + DD * G * Gc * (AcFinal - AcFinal * t + AcStart * gamma * t) * (-T55Final + gamma * T55Start) + CCc * DD * G * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T55Final + gamma * T55Start) - 1. * CCc * Power(G, 2) *

(BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T55Final + gamma ∗ T55Start) + Power(G, 2) ∗ Gc ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) + DD ∗ DDc ∗ G ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) + CCc ∗ Power(G, 2) ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) + CC ∗ G ∗ Gc ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc5Final − Pc5Final ∗ t + gamma ∗ Pc5Start ∗ t) ∗ (−T55Final + gamma ∗ T55Start) − CC ∗ DDc ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − CCc ∗ DD ∗ G ∗ (BcFinal − BcStart ∗ gamma) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − DD ∗ G ∗ (AcFinal − AcStart ∗ gamma) ∗ Gc ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − Power(G, 2) ∗ (BcFinal − BcStart ∗ gamma) ∗ Gc ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − DD ∗ DDc ∗ G ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − CCc ∗ Power(G, 2) ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) − CC ∗ G ∗ Gc ∗ (Pc5Final − gamma ∗ Pc5Start) ∗ (T55Final − t ∗ T55Final + gamma ∗ t ∗ T55Start) + 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal − 1. ∗ AcStart ∗ gamma) ∗ (T55Final − 1. ∗ t ∗ T55Final + gamma ∗ t ∗ T55Start) + 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (T55Final − 1. ∗ t ∗ T55Final + gamma ∗ t ∗ T55Start) + 1. ∗ CC ∗ G ∗ (AcFinal − 1. ∗

AcStart * gamma) * Gc * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + 1. * DD * G * (BcFinal − 1. * BcStart * gamma) * Gc * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + 1. * CCc * DD * G * (Pc5Final − 1. * gamma * Pc5Start) * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + 1. * CC * DDc * G * (Pc5Final − 1. * gamma * Pc5Start) * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + 1. * Power(G, 2) * Gc * (Pc5Final − 1. * gamma * Pc5Start) * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + 1. * CCc * DD * DDc * (AFinal − 1. * AStart * gamma) * Tc35 + CCc * DDc * G * (−AFinal + AStart * gamma) * Tc35 + 1. * DDc * G * (AFinal − 1. * AStart * gamma) * Gc * Tc35 + DD * DDc * (−AFinal + AStart * gamma) * Gc * Tc35 + 1. * CCc * DDc * G * (P5Final − 1. * gamma * P5Start) * Tc35 + 1. * DD * DDc * Gc * (P5Final − 1. * gamma * P5Start) * Tc35 + CCc * DD * DDc * (−P5Final + gamma * P5Start) * Tc35 + DDc * G * Gc * (−P5Final + gamma * P5Start) * Tc35 − CCc * DD * (BcFinal − BcStart * gamma) * (AFinal − AFinal * t + AStart * gamma * t) * Tc35 − G * (BcFinal − BcStart * gamma) * Gc * (AFinal − AFinal * t + AStart * gamma * t) * Tc35 + 1. * CCc * G * (BcFinal − 1. * BcStart * gamma) * (AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc35 + 1. * DD * (BcFinal − 1. * BcStart * gamma) * Gc * (AFinal − 1. * AFinal * t + AStart * gamma * t) * Tc35 − CCc * DD * (AFinal − AStart * gamma) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc35 − G * (AFinal − AStart * gamma) * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc35 − CCc * G * (P5Final − gamma * P5Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc35 − DD * Gc * (P5Final − gamma * P5Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc35 + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * (BcFinal − 1. * BcFinal * t +

BcStart * gamma * t) * Tc35 + 1. * DD * (AFinal − 1. * AStart * gamma) * Gc * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc35 + 1. * CCc * DD * (P5Final − 1. * gamma * P5Start) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc35 + 1. * G * Gc * (P5Final − 1. * gamma * P5Start) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc35 − CCc * G * (BcFinal − BcStart * gamma) * (P5Final − P5Final * t + gamma * P5Start * t) * Tc35 − DD * (BcFinal − BcStart * gamma) * Gc * (P5Final − P5Final * t + gamma * P5Start * t) * Tc35 + 1. * CCc * DD * (BcFinal − 1. * BcStart * gamma) * (P5Final − 1. * P5Final * t + gamma * P5Start * t) * Tc35 + 1. * G * (BcFinal − 1. * BcStart * gamma) * Gc * (P5Final − 1. * P5Final * t + gamma * P5Start * t) * Tc35 + 1. * CCc * DDc * Power(G, 2) * (T55Final − 1. * gamma * T55Start) * Tc35 + 1. * DD * DDc * G * Gc * (T55Final − 1. * gamma * T55Start) * Tc35 + CCc * DD * DDc * G * (−T55Final + gamma * T55Start) * Tc35 + DDc * Power(G, 2) * Gc * (−T55Final + gamma * T55Start) * Tc35 − 1. * CCc * DD * G * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) * Tc35 + CCc * Power(G, 2) * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) * Tc35 + DD * G * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) * Tc35 − 1. * Power(G, 2) * Gc * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) * Tc35 − CCc * Power(G, 2) * (BcFinal − BcStart * gamma) * (T55Final − t * T55Final + gamma * t * T55Start) * Tc35 − DD * G * (BcFinal − BcStart * gamma) * Gc * (T55Final − t * T55Final + gamma * t * T55Start) * Tc35 + 1. * CCc * DD * G * (BcFinal − 1. * BcStart * gamma) * (T55Final − 1. * t * T55Final + gamma * t * T55Start) * Tc35 + 1. * Power(G, 2) * (BcFinal − 1. * BcStart * gamma)

$* Gc * (T55Final - 1. * t * T55Final + gamma * t * T55Start) * Tc35 + 1. * DD * DDc * G * Gc * T35 * (Tc55Final - 1. * gamma * Tc55Start) + 1. * CC * DD * Power(Gc, 2) * T35 * (Tc55Final - 1. * gamma * Tc55Start) + CCc * DD * Gc * (AFinal - AFinal * t + AStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) - 1. * DD * DDc * Gc * (AFinal - AFinal * t + AStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) - 1. * CCc * G * Gc * (AFinal - AFinal * t + AStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) + DDc * G * Gc * (AFinal - AFinal * t + AStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) + CC * DDc * Gc * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) - 1. * DDc * G * Gc * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) - 1. * CC * Power(Gc, 2) * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) + G * Power(Gc, 2) * (BFinal - BFinal * t + BStart * gamma * t) * (-Tc55Final + gamma * Tc55Start) - 1. * CCc * DD * Gc * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) - 1. * CC * DDc * Gc * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) + DD * DDc * Gc * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) + CCc * G * Gc * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) + CC * Power(Gc, 2) * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) - 1. * G * Power(Gc, 2) * (P5Final - P5Final * t + gamma * P5Start * t) * (-Tc55Final + gamma * Tc55Start) + CC * DD * DDc * Gc * T35 * (-Tc55Final + gamma * Tc55Start) + DD * G * Power(Gc, 2) * T35 * (-Tc55Final + gamma * Tc55Start) - 1. * CC * DDc * Gc * (BFinal - BFinal * t + BStart * gamma * t) * T35 * (-Tc55Final + gamma *$

Tc55Start) + DDc * G * Gc * (BFinal − BFinal * t + BStart * gamma * t) * T35 * (−Tc55Final + gamma * Tc55Start) + CC * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T35 * (−Tc55Final + gamma * Tc55Start) − 1. * G * Power(Gc, 2) * (BFinal − BFinal * t + BStart * gamma * t) * T35 * (−Tc55Final + gamma * Tc55Start) − CCc * DD * (AFinal − AStart * gamma) * Gc * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − DDc * G * (AFinal − AStart * gamma) * Gc * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − CC * DDc * (BFinal − BStart * gamma) * Gc * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − G * (BFinal − BStart * gamma) * Power(Gc, 2) * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − DD * DDc * Gc * (P5Final − gamma * P5Start) * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − CCc * G * Gc * (P5Final − gamma * P5Start) * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − CC * Power(Gc, 2) * (P5Final − gamma * P5Start) * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − DDc * G * (BFinal − BStart * gamma) * Gc * T35 * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) − CC * (BFinal − BStart * gamma) * Power(Gc, 2) * T35 * (Tc55Final − t * Tc55Final + gamma * t * Tc55Start) + 1. * DD * DDc * (AFinal − 1. * AStart * gamma) * Gc * (Tc55Final − 1. * t * Tc55Final + gamma * t * Tc55Start) + 1. * CCc * G * (AFinal − 1. * AStart * gamma) * Gc * (Tc55Final − 1. * t * Tc55Final + gamma * t * Tc55Start) + 1. * DDc * G * (BFinal − 1. * BStart * gamma) * Gc * (Tc55Final − 1. * t * Tc55Final + gamma * t * Tc55Start) + 1. * CC * (BFinal − 1. * BStart * gamma) * Power(Gc, 2) * (Tc55Final − 1. * t * Tc55Final + gamma * t * Tc55Start) + 1. * CCc * DD * Gc * (P5Final − 1. * gamma * P5Start) * (Tc55Final − 1. * t * Tc55Final + gamma * t * Tc55Start) + 1. * CC * DDc * Gc * (P5Final − 1. * gamma * P5Start) *

(Tc55Final − 1. ∗ t ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 1. ∗ G ∗ Power(Gc, 2) ∗ (P5Final − 1. ∗ gamma ∗ P5Start) ∗ (Tc55Final − 1. ∗ t ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 1. ∗ CC ∗ DDc ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Gc ∗ T35 ∗ (Tc55Final − 1. ∗ t ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + 1. ∗ G ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Power(Gc, 2) ∗ T35 ∗ (Tc55Final − 1. ∗ t ∗ Tc55Final + gamma ∗ t ∗ Tc55Start);
Ht11 = 1. ∗ F ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) + 1. ∗ Fc ∗ (BFinal − 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P1Final − 1. ∗ gamma ∗ P1Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc1Final − gamma ∗ Pc1Start) ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P1Final − 1. ∗ P1Final ∗ t + gamma ∗ P1Start ∗ t) − (P1Final − gamma ∗ P1Start) ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc1Final − 1. ∗ Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) + F ∗ (−BcFinal + BcStart ∗ gamma) ∗ T31 + 1. ∗ F ∗ (Pc1Final − 1. ∗ gamma ∗ Pc1Start) ∗ T31 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T31 − (Pc1Final − gamma ∗ Pc1Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 − (BFinal − BStart ∗ gamma) ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ H ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) + H ∗ (Pc1Final − Pc1Final ∗ t + gamma ∗ Pc1Start ∗ t) ∗ (−T51Final + gamma ∗ T51Start) − H ∗

(Pc1Final − gamma ∗ Pc1Start) ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ H ∗ (T51Final − 1. ∗ t ∗ T51Final + gamma ∗ t ∗ T51Start) + Fc ∗ (−BFinal + BStart ∗ gamma) ∗ Tc31 + 1. ∗ Fc ∗ (P1Final − 1. ∗ gamma ∗ P1Start) ∗ Tc31 − (P1Final − gamma ∗ P1Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc31 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ Tc31 − (BcFinal − BcStart ∗ gamma) ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ Tc31 + 1. ∗ Fc ∗ H ∗ (T51Final − 1. ∗ gamma ∗ T51Start) ∗ Tc31 + H ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T51Final + gamma ∗ T51Start) ∗ Tc31 − (BcFinal − BcStart ∗ gamma) ∗ H ∗ (T51Final − t ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + 1. ∗ F ∗ Hc ∗ T31 ∗ (Tc51Final − 1. ∗ gamma ∗ Tc51Start) − 1. ∗ Hc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + Hc ∗ (P1Final − P1Final ∗ t + gamma ∗ P1Start ∗ t) ∗ (−Tc51Final + gamma ∗ Tc51Start) + Hc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T31 ∗ (−Tc51Final + gamma ∗ Tc51Start) − Hc ∗ (P1Final − gamma ∗ P1Start) ∗ (Tc51Final − t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − (BFinal − BStart ∗ gamma) ∗ Hc ∗ T31 ∗ (Tc51Final − t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Hc ∗ (Tc51Final − 1. ∗ t ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);
Ht12 = 1. ∗ F ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) + 1. ∗ Fc ∗ (BFinal − 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart

∗ gamma ∗ t) + 1. ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc2Final − gamma ∗ Pc2Start) ∗ (P2Final − P2Final ∗ t + gamma ∗ P2Start ∗ t) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P2Final − 1. ∗ P2Final ∗ t + gamma ∗ P2Start ∗ t) − (P2Final − gamma ∗ P2Start) ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc2Final − 1. ∗ Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) + F ∗ (−BcFinal + BcStart ∗ gamma) ∗ T32 + 1. ∗ F ∗ (Pc2Final − 1. ∗ gamma ∗ Pc2Start) ∗ T32 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T32 − (Pc2Final − gamma ∗ Pc2Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T32 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T32 − (BFinal − BStart ∗ gamma) ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ H ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ (−T52Final + gamma ∗ T52Start) + H ∗ (Pc2Final − Pc2Final ∗ t + gamma ∗ Pc2Start ∗ t) ∗ (−T52Final + gamma ∗ T52Start) − H ∗ (Pc2Final − gamma ∗ Pc2Start) ∗ (T52Final − t ∗ T52Final + gamma ∗ t ∗ T52Start) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ H ∗ (T52Final − 1. ∗ t ∗ T52Final + gamma ∗ t ∗ T52Start) + Fc ∗ (−BFinal + BStart ∗ gamma) ∗ Tc32 + 1. ∗ Fc ∗ (P2Final − 1. ∗ gamma ∗ P2Start) ∗ Tc32 − (P2Final − gamma ∗ P2Start) ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc32 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ Tc32 + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ Tc32 − (BcFinal − BcStart ∗ gamma) ∗ (P2Final − P2Final ∗ t + gamma ∗ P2Start ∗ t) ∗ Tc32 + 1. ∗ Fc ∗ H ∗ (T52Final − 1. ∗ gamma ∗ T52Start) ∗ Tc32 + H ∗ (BcFinal − BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗

(−T52Final + gamma ∗ T52Start) ∗ Tc32 − (BcFinal − BcStart ∗ gamma) ∗ H ∗ (T52Final − t ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + 1. ∗ F ∗ Hc ∗ T32 ∗ (Tc52Final − 1. ∗ gamma ∗ Tc52Start) − 1. ∗ Hc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ (−Tc52Final + gamma ∗ Tc52Start) + Hc ∗ (P2Final − P2Final ∗ t + gamma ∗ P2Start ∗ t) ∗ (−Tc52Final + gamma ∗ Tc52Start) + Hc ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T32 ∗ (−Tc52Final + gamma ∗ Tc52Start) − Hc ∗ (P2Final − gamma ∗ P2Start) ∗ (Tc52Final − t ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − (BFinal − BStart ∗ gamma) ∗ Hc ∗ T32 ∗ (Tc52Final − t ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ Hc ∗ (Tc52Final − 1. ∗ t ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);
Ht13 = 1. ∗ F ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) + 1. ∗ Fc ∗ (BFinal − 1. ∗ BStart ∗ gamma) − 2. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) + 1. ∗ (P3Final − 1. ∗ gamma ∗ P3Start) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) − 2. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) + 1. ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ (BFinal − 1. ∗ BFinal ∗ t + BStart ∗ gamma ∗ t) − (Pc3Final − gamma ∗ Pc3Start) ∗ (P3Final − P3Final ∗ t + gamma ∗ P3Start ∗ t) + 1. ∗ (BcFinal − 1. ∗ BcStart ∗ gamma) ∗ (P3Final − 1. ∗ P3Final ∗ t + gamma ∗ P3Start ∗ t) − (P3Final − gamma ∗ P3Start) ∗ (Pc3Final − Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (Pc3Final − 1. ∗ Pc3Final ∗ t + gamma ∗ Pc3Start ∗ t) + F ∗ (−BcFinal + BcStart ∗ gamma) ∗ T33 + 1. ∗ F ∗ (Pc3Final − 1. ∗ gamma ∗ Pc3Start) ∗ T33 + 1. ∗ (BFinal − 1. ∗ BStart ∗ gamma) ∗ (BcFinal − 1. ∗ BcFinal ∗ t + BcStart ∗ gamma ∗ t) ∗ T33 − (Pc3Final − gamma ∗ Pc3Start) ∗ (BFinal − BFinal ∗ t + BStart ∗ gamma ∗ t) ∗ T33 + 1. ∗ (BcFinal − 1. ∗ BcStart

```
* gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T33 -
(BFinal - BStart * gamma) * (Pc3Final - Pc3Final * t + gamma *
Pc3Start * t) * T33 - 1. * H * (BcFinal - BcFinal * t + BcStart *
gamma * t) * (-T53Final + gamma * T53Start) + H * (Pc3Final - Pc3Final
* t + gamma * Pc3Start * t) * (-T53Final + gamma * T53Start) - H *
(Pc3Final - gamma * Pc3Start) * (T53Final - t * T53Final + gamma * t *
T53Start) + 1. * (BcFinal - 1. * BcStart * gamma) * H * (T53Final - 1.
* t * T53Final + gamma * t * T53Start) + Fc * (-BFinal + BStart *
gamma) * Tc33 + 1. * Fc * (P3Final - 1. * gamma * P3Start) * Tc33 -
(P3Final - gamma * P3Start) * (BcFinal - BcFinal * t + BcStart * gamma
* t) * Tc33 + 1. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1. *
BcFinal * t + BcStart * gamma * t) * Tc33 + 1. * (BcFinal - 1. *
BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) *
Tc33 - (BcFinal - BcStart * gamma) * (P3Final - P3Final * t + gamma *
P3Start * t) * Tc33 + 1. * Fc * H * (T53Final - 1. * gamma * T53Start)
* Tc33 + H * (BcFinal - BcFinal * t + BcStart * gamma * t) *
(-T53Final + gamma * T53Start) * Tc33 - (BcFinal - BcStart * gamma) *
H * (T53Final - t * T53Final + gamma * t * T53Start) * Tc33 + 1. * F *
Hc * T33 * (Tc53Final - 1. * gamma * Tc53Start) - 1. * Hc * (BFinal -
BFinal * t + BStart * gamma * t) * (-Tc53Final + gamma * Tc53Start) +
Hc * (P3Final - P3Final * t + gamma * P3Start * t) * (-Tc53Final +
gamma * Tc53Start) + Hc * (BFinal - BFinal * t + BStart * gamma * t) *
T33 * (-Tc53Final + gamma * Tc53Start) - Hc * (P3Final - gamma *
P3Start) * (Tc53Final - t * Tc53Final + gamma * t * Tc53Start) -
(BFinal - BStart * gamma) * Hc * T33 * (Tc53Final - t * Tc53Final +
gamma * t * Tc53Start) + 1. * (BFinal - 1. * BStart * gamma) * Hc *
(Tc53Final - 1. * t * Tc53Final + gamma * t * Tc53Start);
```

$$Ht14 = 1. * F * (BcFinal - 1. * BcStart * gamma) + 1. * Fc * (BFinal - 1. * BStart * gamma) - 2. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) + 1. * (P4Final - 1. * gamma * P4Start) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) - 2. * (BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) + 1. * (Pc4Final - 1. * gamma * Pc4Start) * (BFinal - 1. * BFinal * t + BStart * gamma * t) - (Pc4Final - gamma * Pc4Start) * (P4Final - P4Final * t + gamma * P4Start * t) + 1. * (BcFinal - 1. * BcStart * gamma) * (P4Final - 1. * P4Final * t + gamma * P4Start * t) - (P4Final - gamma * P4Start) * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) + 1. * (BFinal - 1. * BStart * gamma) * (Pc4Final - 1. * Pc4Final * t + gamma * Pc4Start * t) + F * (-BcFinal + BcStart * gamma) * T34 + 1. * F * (Pc4Final - 1. * gamma * Pc4Start) * T34 + 1. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) * T34 - (Pc4Final - gamma * Pc4Start) * (BFinal - BFinal * t + BStart * gamma * t) * T34 + 1. * (BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) * T34 - (BFinal - BStart * gamma) * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * T34 - 1. * H * (BcFinal - BcFinal * t + BcStart * gamma * t) * (-T54Final + gamma * T54Start) + H * (Pc4Final - Pc4Final * t + gamma * Pc4Start * t) * (-T54Final + gamma * T54Start) - H * (Pc4Final - gamma * Pc4Start) * (T54Final - t * T54Final + gamma * t * T54Start) + 1. * (BcFinal - 1. * BcStart * gamma) * H * (T54Final - 1. * t * T54Final + gamma * t * T54Start) + Fc * (-BFinal + BStart * gamma) * Tc34 + 1. * Fc * (P4Final - 1. * gamma * P4Start) * Tc34 - (P4Final - gamma * P4Start) * (BcFinal - BcFinal * t + BcStart * gamma * t) * Tc34 + 1. * (BFinal - 1. * BStart * gamma) * (BcFinal - 1. *$$

```
BcFinal * t + BcStart * gamma * t) * Tc34 + 1. * (BcFinal - 1. *
BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart * gamma * t) *
Tc34 - (BcFinal - BcStart * gamma) * (P4Final - P4Final * t + gamma *
P4Start * t) * Tc34 + 1. * Fc * H * (T54Final - 1. * gamma * T54Start)
* Tc34 + H * (BcFinal - BcFinal * t + BcStart * gamma * t) *
(-T54Final + gamma * T54Start) * Tc34 - (BcFinal - BcStart * gamma) *
H * (T54Final - t * T54Final + gamma * t * T54Start) * Tc34 + 1. * F *
Hc * T34 * (Tc54Final - 1. * gamma * Tc54Start) - 1. * Hc * (BFinal -
BFinal * t + BStart * gamma * t) * (-Tc54Final + gamma * Tc54Start) +
Hc * (P4Final - P4Final * t + gamma * P4Start * t) * (-Tc54Final +
gamma * Tc54Start) + Hc * (BFinal - BFinal * t + BStart * gamma * t) *
T34 * (-Tc54Final + gamma * Tc54Start) - Hc * (P4Final - gamma *
P4Start) * (Tc54Final - t * Tc54Final + gamma * t * Tc54Start) -
(BFinal - BStart * gamma) * Hc * T34 * (Tc54Final - t * Tc54Final +
gamma * t * Tc54Start) + 1. * (BFinal - 1. * BStart * gamma) * Hc *
(Tc54Final - 1. * t * Tc54Final + gamma * t * Tc54Start);
Ht15 = 1. * F * (BcFinal - 1. * BcStart * gamma) + 1. * Fc * (BFinal -
1. * BStart * gamma) - 2. * (BFinal - 1. * BStart * gamma) * (BcFinal
- 1. * BcFinal * t + BcStart * gamma * t) + 1. * (P5Final - 1. * gamma
* P5Start) * (BcFinal - 1. * BcFinal * t + BcStart * gamma * t) - 2. *
(BcFinal - 1. * BcStart * gamma) * (BFinal - 1. * BFinal * t + BStart
* gamma * t) + 1. * (Pc5Final - 1. * gamma * Pc5Start) * (BFinal - 1.
* BFinal * t + BStart * gamma * t) - (Pc5Final - gamma * Pc5Start) *
(P5Final - P5Final * t + gamma * P5Start * t) + 1. * (BcFinal - 1. *
BcStart * gamma) * (P5Final - 1. * P5Final * t + gamma * P5Start * t)
- (P5Final - gamma * P5Start) * (Pc5Final - Pc5Final * t + gamma *
Pc5Start * t) + 1. * (BFinal - 1. * BStart * gamma) * (Pc5Final - 1. *
```

Pc5Final * t + gamma * Pc5Start * t) + F * (−BcFinal + BcStart * gamma) * T35 + 1. * F * (Pc5Final − 1. * gamma * Pc5Start) * T35 + 1. * (BFinal − 1. * BStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * T35 − (Pc5Final − gamma * Pc5Start) * (BFinal − BFinal * t + BStart * gamma * t) * T35 + 1. * (BcFinal − 1. * BcStart * gamma) * (BFinal − 1. * BFinal * t + BStart * gamma * t) * T35 − (BFinal − BStart * gamma) * (Pc5Final − Pc5Final * t + gamma * Pc5Start * t) * T35 − 1. * H * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) + H * (Pc5Final − Pc5Final * t + gamma * Pc5Start * t) * (−T55Final + gamma * T55Start) − H * (Pc5Final − gamma * Pc5Start) * (T55Final − t * T55Final + gamma * t * T55Start) + 1. * (BcFinal − 1. * BcStart * gamma) * H * (T55Final − 1. * t * T55Final + gamma * t * T55Start) + Fc * (−BFinal + BStart * gamma) * Tc35 + 1. * Fc * (P5Final − 1. * gamma * P5Start) * Tc35 − (P5Final − gamma * P5Start) * (BcFinal − BcFinal * t + BcStart * gamma * t) * Tc35 + 1. * (BFinal − 1. * BStart * gamma) * (BcFinal − 1. * BcFinal * t + BcStart * gamma * t) * Tc35 + 1. * (BcFinal − 1. * BcStart * gamma) * (BFinal − 1. * BFinal * t + BStart * gamma * t) * Tc35 − (BcFinal − BcStart * gamma) * (P5Final − P5Final * t + gamma * P5Start * t) * Tc35 + 1. * Fc * H * (T55Final − 1. * gamma * T55Start) * Tc35 + H * (BcFinal − BcFinal * t + BcStart * gamma * t) * (−T55Final + gamma * T55Start) * Tc35 − (BcFinal − BcStart * gamma) * H * (T55Final − t * T55Final + gamma * t * T55Start) * Tc35 + 1. * F * Hc * T35 * (Tc55Final − 1. * gamma * Tc55Start) − 1. * Hc * (BFinal − BFinal * t + BStart * gamma * t) * (−Tc55Final + gamma * Tc55Start) + Hc * (P5Final − P5Final * t + gamma * P5Start * t) * (−Tc55Final + gamma * Tc55Start) + Hc * (BFinal − BFinal * t + BStart * gamma * t) *

```
T35 * (-Tc55Final + gamma * Tc55Start) - Hc * (P5Final - gamma *
P5Start) * (Tc55Final - t * Tc55Final + gamma * t * Tc55Start) -
(BFinal - BStart * gamma) * Hc * T35 * (Tc55Final - t * Tc55Final +
gamma * t * Tc55Start) + 1. * (BFinal - 1. * BStart * gamma) * Hc *
(Tc55Final - 1. * t * Tc55Final + gamma * t * Tc55Start);
Ht16 = 0;
Ht17 = 0;
Ht18 = 0;
Ht19 = 0;
Ht20 = 0;
// Populating b with the values of -Ht.
complex_t *HtVec = b;
HtVec[0] = -Ht1;
HtVec[1] = -Ht2;
HtVec[2] = -Ht3;
HtVec[3] = -Ht4;
HtVec[4] = -Ht5;
HtVec[5] = -Ht6;
HtVec[6] = -Ht7;
HtVec[7] = -Ht8;
HtVec[8] = -Ht9;
HtVec[9] = -Ht10;
HtVec[10] = -Ht11;
HtVec[11] = -Ht12;
HtVec[12] = -Ht13;
HtVec[13] = -Ht14;
HtVec[14] = -Ht15;
```

```
HtVec[15] = -Ht16;
HtVec[16] = -Ht17;
HtVec[17] = -Ht18;
HtVec[18] = -Ht19;
HtVec[19] = -Ht20;
}
```

B.2.8 PredictorCorrector.cuh

```
#pragma once
#ifndef PREDICTORCORRECTOR_CUH
#define PREDICTORCORRECTOR_CUH
#define _USE_MATH_DEFINES
#include <assert.h>
#include <cublas_v2.h>
#include "Definitions.h"
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include "LinearSolver.cuh"
#include "H.h"
#include "Hx.h"
#include "Ht.cuh"
using namespace thrust;
__global__ void randomizeTaskPositions(complex_t* startParameters,
complex_t* finalParameters, complex_t* startPoints, double*
taskPositions,
double* matrixWorkspace, complex_t* taskPositionWorkspace, double
```

```
kTolX, double kTolY, double kTolTheta, int kNumParametersToCopy, int
kNumTPParametersToCopy, int kNumStartPoints, int kNumTaskPositions);
// Runge-kutta 4-5 predictor.
__device__ void predict(double t, const double stepSize, complex_t *x,
complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L, complex_t *U, complex_t *P,
complex_t *b, complex_t *xPredict, complex_t *predictWorkspace, int
n);
// Uses newton's method to correct the predicted point. Executes
Newton's Method kNumNewtonCorrections times.
__device__ void correct(double t, complex_t *x, complex_t
*startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L,
complex_t *U, complex_t *P, complex_t *b, complex_t *xCorrect, int n,
int kNumNewtonCorrections);
// returns dx/dt at "time" t.
__device__ void getDavidenkoEquation(double t, complex_t *x, complex_t
*startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L, complex_t *U,
complex_t *P, complex_t *b, int n);
__global__ void trackPaths(complex_t* xArray, complex_t* startParams,
complex_t* finalParams,
complex_t gamma, complex_t* JMatWorkspaceArray, complex_t* LArray,
complex_t* UArray, complex_t* PArray,
```

```
complex_t*␣bArray,␣complex_t*␣xPredictArray,
complex_t*␣xWorkspaceArray,␣int␣n,␣int␣numSteps,␣const␣double
stepSize,␣int␣numParameters,␣int*␣solFlagArray,␣int␣kNumStartPoints,
int␣kNumIterations,␣int␣kNumNewtonCorrections);
__global__␣void␣filterAndTransformSolutions(complex_t␣*solutions,
double␣*taskPositionArray,␣int␣*solFlagArray,␣const␣int␣n,␣const␣int
kNumTPParameters);
#endif
```

B.2.9 PredictorCorrector.cu

```
#include "PredictorCorrector.cuh"
#include <curand_kernel.h>
#define Power pow
//const int kNumTaskPositions = 5;
const int kNumParamsPerTP = 3;
const int kLDA = 3; // Leading dimension of our homogeneous transform
matrices
__device__ void isComplexConjugate(complex_t v1, complex_t v2, const
double tol, bool &flag)
{
if (abs(v1.real() -v2.real()) < tol)
{
if (v1.imag() + v2.imag() < tol) // if v1.imag = -v2.imag
{
flag = true; // v1 and v2 are complex conjugate pairs
}
```

```
else
{
flag = false;
}
}
else
{
flag = false;
}
}
__global__ void filterAndTransformSolutions(complex_t *solutions,
double *taskPositionArray, int *solFlagArray, const int n, const int
kNumTPParameters)
{
// Calculating the array offsets for the current thread
int xStart = (blockIdx.x * blockDim.x + threadIdx.x) * n;
complex_t *xPtr = &solutions[xStart];
// checking if solutions are true complex conjugate pairs
complex_t CC = xPtr[0];
complex_t CCc = xPtr[1];
complex_t DD = xPtr[2];
complex_t DDc = xPtr[3];
complex_t F = xPtr[4];
complex_t Fc = xPtr[5];
complex_t G = xPtr[6];
complex_t Gc = xPtr[7];
complex_t H = xPtr[8];
```

```
complex_t Hc = xPtr[9];
complex_t T31 = xPtr[10];
complex_t Tc31 = xPtr[11];
complex_t T32 = xPtr[12];
complex_t Tc32 = xPtr[13];
complex_t T33 = xPtr[14];
complex_t Tc33 = xPtr[15];
complex_t T34 = xPtr[16];
complex_t Tc34 = xPtr[17];
complex_t T35 = xPtr[18];
complex_t Tc35 = xPtr[19];
const double kTol = 0.1; // Tolerance for comparing doubles
bool complexConjugateFlag = false;
int checkSum = 0;
isComplexConjugate(CC, CCc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(DD, DDc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(F, Fc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
```

```
checkSum++;
}
isComplexConjugate(G, Gc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(H, Hc, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(T31, Tc31, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(T32, Tc32, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(T33, Tc33, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
```

```
isComplexConjugate(T34, Tc34, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
isComplexConjugate(T35, Tc35, kTol, complexConjugateFlag);
if (complexConjugateFlag == true)
{
checkSum++;
}
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = checkSum;
//if (checkSum == 10)
//{
//        solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 1;
//}
//else
//{
//        solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = -1;
//}
}
__global__ void randomizeTaskPositions(complex_t *startParameters,
complex_t *finalParameters, complex_t *startPoints, double
*taskPositions,
double *matrixWorkspace, complex_t *taskPositionWorkspace, double
kTolX, double kTolY, double kTolTheta, int kNumParametersToCopy, int
kNumTPParametersToCopy, int kNumStartPoints, int kNumTaskPositions)
{
```

```
int ETI = blockIdx.x * blockDim.x + threadIdx.x;
int EBI = ETI / kNumStartPoints; // Effective block index, this is
what set of task positions the current thread is working on.
// The first block does not require randomization.
if (EBI)
{
// Only the first thread in the "block" needs to do the work
if (threadIdx.x % kNumStartPoints == 0)
{
int tpOffset = EBI * kNumTPParametersToCopy;
double *tpPtr = &taskPositions[tpOffset];
complex_t *tpWorkspacePtr = &taskPositionWorkspace[tpOffset];
int relTPOffset = EBI * kNumParametersToCopy;
complex_t *relTPPtr = &finalParameters[relTPOffset];
int matOffset = EBI * kLDA * kLDA * kNumTaskPositions;
double *matPtr = &matrixWorkspace[matOffset];
// Copy the start Parameters from the first block to the current
block.
for (int i = 0; i < kNumParametersToCopy; i++)
{
startParameters[relTPOffset + i] = startParameters[i];
}
// init RNG library
curandState_t curandState;
curand_init(1234, EBI, 0, &curandState);
// Randomize X, Y and theta
for (int i = 0; i < kNumTaskPositions; i++)
```

```
{
int startTPIdx = (i * kNumParamsPerTP);
int randomTPIdx = startTPIdx + tpOffset;
taskPositions[randomTPIdx] = taskPositions[startTPIdx] + -kTolX +
curand_uniform_double(&curandState) * (2.0f * kTolX); // Randomizing X
taskPositions[randomTPIdx + 1] = taskPositions[startTPIdx + 1] +
-kTolY + curand_uniform_double(&curandState) * (2.0f * kTolY); //
Randomizing Y
taskPositions[randomTPIdx + 2] = taskPositions[startTPIdx + 2] +
-kTolTheta + curand_uniform_double(&curandState) * (2.0f * kTolTheta);
// Randomizing Theta
}
// create 3x3 homogeneous transformation matrices from the task
positions
for (int i = 0; i < kNumTaskPositions; i++)
{
double theta_DEG = *(tpPtr + i * 3);
double currentX = *(tpPtr + i * 3 + 1);
double currentY = *(tpPtr + i * 3 + 2);
double theta_RAD = theta_DEG * (M_PI / 180.0);
const int matOffset = i * 3 * 3;
matPtr[matOffset] = cos(theta_RAD);
matPtr[matOffset + 1] = -sin(theta_RAD);
matPtr[matOffset + 2] = currentX;
matPtr[matOffset + 3] = sin(theta_RAD);
matPtr[matOffset + 4] = cos(theta_RAD);
matPtr[matOffset + 5] = currentY;
```

```
matPtr[matOffset + 6] = 0;
matPtr[matOffset + 7] = 0;
matPtr[matOffset + 8] = 1;
}
//// DEBUGGING: storing the first matrix in the second matrix spot to
check if H^{-1}H = I.
//if (blockIdx.x == 1)
//{
//      const int matOffset = 9;
//      for (int i = 0; i < 9; i++)
//      {
//              matPtr[matOffset + i] = matPtr[i];
//      }
//}
// Invert first matrix, as this is a 3x3 homogeneous transform, the
// inverse has a very specific structure that is easy to compute.
// transpose the first 2x2 rotation submatrix of the first homogeneous
transform
double tempVal;
tempVal = matPtr[matrixToArray(0, 1, 3)];
matPtr[matrixToArray(0, 1, 3)] = matPtr[matrixToArray(1, 0, 3)];
matPtr[matrixToArray(1, 0, 3)] = tempVal;
// set the translational part to be -[rotation submatrix
(transposed)].[(x, y) position vector]
// note: have to store H[[0, 2]] as tempVal as we need the original
value for H[[1, 2]] as well.
tempVal = matPtr[matrixToArray(0, 2, 3)];
```

```
matPtr[matrixToArray(0, 2, 3)] = -(matPtr[matrixToArray(0, 0, 3)] *
matPtr[matrixToArray(0, 2, 3)] + matPtr[matrixToArray(0, 1, 3)] *
matPtr[matrixToArray(1, 2, 3)]);
matPtr[matrixToArray(1, 2, 3)] = -(matPtr[matrixToArray(1, 0, 3)] *
tempVal + matPtr[matrixToArray(1, 1, 3)] * matPtr[matrixToArray(1, 2,
3)]);
// Multiply the rest of the matrices by the first matrix
double cosThetaRel;
double sinThetaRel;
double relX;
double relY;
const int numRelTaskPositions = kNumTaskPositions - 1;
//if (blockIdx.x == 1)
//{
//      for (int i = 0; i < 3; i++)
//      {
//              for (int j = 0; j < 3; j++)
//              {
//                      printf("%f ", *(matPtr + matrixToArray(i, j,
3)));
//              }printf("
");
//      }printf("
");
//      for (int i = 0; i < 3; i++)
//      {
//              for (int j = 0; j < 3; j++)
```

```
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("%f ",␣*(matPtr␣+␣9␣+␣matrixToArray(i,
j,␣3)));
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣}printf("
");
//␣␣␣␣␣␣}printf("
");
//}
for␣(int␣i␣=␣1;␣i␣<␣kNumTaskPositions;␣i++)
{
//␣note:␣this␣part␣is␣doing␣H0^{−1}*H_i␣manually,␣for␣just␣the␣parts
we␣need.␣Wherever␣there␣is␣a␣(matrixOffset␣+)␣refers␣to␣the␣H_i
matrix,
//␣while␣if␣there␣is␣no␣offset␣it␣is␣referencing␣the␣inverse␣of␣the
first␣TP␣matrix.
const␣int␣matOffset␣=␣i␣*␣3␣*␣3;
cosThetaRel␣=␣matPtr[matrixToArray(0,␣0,␣3)]␣*␣matPtr[matOffset␣+
matrixToArray(0,␣0,␣3)]␣+␣matPtr[matrixToArray(0,␣1,␣3)]␣*
matPtr[matOffset␣+␣matrixToArray(1,␣0,␣3)];
sinThetaRel␣=␣matPtr[matrixToArray(1,␣0,␣3)]␣*␣matPtr[matOffset␣+
matrixToArray(0,␣0,␣3)]␣+␣matPtr[matrixToArray(1,␣1,␣3)]␣*
matPtr[matOffset␣+␣matrixToArray(1,␣0,␣3)];
//␣recover␣and␣store␣the␣new␣relative␣(and␣randomized)␣task␣positions
relX␣=␣matPtr[matrixToArray(0,␣0,␣3)]␣*␣matPtr[matOffset␣+
matrixToArray(0,␣2,␣3)]␣+␣matPtr[matrixToArray(0,␣1,␣3)]␣*
matPtr[matOffset␣+␣matrixToArray(1,␣2,␣3)]␣+␣matPtr[matrixToArray(0,
2,␣3)];
```

```
relY = matPtr[matrixToArray(1, 0, 3)] * matPtr[matOffset +
matrixToArray(0, 2, 3)] + matPtr[matrixToArray(1, 1, 3)] *
matPtr[matOffset + matrixToArray(1, 2, 3)] + matPtr[matrixToArray(1,
2, 3)];
// storing T1, P1, T2, P2, ..., in taskPositionWorkspace before
transferring
// them to final parameters in the correct order
tpWorkspacePtr[i * 2 - 2] = complex_t(cosThetaRel, sinThetaRel);
tpWorkspacePtr[i * 2 - 1] = complex_t(relX, relY);
// DEBUGGING: for block index 1, the first multiplied by the second
should be I.
//if (blockIdx.x == 1 && i == 1)
//{
//      printf("theta: %f, x: %f, y: %f
", atan2(sinThetaRel, cosThetaRel), relX, relY);
//}
}
// transferring the new relative task positions to the final
parameters
// expected form is T1, T2, ..., T4, Tc1, Tc2, ..., Tc4, P1, P2, ...,
P4, Pc1, Pc2, ..., Pc4
for (int i = 0; i < numRelTaskPositions; i++)
{
complex_t Ti = tpWorkspacePtr[i * 2];
complex_t Pi = tpWorkspacePtr[i * 2 + 1];
relTPPtr[i] = Ti;
relTPPtr[i + numRelTaskPositions] = complex_t(Ti.real(), -Ti.imag());
```

```
// Tc_i
relTPPtr[i + 2 * numRelTaskPositions] = Pi;
relTPPtr[i + 3 * numRelTaskPositions] = complex_t(Pi.real(),
-Pi.imag()); // Pc_i
}
}
}
if (ETI == 0)
{
printf("Randomization complete from GPU kernel. Starting path
tracking...
");
}
}
// Returns the vector xDot from the Davidenko Differential Equation:
// xDot = -(J)^-1*Ht. Where J is the Jacobian Matrix of the Homotopy
System and
// Ht is the partial derivative of the homotopy system with respect to
t.
//
// The results are stored in JMatWorkspace 0:n
//
// Note: Uses LU Factorization to find xDot instead of finding J^(-1).
This solves the equation Jdx = -Ht for vector dx.
// This is also getting the equation for point x at time t.
//
__device__ void getDavidenkoEquation(double t, complex_t *x, complex_t
```

```
*startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L, complex_t *U,
complex_t *P, complex_t *b, int n)
{
// Populate JMatWorkspace
Hx(t, x, startParams, finalParams, JMatWorkspace, gamma, n);
// Populate bArray with -Ht. Note: the Ht function returns the minus
of dH/dt.
Ht(t, x, startParams, finalParams, b, gamma, n);
// solving Jdx = -Ht for dx.
solveLinearSystem(JMatWorkspace, L, U, P, b, n, t);
//JMatWorkspace should now hold the values of dx.
}
// Runge-Kutta 45 predictor step
// Calculates:  xPredict = x + (1 / 6)*(K1 + 2 * K2 + 2 * K3 + K4),
where:
//       K1 = stepSize * getDavidenkoEquation(t, x, ...);
//       K2 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x +
0.5 * K1, ...);
//       K3 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x +
0.5 * K2, ...);
//       K4 = stepSize * getDavidenkoEquation(t + stepSize, x + K3,
...);
//
// NOTE: This method starts with xPredict = x, then adds K1, K2, K3,
K4 on the fly as they are found.
```

```
// Therefore, xPredict = x + (1/6)K1 + (1/3)K2 + (1/3)K3 + (1/6)K4.
__device__ void predict(double t, const double stepSize, complex_t *x,
complex_t *startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L, complex_t *U, complex_t *P,
complex_t *b, complex_t *xPredict, complex_t *predictWorkspace, int n)
{
// xPredict = x;
for (int i = 0; i < n; i++)
{
xPredict[i] = x[i];
}
//K1 = stepSize * getDavidenkoEquation(t, x, startParams, finalParams,
gamma);
// Using xPredict and predictWorkspace to solve the linear system.
NOTE: the result dx from getDavidenkoEquation()
// is stored in predictWorkspace
getDavidenkoEquation(t, x, startParams, finalParams, gamma,
JMatWorkspace, L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K1 = stepSize * dx.
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/6) * K1;
xPredict[i] += (1 / 6.0) * JMatWorkspace[i];
}
// The input for K2 is x + 0.5 * K1. K1 is currently stored in
```

```
predictWorkspace.
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + 0.5 * JMatWorkspace[i];
}
// K2 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x + 0.5 *
K1, startParams, finalParams, gamma);
// NOTE: t and x are different than the call to getDavidenkoEquation()
above.
getDavidenkoEquation(t + 0.5 * stepSize, predictWorkspace,
startParams, finalParams, gamma, JMatWorkspace, L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K2 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/3) * K2
xPredict[i] += (1 / 3.0) * JMatWorkspace[i];
}
// The input for K3 is x + 0.5 * K2
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + 0.5 * JMatWorkspace[i];
}
//K3 = stepSize * getDavidenkoEquation(t + 0.5 * stepSize, x + 0.5 *
K2, startParams, finalParams, gamma);
getDavidenkoEquation(t + 0.5 * stepSize, predictWorkspace,
startParams, finalParams, gamma, JMatWorkspace, L, U, P, b, n);
```

```
for (int i = 0; i < n; i++)
{
// K3 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/3) * K3
xPredict[i] += (1 / 3.0) * JMatWorkspace[i];
}
// K4's input is just x + K3.
for (int i = 0; i < n; i++)
{
predictWorkspace[i] = x[i] + JMatWorkspace[i];
}
//K4 = stepSize * getDavidenkoEquation(t + stepSize, x + K3,
startParams, finalParams, gamma);
// NOTE: t is different from K3 and K2's t.
getDavidenkoEquation(t + stepSize, predictWorkspace, startParams,
finalParams, gamma, JMatWorkspace, L, U, P, b, n);
for (int i = 0; i < n; i++)
{
// K4 = stepSize * dx
JMatWorkspace[i] = stepSize * JMatWorkspace[i];
// xPredict += (1/6) * K4
xPredict[i] += (1 / 6.0) * JMatWorkspace[i];
}
}
// Uses newton's method to correct the predicted point. Executes
Newton's Method kNumNewtonCorrections times.
```

```
// Saves the result in *x
__device__ void correct(double t, complex_t *x, complex_t
*startParams,
complex_t *finalParams, complex_t gamma, complex_t *JMatWorkspace,
complex_t *L,
complex_t *U, complex_t *P, complex_t *b, complex_t *xCorrect, int n,
int kNumNewtonCorrections)
{
// xCorrect = x
for (int i = 0; i < n; i++)
{
xCorrect[i] = x[i];
}
for (int i = 0; i < kNumNewtonCorrections; i++)
{
// Populating JMatWorkspace with J
Hx(t, xCorrect, startParams, finalParams, JMatWorkspace, gamma, n);
// Populating b with HVec
H_device(t, xCorrect, startParams, finalParams, b, gamma, n);
// solve for J.dx = HVec
solveLinearSystem(JMatWorkspace, L, U, P, b, n, t);
// JMatWorkspace 0:n now holds the solution dx.
// xCorrect -= dx;
for (int j = 0; j < n; j++)
{
xCorrect[j] -= JMatWorkspace[j];
}
```

```
}
// x = xCorrect
for (int i = 0; i < n; i++)
{
x[i] = xCorrect[i];
}
}
// **************************************************************
*********************************************
//
//                                                          MAIN
HOMOTOPY CONTINUATION LOOP
//
// **************************************************************
*********************************************
__global__ void trackPaths(complex_t *xArray, complex_t *startParams,
complex_t *finalParams,
complex_t gamma, complex_t *JMatWorkspaceArray, complex_t *LArray,
complex_t *UArray, complex_t *PArray,
complex_t *bArray, complex_t *xPredictArray,
complex_t *xWorkspaceArray, int n, int numSteps, const double
stepSize, int numParameters, int *solFlagArray, int kNumStartPoints,
int kNumIterations, int kNumNewtonCorrections)
{
int ETI = blockIdx.x * blockDim.x + threadIdx.x;
int EBI = ETI / kNumStartPoints;
if (ETI < kNumIterations * kNumStartPoints) // Run exactly the same
```

```
number of threads as paths
{
// Calculating the array offsets for the current thread
int xStart = (blockIdx.x * blockDim.x  + threadIdx.x) * n;
int bStart = xStart;
complex_t *xPtr = &xArray[xStart];
complex_t *xWorkspacePtr = &xWorkspaceArray[xStart];
complex_t *xPredictPtr = &xPredictArray[xStart];
complex_t *bPtr = &bArray[bStart];
int JStart = (blockIdx.x * blockDim.x + threadIdx.x) * n * n;
complex_t *JPtr = &JMatWorkspaceArray[JStart];
complex_t *LPtr = &LArray[JStart];
complex_t *UPtr = &UArray[JStart];
complex_t *PPtr = &PArray[JStart];
int parameterOffset = EBI * numParameters;
complex_t *finalParamPtr = &finalParams[parameterOffset];
// Main tracking loop
double currentT = 1;
for (int i = 0; i < numSteps; i++)
{
// Take a predict step using Runge-Kutta 4/5 prediction.
predict(currentT, stepSize, xPtr, startParams, finalParamPtr, gamma,
JPtr, LPtr, UPtr, PPtr, bPtr, xPredictPtr, xWorkspacePtr, n);
// xPredictPtr should point to the predicted point.
// Correcting xPredict using newtons method kNumNewtonCorrections
times.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
```

```
UPtr, PPtr, bPtr, xPredictPtr , n, kNumNewtonCorrections);
// xPtr should point to the corrected point.
//H_device(currentT , xPtr, startParams , finalParamPtr , bPtr, gamma,
n);
//complex_t f1 = *(bPtr);
//complex_t f2 = *(bPtr + 1);
//complex_t f3 = *(bPtr + 2);
//complex_t f4 = *(bPtr + 3);
//complex_t f5 = *(bPtr + 4);
//complex_t f6 = *(bPtr + 5);
//complex_t f7 = *(bPtr + 6);
//complex_t f8 = *(bPtr + 7);
//complex_t f9 = *(bPtr + 8);
//complex_t f10 = *(bPtr + 9);
//complex_t f11 = *(bPtr + 10);
//complex_t f12 = *(bPtr + 11);
//complex_t f13 = *(bPtr + 12);
//complex_t f14 = *(bPtr + 13);
//complex_t f15 = *(bPtr + 14);
//complex_t f16 = *(bPtr + 15);
//complex_t f17 = *(bPtr + 16);
//complex_t f18 = *(bPtr + 17);
//complex_t f19 = *(bPtr + 18);
//complex_t f20 = *(bPtr + 19);
//complex_t residual = sqrt(
//      f1 * f1 + f2 * f2 + f3 * f3 + f4 * f4 + f5 * f5 + f6 * f6 + f7
* f7 + f8 * f8 + f9 * f9 + f10 * f10 +
```

```
//      f11 * f11 + f12 * f12 + f13 * f13 + f14 * f14 + f15 * f15 +
f16 * f16 + f17 * f17 + f18 * f18 +
//      f19 * f19 + f20 * f20
//      );
//printf("Effective Thread index: %d, Current T: %f, Residual: (%e,
%e)
", ETI, currentT, residual.real(), residual.imag());
currentT -= stepSize;
}
// Running correct twice (2 * numiterations of newton's method) to
finalize paths.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
H_device(currentT, xPtr, startParams, finalParamPtr, bPtr, gamma, n);
complex_t f1 = *(bPtr);
complex_t f2 = *(bPtr + 1);
complex_t f3 = *(bPtr + 2);
complex_t f4 = *(bPtr + 3);
complex_t f5 = *(bPtr + 4);
complex_t f6 = *(bPtr + 5);
complex_t f7 = *(bPtr + 6);
```

```
complex_t f8 = *(bPtr + 7);
complex_t f9 = *(bPtr + 8);
complex_t f10 = *(bPtr + 9);
complex_t f11 = *(bPtr + 10);
complex_t f12 = *(bPtr + 11);
complex_t f13 = *(bPtr + 12);
complex_t f14 = *(bPtr + 13);
complex_t f15 = *(bPtr + 14);
complex_t f16 = *(bPtr + 15);
complex_t f17 = *(bPtr + 16);
complex_t f18 = *(bPtr + 17);
complex_t f19 = *(bPtr + 18);
complex_t f20 = *(bPtr + 19);
complex_t residual = sqrt(
f1 * f1 + f2 * f2 + f3 * f3 + f4 * f4 + f5 * f5 + f6 * f6 + f7 * f7 +
f8 * f8 + f9 * f9 + f10 * f10 +
f11 * f11 + f12 * f12 + f13 * f13 + f14 * f14 + f15 * f15 + f16 * f16
+ f17 * f17 + f18 * f18 +
f19 * f19 + f20 * f20
);
//printf("Effective Thread index: %d, Residual: (%e, %e)
", ETI, residual.real(), residual.imag());
if (residual.real() < 1e-1 && residual.imag() < 1e-1)
{
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 1;
}
else {
```

```
solFlagArray[blockIdx.x * blockDim.x + threadIdx.x] = 0;
}
}
if (ETI == 0)
{
printf("Path tracking complete from GPU kernel.
");
}
}
// Old Davidenko Method using cublas.
// __device__ void getDavidenkoEquation(double t, complex_t *x,
complex_t *startParams,
//      complex_t *finalParams, complex_t *JMatWorkspace, complex_t
*b, complex_t gamma, cublasHandle_t handle, int n, int *pivot, int
*info)
//{
//
//      complex_t **JPtr = &JMatWorkspace;
//      complex_t **bPtr = &b;
//
//      cublasStatus_t status;
//
//      // Populate JMatWorkspace
//
//      Hx(t, x, startParams, finalParams, JMatWorkspace, gamma, n);
//
//      // Populate bArray with -Ht. Note: the Ht function returns the
```

```
minus␣of␣dH/dt.
//␣␣␣␣␣␣Ht(t,␣x,␣startParams,␣finalParams,␣b,␣gamma,␣n);
//
//␣␣␣␣␣␣cudoubleComplex␣**bPtr_cublas␣=
reinterpret_cast<cudoubleComplex␣**>(bPtr);
//␣␣␣␣␣␣cudoubleComplex␣**JPtr_cublas␣=
reinterpret_cast<cudoubleComplex␣**>(JPtr);
//
//␣␣␣␣␣␣//␣Computing␣the␣LU␣factorization␣of␣all␣Jacobian␣Matrices
with␣getrfbatched()
//␣␣␣␣␣␣if␣(threadIdx.x␣==␣0)
//␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("Calling getRF.
");
//␣␣␣␣␣␣}
//␣␣␣␣␣␣status␣=␣cublasZgetrfBatched(handle,␣n,␣JPtr_cublas,␣n,␣pivot,
info,␣1);
//␣␣␣␣␣␣assert(status␣==␣CUBLAS_STATUS_SUCCESS);
//␣␣␣␣␣␣if␣(threadIdx.x␣==␣0)
//␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("getRF Called.
");
//␣␣␣␣␣␣}
//
//␣␣␣␣␣␣//␣Computing␣dx␣for␣J(dx)␣=␣-Ht␣using␣getrsbatched()
//␣␣␣␣␣␣status␣=␣cublasZgetrsBatched(handle,␣CUBLAS_OP_N,␣n,␣1,
JPtr_cublas,␣n,␣pivot,␣bPtr_cublas,␣n,␣info,␣1);
```

```
//      assert(status == CUBLAS_STATUS_SUCCESS);
//
//      //bArray should now hold dx.
//
//}
```

B.2.10 LinearSolver.cuh

```
#pragma once
#ifndef LINEARSOLVER_CUH
#define LINEARSOLVER_CUH
#include <cublas_v2.h>
#include "Definitions.h"
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
using namespace thrust;
__device__ void solveLinearSystem(complex_t *A, complex_t *L,
complex_t *U, complex_t *P, complex_t *b, int n, double t);
__device__ int matrixToArray(int row, int col, int n);
__device__ void matrixVectorProduct(complex_t *M, complex_t *v,
complex_t *y, int n);
__device__ void matrixMatrixProduct(complex_t *A, complex_t *B,
complex_t *C, int n);
#endif
```

B.2.11 LinearSolver.cu

```
#include "LinearSolver.cuh"
```

```
__device__ int matrixToArray(int row, int col, int n)
{
return (row * n + col);
}
__device__ void matrixVectorProduct(complex_t *M, complex_t *v,
complex_t *y, int n)
{
for (int i = 0; i < n; i++)
{
*(y + i) = 0;
for (int j = 0; j < n; j++)
{
*(y + i) += *(M + matrixToArray(i, j, n)) * *(v + j);
}
}
}
__device__ void matrixMatrixProduct(complex_t *A, complex_t *B,
complex_t *C, int n)
{
for (int i = 0; i < n; i++)
{
for (int j = 0; j < n; j++)
{
complex_t currentSum = complex_t(0, 0);
for (int k = 0; k < n; k++)
{
currentSum += *(A + matrixToArray(i, k, n)) * *(B + matrixToArray(k,
```

```
j, n));
}
*(C + matrixToArray(i, j, n)) = currentSum;
}
}
}
// Decomposes Matrix A into an Upper (stored in U) and a Lower (stored
in L) triangular matrix using
// doolittle factorization.
//
// n is the number of rows/columns of A, A is a row-major matrix.
//
// Code implemented from pesudocode found in Cheney/Kincade,
"Numerical Mathematics", pg 300.
__device__ void LUDecomp(complex_t *A, complex_t *L, complex_t *U, int
n)
{
//printf("A(device) =
");
for (int row = 0; row < n; row++)
{
for (int col = 0; col < n; col++)
{
//printf("%0.2f ", *(A + matrixToArray(row, col, n)));
*(L + matrixToArray(row, col, n)) = complex_t(0, 0);
*(U + matrixToArray(row, col, n)) = complex_t(0, 0);
}
```

```
//printf("
");
}
for (int k = 0; k < n; k++)
{
// L_kk = 1 (set 1 along the diagonal of L)
*(L + matrixToArray(k, k, n)) = 1;
for (int j = k; j < n; j++)
{
complex_t currentSum = 0;
for (int s = 0; s < k; s++)
{
complex_t L_Val = *(L + matrixToArray(k, s, n));
complex_t U_Val = *(U + matrixToArray(s, j, n));;
currentSum += L_Val * U_Val;
}
// U_kj = A_kj - currentSum
*(U + matrixToArray(k, j, n)) = *(A + matrixToArray(k, j, n)) -
currentSum;
}
for (int i = k + 1; i < n; i++)
{
complex_t currentSum = 0;
for (int s = 0; s < k; s++)
{
complex_t L_Val = *(L + matrixToArray(i, s, n));
complex_t U_Val = *(U + matrixToArray(s, k, n));;
```

```
currentSum += L_Val * U_Val;
}
// L_ik = (A_ik - currentSum)/U_kk
*(L + matrixToArray(i, k, n)) = (*(A + matrixToArray(i, k, n)) -
currentSum) / *(U + matrixToArray(k, k, n));
}
}
}
__device__ void matrixVectorProduct(double *M, double *v, double *y,
int n)
{
for (int i = 0; i < n; i++)
{
*(y + i) = 0;
for (int j = 0; j < n; j++)
{
*(y + i) += *(M + matrixToArray(i, j, n)) * *(v + j);
}
}
}
// Solves Ax = b for x, where A has been decomposed into upper (U) and
lower (L) triangular matrices.
//
//       LUx = b -> first, let Ux = z => Lz = b, solve for z.
//
//       Second, solve Ux = z for x.
// Code written from pseudocode in Cheney/Kincaid, pg 301.
```

```
__device__ void linSolve(complex_t *A, complex_t *L, complex_t *U,
complex_t *b, int n)
{
complex_t *x = &A[0];
complex_t *z = &A[n];
for (int i = 0; i < n; i++)
{
*(z + i) = 0;
}
// ******************
// Solve Lz = b for z.
// ******************
// z_0 = b_0.
*(z) = *(b);
// finding the rest of z
for (int i = 1; i < n; i++)
{
complex_t currentSum = 0;
for (int j = 0; j < i; j++)
{
complex_t lVal = *(L + matrixToArray(i, j, n));
complex_t zVal = *(z + j);
currentSum += lVal * zVal;
}
*(z + i) = *(b + i) - currentSum;
}
// ******************
```

```
// Solve Ux = z for x.
// ******************
//x_n = z_n/U_nn
*(x + (n - 1)) = *(z + (n - 1)) / *(U + matrixToArray(n - 1, n - 1,
n));
// finding the rest of x
for (int i = (n - 2); i >= 0; i--)
{
complex_t currentSum = 0;
for (int j = i + 1; j < n; j++)
{
complex_t uVal = *(U + matrixToArray(i, j, n));
complex_t xVal = *(x + j);
currentSum += uVal * xVal;
}
*(x + i) = (*(z + i) - currentSum) / *(U + matrixToArray(i, i, n));
}
}
__device__ void LUDecomp_PartialPivot(complex_t *A, complex_t *L,
complex_t *U, complex_t *P, int n)
{
//printf(”A(device) =
”);
for (int row = 0; row < n; row++)
{
for (int col = 0; col < n; col++)
{
```

```
//printf("%0.2f ", *(A + matrixToArray(row, col, n)));
*(L + matrixToArray(row, col, n)) = complex_t(0, 0);
*(U + matrixToArray(row, col, n)) = complex_t(0, 0);
*(P + matrixToArray(row, col, n)) = complex_t(0, 0);
}
}
for (int k = 0; k < n; k++)
{
*(P + matrixToArray(k, k, n)) = complex_t(1, 0);
}
int pivotRow;
complex_t pivotVal;
complex_t pivotCandidate;
complex_t tempP;
complex_t tempA;
for (int k = 0; k < n; k++)
{
// L_kk = 1 (set 1 along the diagonal of L)
*(L + matrixToArray(k, k, n)) = complex_t(1, 0);
// Pivoting
pivotRow = k;
pivotVal = *(A + matrixToArray(k, k, n));
double pivotNorm = pivotVal.real() * pivotVal.real() + pivotVal.imag()
* pivotVal.imag();
for (int i = k + 1; i < n; i++)
{
pivotCandidate = *(A + matrixToArray(i, k, n));
```

```
double candidateNorm = pivotCandidate.real() * pivotCandidate.real() +
pivotCandidate.imag() * pivotCandidate.imag();
if (candidateNorm > pivotNorm)
{
pivotVal = pivotCandidate;
pivotRow = i;
pivotNorm = candidateNorm;
}
}
// Swapping rows if necessary.
if (pivotRow != k)
{
for (int j = 0; j < n; j++)
{
// swap rows pivotRow and k of P
tempP = *(P + matrixToArray(pivotRow, j, n));
*(P + matrixToArray(pivotRow, j, n)) = *(P + matrixToArray(k, j, n));
*(P + matrixToArray(k, j, n)) = tempP;
// swap rows pivotRow and k of A
tempA = *(A + matrixToArray(pivotRow, j, n));
*(A + matrixToArray(pivotRow, j, n)) = *(A + matrixToArray(k, j, n));
*(A + matrixToArray(k, j, n)) = tempA;
}
if (k >= 1)
{
// interchange rows pivotRow and k in cols 1:k-1 of L
for (int j = 0; j < k; j++)
```

```
{
tempA = *(L + matrixToArray(pivotRow, j, n));
*(L + matrixToArray(pivotRow, j, n)) = *(L + matrixToArray(k, j, n));
*(L + matrixToArray(k, j, n)) = tempA;
}
}
}
// Peforming gaussian elimination
for (int i = k + 1; i < n; i++)
{
// L[i][k] = A[i][k] / A[k][k]
*(L + matrixToArray(i, k, n)) = *(A + matrixToArray(i, k, n)) / *(A +
matrixToArray(k, k, n));
for (int j = k + 1; j < n; j++)
{
// A[i][j] = A[i][j] - L[i][k] * A[k][j]
*(A + matrixToArray(i, j, n)) -= *(L + matrixToArray(i, k, n)) * *(A +
matrixToArray(k, j, n));
}
}
for (int j = k; j < n; j++)
{
//U[k][j] = A[k][j];
*(U + matrixToArray(k, j, n)) = *(A + matrixToArray(k, j, n));
}
}
}
```

```
// Solves the linear system Ax = b.
// Stores the value of x in A from 0:n
__device__ void solveLinearSystem(complex_t *A, complex_t *L,
complex_t *U, complex_t *P, complex_t *b, int n, double t)
{
LUDecomp_PartialPivot(A, L, U, P, n);
//if (threadIdx.x == 0 && t == 1.0)
//{
//      printf("A =
");
//      for (int i = 0; i < n; i++)
//      {
//              for (int j = 0; j < n; j++)
//              {
//                      complex_t currentVal = *(A + (i * n) + j);
//                      printf("(%0.5f,   %0.5f) ", currentVal.real(),
currentVal.imag());
//              }
//              printf("
");
//      }
//      printf("
");
//      printf("L =
");
//      for (int i = 0; i < n; i++)
//      {
```

```
//              for (int j = 0; j < n; j++)
//              {
//                      printf("%f ", *(L + (i * n) + j));
//              }
//              printf("
");
//      }
//      printf("
");
//}
// Permutating b vec by the permutation matrix P
matrixVectorProduct(P, b, &A[0], n);
for (int i = 0; i < n; i++)
{
*(b + i) = *(A + i);
}
linSolve(A, L, U, b, n);
// A from 0:n holds the value of x
}
```

B.2.12 Endgame.cuh

```
#pragma once
#ifndef ENDGAME_CUH
#define ENDGAME_CUH
#include "Definitions.h"
#include "PredictorCorrector.cuh"
```

```
__global__ void samplePathsEndGame(complex_t* xArray, complex_t*
startParams, complex_t* finalParams,
complex_t gamma, complex_t* JMatWorkspaceArray, complex_t* LArray,
complex_t* UArray, complex_t* PArray,
complex_t* bArray, complex_t* xPredictArray, complex_t*
xWorkspaceArray, complex_t* samplePointsArray, complex_t*
sampleDXArray,
int n, int numSteps, int numParameters, int numSamples, complex_t*
tArray, int* tpIndexArray, int numRawSolutions, int kNumIterations,
int kNumNewtonCorrections);
// The sample arrays originally have the strucutre of a column major
matrix x11, x12, x13, x14, ..., x1n
// where xij is the j-th index of the i-th sample, (j=1,...,n),
(i=1,...,numSamples). This yields a n x numSamples matrix.
//
// When creating the interpolating polynomial from the samples, the
data used to create the divided differences table
// is done for each of the (j=1,...,n) elements independently.
Therefore the vectors are needed
// in the form of {x11, x21, x31, x41 ...}, {x21, x22, x32, x42, ...}.
For better alignment, we should transpose our original "matrix"
// of samples so the data lines up for the divided differences
computation.
//
// Effectively, we are transposing the sample matrix and so we can use
the row vectors to create the divided difference tables
// in parallel.
```

```
//__global__ void transposeSampleMatrices(complex_t *xArray, complex_t
*xArrayTransposed, complex_t *dxArray, complex_t *dxArrayTransposed,
int n, int numSamples);
__global__ void runEndGame(complex_t* tArray, complex_t* xArray,
complex_t* dxArray, complex_t* aArray, complex_t* sArray,
complex_t* dsArray, complex_t* xSampleArray, complex_t* xSolsArray,
dim3 numThreadsInterpolation, int currentC,
int kNumSamplesEndGame, int kNumSamplesInterpolation, int
kNumDerivatives, int n);
#endif
```

B.2.13 Endgame.cu

```
#include "EndGame.cuh"
#include <thrust\complex.h>
// Uses the De Moivre theorem to calculate v^n and stores the result
// in out.
__device__ void complexPow(complex_t v, double n, complex_t &out)
{
// converting the complex number to polar form
double r = sqrt(v.real() * v.real() + v.imag() * v.imag());
double theta = atan2(v.imag(), v.real());
out = complex_t(
pow(r, n) * cos(n * theta),
pow(r, n) * sin(n * theta)
);
}
```

```
// Collects the sample points x0, x1, ..., x_{knumSamples} and their
derivatives dx/dt for t= t0, ..., t_{knumSamples}
__global__ void samplePathsEndGame(complex_t *xArray, complex_t
*startParams, complex_t *finalParams,
complex_t gamma, complex_t *JMatWorkspaceArray, complex_t *LArray,
complex_t *UArray, complex_t *PArray,
complex_t *bArray, complex_t *xPredictArray, complex_t
*xWorkspaceArray, complex_t *samplePointsArray, complex_t
*sampleDXArray,
int n, int numSteps, int numParameters, int numSamples, complex_t
*tArray, int *tpIndexArray, int numRawSolutions, int kNumIterations,
int kNumNewtonCorrections)
{
int ETI = (blockIdx.x * blockDim.x + threadIdx.x);
int tpIdx = tpIndexArray[ETI];
if (ETI < kNumIterations * numRawSolutions)
{
// Calculating the array offsets for the current thread
int xStart = ETI * n;
int sampleStart = ETI * n * numSamples;
int bStart = xStart;
complex_t* xPtr = &xArray[xStart];
complex_t* xWorkspacePtr = &xWorkspaceArray[xStart];
complex_t* xPredictPtr = &xPredictArray[xStart];
complex_t* bPtr = &bArray[bStart];
int JStart = ETI * n * n;
complex_t* JPtr = &JMatWorkspaceArray[JStart];
```

```
complex_t* LPtr = &LArray[JStart];
complex_t* UPtr = &UArray[JStart];
complex_t* PPtr = &PArray[JStart];
int parameterOffset = tpIdx * numParameters;
complex_t* finalParamPtr = &finalParams[parameterOffset];
// Since we're starting at the first sample point, save the current x
point to the samplePointsArray and the derivative
// Refining the  point at t=R before saving them by running newtons
method 4 * kNumNewtonCorrections times
// before saving them.
double currentT = tArray[0].real();
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
```

```
for (int j = 0; j < n; j++)
{
samplePointsArray[sampleStart + j] = xPtr[j];
}
// Calculate dx/dt for the sample time
getDavidenkoEquation(tArray[0].real(), xPtr, startParams,
finalParamPtr, gamma, JPtr, LPtr, UPtr, PPtr, bPtr, n);
for (int k = 0; k < n; k++)
{
sampleDXArray[sampleStart + k] = JPtr[k];
}
// Tracking to the points t1, t2, ..., t_{numsamples} and saving both
them and their derivatives.
for (int tIdx = 0; tIdx < numSamples - 1; tIdx++)
{
// NOTE: currentT > nextT in the sample sequence
double currentT = tArray[tIdx].real();
double nextT = tArray[tIdx + 1].real();
double currentStride = currentT - nextT;
double stepSize = currentStride / numSteps;
for (int i = 0; i < numSteps; i++)
{
// Take a predict step using Runge-Kutta 4/5 prediction.
predict(currentT, stepSize, xPtr, startParams, finalParamPtr, gamma,
JPtr, LPtr, UPtr, PPtr, bPtr, xPredictPtr, xWorkspacePtr, n);
// xPredictPtr should point to the predicted point.
// Correcting xPredict using newtons method kNumNewtonCorrections
```

```
times.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
// xPtr should point to the corrected point.
currentT -= stepSize;
}
// Refining the sample points before saving them by running newtons
method 4 * kNumNewtonCorrections times
// before saving them.
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
correct(currentT, xPtr, startParams, finalParamPtr, gamma, JPtr, LPtr,
UPtr, PPtr, bPtr, xPredictPtr, n, kNumNewtonCorrections);
// Store x and dx/dt for calculating the interpolating polynomial
for (int j = 0; j < n; j++)
{
samplePointsArray[sampleStart + ((tIdx + 1) * n) + j] = xPtr[j];
}
// Calculate dx/dt for the sample time
getDavidenkoEquation(currentT, xPtr, startParams, finalParamPtr,
gamma, JPtr, LPtr, UPtr, PPtr, bPtr, n);
for (int k = 0; k < n; k++)
{
```

```
sampleDXArray[sampleStart + ((tIdx + 1) * n) + k] = JPtr[k];
}
}
if (ETI == 0)
{
printf("****FROM_GPU: Sampling finished.
");
}
//if (ETI == 0)
//{
//        for (int i = 0; i < numSamples; i++)
//        {
//                double currentT = tArray[i].real();
//                complex_t* currentPoint =
&samplePointsArray[sampleStart + (i * n)];
//                //printf("current t: %f, current point: ", currentT);
//                //for (int j = 0; j < n; j++)
//                //{
//                //        complex_t v = currentPoint[j];
//                //        printf("(%e, %e) ", v.real(), v.imag());
//                //} printf("
");
//               //complex_t *currentDx = &sampleDXArray[sampleStart +
(i * n)];
//               //printf("current t: %f, current dx/dt: ", currentT);
//               //for (int j = 0; j < n; j++)
//               //{
```

```
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣//␣␣␣␣␣␣complex_t␣v␣=␣currentDx[j];
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣//␣␣␣␣␣␣printf("(%e, %e) ",␣v.real(),␣v.imag());
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣//}␣printf("
");
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣H_device(currentT,␣currentPoint,␣startParams,
finalParamPtr,␣bPtr,␣gamma,␣n);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f1␣=␣*(bPtr);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f2␣=␣*(bPtr␣+␣1);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f3␣=␣*(bPtr␣+␣2);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f4␣=␣*(bPtr␣+␣3);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f5␣=␣*(bPtr␣+␣4);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f6␣=␣*(bPtr␣+␣5);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f7␣=␣*(bPtr␣+␣6);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f8␣=␣*(bPtr␣+␣7);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f9␣=␣*(bPtr␣+␣8);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f10␣=␣*(bPtr␣+␣9);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f11␣=␣*(bPtr␣+␣10);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f12␣=␣*(bPtr␣+␣11);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f13␣=␣*(bPtr␣+␣12);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f14␣=␣*(bPtr␣+␣13);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f15␣=␣*(bPtr␣+␣14);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f16␣=␣*(bPtr␣+␣15);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f17␣=␣*(bPtr␣+␣16);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f18␣=␣*(bPtr␣+␣17);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f19␣=␣*(bPtr␣+␣18);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣f20␣=␣*(bPtr␣+␣19);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣residual␣=␣sqrt(
```

```
//                      f1 * f1 + f2 * f2 + f3 * f3 + f4 * f4 + f5 *
f5 + f6 * f6 + f7 * f7 + f8 * f8 + f9 * f9 + f10 * f10 +
//                      f11 * f11 + f12 * f12 + f13 * f13 + f14 * f14
+ f15 * f15 + f16 * f16 + f17 * f17 + f18 * f18 +
//                      f19 * f19 + f20 * f20
//                      );
//              printf("Effective Thread index: %d, Current T: %f,
Residual: (%e, %e)
", ETI, currentT, residual.real(), residual.imag());
//      }
//}
}
}
// __device__ void transposeMatrix(complex_t *x, complex_t *xT,
complex_t *dx, complex_t *dxT, int numRows, int numCols)
//{
//      for (int row = 0; row < numRows; row++)
//      {
//              for (int col = 0; col < numCols; col++)
//              {
//                      //xT[j][i] = x[i][j]
//                      *(xT + col * n + row) = *(x + row * n + col);
//                      *(dxT + col * n + row) = *(dx + row * n +
col);
//              }
//      }
//}
```

```
//__global__␣void␣transposeSampleMatrices(complex_t␣*xArray,␣complex_t
*xArrayTransposed,␣complex_t␣*dxArray,␣complex_t␣*dxArrayTransposed,
int␣n,␣int␣numSamples)
//{
//␣␣␣␣␣␣//␣each␣thread␣will␣transpose␣a␣matrix
//␣␣␣␣␣␣int␣xStart␣=␣(blockIdx.x␣*␣blockDim.x␣+␣threadIdx.x)␣*␣n␣*
numSamples;
//␣␣␣␣␣␣//printf("ETI: %d,  xStart: %d
",␣blockIdx.x*blockDim.x␣+␣threadIdx.x,␣xStart);
//␣␣␣␣␣␣complex_t␣*xPtr␣=␣&xArray[xStart];
//␣␣␣␣␣␣complex_t␣*xTPtr␣=␣&xArrayTransposed[xStart];
//
//␣␣␣␣␣␣complex_t␣*dxPtr␣=␣&dxArray[xStart];
//␣␣␣␣␣␣complex_t␣*dxTPtr␣=␣&dxArrayTransposed[xStart];
//
//␣␣␣␣␣␣if␣(blockIdx.x␣==␣2␣&&␣threadIdx.x␣==␣0)
//␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("before  transpose:
");
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣(int␣i␣=␣0;␣i␣<␣numSamples;␣i++)
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣(int␣j␣=␣0;␣j␣<␣n;␣j++)
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣v␣=␣*(xPtr␣+␣i␣*␣n␣+␣j);
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("(%0.2f,  %0.2f)  ",␣v.real(),
v.imag());
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}␣printf("
```

```
");
//              } printf("
");
//      }
//      transposeMatrix(xPtr, xTPtr, dxPtr, dxTPtr, numSamples, n);
//      if (blockIdx.x == 2 && threadIdx.x == 0)
//      {
//              printf("after  transpose:
");
//              for (int i = 0; i < n; i++)
//              {
//                      for (int j = 0; j < numSamples; j++)
//                      {
//                              complex_t v = *(xTPtr + i * n + j);
//                              printf("(%0.2f,  %0.2f)  ", v.real(), v.imag());
//                      } printf("
");
//              } printf("
");
//      }
//}
// computes the coefficients of the divided differences table to be used in newtonian polynomial interpolation
__device__ void Coef(complex_t *sArray, complex_t *xArray, complex_t *dsArray, complex_t *aArray, int kNumSamplesInterpolation, int kNumDerivatives)
```

```
{
for (int i = 0; i < kNumSamplesInterpolation * kNumDerivatives; i++)
{
aArray[i] = xArray[i];
}
complex_t ddNum, ddDenom;
for (int j = 1; j < kNumSamplesInterpolation * kNumDerivatives; j++)
{
for (int i = kNumSamplesInterpolation * kNumDerivatives - 1; i > j -
1; i--)
{
// Fill in the element of the divided difference table
ddNum = aArray[i] - aArray[i - 1];
ddDenom = sArray[i] - sArray[i - j];
aArray[i] = ddNum / ddDenom;
// Old Hermite interpolation code
//if (sArray[i] == sArray[i - j])
//{
//       aArray[i] = dsArray[i];
//}
//else
//{
//       // Fill in the element of the divided difference table
//       ddNum = aArray[i] - aArray[i - 1];
//       ddDenom = sArray[i] - sArray[i - j];
//       aArray[i] = ddNum / ddDenom;
//}
```

```
}
}
}
// Interpolates x(tPredict) from the array of sample t values and the
divided differences table.
// Returns: x(tPredictPoint) as tPredictVal
__device__ void Eval(complex_t *tArray, complex_t *aArray, complex_t
tPredictPoint, complex_t &tPredictVal, int kNumSamplesInterpolation,
int kNumDerivatives)
{
int maxNumIdx = 0;
for (int j = 0; j < kNumSamplesInterpolation * kNumDerivatives; j++)
{
if (!isnan(aArray[j].real()))
{
maxNumIdx = j;
}
}
tPredictVal = aArray[maxNumIdx];
for (int i = maxNumIdx - 1; i > -1; i--)
{
tPredictVal = tPredictVal * (tPredictPoint - tArray[i]) + aArray[i];
}
}
__global__ void runEndGame(complex_t *tArray, complex_t *xArray,
complex_t *dxArray, complex_t *aArray, complex_t *sArray,
complex_t *dsArray, complex_t *xSampleArray, complex_t *xSolsArray,
```

```
dim3 numThreadsInterpolation , int currentC ,
int kNumSamplesEndGame, int kNumSamplesInterpolation , int
kNumDerivatives , int n)
{
// Calculating the Effective Thread Index , that is ,
// the absolute index of the current thread reguardless
// of what block it is in .
int ETI = blockIdx . x * blockDim . x + threadIdx . x;
if (ETI < numThreadsInterpolation . x)
{
// find c and which row to interpolate
//int currentC = ((ETI / n) % kMaxC) + 1;
int currentRow = ETI % n;
// Since each set of TPs is a block , we can read which samples we
// need just by identifying the block index and then offsetting
// the start by how many numbers are in each sample set (n * num
samples ).
int pathIdx = ETI / n;
//if (pathIdx == 0 && currentC == 2 && currentRow == 0)
//{
//       printf("ETI: %d, pathIdx: %d, currentC: %d, currentRow: %d
", ETI, pathIdx , currentC , currentRow );
//}
//printf("ETI: %d, pathIdx: %d, currentC: %d, currentRow: %d
", ETI, pathIdx , currentC , currentRow );
// calculating where to read the xArray start values
// the start is pathIdx * n * kNumSamplesEndGame for the whole
```

```
// sample array.
int sampleStartIdx = pathIdx * n * kNumSamplesEndGame;
//printf("Thread :%d, pathIdx: %d, c: %d, row: %d
", ETI, pathIdx, currentC, currentRow);
complex_t* xPtr = &xArray[sampleStartIdx];
complex_t* dxPtr = &dxArray[sampleStartIdx];
//if (pathIdx == 0 && currentC == 1 && currentRow == 2)
//{
//      //printf("x sample (block %d):
"), blockIdx.x;
//      printf("x sample:
");
//      for (int i = 0; i < kNumSamplesEndGame - 1; i++)
//      {
//              complex_t v = xPtr[i];
//              printf("(%0.2f, %0.2f) ", v.real(), v.imag());
//      } printf("
");
//}
// calculating where to output the transformed
// s values for the current c and row index
int sStartIdx = pathIdx * (n * kNumSamplesEndGame * kNumDerivatives) +
n * kNumSamplesEndGame * kNumDerivatives + kNumSamplesEndGame *
kNumDerivatives * currentRow;
//printf("Thread :%d, pathIdx: %d, c: %d, row: %d, sStart: %d
", ETI, pathIdx, currentC, currentRow, sStartIdx);
//int sIdx = pathIdx * n * kMaxC * kNumSamplesEndGame + (currentC - 1)
```

```
*  n * kNumSamplesEndGame +  n * currentRow;
complex_t* sPtr = &sArray[sStartIdx];
complex_t* xSamplesPtr = &xSampleArray[sStartIdx];
complex_t* dsPtr = &dsArray[sStartIdx];
//if (pathIdx == 24)
//{
//       printf("c: %d, row: %d, sIdx: %d
", currentC, currentRow, sIdx);
//}
// Normalizing the s vals to the last s val
complex_t scalingFactor;
complexPow(tArray[0], (double)1 / currentC, scalingFactor);
scalingFactor = 1.0 / scalingFactor;
//scalingFactor = complex_t(1.0, 0);
//if (pathIdx == 0 && currentC == 1 && currentRow == 0)
//{
//       printf("Scaling  factor: (%e, %e)
", scalingFactor.real(), scalingFactor.imag());
//}
// transforming s_i = (t_i)^1/c * scalingFactor
for (int i = 0; i < kNumSamplesEndGame; i++)
{
complexPow(tArray[i], (double)1 / currentC, sPtr[kNumDerivatives *
i]);
//complexPow(tArray[i], (double)1 / currentC, sPtr[kNumDerivatives * i
+ 1]);
// Normalizing the s values
```

```
sPtr[kNumDerivatives * i] *= scalingFactor;
//sPtr[kNumDerivatives *i + 1] *= scalingFactor;
xSamplesPtr[kNumDerivatives * i] = xPtr[i * n + currentRow];
//xSamplesPtr[kNumDerivatives * i + 1] = xPtr[i * n + currentRow];
}
// transforming dx_i/ds_i = dx_i/dx*c*s_i^(c-1)
complex_t currentC_complex(currentC, 0);
for (int i = 0; i < kNumSamplesEndGame; i++)
{
//dsPtr[kNumDerivatives * i] = dxPtr[i * n + currentRow];
//dsPtr[kNumDerivatives * i + 1] = dxPtr[i * n + currentRow];
//dsPtr[kNumDerivatives * i] = dxPtr[i * n + currentRow] *
currentC_complex * thrust::pow(sPtr[i], (double)currentC - 1);
//dsPtr[kNumDerivatives * i + 1] = dxPtr[i * n + currentRow] *
currentC_complex * thrust::pow(sPtr[i], (double)currentC - 1);
complexPow(sPtr[kNumDerivatives * i], (double)currentC - 1,
dsPtr[kNumDerivatives * i]);
dsPtr[kNumDerivatives * i] *= dxPtr[i * n + currentRow] *
currentC_complex;
//complexPow(sPtr[kNumDerivatives * i], (double)currentC - 1,
dsPtr[kNumDerivatives * i + 1]);
//dsPtr[kNumDerivatives * i + 1] *= dxPtr[i * n + currentRow] *
currentC_complex;
}
//if (pathIdx == 0 && currentC == 1 && currentRow == 0)
//{
//      for (int i = 0; i < kNumSamplesEndGame ; i++)
```

```
//␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣v␣=␣xPtr[i␣*␣n␣+␣currentRow];
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("(%0.2f, %0.2f) ",␣v.real(),␣v.imag());
//␣␣␣␣␣␣}␣printf("
");
//␣␣␣␣␣␣printf("s sample:
");
//␣␣␣␣␣␣for␣(int␣i␣=␣0;␣i␣<␣kNumSamplesEndGame␣*␣kNumDerivatives;␣i++)
//␣␣␣␣␣␣{
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣v␣=␣xSamplesPtr[i];
//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("(%0.2f, %0.2f) ",␣v.real(),␣v.imag());
//␣␣␣␣␣␣}␣printf("
");
//}
//if␣(pathIdx␣==␣0␣&&␣currentC␣==␣1␣&&␣currentRow␣==␣0)
//{
//␣␣␣␣␣␣//printf("x samples:
");
//␣␣␣␣␣␣//for␣(int␣i␣=␣0;␣i␣<␣kNumSamplesEndGame;␣i++)
//␣␣␣␣␣␣//{
//␣␣␣␣␣␣//␣␣␣␣␣␣for␣(int␣j␣=␣0;␣j␣<␣n;␣j++)
//␣␣␣␣␣␣//␣␣␣␣␣␣{
//␣␣␣␣␣␣//␣␣␣␣␣␣␣␣␣␣␣␣␣␣complex_t␣v␣=␣xPtr[i␣*␣n␣+␣j];
//␣␣␣␣␣␣//␣␣␣␣␣␣␣␣␣␣␣␣␣␣printf("(%0.2f, %0.2f) ",␣v.real(),␣v.imag());
//␣␣␣␣␣␣//␣␣␣␣␣␣}␣printf("
");
//␣␣␣␣␣␣//}␣printf("
```

```
");
//      //printf("dx samples:
");
//      //for (int i = 0; i < kNumSamplesEndGame; i++)
//      //{
//      //      for (int j = 0; j < n; j++)
//      //      {
//      //              complex_t v = dxPtr[i * n + j];
//      //              printf("(%e, %e) ", v.real(), v.imag());
//      //      } printf("
");
//      //} printf("
");
//}
// each row needs num derivatives * num samples spaces
int aStart = pathIdx * (kNumDerivatives * kNumSamplesInterpolation *
n) + n * kNumDerivatives * kNumSamplesInterpolation + kNumDerivatives
* kNumSamplesInterpolation * currentRow;
complex_t* aPtr = &aArray[aStart];
// fill the dd table for the rows
Coef(sPtr, xSamplesPtr, dsPtr, aPtr, kNumSamplesInterpolation,
kNumDerivatives);
//if (pathIdx == 0 && currentRow == 2 && currentC == 1)
//{
//      printf("t array:
");
//      for (int i = 0; i < kNumSamplesEndGame; i++)
```

```
//      {
//              printf("(%e, %e) ", tArray[i].real(),
tArray[i].imag());
//      }printf("
");
//      printf("s array:
");
//      for (int i = 0; i < kNumDerivatives * kNumSamplesEndGame; i++)
//      {
//              complex_t currentV = sPtr[i];
//              printf("(%e, %e) ", currentV.real(), currentV.imag());
//      } printf("
");
//      printf("ds array:
");
//      for (int i = 0; i < kNumDerivatives *
kNumSamplesInterpolation; i++)
//      {
//              complex_t currentV = dsPtr[i];
//              printf("(%e, %e) ", currentV.real(), currentV.imag());
//      } printf("
");
//      printf("xSamples array:
");
//      for (int i = 0; i < kNumDerivatives * kNumSamplesEndGame; i++)
//      {
//              complex_t currentV = xSamplesPtr[i];
```

```
//              printf("(%e, %e) ", currentV.real(), currentV.imag());
//      } printf("
");
//      printf("a array:
");
//      for (int i = 0; i < kNumDerivatives *
kNumSamplesInterpolation; i++)
//      {
//              complex_t currentV = aPtr[i];
//              printf("(%0.10f, %0.10f) ", currentV.real(),
currentV.imag());
//      }printf("
");
//}
// predict the last x as a function of the s_N = t_N^{1/c}
int predictIdx = kNumSamplesEndGame - 1;
complex_t sPredict;
complexPow(tArray[predictIdx], (double)1 / currentC, sPredict);
sPredict *= scalingFactor;
Eval(sPtr, aPtr, sPredict, dsPtr[0], kNumSamplesInterpolation,
kNumDerivatives);
// Calculating the error for this particular row and c value
complex_t x = xPtr[predictIdx * n + currentRow];
complex_t xPredict = dsPtr[0];
complex_t e = x - xPredict;
e = e * e;
dsPtr[1] = e;
```

```
// Calculating the predicted solution at t=0
Eval(sPtr, aPtr, complex_t(0, 0), dsPtr[3], kNumSamplesInterpolation,
kNumDerivatives);
//if (pathIdx == 0)
//{
//      //printf("Thread :%d, pathIdx: %d, c: %d, row: %d, sStart: %d,
aStart: %d
", ETI, pathIdx, currentC, currentRow, sStartIdx, aStart);
//      complex_t lastT = tArray[kNumSamplesEndGame - 1];
//      printf("current C: %d, t_N: (%e, %e), s_N: (%e, %e)
", currentC, lastT.real(), lastT.imag(), sPredict.real(),
sPredict.imag());
//}
if (pathIdx == 0 && currentC == 1)
{
//printf("SPredict: (%0.2f, %0.2f)
", sPredict.real(), sPredict.imag());
//x = xPtr[predictIdx * n + currentRow];
//complex_t xPredict = dsPtr[0];
//complex_t e = dsPtr[1];
//printf("residual: (%e, %e)
", e.real(), e.imag());
//printf("pathIdx: %d, c: %d, current row: %d, x: (%e, %e), xPredict:
(%e, %e), residual: (%0.2e, %0.2e)
", pathIdx, currentC, currentRow, x.real(), x.imag(), xPredict.real(),
xPredict.imag(), e.real(), e.imag());
//printf("pathIdx: %d, c: %d, current row: %d, x: (%e, %e), xPredict:
```

```
(%e, %e), residual: (%0.2e, %0.2e)
", pathIdx, currentC, currentRow, x.real(), x.imag(), xPredict.real(),
xPredict.imag(), e.real(), e.imag());
//Eval(sPtr, aPtr, complex_t(0, 0), dsPtr[0]);
//xPredict = dsPtr[0];
//printf("pathIdx: %d, c: %d, current row: %d, Endpoint estimate: (%e,
%e)
", pathIdx, currentC, currentRow, xPredict.real(), xPredict.imag());
//printf("(%0.2e, %0.2e) ", e.real(), e.imag());
}
// Calculating the error || phi_c - x_N ||
if (currentRow == 0) // Only need one of the threads per c to
calculate the error for the current C
{
//This threads dsPtr[1] has the error^2 for row 0
complex_t residualError = dsPtr[1];
for (int i = 1; i < n; i++)
{
int currentRowSIdx = pathIdx * (n * kNumSamplesEndGame *
kNumDerivatives) + n * kNumSamplesEndGame * kNumDerivatives +
kNumSamplesEndGame * kNumDerivatives * i;
dsPtr = &dsArray[currentRowSIdx];
residualError += dsPtr[1];
}
residualError = sqrt(residualError);
dsPtr = &dsArray[sStartIdx];
dsPtr[2] = residualError;
```

```
}
//// Finding which C gives the minimum residual error and saving the
predicted solution
//if (currentRow == 0 && currentC == 1)
//{
//      int minC = 1;
//      complex_t currentResidual = dsPtr[2];
//      double minError = currentResidual.real() *
currentResidual.real() + currentResidual.imag() *
currentResidual.imag();
//      double currentError = 0;
//      for (int c = 2; c < kMaxC + 1; c++)
//      {
//              int currentRowSIdx = pathIdx * (kMaxC * n *
kNumSamplesEndGame * kNumDerivatives) + (c - 1) * n *
kNumSamplesEndGame * kNumDerivatives + kNumSamplesEndGame *
kNumDerivatives * currentRow;
//              //printf("%d
", currentRowSIdx);
//              dsPtr = &dsArray[currentRowSIdx];
//              currentResidual = dsPtr[2];
//              currentError = currentResidual.real() *
currentResidual.real() + currentResidual.imag() *
currentResidual.imag();
//              if (currentError < minError)
//              {
//                      //minC = c;
```

```
//                       //minError = currentError;
//               }
//
//       }
//       // Save the solution with the c that gave the minimum residual
error.
//       int xSolIdx = pathIdx * n;
//       complex_t *xSolPtr = &xSolsArray[xSolIdx];
//       for (int i = 0; i < n; i++)
//       {
//               int minCRowIdx = pathIdx * (kMaxC * n *
kNumSamplesEndGame * kNumDerivatives) + (minC - 1) * n *
kNumSamplesEndGame * kNumDerivatives + kNumSamplesEndGame *
kNumDerivatives * i;
//               dsPtr = &dsArray[minCRowIdx];
//               xSolPtr[i] = dsPtr[3];
//               //xSolPtr[i] = complex_t(0, 0);
//       }
//}
}
if (ETI == 0)
{
printf("*****FROM GPU: Interpolation finished for c=%d.
", currentC);
}
}
```

B.2.14 kernel.cu

```
//v1 - Moved the Homotopy functions into their own separate files as
compiling them took a long time since they are so complex.
#define _USE_MATH_DEFINES
#include "cuda_runtime.h"
#include "cublas_v2.h"
#include "device_launch_parameters.h"
#include "Definitions.h"
#include "EndGame.cuh"
#include "SixBarAnalysis.h"
#include "PredictorCorrector.cuh"
#include <assert.h>
#include <chrono>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <math.h>
#include <random>
#include <stdio.h>
#include <vector>
#define Power pow
// ****************************************
//
//                    PROBLEM-SPECIFIC CONSTANTS
//
// ****************************************
const int kNumParameters = 19; // Number of start and final parameters
```

```
const unsigned kNumLinesPerPoint = 20; // Number of unknowns (used in
file reading)
const unsigned n = kNumLinesPerPoint; // Number of unknowns (used in
code)
const int kNumStartPoints = 5743; // Number of start points as
calculated by Bertini
const unsigned kNumPointsAnalysis = n / 2; // unknowns converted to
points (x, y)
const int kNumTaskPositions = 6; // Number of task positions in the
global frame.
const int kNumTPParameters = 22; // Number of parameters (angle, x, y)
per task position
const double kResolution = 0.1 * M_PI / 180.0; // Angle resolution for
the analysis code
// ****************************************
//
//                    RANDOMIZATION PARAMETERS
//
// ****************************************
const int kNumIterations = 1; // Number of iterations for
randomization
const double kTolX = 0.05; // x tolerance for randomization
const double kTolY = 0.05; // y tolerance for randomization
const double kTolTheta = 0.5; // theta tolerance for randomization
// ****************************************
//
//                    PATH-TRACKING CONSTANTS
```

```
//
// ****************************************
const int kNumNewtonCorrections = 2; // Number of corrections to run
in the correction step
const double kEndGameBoundary = 0.1; // path tracking from 1 to
endGameBoundary
const double kNumSteps = 400; // number of steps to take in the path
tracking
const double kStepSize = (1 - kEndGameBoundary) / kNumSteps; // Step
size for path tracking
// ****************************************
//
//                    ENDGAME CONSTANTS
//
// ****************************************
const int kNumSamplesEndGame = 25; // Number of sample points to use
for the end game
const double kEndGameLambda = 0.8; // Lambda to use to generate the
series R, lambda*R, lambda^2*R, ...
const int kNumSamplesInterpolation = kNumSamplesEndGame - 1; // Number
of sample points to use to predict the final sample point in the end
game.
const double kNumStepsEndGame = 10; // Number of steps to take between
sample times for the end game
const int kMaxC = 4; // Max winding number to test in the endgame
const int kNumDerivatives = 1; // using the 0-th and 1-st derivative
in hermite interpolation, just the 0-th in newton interpolation (just
```

```
the sample points themselves)
// Transforms the 6 task positions to 5 task positions relative to the
first one, then writes
// the relative positions to host_final_parameters. Note the angles
are in degrees.
void transformTaskPositions(std::vector<double> tps, complex_t*
host_final_parameters)
{
// Task positions are of the form (theta, x, y) x 6 for the task
positions,
// followed by the (x, y) coordinates of fixed pivots A and B.
// converting the angles from degrees to radians
double t1 = tps[0] * M_PI / 180.0;
double t2 = tps[3] * M_PI / 180.0;
double t3 = tps[6] * M_PI / 180.0;
double t4 = tps[9] * M_PI / 180.0;
double t5 = tps[12] * M_PI / 180.0;
double t6 = tps[15] * M_PI / 180.0;
complex_t p1(tps[1], tps[2]);
complex_t p2(tps[4], tps[5]);
complex_t p3(tps[7], tps[8]);
complex_t p4(tps[10], tps[11]);
complex_t p5(tps[13], tps[14]);
complex_t p6(tps[16], tps[17]);
complex_t A(tps[18], tps[19]);
complex_t B(tps[20], tps[21]);
complex_t ABar = conj(A);
```

```
complex_t BBar = conj(B);
// Converting tps to relative task positions
double t1Rel = t2 - t1;
double t2Rel = t3 - t1;
double t3Rel = t4 - t1;
double t4Rel = t5 - t1;
double t5Rel = t6 - t1;
complex_t p1Rel = p2 - p1;
complex_t p2Rel = p3 - p1;
complex_t p3Rel = p4 - p1;
complex_t p4Rel = p5 - p1;
complex_t p5Rel = p6 - p1;
complex_t p1RelBar = conj(p1Rel);
complex_t p2RelBar = conj(p2Rel);
complex_t p3RelBar = conj(p3Rel);
complex_t p4RelBar = conj(p4Rel);
complex_t p5RelBar = conj(p5Rel);
complex_t T1Rel = complex_t(cos(t1Rel), sin(t1Rel));
complex_t T2Rel = complex_t(cos(t2Rel), sin(t2Rel));
complex_t T3Rel = complex_t(cos(t3Rel), sin(t3Rel));
complex_t T4Rel = complex_t(cos(t4Rel), sin(t4Rel));
complex_t T5Rel = complex_t(cos(t5Rel), sin(t5Rel));
// Writing the new task positions to the final parameters array
host_final_parameters[0] = p1Rel;
host_final_parameters[1] = p2Rel;
host_final_parameters[2] = p3Rel;
host_final_parameters[3] = p4Rel;
```

```
host_final_parameters[4] = p5Rel;
host_final_parameters[5] = p1RelBar;
host_final_parameters[6] = p2RelBar;
host_final_parameters[7] = p3RelBar;
host_final_parameters[8] = p4RelBar;
host_final_parameters[9] = p5RelBar;
host_final_parameters[10] = T1Rel;
host_final_parameters[11] = T2Rel;
host_final_parameters[12] = T3Rel;
host_final_parameters[13] = T4Rel;
host_final_parameters[14] = T5Rel;
host_final_parameters[15] = A;
host_final_parameters[16] = ABar;
host_final_parameters[17] = B;
host_final_parameters[18] = BBar;
}
double H_CPU(double t, complex_t *x, complex_t *startParams,
complex_t *finalParams, complex_t gamma, int n)
{
complex_t P1Start, P2Start, P3Start, P4Start, P5Start, Pc1Start,
Pc2Start, Pc3Start, Pc4Start, Pc5Start;
complex_t T51Start, T52Start, T53Start, T54Start, T55Start, Tc51Start,
Tc52Start, Tc53Start, Tc54Start, Tc55Start;
complex_t AStart, AcStart, BStart, BcStart;
P1Start = startParams[0]; P2Start = startParams[1]; P3Start =
startParams[2]; P4Start = startParams[3]; P5Start = startParams[4];
Pc1Start = startParams[5]; Pc2Start = startParams[6]; Pc3Start =
```

```
startParams[7]; Pc4Start = startParams[8]; Pc5Start = startParams[9];
T51Start = startParams[10]; T52Start = startParams[11]; T53Start =
startParams[12]; T54Start = startParams[13]; T55Start =
startParams[14];
Tc51Start = 1.0 / T51Start; Tc52Start = 1.0 / T52Start; Tc53Start =
1.0 / T53Start; Tc54Start = 1.0 / T54Start; Tc55Start = 1.0 /
T55Start;
//Tc51Start = conj(T51Start); Tc52Start = conj(T52Start); Tc53Start =
conj(T53Start); Tc54Start = conj(T54Start); Tc55Start =
conj(T55Start);
AStart = startParams[15]; AcStart = startParams[16]; BStart =
startParams[17]; BcStart = startParams[18];
complex_t P1Final, P2Final, P3Final, P4Final, P5Final, Pc1Final,
Pc2Final, Pc3Final, Pc4Final, Pc5Final;
complex_t T51Final, T52Final, T53Final, T54Final, T55Final, Tc51Final,
Tc52Final, Tc53Final, Tc54Final, Tc55Final;
complex_t AFinal, AcFinal, BFinal, BcFinal;
P1Final = finalParams[0]; P2Final = finalParams[1]; P3Final =
finalParams[2]; P4Final = finalParams[3]; P5Final = finalParams[4];
Pc1Final = finalParams[5]; Pc2Final = finalParams[6]; Pc3Final =
finalParams[7]; Pc4Final = finalParams[8]; Pc5Final = finalParams[9];
T51Final = finalParams[10]; T52Final = finalParams[11]; T53Final =
finalParams[12]; T54Final = finalParams[13]; T55Final =
finalParams[14];
Tc51Final = 1.0 / T51Final; Tc52Final = 1.0 / T52Final; Tc53Final =
1.0 / T53Final; Tc54Final = 1.0 / T54Final; Tc55Final = 1.0 /
T55Final;
```

```
//Tc51Final = conj(T51Final); Tc52Final = conj(T52Final); Tc53Final =
conj(T53Final); Tc54Final = conj(T54Final); Tc55Final =
conj(T55Final);
AFinal = finalParams[15]; AcFinal = finalParams[16]; BFinal =
finalParams[17]; BcFinal = finalParams[18];
complex_t CC, CCc, DD, DDc, F, Fc, G, Gc, H, Hc, T31, Tc31, T32, Tc32,
T33, Tc33, T34, Tc34, T35, Tc35;
CC = x[0], CCc = x[1], DD = x[2], DDc = x[3], F = x[4], Fc = x[5], G =
x[6], Gc = x[7], H = x[8], Hc = x[9];
T31 = x[10], Tc31 = x[11], T32 = x[12], Tc32 = x[13], T33 = x[14],
Tc33 = x[15], T34 = x[16], Tc34 = x[17], T35 = x[18], Tc35 = x[19];
complex_t f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14,
f15, f16, f17, f18, f19, f20;
f1 = DDc * G + DD * Gc - 1. * DD * (BcFinal * (1 - t) + BcStart *
gamma * t) - 1. * DDc * (BFinal * (1 - t) + BStart * gamma * t) + 2. *
(BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart
* gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) *
(P1Final * (1 - t) + gamma * P1Start * t) - 1. * (BFinal * (1 - t) +
BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) +
(P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) + DD * (BcFinal * (1 - t) + BcStart * gamma * t)
* T31 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1
- t) + BStart * gamma * t) * T31 - 1. * DD * (Pc1Final * (1 - t) +
gamma * Pc1Start * t) * T31 + (BFinal * (1 - t) + BStart * gamma * t)
* (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * G *
(BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final +
gamma * t * T51Start) + G * (Pc1Final * (1 - t) + gamma * Pc1Start *
```

t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DD ∗ Gc ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T31 ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start);

f2 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 − 1. ∗ DD ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) ∗ T32 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + G ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗

t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ DD ∗ Gc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

f3 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ DD ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗

t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) − 1. ∗ DD ∗ Gc ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 ∗ ((1 − t) ∗ Tc53Final + gamma ∗ t ∗ Tc53Start);

f4 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 − 1. ∗ DD ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + G ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗

t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ DDc ∗ G ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Gc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ DD ∗ Gc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);
f5 = DDc ∗ G + DD ∗ Gc − 1. ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ DD ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗

t) * ((1 - t) * T55Final + gamma * t * T55Start) + DDc * (BFinal * (1
- t) + BStart * gamma * t) * Tc35 - 1. * (BcFinal * (1 - t) + BcStart
* gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. *
DDc * (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + (BcFinal * (1
- t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start *
t) * Tc35 - 1. * DDc * G * ((1 - t) * T55Final + gamma * t * T55Start)
* Tc35 + G * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T55Final + gamma * t * T55Start) * Tc35 - 1. * Gc * (BFinal * (1 - t)
+ BStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start)
+ Gc * (P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) - 1. * DD * Gc * T35 * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + Gc * (BFinal * (1 - t) + BStart *
gamma * t) * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start);
f6 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD
* DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC
* DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart
* gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma
* t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 -
t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart

* gamma * t) − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) *
(BcFinal * (1 − t) + BcStart * gamma * t) − 1. * CC * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma *
t) + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal *
(1 − t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 − t) + AcStart
* gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * G * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart
* gamma * t) + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P1Final * (1 − t) + gamma * P1Start * t) − 1. * DD * DDc * (AcFinal *
(1 − t) + AcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start
* t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) *
(P1Final * (1 − t) + gamma * P1Start * t) + DD * Gc * (AcFinal * (1 −
t) + AcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t)
+ CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1
− t) + gamma * P1Start * t) − 1. * CCc * G * (BcFinal * (1 − t) +
BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1.
* DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1
− t) + gamma * P1Start * t) + G * Gc * (BcFinal * (1 − t) + BcStart *
gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) + CCc * DD *
(AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) + gamma
* Pc1Start * t) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma *
t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) − 1. * CCc * G *
(AFinal * (1 − t) + AStart * gamma * t) * (Pc1Final * (1 − t) + gamma
* Pc1Start * t) + DDc * G * (AFinal * (1 − t) + AStart * gamma * t) *
(Pc1Final * (1 − t) + gamma * Pc1Start * t) + CC * DDc * (BFinal * (1
− t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start *
t) − 1. * DDc * G * (BFinal * (1 − t) + BStart * gamma * t) *

$(Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CCc * DD * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CC * DDc * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + DD * DDc * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + CCc * G * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) + CC * Gc * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * G * Gc * (P1Final * (1 - t) + gamma * P1Start * t) * (Pc1Final * (1 - t) + gamma * Pc1Start * t) - 1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 + DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 + CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 - 1. * DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T31 + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T31 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T31 - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T31 + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T31 + CC * DD * DDc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * DD * DDc * G * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * CC * DD * Gc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 + DD * G * Gc * (Pc1Final * (1 - t) + gamma * Pc1Start * t) * T31 - 1. * CC * DDc * (BFinal * (1 - t) +$

BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ T31 + CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + DD ∗ DDc ∗ G ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CCc ∗ Power(G, 2) ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) + CC ∗ G ∗ Gc ∗ (Pc1Final ∗ (1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc1Final ∗

(1 − t) + gamma ∗ Pc1Start ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) − 1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc31 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc31 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc31 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc31 + CCc ∗ DD ∗ DDc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + DDc ∗ G ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ Tc31 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final +

gamma ∗ t ∗ T51Start) ∗ Tc31 + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T51Final + gamma ∗ t ∗ T51Start) ∗ Tc31 + CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + DD ∗ DDc ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CCc ∗ G ∗ Gc ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) + CC ∗ Power(Gc, 2) ∗ (P1Final ∗ (1 − t) + gamma ∗ P1Start ∗ t) ∗ ((1 − t) ∗ Tc51Final + gamma ∗ t ∗ Tc51Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P1Final ∗ (1 − t) + gamma ∗

P1Start * t) * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + CC * DD
* DDc * Gc * T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) − 1.
* DD * DDc * G * Gc * T31 * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) − 1. * CC * DD * Power(Gc, 2) * T31 * ((1 − t) * Tc51Final
+ gamma * t * Tc51Start) + DD * G * Power(Gc, 2) * T31 * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * CC * DDc * Gc * (BFinal * (1
− t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t *
Tc51Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) *
T31 * ((1 − t) * Tc51Final + gamma * t * Tc51Start) + CC * Power(Gc,
2) * (BFinal * (1 − t) + BStart * gamma * t) * T31 * ((1 − t) *
Tc51Final + gamma * t * Tc51Start) − 1. * G * Power(Gc, 2) * (BFinal *
(1 − t) + BStart * gamma * t) * T31 * ((1 − t) * Tc51Final + gamma * t
* Tc51Start);
f7 = −1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) − 1. * CC * DD
* DDc * Gc + 2. * DD * DDc * G * Gc − 1. * DDc * Power(G, 2) * Gc + CC
* DD * Power(Gc, 2) − 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DDc * G *
(AcFinal * (1 − t) + AcStart * gamma * t) − 1. * CC * DD * Gc *
(AcFinal * (1 − t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal *
(1 − t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 − t) +
AStart * gamma * t) − 1. * CCc * DDc * G * (AFinal * (1 − t) + AStart
* gamma * t) − 1. * CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma
* t) + CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) − 1. *
CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 −
t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 − t) + AStart *
gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart

∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart
∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart
∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start
∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t)
+ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1
− t) + gamma ∗ P2Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) − 1.
∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1
− t) + gamma ∗ P2Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) + CCc ∗ DD ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma
∗ Pc2Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗
t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) − 1. ∗ CCc ∗ G ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma
∗ Pc2Start ∗ t) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1
− t) + BStart ∗ gamma ∗ t) ∗ (Pc2Final ∗ (1 − t) + gamma ∗ Pc2Start ∗
t) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

(Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * CCc * DD * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * CC * DDc * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + DD * DDc * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + CCc * G * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + CC * Gc * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * G * Gc * (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) - 1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 - 1. * DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T32 + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 + CC * DD * DDc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * DD * DDc * G * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * CC * DD * Gc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 + DD * G * Gc * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * CC * DDc * (BFinal * (1 - t) +

BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + DDc * G * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 − 1. * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * T32 + CC * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CC * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CCc * DD * G * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CCc * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * DD * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) + Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CCc * DD * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * CC * DDc * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + DD * DDc * G * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CCc * Power(G, 2) * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) + CC * G * Gc * (Pc2Final * (1 − t) + gamma * Pc2Start * t) * ((1 − t) * T52Final + gamma * t * T52Start) − 1. * Power(G, 2) * Gc * (Pc2Final *

(1 − t) + gamma ∗ Pc2Start ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) − 1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc32 + CCc ∗ DD ∗ DDc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + DDc ∗ G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final +

gamma ∗ t ∗ T52Start) ∗ Tc32 + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t)
+ BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗
Tc32 + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ Power(G, 2) ∗ Gc ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final +
gamma ∗ t ∗ T52Start) ∗ Tc32 + CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) +
AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) −
1. ∗ DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal
∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗
Tc52Start) + DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ DDc ∗ Gc ∗
(BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma
∗ t ∗ Tc52Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗
gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗
Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗
Tc52Final + gamma ∗ t ∗ Tc52Start) + G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 −
t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗
Tc52Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start
∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ CC ∗ DDc ∗
Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final
+ gamma ∗ t ∗ Tc52Start) + DD ∗ DDc ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma
∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + CCc ∗
G ∗ Gc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗
Tc52Final + gamma ∗ t ∗ Tc52Start) + CC ∗ Power(Gc, 2) ∗ (P2Final ∗ (1
− t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗
Tc52Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P2Final ∗ (1 − t) + gamma ∗

P2Start * t) * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC * DD * DDc * Gc * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * DD * DDc * G * Gc * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * CC * DD * Power(Gc, 2) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + DD * G * Power(Gc, 2) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) + CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start) - 1. * G * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T32 * ((1 - t) * Tc52Final + gamma * t * Tc52Start);

f8 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart

* gamma * t) − 1. * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * CCc * G * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) − 1. * DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P3Final * (1 − t) + gamma * P3Start * t) + CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * DD * DDc * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CCc * G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + DDc * G * (AFinal * (1 − t) + AStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CC * DDc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * DDc * G * (BFinal * (1 − t) + BStart * gamma * t) *

(Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CCc * DD * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DDc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + DD * DDc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CCc * G * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) + CC * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * (Pc3Final * (1 − t) + gamma * Pc3Start * t) − 1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 − 1. * DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T33 + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T33 + CC * DD * DDc * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * DD * DDc * G * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * CC * DD * Gc * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 + DD * G * Gc * (Pc3Final * (1 − t) + gamma * Pc3Start * t) * T33 − 1. * CC * DDc * (BFinal * (1 − t) +

BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗
T33 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗
(1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗
T33 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + CC ∗ DDc ∗ G ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) −
1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ Power(G, 2) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) +
Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc3Final
∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗
T53Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start
∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + DD ∗ DDc ∗ G ∗
(Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final +
gamma ∗ t ∗ T53Start) + CCc ∗ Power(G, 2) ∗ (Pc3Final ∗ (1 − t) +
gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) +
CC ∗ G ∗ Gc ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗
T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc3Final ∗

(1 − t) + gamma ∗ Pc3Start ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) − 1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc33 + CCc ∗ DD ∗ DDc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + DDc ∗ G ∗ Gc ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ Tc33 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) ∗ Tc33 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final +

gamma * t * T53Start) * Tc33 + CCc * Power(G, 2) * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + DD * G * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 − 1. * Power(G, 2) * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T53Final + gamma * t * T53Start) * Tc33 + CCc * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CCc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DDc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CCc * DD * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * CC * DDc * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + DD * DDc * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CCc * G * Gc * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc, 2) * (P3Final * (1 − t) + gamma * P3Start * t) * ((1 − t) * Tc53Final + gamma * t * Tc53Start) − 1. * G * Power(Gc, 2) * (P3Final * (1 − t) + gamma *

```
P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * DD
* DDc * Gc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1.
* DD * DDc * G * Gc * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) - 1. * CC * DD * Power(Gc, 2) * T33 * ((1 - t) * Tc53Final
+ gamma * t * Tc53Start) + DD * G * Power(Gc, 2) * T33 * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * CC * DDc * Gc * (BFinal * (1
- t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t *
Tc53Start) + DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) *
T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + CC * Power(Gc,
2) * (BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) *
Tc53Final + gamma * t * Tc53Start) - 1. * G * Power(Gc, 2) * (BFinal *
(1 - t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t
* Tc53Start);
f9 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD
* DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC
* DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G *
(AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc *
(AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal *
(1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) +
AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart
* gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma
* t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. *
CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 -
t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart *
gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc *
(AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart
```

∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) − 1. ∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

(Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * CC * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) + G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * CCc * DD * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * CC * DDc * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) + DD * DDc * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) + CCc * G * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) + CC * Gc * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * G * Gc * (P4Final * (1 − t) + gamma * P4Start * t) * (Pc4Final * (1 − t) + gamma * Pc4Start * t) − 1. * CC * DD * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + DD * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + CC * DD * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 − 1. * DD * G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * T34 + CC * DDc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 − 1. * DDc * G * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 − 1. * CC * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + G * Gc * (AcFinal * (1 − t) + AcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T34 + CC * DD * DDc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * DD * DDc * G * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DD * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 + DD * G * Gc * (Pc4Final * (1 − t) + gamma * Pc4Start * t) * T34 − 1. * CC * DDc * (BFinal * (1 − t) +

BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗
T34 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗
(1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) +
BStart ∗ gamma ∗ t) ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗
T34 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗
(Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ T34 + CC ∗ DDc ∗ G ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final +
gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) +
AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) −
1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗
((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CCc ∗ Power(G, 2) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final +
gamma ∗ t ∗ T54Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) +
Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 −
t) ∗ T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc4Final
∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗
T54Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start
∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + DD ∗ DDc ∗ G ∗
(Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final +
gamma ∗ t ∗ T54Start) + CCc ∗ Power(G, 2) ∗ (Pc4Final ∗ (1 − t) +
gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) +
CC ∗ G ∗ Gc ∗ (Pc4Final ∗ (1 − t) + gamma ∗ Pc4Start ∗ t) ∗ ((1 − t) ∗
T54Final + gamma ∗ t ∗ T54Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc4Final ∗

(1 − t) + gamma * Pc4Start * t) * ((1 − t) * T54Final + gamma * t *
T54Start) − 1. * CCc * DD * DDc * (AFinal * (1 − t) + AStart * gamma *
t) * Tc34 + CCc * DDc * G * (AFinal * (1 − t) + AStart * gamma * t) *
Tc34 + DD * DDc * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc34
− 1. * DDc * G * Gc * (AFinal * (1 − t) + AStart * gamma * t) * Tc34 +
CCc * DD * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal * (1 −
t) + BcStart * gamma * t) * Tc34 − 1. * CCc * G * (AFinal * (1 − t) +
AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc34
− 1. * DD * Gc * (AFinal * (1 − t) + AStart * gamma * t) * (BcFinal *
(1 − t) + BcStart * gamma * t) * Tc34 + G * Gc * (AFinal * (1 − t) +
AStart * gamma * t) * (BcFinal * (1 − t) + BcStart * gamma * t) * Tc34
+ CCc * DD * DDc * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34 −
1. * CCc * DDc * G * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34
− 1. * DD * DDc * Gc * (P4Final * (1 − t) + gamma * P4Start * t) *
Tc34 + DDc * G * Gc * (P4Final * (1 − t) + gamma * P4Start * t) * Tc34
− 1. * CCc * DD * (BcFinal * (1 − t) + BcStart * gamma * t) * (P4Final
* (1 − t) + gamma * P4Start * t) * Tc34 + CCc * G * (BcFinal * (1 − t)
+ BcStart * gamma * t) * (P4Final * (1 − t) + gamma * P4Start * t) *
Tc34 + DD * Gc * (BcFinal * (1 − t) + BcStart * gamma * t) * (P4Final
* (1 − t) + gamma * P4Start * t) * Tc34 − 1. * G * Gc * (BcFinal * (1
− t) + BcStart * gamma * t) * (P4Final * (1 − t) + gamma * P4Start *
t) * Tc34 + CCc * DD * DDc * G * ((1 − t) * T54Final + gamma * t *
T54Start) * Tc34 − 1. * CCc * DDc * Power(G, 2) * ((1 − t) * T54Final
+ gamma * t * T54Start) * Tc34 − 1. * DD * DDc * G * Gc * ((1 − t) *
T54Final + gamma * t * T54Start) * Tc34 + DDc * Power(G, 2) * Gc * ((1
− t) * T54Final + gamma * t * T54Start) * Tc34 − 1. * CCc * DD * G *
(BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T54Final +

gamma * t * T54Start) * Tc34 + CCc * Power(G, 2) * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + DD * G * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 - 1. * Power(G, 2) * Gc * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) * Tc34 + CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * DD * DDc * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + G * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CCc * DD * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC * DDc * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DD * DDc * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + CCc * G * Gc * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + CC * Power(Gc, 2) * (P4Final * (1 - t) + gamma * P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * G * Power(Gc, 2) * (P4Final * (1 - t) + gamma *

P4Start * t) * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + CC * DD * DDc * Gc * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * DD * DDc * G * Gc * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC * DD * Power(Gc, 2) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DD * G * Power(Gc, 2) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * CC * DDc * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + DDc * G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) + CC * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start) - 1. * G * Power(Gc, 2) * (BFinal * (1 - t) + BStart * gamma * t) * T34 * ((1 - t) * Tc54Final + gamma * t * Tc54Start);
f10 = -1. * CCc * DD * DDc * G + CCc * DDc * Power(G, 2) - 1. * CC * DD * DDc * Gc + 2. * DD * DDc * G * Gc - 1. * DDc * Power(G, 2) * Gc + CC * DD * Power(Gc, 2) - 1. * DD * G * Power(Gc, 2) + CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) - 1. * CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CC * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) + CCc * DD * DDc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DDc * G * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) + CCc * G * Gc * (AFinal * (1 - t) + AStart * gamma * t) - 1. * CCc * DD * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + CCc * G * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart * gamma * t) + DD * Gc * (AFinal * (1 - t) + AStart * gamma * t) * (BcFinal * (1 - t) + BcStart

∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ CC ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗
t) + DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗
(1 − t) + BStart ∗ gamma ∗ t) + CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart
∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ G ∗ Gc ∗
(AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart
∗ gamma ∗ t) + CC ∗ DDc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AcFinal ∗
(1 − t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start
∗ t) − 1. ∗ CC ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗
(P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + DD ∗ Gc ∗ (AcFinal ∗ (1 −
t) + AcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t)
+ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1
− t) + gamma ∗ P5Start ∗ t) − 1. ∗ CCc ∗ G ∗ (BcFinal ∗ (1 − t) +
BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1.
∗ DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1
− t) + gamma ∗ P5Start ∗ t) + G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗
gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) + CCc ∗ DD ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma
∗ Pc5Start ∗ t) − 1. ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗
t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) − 1. ∗ CCc ∗ G ∗
(AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma
∗ Pc5Start ∗ t) + DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗
(Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + CC ∗ DDc ∗ (BFinal ∗ (1
− t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗
t) − 1. ∗ DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗

(Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * CC * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + G * Gc * (BFinal * (1 - t) + BStart * gamma * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * CCc * DD * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * CC * DDc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + DD * DDc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + CCc * G * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) + CC * Gc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * G * Gc * (P5Final * (1 - t) + gamma * P5Start * t) * (Pc5Final * (1 - t) + gamma * Pc5Start * t) - 1. * CC * DD * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 + DD * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 + CC * DD * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 - 1. * DD * G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * T35 + CC * DDc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T35 - 1. * DDc * G * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T35 - 1. * CC * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T35 + G * Gc * (AcFinal * (1 - t) + AcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T35 + CC * DD * DDc * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 - 1. * DD * DDc * G * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 - 1. * CC * DD * Gc * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 + DD * G * Gc * (Pc5Final * (1 - t) + gamma * Pc5Start * t) * T35 - 1. * CC * DDc * (BFinal * (1 - t) +

BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + DDc ∗ G ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + CC ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ DDc ∗ G ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CC ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ G ∗ Gc ∗ (AcFinal ∗ (1 − t) + AcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CCc ∗ DD ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CC ∗ DDc ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + DD ∗ DDc ∗ G ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CCc ∗ Power(G, 2) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + CC ∗ G ∗ Gc ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ Power(G, 2) ∗ Gc ∗ (Pc5Final ∗

(1 − t) + gamma ∗ Pc5Start ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) − 1. ∗ CCc ∗ DD ∗ DDc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 + CCc ∗ DDc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 + DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ Tc35 + CCc ∗ DD ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ G ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 − 1. ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 + G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ Tc35 + CCc ∗ DD ∗ DDc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ DDc ∗ G ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DDc ∗ G ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ CCc ∗ DD ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + CCc ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + DD ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 − 1. ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ Tc35 + CCc ∗ DD ∗ DDc ∗ G ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ CCc ∗ DDc ∗ Power(G, 2) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ DD ∗ DDc ∗ G ∗ Gc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DDc ∗ Power(G, 2) ∗ Gc ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ CCc ∗ DD ∗ G ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final +

gamma ∗ t ∗ T55Start) ∗ Tc35 + CCc ∗ Power(G, 2) ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + DD ∗ G ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 − 1. ∗ Power(G, 2) ∗ Gc ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) ∗ Tc35 + CCc ∗ DD ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DD ∗ DDc ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CCc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DDc ∗ G ∗ Gc ∗ (AFinal ∗ (1 − t) + AStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ DDc ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ DDc ∗ G ∗ Gc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + G ∗ Power(Gc, 2) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CCc ∗ DD ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ CC ∗ DDc ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + DD ∗ DDc ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CCc ∗ G ∗ Gc ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) + CC ∗ Power(Gc, 2) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ ((1 − t) ∗ Tc55Final + gamma ∗ t ∗ Tc55Start) − 1. ∗ G ∗ Power(Gc, 2) ∗ (P5Final ∗ (1 − t) + gamma ∗

P5Start * t) * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC * DD * DDc * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * DD * DDc * G * Gc * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC * DD * Power(Gc, 2) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + DD * G * Power(Gc, 2) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * CC * DDc * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + DDc * G * Gc * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) + CC * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start) − 1. * G * Power(Gc, 2) * (BFinal * (1 − t) + BStart * gamma * t) * T35 * ((1 − t) * Tc55Final + gamma * t * Tc55Start);

f11 = Fc * H + F * Hc − 1. * F * (BcFinal * (1 − t) + BcStart * gamma * t) − 1. * Fc * (BFinal * (1 − t) + BStart * gamma * t) + 2. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (P1Final * (1 − t) + gamma * P1Start * t) − 1. * (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + (P1Final * (1 − t) + gamma * P1Start * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) + F * (BcFinal * (1 − t) + BcStart * gamma * t) * T31 − 1. * (BcFinal * (1 − t) + BcStart * gamma * t) * (BFinal * (1 − t) + BStart * gamma * t) * T31 − 1. * F * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 + (BFinal * (1 − t) + BStart * gamma * t) * (Pc1Final * (1 − t) + gamma * Pc1Start * t) * T31 − 1. * H * (BcFinal * (1 − t) + BcStart * gamma * t) * ((1 − t) * T51Final + gamma * t * T51Start) + H * (Pc1Final * (1 − t) + gamma * Pc1Start *

t) * ((1 - t) * T51Final + gamma * t * T51Start) + Fc * (BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc31 - 1. * Fc * (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P1Final * (1 - t) + gamma * P1Start * t) * Tc31 - 1. * Fc * H * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T51Final + gamma * t * T51Start) * Tc31 - 1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + Hc * (P1Final * (1 - t) + gamma * P1Start * t) * ((1 - t) * Tc51Final + gamma * t * Tc51Start) - 1. * F * Hc * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * T31 * ((1 - t) * Tc51Final + gamma * t * Tc51Start);

f12 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P2Final * (1 - t) + gamma * P2Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + (P2Final * (1 - t) + gamma * P2Start * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) + F * (BcFinal * (1 - t) + BcStart * gamma * t) * T32 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T32 - 1. * F * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 + (BFinal * (1 - t) + BStart * gamma * t) * (Pc2Final * (1 - t) + gamma * Pc2Start * t) * T32 - 1. * H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T52Final + gamma * t * T52Start) + H * (Pc2Final * (1 - t) + gamma * Pc2Start *

t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) + Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc32 − 1. ∗ Fc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ Tc32 − 1. ∗ Fc ∗ H ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 + H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T52Final + gamma ∗ t ∗ T52Start) ∗ Tc32 − 1. ∗ Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Hc ∗ (P2Final ∗ (1 − t) + gamma ∗ P2Start ∗ t) ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) − 1. ∗ F ∗ Hc ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start) + Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T32 ∗ ((1 − t) ∗ Tc52Final + gamma ∗ t ∗ Tc52Start);

f13 = Fc ∗ H + F ∗ Hc − 1. ∗ F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + (P3Final ∗ (1 − t) + gamma ∗ P3Start ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) + F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T33 − 1. ∗ F ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗ t) ∗ T33 − 1. ∗ H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T53Final + gamma ∗ t ∗ T53Start) + H ∗ (Pc3Final ∗ (1 − t) + gamma ∗ Pc3Start ∗

t) * ((1 - t) * T53Final + gamma * t * T53Start) + Fc * (BFinal * (1 - t) + BStart * gamma * t) * Tc33 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc33 - 1. * Fc * (P3Final * (1 - t) + gamma * P3Start * t) * Tc33 + (BcFinal * (1 - t) + BcStart * gamma * t) * (P3Final * (1 - t) + gamma * P3Start * t) * Tc33 - 1. * Fc * H * ((1 - t) * T53Final + gamma * t * T53Start) * Tc33 + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T53Final + gamma * t * T53Start) * Tc33 - 1. * Hc * (BFinal * (1 - t) + BStart * gamma * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + Hc * (P3Final * (1 - t) + gamma * P3Start * t) * ((1 - t) * Tc53Final + gamma * t * Tc53Start) - 1. * F * Hc * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start) + Hc * (BFinal * (1 - t) + BStart * gamma * t) * T33 * ((1 - t) * Tc53Final + gamma * t * Tc53Start);

f14 = Fc * H + F * Hc - 1. * F * (BcFinal * (1 - t) + BcStart * gamma * t) - 1. * Fc * (BFinal * (1 - t) + BStart * gamma * t) + 2. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (P4Final * (1 - t) + gamma * P4Start * t) - 1. * (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + (P4Final * (1 - t) + gamma * P4Start * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) + F * (BcFinal * (1 - t) + BcStart * gamma * t) * T34 - 1. * (BcFinal * (1 - t) + BcStart * gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * T34 - 1. * F * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 + (BFinal * (1 - t) + BStart * gamma * t) * (Pc4Final * (1 - t) + gamma * Pc4Start * t) * T34 - 1. * H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) * T54Final + gamma * t * T54Start) + H * (Pc4Final * (1 - t) + gamma * Pc4Start *

t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) + Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ Tc34 − 1. ∗ Fc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 + (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ Tc34 − 1. ∗ Fc ∗ H ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 + H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T54Final + gamma ∗ t ∗ T54Start) ∗ Tc34 − 1. ∗ Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Hc ∗ (P4Final ∗ (1 − t) + gamma ∗ P4Start ∗ t) ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) − 1. ∗ F ∗ Hc ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start) + Hc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T34 ∗ ((1 − t) ∗ Tc54Final + gamma ∗ t ∗ Tc54Start);
f15 = Fc ∗ H + F ∗ Hc − 1. ∗ F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) − 1. ∗ Fc ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) + 2. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) − 1. ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + (P5Final ∗ (1 − t) + gamma ∗ P5Start ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) + F ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ T35 − 1. ∗ F ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 + (BFinal ∗ (1 − t) + BStart ∗ gamma ∗ t) ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗ t) ∗ T35 − 1. ∗ H ∗ (BcFinal ∗ (1 − t) + BcStart ∗ gamma ∗ t) ∗ ((1 − t) ∗ T55Final + gamma ∗ t ∗ T55Start) + H ∗ (Pc5Final ∗ (1 − t) + gamma ∗ Pc5Start ∗

```
t) * ((1 - t) * T55Final + gamma * t * T55Start) + Fc * (BFinal * (1 -
t) + BStart * gamma * t) * Tc35 - 1. * (BcFinal * (1 - t) + BcStart *
gamma * t) * (BFinal * (1 - t) + BStart * gamma * t) * Tc35 - 1. * Fc
* (P5Final * (1 - t) + gamma * P5Start * t) * Tc35 + (BcFinal * (1 -
t) + BcStart * gamma * t) * (P5Final * (1 - t) + gamma * P5Start * t)
* Tc35 - 1. * Fc * H * ((1 - t) * T55Final + gamma * t * T55Start) *
Tc35 + H * (BcFinal * (1 - t) + BcStart * gamma * t) * ((1 - t) *
T55Final + gamma * t * T55Start) * Tc35 - 1. * Hc * (BFinal * (1 - t)
+ BStart * gamma * t) * ((1 - t) * Tc55Final + gamma * t * Tc55Start)
+ Hc * (P5Final * (1 - t) + gamma * P5Start * t) * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) - 1. * F * Hc * T35 * ((1 - t) *
Tc55Final + gamma * t * Tc55Start) + Hc * (BFinal * (1 - t) + BStart *
gamma * t) * T35 * ((1 - t) * Tc55Final + gamma * t * Tc55Start);
f16 = -1. + T31 * Tc31;
f17 = -1. + T32 * Tc32;
f18 = -1. + T33 * Tc33;
f19 = -1. + T34 * Tc34;
f20 = -1. + T35 * Tc35;
complex_t residual = sqrt(
f1*f1 + f2*f2 + f3*f3 + f4*f4 + f5*f5 + f6*f6 + f7*f7 + f8*f8 + f9*f9
+ f10*f10 +
f11*f11 + f12*f12 + f13*f13 + f14*f14 + f15*f15 + f16*f16 + f17*f17 +
f18*f18 +
f19*f19 + f20*f20
);
double e = residual.real() * residual.real() + residual.imag() *
residual.imag();
```

```
return e;
}
int main()
{
auto t0 = std::chrono::high_resolution_clock::now();
cudaError_t cudaStatus;
cudaStatus = cudaSetDevice(0);
assert(cudaStatus == cudaSuccess);
std::vector<complex_t> startParameters(kNumParameters);
std::vector<complex_t> finalParameters(kNumParameters);
std::vector<std::vector<complex_t>> startPoints;
std::vector<std::string> inputBuffer;
std::vector<double> taskPositions(kNumTPParameters); // task positions
are of the form: (angle in degrees, x, y)
std::ifstream inputStream("start_parameters", std::ios::in);
if (inputStream.is_open())
{
std::cout << "Start File is open" << std::endl;
std::string line;
while (getline(inputStream, line))
{
inputBuffer.push_back(line);
}
inputStream.close();
}
else
{
```

```
std::cout << "Failed to open start parameters file." << std::endl;
exit(-1);
}
for (unsigned i = 1; i < inputBuffer.size(); i++)
{
std::istringstream inputSS(inputBuffer[i]);
std::vector<std::string>
stringSplit(std::istream_iterator<std::string>{inputSS},
std::istream_iterator<std::string>());
std::string realStr = stringSplit[0];
std::string imgStr = stringSplit[1];
std::istringstream realStrStream(realStr);
std::istringstream imgStrStream(imgStr);
double realPart, imgPart;
realStrStream >> realPart;
imgStrStream >> imgPart;
complex_t currentParam(realPart, imgPart);
startParameters[i - 1] = currentParam;
}
inputBuffer.clear();
//inputStream.open("final_parameters");
//if (inputStream.is_open())
//{
//        std::cout << "Final Parameters File is open" << std::endl;
//        std::string line;
//        while (getline(inputStream, line))
//        {
```

```
//                    inputBuffer.push_back(line);
//          }
//          inputStream.close();
//}
//else
//{
//          std::cout << "Failed to open final parameters file." <<
std::endl;
//          exit(-1);
//}
//for (unsigned i = 1; i < inputBuffer.size(); i++)
//{
//          std::istringstream inputSS(inputBuffer[i]);
//          std::vector<std::string>
stringSplit(std::istream_iterator<std::string>{inputSS},
std::istream_iterator<std::string>());
//          std::string realStr = stringSplit[0];
//          std::string imgStr = stringSplit[1];
//          std::istringstream realStrStream(realStr);
//          std::istringstream imgStrStream(imgStr);
//          double realPart, imgPart;
//          realStrStream >> realPart;
//          imgStrStream >> imgPart;
//          complex_t currentParam(realPart, imgPart);
//          finalParameters[i - 1] = currentParam;
//}
//inputBuffer.clear();
```

```
inputStream.open("start");
if (inputStream.is_open())
{
std::cout << "Start Points File is open" << std::endl;
std::string line;
while (getline(inputStream, line))
{
if (!line.empty())
{
inputBuffer.push_back(line);
}
}
inputStream.close();
}
else
{
std::cout << "Failed to open start points file." << std::endl;
exit(-1);
}
unsigned numStartPoints = (inputBuffer.size() - 1) /
kNumLinesPerPoint;
for (unsigned pointIdx = 0; pointIdx < numStartPoints; pointIdx++)
{
std::vector<complex_t> currentStartPoint(kNumLinesPerPoint);
for (unsigned lineIdx = 0; lineIdx < kNumLinesPerPoint; lineIdx++)
{
// the +1 at the end there is to skip the first line of the start file
```

```
(which just says the number of start points in the file).
std::istringstream inputSS(inputBuffer[pointIdx * kNumLinesPerPoint +
lineIdx + 1]);
std::vector<std::string>
stringSplit(std::istream_iterator<std::string>{inputSS},
std::istream_iterator<std::string>());
std::string realStr = stringSplit[0];
std::string imgStr = stringSplit[1];
std::istringstream realStrStream(realStr);
std::istringstream imgStrStream(imgStr);
double realPart, imgPart;
realStrStream >> realPart;
imgStrStream >> imgPart;
complex_t currentParam(realPart, imgPart);
currentStartPoint[lineIdx] = currentParam;
}
startPoints.push_back(currentStartPoint);
}
inputBuffer.clear();
inputStream.open("task_positions");
if (inputStream.is_open())
{
std::cout << "Task positions file is open." << std::endl;
std::string line;
while (getline(inputStream, line))
{
inputBuffer.push_back(line);
```

```
}
inputStream.close();
}
else
{
std::cout << "Failed to open task positions file." << std::endl;
exit(-1);
}
for (unsigned i = 0; i < inputBuffer.size(); i++)
{
std::istringstream inputSS(inputBuffer[i]);
double currentVal;
inputSS >> currentVal;
taskPositions[i] = currentVal;
}
// Seeding the new random value of gamma on the unit circle.
std::chrono::system_clock::rep seed =
std::chrono::system_clock::now().time_since_epoch().count();
std::default_random_engine generator(seed);
std::uniform_real_distribution<double> distribution(0, 2 * M_PI);
double thetaRand = distribution(generator);
//complex_t gamma(cos(thetaRand), sin(thetaRand));
complex_t gamma(1, 0);
std::cout << "theta rand: " << thetaRand << std::endl;
std::cout << "Seeded gamma: " << gamma << std::endl;
// ***********************************
//                  MEMORY ALLOCATION
```

```
// ********************************
// Transfer the start parameters, final parameters, and start points
to the device.
const int kBytesPerStartPoint = numStartPoints * kNumLinesPerPoint *
sizeof(complex_t);
const int kBytesPerParameterSet = kNumParameters * sizeof(complex_t);
const int kStartBytesTotal = kBytesPerStartPoint * kNumIterations;
const int kParameterBytesTotal = kBytesPerParameterSet *
kNumIterations;
const int kTPSetBytesTotal = kNumTPParameters * kNumIterations *
sizeof(double);
complex_t* host_start = (complex_t*)malloc(kStartBytesTotal);
complex_t* host_start_parameters =
(complex_t*)malloc(kParameterBytesTotal);
complex_t* host_final_parameters =
(complex_t*)malloc(kParameterBytesTotal);
double* host_task_positions = (double*)malloc(kTPSetBytesTotal);
for (unsigned i = 0; i < startParameters.size(); i++)
{
*(host_start_parameters + i) = startParameters[i];
*(host_final_parameters + i) = finalParameters[i];
}
for (unsigned i = 0; i < startPoints.size(); i++)
{
std::vector<complex_t> currentPoint = startPoints[i];
for (unsigned j = 0; j < kNumLinesPerPoint; j++)
{
```

```
*(host_start + i * kNumLinesPerPoint + j) = currentPoint[j];
}
}
for (unsigned i = 0; i < taskPositions.size(); i++)
{
*(host_task_positions + i) = taskPositions[i];
}
complex_t* device_start = NULL;
cudaStatus = cudaMalloc((void**)&device_start, kStartBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_start_parameters = NULL;
cudaStatus = cudaMalloc((void**)&device_start_parameters,
kParameterBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_final_parameters = NULL;
cudaStatus = cudaMalloc((void**)&device_final_parameters,
kParameterBytesTotal);
assert(cudaStatus == cudaSuccess);
double* device_task_positions = NULL;
cudaStatus = cudaMalloc((void**)&device_task_positions,
kTPSetBytesTotal);
assert(cudaStatus == cudaSuccess);
// Transferring start/final parameter data from host to device.
// For the start points, copy the start point values to each block.
for (int i = 0; i < kNumIterations; i++)
{
int copySize = numStartPoints * kNumLinesPerPoint * sizeof(complex_t);
```

```
int copyOffset = i * numStartPoints * kNumLinesPerPoint;
cudaStatus = cudaMemcpy(&device_start[copyOffset], host_start,
copySize, cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
}
cudaStatus = cudaMemcpy(device_start_parameters,
host_start_parameters, kNumParameters * sizeof(complex_t),
cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
// Transfering original set of task positions to beginning of
device_task_positions
cudaStatus = cudaMemcpy(device_task_positions, host_task_positions,
kNumTPParameters * sizeof(double), cudaMemcpyHostToDevice);
// Calculating the workspace for needed
// For each start point, there will be an n x n jacobian matrix
unsigned batchSize = kNumStartPoints;
const int kBytesPerMatrix = batchSize * n * n * sizeof(complex_t);
const int kMatrixBytesTotal = kBytesPerMatrix * kNumIterations;
// next, allocate an array of pointers on the device
complex_t* device_JArray = NULL;
cudaStatus = cudaMalloc((void**)&device_JArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);
// bArray is an n x batchSize array to be filled in by Ht/H during the
predict/correct stages
// and are to be used in cublasgetrs.
const int kBytesPerVector = batchSize * n * sizeof(complex_t);
const int kVectorBytesTotal = kBytesPerVector * kNumIterations;
```

```
complex_t* device_bArray = NULL;
cudaStatus = cudaMalloc((void**)&device_bArray, kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);
// allocating the xPredictArray and xWorkspaceArray.
complex_t* device_xPredictArray = NULL;
cudaStatus = cudaMalloc((void**)&device_xPredictArray,
kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_xWorkspaceArray = NULL;
cudaStatus = cudaMalloc((void**)&device_xWorkspaceArray,
kVectorBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_LArray = NULL;
cudaStatus = cudaMalloc((void**)&device_LArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_UArray = NULL;
cudaStatus = cudaMalloc((void**)&device_UArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);
complex_t* device_PArray = NULL;
cudaStatus = cudaMalloc((void**)&device_PArray, kMatrixBytesTotal);
assert(cudaStatus == cudaSuccess);
int* device_solFlag_array = NULL;
cudaStatus = cudaMalloc((void**)&device_solFlag_array, kNumIterations
* numStartPoints * sizeof(int));
assert(cudaStatus == cudaSuccess);
int* host_solFlag_array = NULL;
host_solFlag_array = (int*)malloc(kNumIterations * numStartPoints *
```

```
sizeof(int));
//H_CPU(1, host_start, host_start_parameters, host_final_parameters,
complex_t(1, 0), n);
// ***************************************************************
**********
//        --   --        -          -                       -
- -
//        |  \/  |      (_)        | |                       | |
| | |
//        | \  / | __ _ _ _ __     | | _____ _ __ _ __    ___| |    ___ __
_| | |___
//        | |\/| |/ _` | | '_ \    | |/ / _ \ '__| '_ \ / _ \ |  / __/ _`
| | / __|
//       | |  | | (_| | | | | | |   <  __/ |  | | | |  __/ | | (_| (_|
| | \__    //        |_|  |_|\__,_|_|_| |_| |_|\_\___|_|  |_| |_|\___|_|
\___\__,_|_|_|___/
//
// ***************************************************************
**********
// ***************************************
//
//        BLOCKS/NUM THREADS PER BLOCK CONSTANTS
//
// ***************************************
const int kCurrentThreadsPerBlock = 256; // Should be less than 1024
for the current architecture (maxwell)
dim3 kNumThreadsPerBlock = kCurrentThreadsPerBlock; // How many
```

```
threads per block to use currently (non-interpolation)
dim3 kNumBlocks = (kNumStartPoints / kCurrentThreadsPerBlock) + 1; //
How many blocks (non-interpolation)
std::cout << "NumThreads per block: " << kNumThreadsPerBlock.x << "
Num Blocks: " << kNumBlocks.x << std::endl;
//float t = 1.0;
//H_global <<<kNumBlocks, kNumThreadsPerBlock>>> (t, device_start,
device_start_parameters, device_final_parameters, device_bArray,
gamma, n, device_solFlag_array, kNumParameters);
//std::cout << "***********Skipping randomization." << std::endl;
//std::cout << "***********Skipping path tracking." << std::endl;
//std::cout << "***********Skipping endgame sampling." << std::endl;
double* device_LArray_floatPtr =
reinterpret_cast<double*>(device_JArray);
//printf("Starting Randomization. Num Iterations: %i, TolX: %0.2f,
TolY: %0.2f, TolTheta: %0.2f
", kNumIterations, kTolX, kTolY, kTolTheta);
std::cout << "Transforming task positions." << std::endl;
transformTaskPositions(taskPositions, host_final_parameters);
std::cout << "Transformed task positions: ";
for (int i = 0; i < kNumParameters; i++)
{
std::cout << host_final_parameters[i] << " ";
} std::cout << std::endl;
cudaStatus = cudaMemcpy(device_final_parameters,
host_final_parameters, kNumParameters * sizeof(complex_t),
cudaMemcpyHostToDevice);
```

```
assert(cudaStatus == cudaSuccess);
//// TP RANDOMIZATION KERNEL CALL. Note casting LArray as a pointer to
a float, which is fine because it is half the size (in bytes) as the
original complex_t array.
//randomizeTaskPositions <<<kNumBlocks, kNumThreadsPerBlock >>>
(device_start_parameters, device_final_parameters, device_start,
device_task_positions,
//      device_LArray_floatPtr, device_bArray, kTolX, kTolY,
kTolTheta, kNumParameters, kNumTPParameters);
//cudaStatus = cudaDeviceSynchronize();
//assert(cudaStatus == cudaSuccess);
//std::cout << "*******************SKIPPING TP RANDOMIZATION
*************" << std::endl;
//// ************** DEBUG **************
// // ****** CHECKING START POINTS AT H(x, 1)
//cudaStatus = cudaMemcpy(host_start, device_start, kNumIterations *
numStartPoints * n * sizeof(complex_t), cudaMemcpyDeviceToHost);
//assert(cudaStatus == cudaSuccess);
//cudaStatus = cudaMemcpy(host_final_parameters,
device_final_parameters, kParameterBytesTotal,
cudaMemcpyDeviceToHost);
//assert(cudaStatus == cudaSuccess);
//for (int i = 0; i < kNumIterations * numStartPoints; i++)
//{
//      complex_t *xStart = &host_start[i * n];
//      complex_t *finalParams = &host_final_parameters[i /
kNumStartPoints * kNumParameters];
```

```
//      std::cout << "Current sol: " << i << " Param offset: " << i /
kNumStartPoints * kNumParameters << std::endl;
//      H_CPU(1, xStart, host_start_parameters, finalParams, gamma,
n);
//}
//// *********** END DEBUG ************
// ***********************************
//                      PATH TRACKING
// ***********************************
size_t freeMem = 0;
size_t totMem = 0;
std::cout << "Starting path tracking..." << std::endl;
auto pathTrackingStart = std::chrono::high_resolution_clock::now();
trackPaths << <kNumBlocks, kNumThreadsPerBlock >> > (
device_start, device_start_parameters, device_final_parameters, gamma,
device_JArray,
device_LArray, device_UArray, device_PArray, device_bArray,
device_xPredictArray, device_xWorkspaceArray,
n, kNumSteps, kStepSize, kNumParameters, device_solFlag_array,
kNumStartPoints, kNumIterations, kNumNewtonCorrections);
//std::cout << "************** SKIPPING PATH TRACKING ************" <<
std::endl;
cudaStatus = cudaDeviceSynchronize();
if (cudaStatus != cudaSuccess)
{
std::cout << "CUDA error in path tracking, message: " <<
cudaGetErrorString(cudaStatus) << std::endl;
```

```
exit(-1);
}
auto pathTrackingEnd = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> dtPath = pathTrackingEnd -
pathTrackingStart;
std::chrono::seconds dtSecondsPT =
std::chrono::duration_cast<std::chrono::seconds>(dtPath);
std::cout << "Path tracking runtime: " << dtSecondsPT.count() << "
seconds." << std::endl;
// ***********************************
//                               ENDGAME
// ***********************************
// Saving the paths that converged at t = R, as to only run the
endgame on those paths.
// This is done to save memory.
cudaStatus = cudaMemcpy(host_solFlag_array, device_solFlag_array,
kNumStartPoints * kNumIterations * sizeof(int),
cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(host_start, device_start, kNumStartPoints *
kNumIterations * n * sizeof(complex_t), cudaMemcpyDeviceToHost);
std::vector<complex_t> rawSols;
std::vector<int> rawFinalParamsIdx;
int numRawSolutions = 0;
for (int threadIdx = 0; threadIdx < kNumStartPoints * kNumIterations;
threadIdx++)
{
```

```
if (host_solFlag_array[threadIdx] == 1)
{
int pointStartIdx = threadIdx * n;
for (int i = 0; i < n; i++)
{
rawSols.push_back(host_start[pointStartIdx + i]);
}
rawFinalParamsIdx.push_back(threadIdx / kNumStartPoints);
}
numRawSolutions += host_solFlag_array[threadIdx];
}
std::cout << "***Number of raw solutions: " << numRawSolutions <<
std::endl;
//std::cout << "********* SKIPPING NUM RAW SOLS CHECK
****************** " << std::endl;
// Free all old path tracking memory and re-allocate it with the new
(and smaller) memory necessary
// for endgame sampling.
cudaFree(device_start);
//cudaFree(device_final_parameters);
cudaFree(device_JArray);
cudaFree(device_LArray);
cudaFree(device_UArray);
cudaFree(device_PArray);
cudaFree(device_bArray);
cudaFree(device_xPredictArray);
cudaFree(device_xWorkspaceArray);
```

```
// Clearing the stack memory used by the path tracker
cudaStatus = cudaDeviceSetLimit(cudaLimitStackSize, 0);
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_start, n * numRawSolutions *
sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_JArray, n * n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_LArray, n * n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_UArray, n * n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_PArray, n * n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_bArray, n * numRawSolutions *
sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_xPredictArray, n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMalloc((void**)&device_xPredictArray, n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
```

```
cudaStatus = cudaMalloc((void**)&device_xWorkspaceArray, n *
numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(device_start, rawSols.data(), numRawSolutions
* n * sizeof(complex_t), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
int* device_tpIndexArray = NULL;
cudaStatus = cudaMalloc((void**)&device_tpIndexArray, numRawSolutions
* sizeof(int));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(device_tpIndexArray, rawFinalParamsIdx.data(),
numRawSolutions * sizeof(int), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
// Creating the array for sample times t = lambda^0*R, lambda^1*R,
..., lambda^{numsamples-1}*R, where R is the endgame boundary
complex_t* host_tArray = NULL;
host_tArray = (complex_t*)malloc(kNumSamplesEndGame *
sizeof(complex_t));
//std::uniform_real_distribution<double> tDistribution(0, 1);
//double lambda = tDistribution(generator);
std::cout << "Starting Endgame. Endgame lambda: " << kEndGameLambda <<
". Number of samples: " << kNumSamplesEndGame << std::endl;
std::cout << "Powerseries interpolation method: ";
if (kNumDerivatives == 1)
{
std::cout << "Newton.";
}
```

```
else if (kNumDerivatives == 2)
{
std::cout << "Hermite.";
}
else
{
std::cout << "Error: Method unrecognized.";
}
std::cout << std::endl;
for (int i = 0; i < kNumSamplesEndGame; i++)
{
host_tArray[i] = complex_t(pow(kEndGameLambda, i) * kEndGameBoundary,
0);
}
complex_t* device_tArray = NULL;
cudaStatus = cudaMalloc((void**)&device_tArray, kNumSamplesEndGame *
sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(device_tArray, host_tArray, kNumSamplesEndGame
* sizeof(complex_t), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
// Endgame prep, taking a sampling of points and their derivatives
complex_t* device_sampleArray = NULL;
cudaStatus = cudaMalloc((void**)&device_sampleArray, kNumIterations *
numRawSolutions * n * kNumSamplesEndGame * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
complex_t* device_sampleDXArray = NULL;
```

```
cudaStatus = cudaMalloc((void**)&device_sampleDXArray, kNumIterations
* numRawSolutions * n * kNumSamplesEndGame * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
kNumBlocks = (numRawSolutions / kNumThreadsPerBlock.x) + 1;
std::cout << "Starting endgame sampling. Num threads per block: " <<
kNumThreadsPerBlock.x << " Num blocks: " << kNumBlocks.x << std::endl;
auto egSamplingStart = std::chrono::high_resolution_clock::now();
samplePathsEndGame <<<kNumBlocks, kNumThreadsPerBlock >>> (
device_start, device_start_parameters, device_final_parameters, gamma,
device_JArray,
device_LArray, device_UArray, device_PArray, device_bArray,
device_xPredictArray,
device_xWorkspaceArray, device_sampleArray, device_sampleDXArray, n,
kNumStepsEndGame,
kNumParameters, kNumSamplesEndGame, device_tArray,
device_tpIndexArray, numRawSolutions, kNumIterations,
kNumNewtonCorrections);
/*std::cout << "*********** SKIPPING ENDGAME SAMPLING ************" <<
std::endl;*/
cudaStatus = cudaDeviceSynchronize();
if (cudaStatus != cudaSuccess)
{
std::cout << "CUDA error in endgame sampling, message: " <<
cudaGetErrorString(cudaStatus) << std::endl;
exit(-1);
}
auto egSamplingEnd = std::chrono::high_resolution_clock::now();
```

```
std::chrono::duration<double> dtEGSamp = egSamplingEnd -
egSamplingStart;
std::chrono::seconds dtSecondsEGSamp =
std::chrono::duration_cast<std::chrono::seconds>(dtEGSamp);
std::cout << "Endgame sampling runtime: " << dtSecondsEGSamp.count()
<< " seconds." << std::endl;
//freeing all the matrix memory as its no longer needed
cudaFree(device_JArray);
cudaFree(device_LArray);
cudaFree(device_UArray);
cudaFree(device_PArray);
// Clearing the stack memory used by the end game sampler
cudaStatus = cudaDeviceSetLimit(cudaLimitStackSize, 0);
assert(cudaStatus == cudaSuccess);
//complex_t* host_sampleArray;
//host_sampleArray = (complex_t*)malloc(kNumIterations *
numRawSolutions * n * kNumSamplesEndGame * sizeof(complex_t));
//cudaStatus = cudaMemcpy(host_sampleArray, device_sampleArray,
kNumIterations * numRawSolutions * n * kNumSamplesEndGame *
sizeof(complex_t), cudaMemcpyDeviceToHost);
//assert(cudaStatus == cudaSuccess);
//std::cout << "Writing sample files...." << std::endl;
//std::ofstream outFile;
//outFile.open("SamplesOut.txt");
//for (int i = 0; i < numRawSolutions; i++)
//{
//      complex_t *sampleStart = &host_sampleArray[i * n *
```

```
kNumSamplesEndGame];
//      for (int j = 0; j < kNumSamplesEndGame; j++)
//      {
//              for (int k = 0; k < n; k++)
//              {
//                      complex_t v = sampleStart[j * n + k];
//                      outFile << v.real() << " " << v.imag() << " ";
//              } outFile << std::endl;
//      }
//}
//inputBuffer.clear();
//std::ifstream inStream;
//inStream.open("SamplesOut.txt");
//if (inStream.is_open())
//{
//      std::cout << "Samples Out is open" << std::endl;
//      std::string line;
//      while (getline(inStream, line))
//      {
//              inputBuffer.push_back(line);
//      }
//      inputStream.close();
//}
//else
//{
//      std::cout << "Failed to open samples outfile." << std::endl;
//      exit(-1);
```

```
//}
//complex_t* host_sampleArray;
//host_sampleArray = (complex_t*)malloc(kNumIterations *
numRawSolutions * n * kNumSamplesEndGame * sizeof(complex_t));
//cudaStatus = cudaMemcpy(host_sampleArray, device_sampleArray,
kNumIterations * numRawSolutions * n * kNumSamplesEndGame *
sizeof(complex_t), cudaMemcpyDeviceToHost);
//assert(cudaStatus == cudaSuccess);
//for (unsigned i = 0; i < inputBuffer.size(); i++)
//{
//      std::istringstream inputSS(inputBuffer[i]);
//      std::vector<std::string>
stringSplit(std::istream_iterator<std::string>{inputSS},
std::istream_iterator<std::string>());
//      for (unsigned j = 0; j < stringSplit.size(); j += 2)
//      {
//              std::string realStr = stringSplit[j];
//              std::string imgStr = stringSplit[j + 1];
//              std::istringstream realStrStream(realStr);
//              std::istringstream imgStrStream(imgStr);
//              double realPart, imgPart;
//              realStrStream >> realPart;
//              imgStrStream >> imgPart;
//              complex_t currentVal(realPart, imgPart);
//              host_sampleArray[i * n + j] = currentVal;
//      }
//}
```

```
//numRawSolutions = inputBuffer.size() / 15;
//std::cout << "num raw solutions injected: " << numRawSolutions <<
std::endl;
//cudaStatus = cudaMemcpy(device_sampleArray, host_sampleArray,
kNumIterations * numRawSolutions * n * kNumSamplesEndGame *
sizeof(complex_t), cudaMemcpyHostToDevice);
//assert(cudaStatus == cudaSuccess);
complex_t* device_sArray = NULL;
cudaStatus = cudaMalloc((void**)&device_sArray, numRawSolutions * n *
kNumSamplesEndGame * kNumDerivatives * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
complex_t* device_xSampleArray = NULL;
cudaStatus = cudaMalloc((void**)&device_xSampleArray, numRawSolutions
* n * kNumSamplesEndGame * kNumDerivatives * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
complex_t* device_dsArray = NULL;
cudaStatus = cudaMalloc((void**)&device_dsArray, numRawSolutions * n *
kNumSamplesEndGame * kNumDerivatives * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
complex_t* device_aArray = NULL;
cudaStatus = cudaMalloc((void**)&device_aArray, kNumDerivatives *
kNumSamplesInterpolation * n * numRawSolutions * sizeof(complex_t));
assert(cudaStatus == cudaSuccess);
// repurposing the original device_sampleArray and device_DXArray to
hold the values
// of S and dS in the interpolation.
cudaStatus = cudaMemGetInfo(&freeMem, &totMem);
```

```
std::cout << cudaGetErrorString(cudaStatus) << std::endl;
assert(cudaStatus == cudaSuccess);
std::cout << "Before interpolation, Memory (free, tot) = " << freeMem
<< " " << totMem << std::endl;
const int kNumThreadsInterpolation = numRawSolutions * n; // Number of
threads to use in the interpolation kernel call.
dim3 kNumThreadsPerBlock_Interpolation = kCurrentThreadsPerBlock;
dim3 kNumBlocks_Interpolation = (kNumThreadsInterpolation /
kCurrentThreadsPerBlock) + 1; // Number of blocks (interpolation)
std::cout << "Starting Interpolation, Num Threads per block: " <<
kNumThreadsPerBlock_Interpolation.x << " Num Blocks (interpolation) "
<< kNumBlocks_Interpolation.x << std::endl;
auto egInterpStart = std::chrono::high_resolution_clock::now();
int* minCArray = NULL;
minCArray = (int *)malloc(numRawSolutions * sizeof(int));
double *minResidualArray = NULL;
minResidualArray = (double *)malloc(numRawSolutions * sizeof(double));
complex_t *minCSols = NULL;
minCSols = (complex_t *)malloc(numRawSolutions * n *
sizeof(complex_t));
complex_t* host_dsArray = NULL;
host_dsArray = (complex_t *)malloc(numRawSolutions * n *
kNumSamplesEndGame * kNumDerivatives * sizeof(complex_t));
complex_t* dsPtr = NULL;
// Running the interpolation code testing c = 1, ..., kMaxC
for (int c = 1; c <= kMaxC; c++)
{
```

```
runEndGame <<< kNumBlocks_Interpolation ,
kNumThreadsPerBlock_Interpolation >>> (
device_tArray , device_sampleArray , device_sampleDXArray ,
device_aArray , device_sArray ,
device_dsArray , device_xSampleArray , device_start ,
kNumThreadsInterpolation , c , kNumSamplesEndGame,
kNumSamplesInterpolation , kNumDerivatives , n);
cudaStatus = cudaDeviceSynchronize();
assert(cudaStatus == cudaSuccess);
// Copying the data from the endgame back to find approximated
solutions
cudaStatus = cudaMemcpy(host_dsArray , device_dsArray , kNumIterations *
numRawSolutions * n * kNumSamplesEndGame * kNumDerivatives *
sizeof(complex_t), cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
// Finding which which c is minimal and saving the solution
for (int pathIdx = 0; pathIdx < numRawSolutions; pathIdx++)
{
int currentRowSIdx = pathIdx * (n * kNumSamplesEndGame *
kNumDerivatives);
dsPtr = &host_dsArray[currentRowSIdx];
complex_t currentResidual = dsPtr[2];
double currentError = currentResidual.real() * currentResidual.real()
+ currentResidual.imag() * currentResidual.imag();
if (c == 1) // min arrays are unpopulated, so populate it with the
first error
{
```

```
minCArray[pathIdx] = c;
minResidualArray[pathIdx] = currentError;
int xSolIdx = pathIdx * n;
complex_t* xSolPtr = &minCSols[xSolIdx];
for (int i = 0; i < n; i++)
{
int minCRowIdx = pathIdx * (n * kNumSamplesEndGame * kNumDerivatives)
+ (c - 1) * n * kNumSamplesEndGame * kNumDerivatives +
kNumSamplesEndGame * kNumDerivatives * i;
dsPtr = &host_dsArray[minCRowIdx];
xSolPtr[i] = dsPtr[3];
}
}
else
{
if (currentError < minResidualArray[pathIdx])
{
minCArray[pathIdx] = c;
minResidualArray[pathIdx] = currentError;
int xSolIdx = pathIdx * n;
complex_t* xSolPtr = &minCSols[xSolIdx];
for (int i = 0; i < n; i++)
{
int minCRowIdx = pathIdx * (n * kNumSamplesEndGame * kNumDerivatives)
+ (c - 1) * n * kNumSamplesEndGame * kNumDerivatives +
kNumSamplesEndGame * kNumDerivatives * i;
dsPtr = &host_dsArray[minCRowIdx];
```

```
xSolPtr[i] = dsPtr[3];
}
}
}
}
}
auto egInterpEnd = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> dtEGInterp = egInterpEnd -
egInterpStart;
std::chrono::seconds dtSecondsEGInterp =
std::chrono::duration_cast<std::chrono::seconds>(dtEGInterp);
std::cout << "Endgame interpolation runtime: " <<
dtSecondsEGInterp.count() << " seconds." << std::endl;
//for (int pathIdx = 0; pathIdx < numRawSolutions; pathIdx++)
//{
//      int minC = minCArray[pathIdx];
//      double minError = minResidualArray[pathIdx];
//      std::cout << "Current Path: " << pathIdx << " min c: " << minC
<< " min error: " << minError << std::endl;
//}
//      //std::cout << "Current Path: " << pathIdx << " min c: " <<
minC << " min error: " << minError << std::endl;
//      // Save the solution with the c that gave the minimum residual
error.
//      int xSolIdx = pathIdx * n;
//      complex_t* xSolPtr = &host_start[xSolIdx];
//      for (int i = 0; i < n; i++)
```

```
//       {
//               int minCRowIdx = pathIdx * (kMaxC * n *
kNumSamplesEndGame * kNumDerivatives) + (minC - 1) * n *
kNumSamplesEndGame * kNumDerivatives + kNumSamplesEndGame *
kNumDerivatives * i;
//               dsPtr = &host_dsArray[minCRowIdx];
//               xSolPtr[i] = dsPtr[3];
//       }
//}
//cudaStatus = cudaMemcpy(host_solFlag_array, device_solFlag_array,
kNumIterations * numStartPoints * sizeof(int),
cudaMemcpyDeviceToHost);
//assert(cudaStatus == cudaSuccess);
//int numSolutionsPolynomialSystem = 0;
//for (int i = 0; i < kNumIterations * numStartPoints; i++)
//{
//       numSolutionsPolynomialSystem += host_solFlag_array[i];
//}
//printf("Path tracking complete. Number of unfiltered solutions: %d
", numSolutionsPolynomialSystem);
std::ofstream outFile("rawSolsCPP.txt");
//float injectedSol[40] = { 0.1088303386319257, 0.4911571324181687,
0.1088303387261076, -0.4911571324531193, 0.07117415413355943,
0.3341690495591053, 0.07117415426176278, -0.3341690497753884,
0.2741863557240121, 0.4663521127922585, 0.2741863556175026,
-0.4663521128679312, 0.3136778605556542, 0.4972361568776717,
0.3136778606593763, -0.4972361568866053, 0.001415581318431086,
```

```
0.4874874795569437, 0.001415581206383243, -0.4874874797105733,
0.6522271756891201, 0.7580235553954101, 0.6522271761837017,
-0.7580235559702146, 0.1186133680115465, 0.9929405158257079,
0.1186133681255039, -0.9929405167796594, -0.8608714601010493,
0.5088224910491206, -0.8608714616386132, -0.5088224919579052,
-0.9750630480598933, 0.2219280282629836, -0.9750630505750243,
-0.2219280288354361, -0.7533699643505547, -0.6575969081920536,
-0.753369966724192, 0.6575969102639376 };
//for (int i = 0; i < n; i++)
//{
//      complex_t v(injectedSol[2 * i], injectedSol[2 * i + 1]);
//      minCSols[i] = v;
//}
// ***** CHECKING SOLUTIONS COMING OUT OF THE ENDGAME
int numHomotopySols = 0;
std::cout << "*************** Endgame check assuming only one
iteration here, fix later ***************" << std::endl;
for (int i = 0; i < numRawSolutions; i++)
{
complex_t *xStart = &minCSols[i * n];
// complex_t *finalParams =
&host_final_parameters[rawFinalParamsIdx[i]];
complex_t *finalParams = &host_final_parameters[0];
//std::cout << "Current sol: " << i << std::endl;
double e = H_CPU(0, xStart, host_start_parameters, finalParams, gamma,
n);
//double e1 = H_CPU(0, xStart, host_start_parameters, finalParams,
```

```
complex_t(1,0), n);
if (i == 0)
{
std::cout << "Injected solution residual: " << e << std::endl;
}
//std::cout << "Solution: " << i << ", Residual error (with gamma): "
<< e << std::endl;
//std::cout << "Solution: " << i << ", Residual error (with gamma=1):
" << e1 << std::endl;
if (e < 1e-3)
{
//std::cout << "Sol vals: ";
for (int j = 0; j < n; j++)
{
complex_t v = xStart[j];
if (j != (n - 1))
{
outFile << v.real() << " " << v.imag() << " ";
}
else
{
outFile << v.real() << " " << v.imag();
}
} outFile << std::endl;
numHomotopySols++;
}
}
```

```
std::cout << "***Number of homotopy solutions at t=0: " <<
numHomotopySols << std::endl;
cudaStatus = cudaMemcpy(device_start, minCSols, numRawSolutions * n *
sizeof(complex_t), cudaMemcpyHostToDevice);
assert(cudaStatus == cudaSuccess);
std::cout << "Starting Filtering, Num Threads per block: " <<
kNumThreadsPerBlock.x << " Num Blocks (filtering) " << kNumBlocks.x <<
std::endl;
filterAndTransformSolutions << <kNumBlocks, kNumThreadsPerBlock >> >
(device_start, device_task_positions, device_solFlag_array, n,
kNumTPParameters);
cudaStatus = cudaDeviceSynchronize();
if (cudaStatus != cudaSuccess)
{
std::cout << "CUDA error in filtering, message: " <<
cudaGetErrorString(cudaStatus) << std::endl;
exit(-1);
}
printf("Filtering complete. Transferring successful solutions to CPU.
");
cudaStatus = cudaMemcpy(host_task_positions, device_task_positions,
kTPSetBytesTotal, cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
cudaStatus = cudaMemcpy(host_solFlag_array, device_solFlag_array,
numRawSolutions * sizeof(int), cudaMemcpyDeviceToHost);
assert(cudaStatus == cudaSuccess);
complex_t* solBuf = NULL;
```

```
solBuf = (complex_t*)malloc(n * sizeof(complex_t));
std::vector<solution_t> solVec;
for (unsigned i = 0; i < numRawSolutions; i++)
{
if (host_solFlag_array[i] == 10)
{
complex_t* solPtr = &minCSols[i * n];
std::cout << "Solution " << i << " checksum: 10" << std::endl;
for (int j = 0; j < n; j++)
{
std::cout << solPtr[j] << " ";
} std::cout << std::endl;
solution_t currentSol;
currentSol.ptA = host_final_parameters[15];
currentSol.ptB = host_final_parameters[17];
currentSol.ptC = solPtr[0];
currentSol.ptD = solPtr[2];
currentSol.ptF = solPtr[4];
currentSol.ptG = solPtr[6];
currentSol.ptH = solPtr[8];
std::vector<double> currentRelativeAngles;
for (unsigned j = 10; j < n; j+=2)
{
complex_t currentAng = solPtr[j];
double currentPsiRel = atan2(currentAng.imag(), currentAng.real());
currentRelativeAngles.push_back(currentPsiRel);
}
```

```
currentSol.relativePsiAngles = currentRelativeAngles;
solVec.push_back(currentSol);
}
//else if (host_solFlag_array[i] > 0)
//{
//      complex_t* solPtr = &minCSols[i * n];
//      std::cout << "Solution " << i << " checksum: " <<
host_solFlag_array[i] << std::endl;
//      for (int j = 0; j < n; j++)
//      {
//              std::cout << solPtr[j] << " ";
//      } std::cout << std::endl;
//}
}
printf("%zd solutions passed filtering.
", solVec.size());
//for (int solIdx = 0; solIdx < 2; solIdx++)
//{
//      printf("Solution %d:
", solIdx);
//      solution_t currSol = solVec[solIdx];
//      for (int i = 0; i < n; i++)
//      {
//              printf("%0.5f ", currSol.solutionCoordinates[i]);
//      }
//      printf("
");
```

```
//}
// ANALYSIS OF THE FILTERED SOLUTIONS
std::vector<solution_t>designsVec;
for (unsigned i = 0; i < solVec.size(); i++)
{
solution_t currentSol = solVec[i];
bool goodSol = sixBarTwoFixedPivotAnalysis(currentSol, taskPositions);
if (goodSol)
{
designsVec.push_back(currentSol);
}
}
printf("%d design candidates passed analysis.
", designsVec.size());
// Writing the solutions to a text file so they can be drawn by
Mathematica
std::ofstream solFile("solutions.txt");
std::ofstream tpsFile("taskpositions.txt");
for (unsigned i = 0; i < designsVec.size(); i++)
{
solution_t currentSol = designsVec[i];
for (int j = 0; j < n; j++)
{
solFile << currentSol.ptA.real() << " " << currentSol.ptA.imag() <<
std::endl;
solFile << currentSol.ptB.real() << " " << currentSol.ptB.imag() <<
std::endl;
```

```
solFile << currentSol.ptC.real() << " " << currentSol.ptC.imag() <<
std::endl;
solFile << currentSol.ptD.real() << " " << currentSol.ptD.imag() <<
std::endl;
solFile << currentSol.ptF.real() << " " << currentSol.ptF.imag() <<
std::endl;
solFile << currentSol.ptG.real() << " " << currentSol.ptG.imag() <<
std::endl;
solFile << currentSol.ptH.real() << " " << currentSol.ptH.imag() <<
std::endl;
}
for (int k = 0; k < kNumTPParameters; k++)
{
tpsFile << taskPositions[i] << " ";
}
solFile << std::endl;
tpsFile << std::endl;
}
solFile.close();
tpsFile.close();
cudaFree(device_start);
cudaFree(device_start_parameters);
cudaFree(device_final_parameters);
cudaFree(device_JArray);
cudaFree(device_bArray);
cudaFree(device_xPredictArray);
cudaFree(device_xWorkspaceArray);
```

```
cudaFree(device_PArray);
cudaFree(device_LArray);
cudaFree(device_UArray);
cudaFree(device_task_positions);
cudaFree(device_solFlag_array);
free(host_start);
free(host_start_parameters);
free(host_final_parameters);
free(host_task_positions);
free(solBuf);
free(host_solFlag_array);
// cudaDeviceReset must be called before exiting in order for
profiling and
// tracing tools such as Nsight and Visual Profiler to show complete
traces.
cudaStatus = cudaDeviceReset();
if (cudaStatus != cudaSuccess) {
fprintf(stderr, "cudaDeviceReset failed!");
return 1;
}
auto tf = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> dt = tf - t0;
std::chrono::seconds dtSeconds =
std::chrono::duration_cast<std::chrono::seconds>(dt);
std::cout << "runtime: " << dtSeconds.count() << " seconds." <<
std::endl;
return 0;
```

```
}
```