UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Aircraft Design Optimization as a Geometric Program

Permalink
https://escholarship.org/uc/item/0928939m

Author
Hoburg, Warren Woodrow

Publication Date
2013

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Aircraft Design Optimization as a Geometric Program

by

Warren Woodrow Hoburg

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Alexandre Bayen
Professor Laurent El Ghaoui
Professor Andrew Packard

Fall 2013

Aircraft Design Optimization as a Geometric Program

Copyright 2013
by
Warren Woodrow Hoburg

Abstract

Aircraft Design Optimization as a Geometric Program

by

Warren Woodrow Hoburg

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Pieter Abbeel, Chair

Recent advances in convex optimization make it possible to solve certain classes of constrained optimization problems reliably and efficiently. These techniques offer significant advantages over general nonlinear optimization methods. In this thesis, conceptual-stage aircraft design problems are formulated as geometric programs (GPs), which are a specific type of convex optimization problem. Modern GP solvers are extremely fast, even on large problems, require no initial guesses or tuning of solver parameters, and guarantee *globally* optimal solutions. They also return optimal dual variables, which encode sensitivity information that is especially relevant in an aircraft design context. These benefits come at a price: all objective and constraint functions – the mathematical models that describe aircraft design relations – must be expressed within the restricted functional forms of GP. Perhaps surprisingly, this restricted set of functional forms appears again and again in prevailing physics-based models for aircraft systems. Moreover, for models that cannot be manipulated algebraically into the forms required by GP, one can use methods developed in this thesis to fit compact GP models that accurately approximate the original models. Each of these ideas is illustrated by way of concrete examples from aircraft design.

To my parents, who always supported me no matter what.

Contents

Contents **ii**

1 Introduction **1**
1.1 The Aircraft Design Process 1
1.2 Why Geometric Programming? 6
1.3 Thesis Overview and Summary of Contributions 7

2 The GP Design Paradigm **10**
2.1 Geometric Programming 10
2.2 A Simple Example 16
2.3 GP Modeling 21
2.4 Exploring Tradeoffs 25

3 Sensitivity Analysis and the Power of Lagrange Duality **27**
3.1 Maximum Entropy Dual of a Geometric Program 28
3.2 Sensitivity Analysis 30
3.3 Fixed Variable Sensitivities 33
3.4 Aside: Recovering known Scaling Laws 34
3.5 Linearized Propagation of Log-Normal Uncertainties 37

4 Fitting GP Models to Data **39**
4.1 The Convex Regression Problem 39
4.2 Some Convex Function Classes 41
4.3 Application to Geometric Programming 45
4.4 Fitting Model Parameters 47
4.5 Numerical Examples and Comparisons 53
4.6 Conclusions 58

5 Selected GP-Compatible Aircraft Design Models **61**
5.1 Steady Flight Relations 62
5.2 Weight, Drag, and Efficiency Breakdowns 63
5.3 Performance Metrics 65
5.4 Propulsive Efficiency 70

5.5 Lifting Surface Structural Models 71
5.6 Stall 78

6 Aircraft Design Example **80**

7 Current Limitations and Future Perspectives **86**

Bibliography **91**

A Airfoil Area and Inertia Calculations **98**

Acknowledgments

I consider myself incredibly fortunate to be surrounded by a number of extraordinary people. First, I am grateful to my adviser, Pieter Abbeel, for his never-ending supply of generosity, patience, and good ideas. Pieter has always wanted the very best for his students, and I have benefited from and enjoyed our many discussions of challenging applied math problems. He has left an invaluable mark on my professional development.

At Berkeley, I am especially thankful for unforgettable lunches with Claire Thomas, and happy hours with Aude Hofleitner. It has been a pleasure to be surrounded by so many brilliant and interesting students, such as Tim Hunter, Pat Virtue, Arjun Singh, Jeremy Maitin-Shepard, Stephen Miller, Teodor Moldovan, Jon Barron, Bharath Hariharan, Greg Durrett, and John Duchi.

My thesis committee members Laurent El Ghaoui, Alex Bayen, and Andy Packard have been a pleasure to work with. They made time to meet with me despite their busy schedules, and took a genuine interest in my research.

Thank you to Ana S. Rufino Ferreira, Max Balandat, Sara Alspaugh, Tony Kim, and Maude David for all the hard-won volleyball victories (and defeats).

Graduate school would not have been the same without spending time on YOSAR and BAMRU. I shared amazing camraderie and friendships with my teammates, and am constantly impressed by their skill, professionalism, and dedication to service.

Through most of my time at Berkeley, Genita Metzler has been a wonderful friend and companion. I would also like to thank my long-time housemates Kieren Patel and Nick Rey. They have been incredibly positive influences in many ways.

At Boeing, Adam Marshall, Tom Grandine, and Matt Patterson were extraordinary managers and mentors, and gave me exposure to challenging problems in manufacturing optimization.

Before graduate school, I was shaped by several mentors. Karen Willcox continues to inspire me year after year. Russ Tedrake taught me a great deal about good research, good science, and how to run a lab. Sheila Widnall has helped to guide me throughout my career; I will be pleased if I do half as many things in my life as she has done in hers.

My undergraduate years at MIT were filled with climbing successes and failures with my friend Darren, who is always up for a challenge. He and I also raced for the ski team, where my coach Todd Dumond taught me a lot about hard work.

I am grateful to Daniel Limonadi at JPL for taking me under his wing as a summer intern, and for unforgettable weekly coffee discussions. Ben Ingram and Mike Anderberg, formerly at Frontier Systems/Boeing Phantom Works, greatly shaped who I am as an engineer.

Before I ever set foot on a university campus, I spent my summers building high power amateur rockets. Mark Mazzon and Kreig Williams took me under their wing in those formative years, with the utmost patience and generosity. They profoundly influenced me in the most positive way.

Finally, my brother, mom, and dad have given me unconditional love and support; they are the best family imaginable.

Nomenclature

α	Softness parameter	$\bar{\boldsymbol{A}}$	fixed variable exponents
δ	wing tip deflection [m]	b	wing span [m]
η	nondimensional spanwise coordinate	$\boldsymbol{b}$	$\equiv \log \boldsymbol{c}$
η_0	overall efficiency	$\boldsymbol{c}$	vector of monomial coefficients
η_0	overall efficiency	C_D	total drag coefficient
η_i	inviscid propeller efficiency	c_d	2D profile drag coefficient
η_v	viscous propeller efficiency	C_f	skin friction coefficient
η_{eng}	engine efficiency	C_L	lift coefficient
η_{prop}	propeller efficiency	C_{Dp}	profile drag coefficient
γ	climb angle	$C_{L,\max}$	max lift coefficient
$\bar{\gamma}$	descent angle, $-\gamma$	CDA_0	non-wing drag area [m^2]
λ	wing taper ratio	D	drag force [N]
$\boldsymbol{\lambda}$	dual Lagrange multipliers	e	Oswald efficiency factor
μ	fluid viscosity [kg/(s·m)]	E_{bat}	available battery energy [J]
ν	$\equiv (1+\lambda+\lambda^2)/(1+\lambda)^2$	g	gravitational constant, 9.8 m/s^2
$\boldsymbol{\nu}$	dual variables (see 3.1)	h	altitude [m]
ρ	air density [kg/m^3]	h_{fuel}	fuel heating value [J/kg]
τ	wing thickness ratio	$\bar{h}_{\text{rms}}$	root mean square spar box height
θ_{fuel}	fuel fraction, $W_{\text{fuel}}/W_{\text{zfw}}$	I_r	root area moment of inertia [m^4]
ξ	takeoff drag-to-thrust ratio	$\bar{I}_{\text{cap}}$	area moment of inertia per chord^4
ζ	stall margin	k	pressure drag form factor
A	aspect ratio	K_i	number of terms in posynomial i
$\boldsymbol{A}$	matrix of monomial exponents	L	lift force [N]

L'	lift force per unit span [N/m]
m	number of constraints
M_r	root moment [N·m]
$\dot{m}_{\text{fuel}}$	fuel mass flow rate [kg/s]
$\bar{M}_r$	root moment per chord, M_r/c_r
n	number of decision variables
N_{lift}	ultimate load factor
p	$\equiv 1 + 2\lambda$
P_{fuel}	fuel power, $\dot{m}_{\text{fuel}} h_{\text{fuel}}$ [W]
P_{max}	max engine output power [W]
q	$\equiv 1 + \lambda$
R	Range [m]
Re	Reynolds number
S	wing area [m^2]
S_r	root section modulus [m^3]
$s_{\bar{\gamma}}$	$\sin(-\gamma)$
s_{γ}	$\sin(\gamma)$
$\mathcal{S}$	log-space sensitivity
T	thrust force [N]
t	number of monomial terms in GP
$\bar{t}_{\text{cap}}$	spar cap thickness per unit chord
$\bar{t}_{\text{web}}$	shear web thickness per unit chord
$\boldsymbol{u}$	vector of decision variables
V	flight speed [m/s]
V_{S0}	stall speed, flaps extended [m/s]
W	operating weight [N]
W_0	fixed weight [N]
W_{cap}	spar cap weight [N]
W_{eng}	engine weight [N]
$W_{\text{fuel,out}}$	weight of fuel burned, outbound [N]
$W_{\text{fuel,ret}}$	weight of fuel burned, return [N]
W_{MTO}	maximum takeoff weight [N]
W_{pay}	payload weight [N]
W_{web}	shear web weight [N]
W_{w}	wing weight [N]
W_{zfw}	zero fuel weight [N]
$\tilde{W}$	weight excluding wing [N]
$\boldsymbol{x}$	log-transformed decision variables
x_{TO}	takeoff distance [m]
$\bar{\boldsymbol{x}}$	fixed variables (log-transformed)
z	helper variable
$\boldsymbol{z}$	$\equiv \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$
z_{bre}	Breguet parameter

Chapter 1

Introduction

"Aeronautics was neither an industry nor even a science... it was a miracle."

– Igor Sikorsky

Today's aircraft are some of the most complex engineering systems ever conceived and built. Designing, testing, certifying, and producing an aircraft is a monumental undertaking requiring millions of decisions and years of effort. For each stakeholder involved, be they aircraft manufacturers, airlines, government operators, regulatory agencies, or investors, the stakes are high – decisions made early in the design process can lock in operational costs, marketability, mission constraints, and manufacturing costs for decades to come. With such a wide range of engineering and management disciplines governing their design, aircraft exemplify the challenges of modern engineering design.

1.1 The Aircraft Design Process

The aircraft design process has evolved considerably since the Wright Brothers' historic first flight in 1903. Early on, many design decisions were guided by trial-and-error, wind tunnel testing, and empirical studies. Analytical theory gradually entered the picture, and by the 1960's many of the major aerodynamic theories, including potential flow theory [69], thin airfoil theory [1], slender body theory [6], flight dynamics [26], and blade-element/vortex propeller theory [29, 70, 27, 2] were established. These theories continue to carry weight

today – they are important sanity checks for higher fidelity analyses, and provide useful intuition about scaling and tradeoffs among relevant parameters.

The advent of the digital computer created nothing short of a revolution in aerospace analysis. In aerodynamics, a spectrum of numerical tools – commonly referred to as *computational fluid dynamics* (CFD) – were developed to solve the governing equations of fluid dynamics. These tools generally provide faster turnaround than wind tunnel testing, and thus have enabled expert designers to explore a greater number of design configurations and further refine their solutions. Examples of CFD methods, which represent a spectrum of computational complexity and accuracy of solution, include panel methods [39], potential methods with viscous corrections [23], Euler methods [17], and Reynolds-Averaged-Navier-Stokes (RANS) [43]. Similarly, in structural engineering, finite element methods have enabled numerical analysis of a wide range of configurations [64]. In guidance and control, computation is widely relied upon to analyze control systems with large numbers of inputs, outputs, and states. Across all disciplines, computation and numerical simulation have become standard tools.

In order to appreciate the context in which these tools are applied, the typical stages of aircraft development must be understood. For many decades, engineers and managers have thought of the aircraft design process in terms of three distinct stages:

Conceptual design is where the initial business case for an aircraft program is made. Design missions are defined, performance requirements are agreed upon, and initial sizing studies are conducted. Engineering development efforts are prioritized. Depending on the scope of the project, conceptual design may last several years, and could involve fairly detailed analysis and mathematical modeling. By the end of conceptual design, the basic aircraft configuration ('what goes where') is known, sizing of major elements is mostly fixed, and performance figures, major component weights, direct operating costs, and manufacturing costs have all been estimated.

Preliminary design involves a steadily-increasing level of understanding of the selected aircraft configuration. More detailed studies of each subsystem are conducted; structural instabilities and aerodynamic interferences are identified and corrected; the outer mold line is lofted (modeled mathematically); mockups (simulated or physical) are

created. At some point during preliminary design, the basic aircraft configuration is *frozen*. The purpose of this design freeze is to stop modifying major coupling variables, such that subsystem design teams can move on to detailed design work independently, without repeatedly affecting the basic assumptions of other teams.

Detailed design involves the engineering definition (e.g., CAD drawing) of every part on the aircraft; detailed stress analysis of all load bearing parts; routing of electrical, hydraulic, and fuel lines; and fabrication of production tooling. The number of parts under consideration can be quite large – for example, according to Boeing, there are approximately 3 million parts in a 777, provided by 500 suppliers worldwide.

Much of the activity that occurs during each of these design stages amounts (notionally) to solving (or iterating on) a large, constrained optimization problem. As a design program progresses through conceptual, preliminary, and detailed design, there is a natural growth in complexity, size of organization, and cost – the number of decision variables and constraints increases, dramatically at times. Moreover, because it is so incredibly complex and nuanced, the full 'optimization problem' is usually solved not by a single numerical method on a computer, but rather by teams of engineers. This introduces a decomposition structure into the notional optimization problem, creating significant organizational and communication challenges. Information sharing might be easier if one engineer were in charge of the entire design, but it can become quite encumbering with separate aerodynamics and structures teams whose analyses depend on shared variables like spanwise wing thickness distribution or flap hinge location. Systems engineers typically pass interdependencies among subsystem teams in the form of requirements and specifications that they revise as designs progress. Disciplinary designs proceed in parallel, with periodic design reviews to synchronize requirements. Between reviews, engineering teams are challenged to predict performance and optimize designs despite uncertainty about other subsystem designs. Because each design iteration is lengthy and costly, some programs only complete one or two formal design iterations before freezing the design [44].

The widespread introduction of computational models in all disciplines, combined with ever-improving computational power, have led researchers in academia and industry to seek solutions to the challenges outlined above. There has been significant excitement about

developing methods that would combine analysis tools and models from multiple disciplines into a coordinated numerical optimization. These research efforts, which fall broadly under the field of *multidisciplinary design optimization*, have made significant headway, but also continue to uncover serious challenges [33].

Multidisciplinary Design Optimization

The field of *Multidisciplinary Design Optimization* (MDO) emerged in the 1980's in response to the growing complexity of computational models in engineering design. The goal of MDO methods is to simultaneously incorporate models from multiple engineering disciplines into a coordinated design optimization. The hope is that by optimizing with respect to all relevant models, better designs can be found earlier in the design process.

Generally, MDO methods start with a set of black box computational routines, often segregated into *disciplines*, and define an *architecture* for coordinating calls to each of the routines. Numerous MDO architectures have emerged, with various breakdowns of computation into subproblems, communication schemes for passing coupling variables among computational blocks, and varied degrees of feasibility enforced after each iteration [15, 51, 3, 40, 44, 42, 66]. The effectiveness of these methods depends in some part on what is inside each of the black boxes. The most desirable situation is a set of computational procedures that, in addition to their outputs, return *gradients* of their output with respect to their input. Although the importance of gradient information seems obvious from an optimization point of view, many analysis routines in widespread use do not return any gradient information. This makes automatic differentiation a relevant area in MDO [50]. Even then, some black box routines introduce other difficulties such as nearest-neighbor table lookups that drive finite difference estimates of gradients to the uninformative value of zero. Dealing with these and related challenges continues to be a major challenge in MDO.

Another theme in MDO research is reducing, to the extent possible, the number of calls made to expensive solvers within the optimization loop. Examples include surrogate modeling via Kriging and response surface methods [65, 48]. Reduced order modeling is another area of great interest, where complex, high dimensional models are replaced by cheaper models that still capture relevant input/output relationships [13, 72]. Finally, there

is recent interest in *multifidelity* methods [63], where inexpensive computational models could be used to better guide which high-fidelity analyses to conduct.

In high-fidelity optimization, one of the great success stories of the past few decades has been the development of *adjoint methods*. This line of research started in the mid 1980's with Pironneau's study of control theory applied to shape optimization for systems governed by elliptic partial differential equations [59], and was soon followed by Jameson's seminal work on adjoint methods for aerodynamic design [34, 38, 35]. Adjoint methods are an efficient way to compute gradients in PDE-constrained optimization. In particular, when a problem is governed by a PDE, adjoint methods make it possible to compute the derivative of a cost function (drag, say) with respect to a set of design parameters, for approximately the same cost as solving the PDE once. These methods have evolved considerably thanks to significant research attention over the past two decades [37, 8, 25]. They have enabled gradient-based optimization in a wide range of applications, including designs involving multiple engineering disciplines (often aerodynamics and structures). Indeed, adjoint methods are the state of the art for high-fidelity aero-structural optimization [36, 49, 61].

Despite these successes, high-fidelity aerodynamic analysis and optimization tools remain just that – tools that an expert designer can use to make informed decisions. Adjoint methods provide remarkable capability for calculating sensitivities, but their computational cost still scales with the cost of solving the underlying PDE discretization. Depending on the scale of the problem, Euler and RANS solutions can require hours or even days. Indeed, almost all multidisciplinary design tools in commercial or industry use today either solve very specific problems involving a few disciplines (such as an engine design subroutine or a high-fidelity aero-structural wing optimization), or target arbitrary instances of very general design problems, and therefore take a long time (i.e. days or weeks) to arrive at a solution.

Days or weeks to improve aerodynamic performance is well worth the cost once the rough wing configuration is known, but this fidelity is unnecessary and even counterproductive in the early stages of conceptual design. When an aircraft configuration is first evaluated, the goal is to understand tradeoffs among various facets of the aircraft and mission. In many cases, the objectives (design missions) are not even defined, so the goal is to understand the shape of a Pareto frontier as opposed to optimizing a known objective. Moreover, beyond analysis of flight performance, today's design decisions are increasingly informed by a wider

range of considerations. Examples include designing for manufacturability, economic models for forecasting demand, and logistics models for improving supply chains. The wide range of models involved, along with the need to solve many similar design problems to sweep out tradeoff curves, calls for lower fidelity physics-based models that provide reasonable approximations over a wide range of parameter inputs.

Lower fidelity analysis also plays a vital role in sanity-checking high fidelity results. Without an optimization expert to set up the right problem and correctly interpret the results, high fidelity analysis can prove misleading [43]. Also, if the objective (design cases) are not chosen carefully, high fidelity optimizers may overfit their designs to peculiarities only present at the design case(s) under consideration. For example, it has been shown that even in the simple case of 2D airfoil optimization, unless the objective or design cases are carefully smoothed, optimizers tune airfoil shape to exploit the exact chordwise location of separation in the chosen design case(s), to the detriment of performance at other flight conditions [18].

For all these reasons, optimization of simple, reliable, physics-based models remains an important tool in modern aircraft design.

1.2 Why Geometric Programming?

As researchers continue studying design of engineering systems, computational tractability and scalability remain pressing issues.

Unfortunately, most optimization problems, including those that arise in aircraft design optimization, simply cannot be solved reliably or efficiently. The most common approach is to model the problem as a general nonlinear program (NLP) and use an algorithm such as sequential quadratic programming to find a local optimum. While such a framework can model extremely general problems, the solution techniques tend not to meet the reliability and efficiency bar needed for widespread adoption and day-to-day use in industry (exceptions typically involve either laborious tuning of solver parameters to a specific problem, or an optimization expert overseeing the solver). On the other hand, more specialized optimization problems (consider linear programs, for example) are easy to solve, but can be poor fits to the actual functions of interest.

Linear programs, however, are not the only type of optimization problem that is easy to solve. One of the quiet breakthroughs of the early 21st century has been the maturation and commercialization of algorithms for solving a much broader class of *convex optimization problems*. Thanks to these reliable and efficient algorithms, many general classes of convex program can now be solved globally on a desktop computer. One such general class, which appears particularly suited to problems in engineering design, is the *geometric program* (GP) [24, 9, 71, 10].

In GP formulations, a design problem is written as a numerical optimization problem of the form

$$\begin{aligned} \text{minimize} \quad & f_0(\boldsymbol{u}) \\ \text{subject to} \quad & f_i(\boldsymbol{u}) \leq 1, \;\; i = 1, ..., m, \end{aligned}$$

where the functions f_i have a special *posynomial* form that makes them compatible with geometric programming. The list of constraints typically includes mathematical models for the physics or governing relationships in the problem, as well as engineering requirements or performance specifications on the system.

The GP approach is not universal. By restricting ourselves to special functional forms, we give up the ability to model arbitrary nonlinear relationships. In return, we get something extremely powerful. Unlike solving a general nonlinear optimization problem, which is extremely hard, solving a GP is fast and easy. Among other benefits, modern solvers find *globally* optimal solutions, with fast solution times that scale to large problems.

1.3 Thesis Overview and Summary of Contributions

This thesis introduces the idea of formulating *conceptual* stage aircraft design problems as geometric programs.

Currently, the methods presented in this thesis apply to low-fidelity, physics-based models of the type that might be used in very early conceptual stage design [22]. They are not intended to replace higher fidelity analysis and optimization, which are vital tools later in the aircraft design process.

Compared with MDO, the methods of this thesis represent a unique approach. In both cases, the ultimate goal is to efficiently arrive at a solution that is supported by accurate modeling. Much of MDO starts with extremely accurate models – the very best – and makes sacrifices in efficiency or quality of optimization (e.g., accepting a local instead of global optimum). In contrast, the proposed approach starts with extremely reliable and efficient optimization – again, the very best – and makes sacrifices in the accuracy or fidelity of the models one can optimize over. The unique efficiency and reliability of GP methods makes them a powerful tool for optimizing large, multidisciplinary systems of low-order models.

The remainder of this thesis is organized as follows:

1. Introduction - Explains current challenges in system-level aircraft design. Outlines opportunities created by recent breakthroughs in convex optimization.

2. The GP Design Paradigm - Defines geometric programming and introduces its application to aircraft design problems. Explains general problem formulation techniques, including basic elements of a GP design formulation, simultaneous analysis of multiple flight conditions, multi-objective optimization, and Pareto frontier exploration. The ideas are grounded in a concrete aircraft design example.

3. Sensitivity Analysis and the Power of Lagrange Duality - Leverages the optimal values of the *Lagrange dual variables* – information that is determined *for free* when a GP is solved – to understand how tightening or loosening each constraint would affect the optimal objective value. Extends these ideas to cover sensitivity analysis of fixed problem variables and propagation of parameter uncertainties. Shows by way of example how sensitivity information from a single solution can be used to construct a surrogate Pareto frontier for design problems with multiple objectives.

4. Fitting GP Models to Data - Studies the problem of approximating black-box data sets with GP-compatible functions. Introduces two new function classes, softmax-affine and implicit softmax-affine, which provide provably better fits than existing max-affine methods. Discusses fitting algorithms and practical implementation considerations.

5. Selected GP-Compatible Aircraft Design Models - Gives examples of aircraft subsystem models and design relations that admit a GP-compatible formulation.

6. **Aircraft Design Example -** Solves an example aircraft design problem by formulating it as a GP, illustrating all of the concepts from previous chapters in the process.

7. **Current Limitations and Future Perspectives -** Describes current limitations and ongoing research directions.

Chapter 2

The GP Design Paradigm

This chapter describes a basic PG-powered conceptual design framework. After defining GP and discussing current solution technology, the chapter presents a simple design example, discusses general problem formulation, and describes characterization of Pareto frontiers in multiobjective optimization.

2.1 Geometric Programming

First introduced in 1967 by Duffin, Peterson, and Zener [24], a GP is a specific type of constrained optimization problem that becomes a convex optimization problem after a logarithmic change of variables. Despite significant work on early applications in structural design [53], network flow [58], and optimal control [9, 71], reliable and efficient numerical methods for solving GPs were not available until the 1990's [55]. GP has recently begun a resurgence as researchers discover promising applications in statistics [11], digital circuit design [12], antenna optimization [7], communication systems [14], and most recently, aircraft design [32].

Definition

This section uses *power law notation*: for two vectors $\boldsymbol{u}, \boldsymbol{a} \in \mathbb{R}^n$,

$$\boldsymbol{u}^{\boldsymbol{a}} \equiv \prod_{j=1}^{n} u_j^{a_j}. \tag{2.1}$$

Geometric programs are constrained optimization problems where the objective and constraints are made up of *monomial* and *posynomial* functions. We will start by defining these two special function classes.

Monomial Functions

In geometric programming, a *monomial*[1] is a function $h(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{R}^n_{++} \to \mathbb{R}_{++}$ of the form

$$h(\boldsymbol{u}) = c\boldsymbol{u}^{\boldsymbol{a}}, \tag{2.2}$$

where $\boldsymbol{a} \in \mathbb{R}^n$, and $c \in \mathbb{R}_{++}$. For instance, the familiar expression for lift, $\frac{1}{2}\rho V^2 C_L S$, is a monomial in $\boldsymbol{u} = (\rho, V, C_L, S)$, with $c = 1/2$ and $\boldsymbol{a} = (1, 2, 1, 1)$. Since the powers a_i in (2.2) may be negative and non-integer, expressions like $\frac{u_1 u_2^{0.7}\sqrt{u_3}}{u_4}$ are also monomials.

Posynomial Functions

Posynomials are functions $f(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{R}^n_{++} \to \mathbb{R}_{++}$, of the form

$$f(\boldsymbol{u}) = \sum_{k=1}^{K} c_k \boldsymbol{u}^{\boldsymbol{a}_k}, \tag{2.3}$$

where $\boldsymbol{a_k} \in \mathbb{R}^n$, and $c_k \in \mathbb{R}_{++}$. Thus, a posynomial is simply a sum of monomial terms, and all monomials are also posynomials (with just one term). The expression $0.23 + u_1^2 + 0.1 u_1 u_2^{-0.8}$ is an example of a posynomial in $\boldsymbol{u} = (u_1, u_2)$, whereas $2u_1 - u_2^{1.5}$ is not a posynomial because negative leading coefficients c_k are not allowed.

[1] As noted in [10], the term *monomial* carries a special meaning in GP; the term used in algebra is slightly different.

Geometric Program in Standard Form

A *geometric program in standard form* (also called a GP in posynomial form) is a non-linear, non-convex optimization problem of the form

$$\begin{aligned} \text{minimize} \quad & f_0(\boldsymbol{u}) \\ \text{subject to} \quad & f_i(\boldsymbol{u}) \leq 1, \;\; i = 1, ..., m, \\ & h_i(\boldsymbol{u}) = 1, \;\; i = 1, ..., m_e, \end{aligned} \tag{2.4}$$

where the f_i are posynomial (or monomial) functions, the h_i are monomial functions, and $\boldsymbol{u} \in \mathbb{R}^n_{++}$ are the *decision variables*. In plain English, a GP minimizes a posynomial objective function, subject to monomial equality constraints and posynomial inequality constraints. Monomials and posynomials are both closed under monomial division, so constraints of the form (posynomial $\leq$ monomial) or (monomial $=$ monomial) are easily converted into the form in (2.4). Also, any monomial equality constraint $h(\boldsymbol{u}) = 1$ may be expressed equivalently as two monomial inequality constraints: $h(\boldsymbol{u}) \leq 1$ and $1/h(\boldsymbol{u}) \leq 1$. Thus without loss of generality, a geometric program in standard form can always be written in the inequality-constrained form

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K_0} c_{0k} \boldsymbol{u}^{\boldsymbol{a}_{0k}} \\ \text{subject to} \quad & \sum_{k=1}^{K_i} c_{ik} \boldsymbol{u}^{\boldsymbol{a}_{ik}} \leq 1, \;\; i = 1, ..., m. \end{aligned} \tag{2.5}$$

This form is expected by some modern commercial solvers. The objective and constraints contain a combined total of $t = \sum_{i=0}^{m} K_i$ monomial terms. The entire GP is therefore parameterized by a vector of constants $\boldsymbol{c} \in \mathbb{R}^t$, an (often sparse) matrix of exponents $\boldsymbol{A} \in \mathbb{R}^{t \times n}$, and a mapping that encodes which of the $m + 1$ posynomials each of the t monomial terms resides in.

GP with Fixed Variables

When an engineering design problem is modeled as a GP, it is common for a subset of the variables to be fixed to constant values. For example, in a monomial model for lift, air

density might be set to a constant. Or, in a model for stress at a wing root, material density or Young's modulus might be set to a constant. More generally, a GP with *fixed variables* can be written in the form

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K_0} c_{0k} \bar{\boldsymbol{u}}^{\bar{\boldsymbol{a}}_{0k}} \boldsymbol{u}^{\boldsymbol{a}_{0k}} \\ \text{subject to} \quad & \sum_{k=1}^{K_i} c_{ik} \bar{\boldsymbol{u}}^{\bar{\boldsymbol{a}}_{ik}} \boldsymbol{u}^{\boldsymbol{a}_{ik}} \leq 1, \;\; i = 1, ..., m, \end{aligned} \tag{2.6}$$

where the vector $\bar{\boldsymbol{u}}$ contains the constant values of the fixed variables, and the vectors $\bar{\boldsymbol{a}}_{ik}$ contain the corresponding monomial exponents. We will refer back to this form in Chapter 3.

Geometric Program in Convex Form

The power of geometric programming lies in a transformation that converts GPs into *convex optimization problems*. In particular, consider the change of variables

$$\boldsymbol{x} = \log \boldsymbol{u}. \tag{2.7}$$

The logarithm of a monomial function becomes affine when written in terms of $\boldsymbol{x}$,

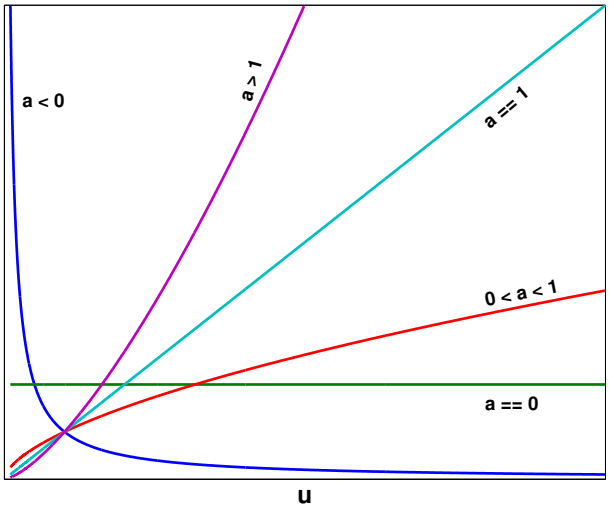
$$\log c\boldsymbol{u}^{\boldsymbol{a}} = \log c + \boldsymbol{a}^{\mathrm{T}} \boldsymbol{x}. \tag{2.8}$$

The logarithm of a posynomial function becomes the logarithm of a sum of exponentials of affine functions of $\boldsymbol{x}$,

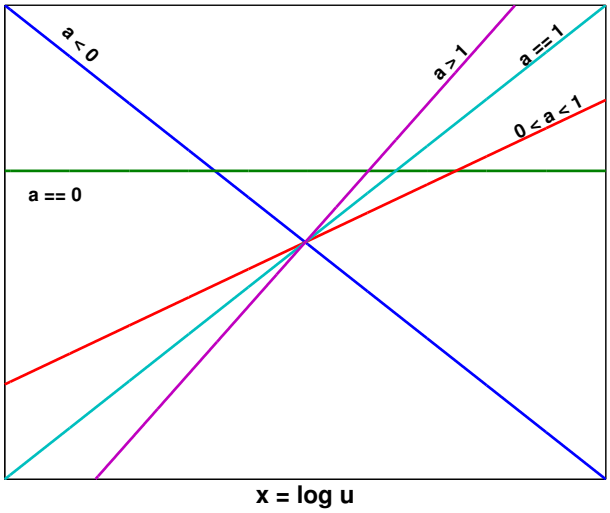
$$\log \sum_{k=1}^{K} c_k \boldsymbol{u}^{\boldsymbol{a}_k} = \log \sum_{k=1}^{K} \exp(\log c_k + \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{x}). \tag{2.9}$$

These transformations are illustrated in Figure 2.1. Taking the logarithm of the objective and every constraint, the GP (2.5) becomes

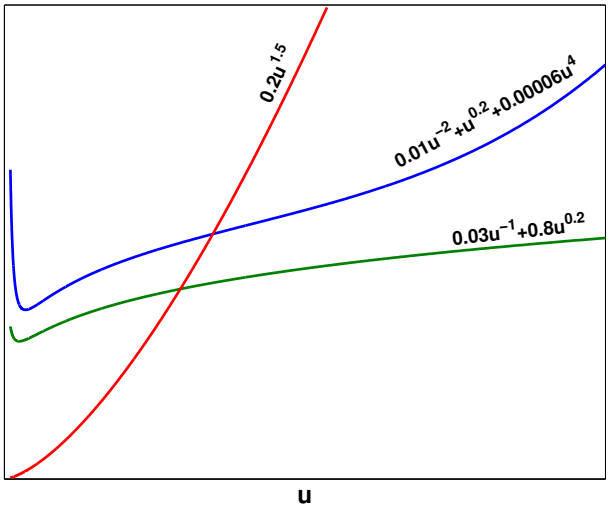
$$\begin{aligned} \text{minimize} \quad & \log \sum_{k=1}^{K_0} \exp(\boldsymbol{a}_{0k}^{\mathrm{T}} \boldsymbol{x} + b_{0k}) \\ \text{subject to} \quad & \log \sum_{k=1}^{K_i} \exp(\boldsymbol{a}_{ik}^{\mathrm{T}} \boldsymbol{x} + b_{ik}) \leq 0, \;\; i = 1, ..., m, \end{aligned} \tag{2.10}$$



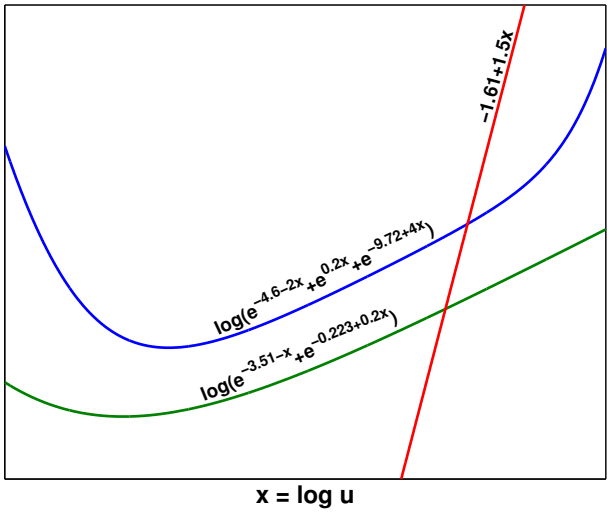
(a) Scalar monomials, cu^a



(b) Corresponding log-space monomials, $\log c + ax$



(c) Scalar posynomials, $\sum_k c_k u^{a_k}$



(d) Corresponding log-space posynomials, $\log \sum_k e^{\log c_k + a_k x}$

Figure 2.1: Qualitative shape of scalar monomial and posynomial functions. Monomials are *log-affine*, whereas posynomials are *log-convex*. These behaviors extend to arbitrarily high-dimensional spaces.

where $b_{ik} \equiv \log c_{ik}$. Log-sum-exp is known to be a convex[2] function class, and convexity is preserved under affine transformations, so (2.10) is a convex optimization problem [11].

Solving GPs

Over the past two decades, technology for solving GPs has become extremely reliable and efficient. At their core, today's state-of-the-art solvers implement *primal-dual interior point methods* [52, 55]. When applied to GPs, these methods provide remarkable capabilities:

Optimality - guaranteed convergence to a *global* optimum (or a certificate of infeasibility, if it is impossible to simultaneously satisfy all the constraints).

Robustness - no need for 'initial guesses' or hand tuning of optimizer hyperparameters.

Speed - approaching that of linear program (LP) solvers. As of 2005, a GP with thousands of decision variables and tens of thousands of constraints could be solved on a desktop computer in minutes [11], with additional gains if the problem is *sparse*.

Strong Duality - simultaneous determination of globally optimal *dual* variables (leveraged extensively in Chapter 3).

This level of effectiveness is a stark contrast from methods for general nonlinear optimization, which typically require initial guesses, often require problem-specific hand-tuning of optimizer parameters, and at best guarantee finding local, not global, optima. Because general nonlinear methods require so many problem-specific inputs and tweaks, many practitioners elect to implement their own optimization code. GP solvers, on the other hand, are robust and general enough for designers to confidently leave the optimization process to standard software packages. Moreover, active research in the applied mathematics community continually improves solution methods, enabling trickle-down benefits not possible with specialized aircraft-specific optimization routines.

[2] A function $f(\boldsymbol{x})$ is convex if the property $f(\theta \boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2) \leq \theta f(\boldsymbol{x}_1) + (1-\theta) f(\boldsymbol{x}_2)$ holds for all $\theta \in [0, 1]$ and $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ in the domain of f.

2.2 A Simple Example

As an initial warm-up to build familiarity with GP formulations, consider the following simple wing design example, adapted from the course materials of the class *Introduction to MDO* at Stanford [4]. A more complex design example will appear in Chapter 6.

The challenge is to size a wing with total area S, span b, and aspect ratio $A = b^2/S$. These parameters should be chosen to minimize the total drag, $D = \frac{1}{2}\rho V^2 C_D S$. The drag coefficient is modeled as the sum of fuselage parasite drag, wing parasite drag, and induced drag,

$$C_D = \frac{(\mathrm{CDA}_0)}{S} + kC_f\frac{S_{\mathrm{wet}}}{S} + \frac{C_L^2}{\pi Ae}, \tag{2.11}$$

where (CDA_0) is the fuselage drag area, k is a form factor that accounts for pressure drag, S_{wet}/S is the wetted area ratio, and e is the Oswald efficiency factor. For a fully turbulent boundary layer, the skin friction coefficient C_f can be approximated as

$$C_f = \frac{0.074}{\mathrm{Re}^{0.2}}, \tag{2.12}$$

where $\mathrm{Re} = \frac{\rho V}{\mu}\sqrt{\frac{S}{A}}$ is the Reynolds number at the mean chord $\bar{c} = \sqrt{S/A}$. The total aircraft weight W is modeled as the sum of a fixed weight W_0 and the wing weight,

$$W = W_0 + W_w. \tag{2.13}$$

The wing weight is modeled as

$$W_w = 45.42S + 8.71 \times 10^{-5}\frac{N_{\mathrm{lift}} b^3\sqrt{W_0 W}}{S\tau}, \tag{2.14}$$

where N_{lift} is the ultimate load factor for structural sizing, and τ is the airfoil thickness to chord ratio. The weight equations are coupled to the drag equations by the constraint that lift equals weight,

$$W = \frac{1}{2}\rho V^2 C_L S. \tag{2.15}$$

Finally, for safe landing, the aircraft should have a sufficiently slow flaps extended stall speed V_{S0}, where

$$\frac{2W}{\rho V_{\mathrm{S0}}^2 S} \leq C_{L,\max}. \tag{2.16}$$

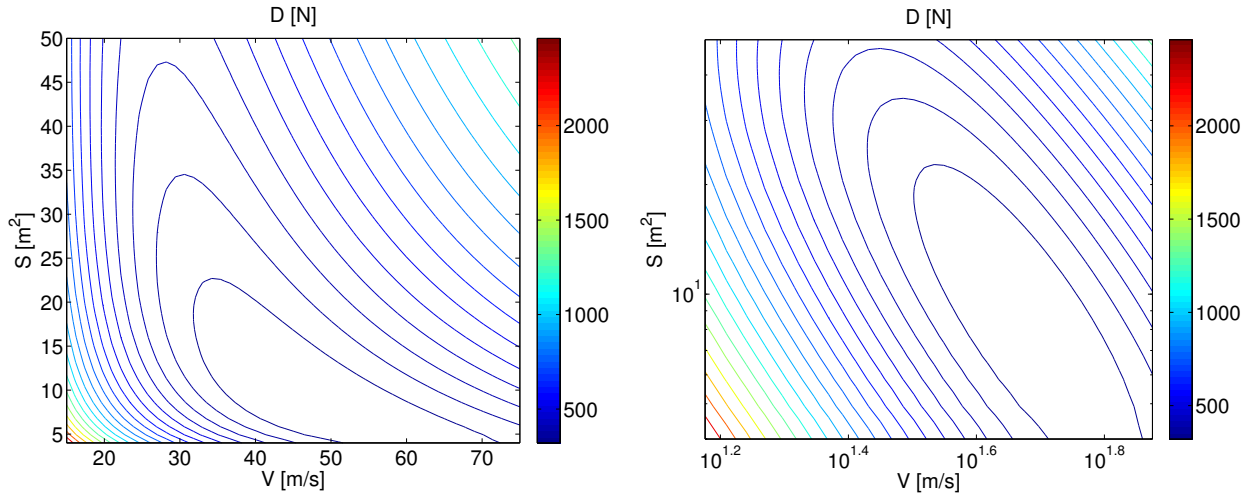
We must choose values of S, A, and V that minimize drag, subject to all the relations in the

Table 2.1: Fixed constants for the simple example problem in Section 2.2.

Quantity	Value	Units	Description
(CDA_0)	0.031	m^2	fuselage drag area
ρ	1.23	$\mathrm{kg/m}^3$	density of air
μ	1.78×10^{-5}	kg/(ms)	viscosity of air
S_{wet}/S	2.05		wetted area ratio
k	1.2		form factor
e	0.95		Oswald efficiency factor
W_0	4940	N	aircraft weight excluding wing
N_{lift}	3.8		ultimate load factor
τ	0.12		airfoil thickness to chord ratio
V_{S0}	22	m/s	flaps extended stall speed
$C_{L,\max}$	1.5		max C_L, flaps down

preceding text. Constant parameters are given in Table 2.1. Note that this problem, while simplistic, exhibits several elements commonly associated with difficult optimization problems. The equations are highly nonlinear. Equation (2.16) is a hard inequality constraint. In Equation (2.14), the wing weight depends on itself through the total aircraft weight W, meaning that iteration would be necessary to bring the weights into agreement.

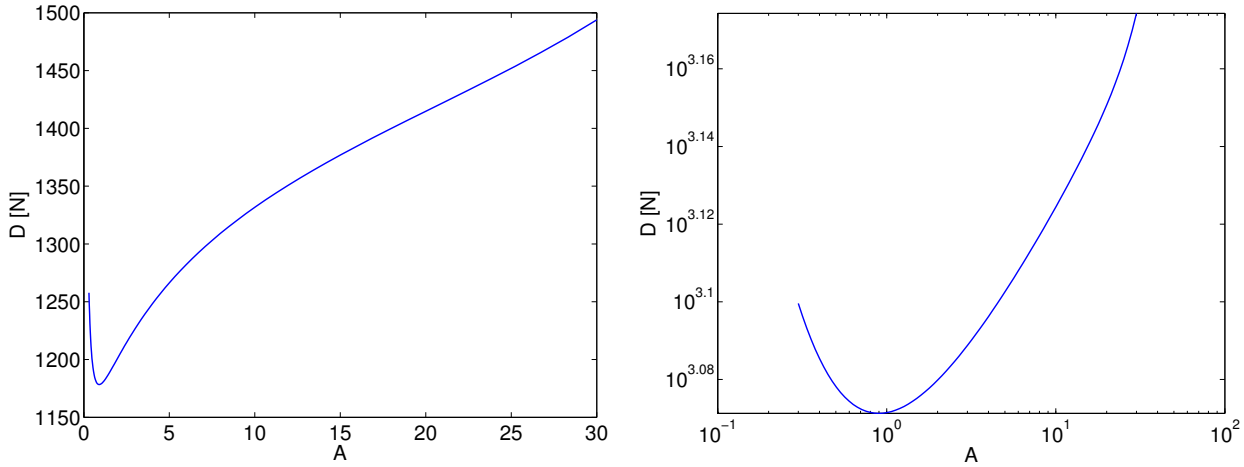
Despite these apparent complexities, it turns out that the *global* optimum can be found reliably on a laptop computer in a few milliseconds. The key lies in recognizing that all the models consist of monomial and posynomial expressions. In fact, the entire optimization problem can be expressed exactly as a GP:



(a) Original parameterization (b) Logarithmic parameterization

Figure 2.2: Slice of the design space for the simple example problem. The contours represent the drag objective as a function of two of the design variables: cruise velocity and wing area. In the original parameterization, the contours are not convex, and are not well-approximated by level sets of a quadratic function. This implies that iterative Newton descent algorithms (SQP, for example) could require many iterations to converge (especially if the starting point is not well-chosen). In contrast, after the log transformation, the level sets are convex and appear to be well-approximated by quadratic level sets (ellipses).

$$
\begin{aligned}
\underset{A,S,C_D,C_L,C_f,\mathrm{Re},W,W_w,V}{\text{minimize}} \quad & \frac{1}{2}\rho V^2 C_D S \\
\text{subject to} \quad & \\
\texttt{1. CD breakdown} \quad & 1 \geq \frac{(\mathrm{CDA}_0)}{C_D S} + \frac{kC_f}{C_D}\frac{S_{\mathrm{wet}}}{S} + \frac{C_L^2}{C_D \pi A e} \\
\texttt{2. Cf definition} \quad & 1 \geq \frac{0.074}{C_f \mathrm{Re}^{0.2}} \\
\texttt{3. Re definition} \quad & 1 \geq \frac{\mu \mathrm{Re}}{\rho V}\sqrt{\frac{A}{S}} \\
\texttt{4. CL definition} \quad & 1 \geq \frac{2W}{\rho V^2 C_L S} \\
\texttt{5. weight breakdown} \quad & 1 \geq \frac{W_0}{W} + \frac{W_w}{W} \\
\texttt{6. wing weight model} \quad & 1 \geq 45.42\frac{S}{W_w} + 8.71 \times 10^{-5}\frac{N_{\mathrm{lift}} A^{3/2}\sqrt{W_0 W S}}{W_w \tau} \\
\texttt{7. stall speed} \quad & 1 \geq \frac{2W}{\rho V_{\mathrm{S0}}^2 S C_{L,\max}}
\end{aligned}
\tag{2.17}
$$



(a) Original parameterization

(b) Logarithmic parameterization

Figure 2.3: Slice of the example problem design space. Drag is plotted as a function of aspect ratio, with $S = 16.45\text{m}^2$ and $V = 120\text{m/s}$ held fixed. In the original parameterization, there clearly exist portions of the design space over which the second-derivative of drag with respect to aspect ratio is negative. Said another way, the objective function is locally concave. As a result, Newton's method cannot compute an accurate second-order approximation. This situation forces solution algorithms to resort to first-order gradient descent, which can be poorly conditioned and require many iterations to converge. This figure shows just one dimension, but the problem is exacerbated in higher dimensions. In contrast, the GP logarithmic parameterization of the same relationships makes the entire design space (including the slice illustrated here) convex, and thus straightforward to globally optimize.

The problem data for this GP can be encoded in a map vector, constant vector $\boldsymbol{c}$, matrix

of fixed variable exponents $\bar{\boldsymbol{A}}$, and matrix of decision variable exponents $\boldsymbol{A}$ as follows:

map	$\boldsymbol{c}$	$\bar{\boldsymbol{A}}$: CDA_0	ρ	μ	$\frac{S_{\mathrm{wet}}}{S}$	k	e	W_0	N_{lift}	τ	V_{S0}	$C_{L_{\max}}$	$\boldsymbol{A}$: A	S	C_D	C_L	Cf	Re	W	W_w	V
0	1/2		1											1	1						2
1	1	1												−1	−1						
1	1				1	1									−1		1				
1	$1/\pi$						−1						−1		−1	2					
2	0.074																−1	−0.2			
3	1		−1	1									1/2	−1/2				1			−1
4	2		−1											−1		−1			1		−2
5	1							1											−1		
5	1																		−1	1	
6	45.42													1						−1	
6	8.71e-5							1/2	1	−1			3/2	1/2					1/2	−1	
7	2		−1								−2	−1		−1					1		

This is the standard parameterization used as input to a solver. For a design engineer wishing to solve this problem, no further algorithmic work is required. A GP solver [54] reliably and quickly finds the solution:

```
VARIABLE   OPTIMAL VALUE   UNITS
A           8.457
S           16.45           m^2
CD          0.02059
CL          0.4987
Cf          0.003599
Re          3.677e+06
W           7344            N
Wwing       2404            N
V           38.16           m/s
```

This design is *feasible*, meaning that it satisfies all the previously defined design constraints. It is also *globally optimal*, meaning that no other set of feasible decision variables could possibly achieve a lower value of the objective, $1/2\rho V^2 C_D S$.

2.3 GP Modeling

In general, the process of formulating a practical problem as a GP (so that it can be solved reliably and efficiently) is called GP modeling [10]. One of the key steps in GP modeling is expressing or approximating physical relationships – the disciplinary models that govern the design space – in terms of monomial and posynomial functions. When this is possible, the formulations described in this section make it straightforward to analyze and optimize a wide range of missions, requirements, and objectives.

Elements of a GP Problem Formulation

Decision Variables

Referring to the definition of a GP (2.5), the *decision variables* are a vector of unknowns $\boldsymbol{u} \in \mathbb{R}^n_{++}$, implicitly constrained to be positive[3]. In the previous example (2.17), the decision variables were $\boldsymbol{u} = (A, S, C_D, C_L, C_f, \mathrm{Re}, W, W_w, V)$. More generally, the decision variables consist of every quantity whose value is to be determined by the optimizer. Once the problem has been solved by a GP solver, each element of $\boldsymbol{u}$ will be assigned an optimal value.

Clearly, the decision variables cannot take on arbitrary values; they must obey physics (or models thereof). These relationships and other limits are quantified by *constraints* on the feasible set of the GP.

Constraints

In the context of GP-based design, constraints serve several purposes. Examples include:

- GP-compatible models of design relations, such as those appearing in Section 5, govern the combinations of design variables that respect physics. In order to consider a particular decision variable in the optimization, the trade-offs governing that variable must be expressed as constraints.

[3]The restriction $\boldsymbol{u} > 0$ is not as limiting as one might initially assume, since in the context of aircraft design, decision variables tend to be physical quantities such as weight, drag, or component sizes.

- Practical or manufacturing limitations, imposed on decision variables such as material gauges, structural stresses, deflections, part sizes for FOD minimization, margins of safety, etc.

- Requirements or system-level performance bounds, imposed by the design team or customer. These constraints are often simple expressions involving a single decision variable, e.g. Range $>$ 5000 km, or $W_{\text{payload}} > 49000$ N.

Objective

What is a 'good' airplane? How do we assign values to two different designs? While this is an interesting research question in and of itself, for the purposes of this thesis we will take a simple yet powerful approach.

In practice, typical design problems involve multiple criteria of interest. Without loss of generality, we can assume that each of these criteria is a decision variable (if it is instead a posynomial expression, we can simply add a decision variable and the corresponding posynomial constraint).

The objective may then be chosen in one of two ways:

- Construct an aggregate objective function (AOF) – a (posynomial) weighted sum of the individual criteria [67]. The weights allow a designer to specify (and refine) his or her relative weightings of the various criteria. To *reward* criteria, such as velocity or efficiency, that term's inverse is penalized[4].

- Choose *one* of the criteria as the (monomial) objective function, and set the other criteria to desired levels using monomial constraints. This approach gives more precise control than the weighted method. However, one must specify a combination of objective values that are *feasible*. The optimizer then sets the remaining monomial objective to the most extreme value possible such that the decision variables all remain feasible.

The role of these formulations in exploring Pareto frontiers (tradeoffs) is discussed in Section 2.4.

[4]Here we use the terms *penalize* and *reward* as opposed to *minimize* and *maximize* to distinguish individual criteria, which are penalized, from the overall objective, which is globally minimized.

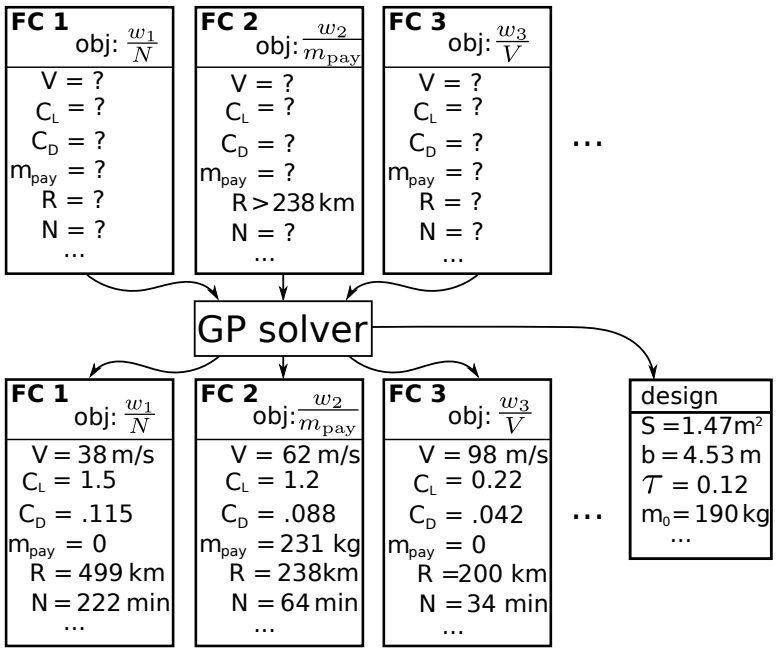


Figure 2.4: Use of flight conditions to define an aircraft design problem. Here the overall objective trades off endurance N, payload m_{pay}, and maximum velocity V. Three flight conditions are defined, and a range constraint $R(2) > 238$ km is placed on the maximum payload flight condition. Onboard fuel is fixed to a constant for simplicity. After optimization, the flight variables that optimize each flight condition objective are found, along with the unique optimal design variable values. Only a subset of the decision variables are shown.

Flight Conditions

The notion of *flight conditions* addresses the problem that designers typically want 'good' performance across a wide range of velocities, payloads, and other *flight variables*. Each flight condition represents a design case the designer wishes to analyze. Variables that may differ from one flight condition to the next, such as velocity or payload, become vectors instead of scalars. In this way, flight conditions make it possible to analyze performance across several design cases within simultaneously (within a single GP optimization). Importantly, design variables that do not change in flight, such as wing area, must remain scalars. An example is shown in Figure 2.4.

Posynomial Inequality Relaxation

Posynomial equality relaxation is a GP modeling technique that is central to the GP design paradigm. The basic idea is to relax selected posynomial equality constraints into inequality constraints, thereby making them GP-compatible. Under certain conditions, an equality relationship will hold at the optimum despite the relaxation [10].

Consider as an example the following drag model, which breaks down C_D into a profile drag component and an induced drag component:

$$C_D = C_{d_0} + \frac{C_L^2}{\pi e A} \tag{2.18}$$

Although the posynomial structure in this model is obvious, the model is *not* GP-compatible, because posynomial *equality* constraints are not allowed in GP. Indeed, adding a posynomial equality constraint can turn an otherwise harmless GP into a very difficult combinatorial optimization problem. However, thanks to our knowledge of the variables involved, we can relax (2.18) to create a GP-compatible inequality constraint:

$$C_D \geq C_{d_0} + \frac{C_L^2}{\pi e A} \tag{2.19}$$

Even though the constraint has been relaxed, the original *equality* relationship (2.18) will be globally optimal (i.e. the relaxation will not change the optimum) under certain conditions on the functional behavior of the objective and constraints with respect to C_D.

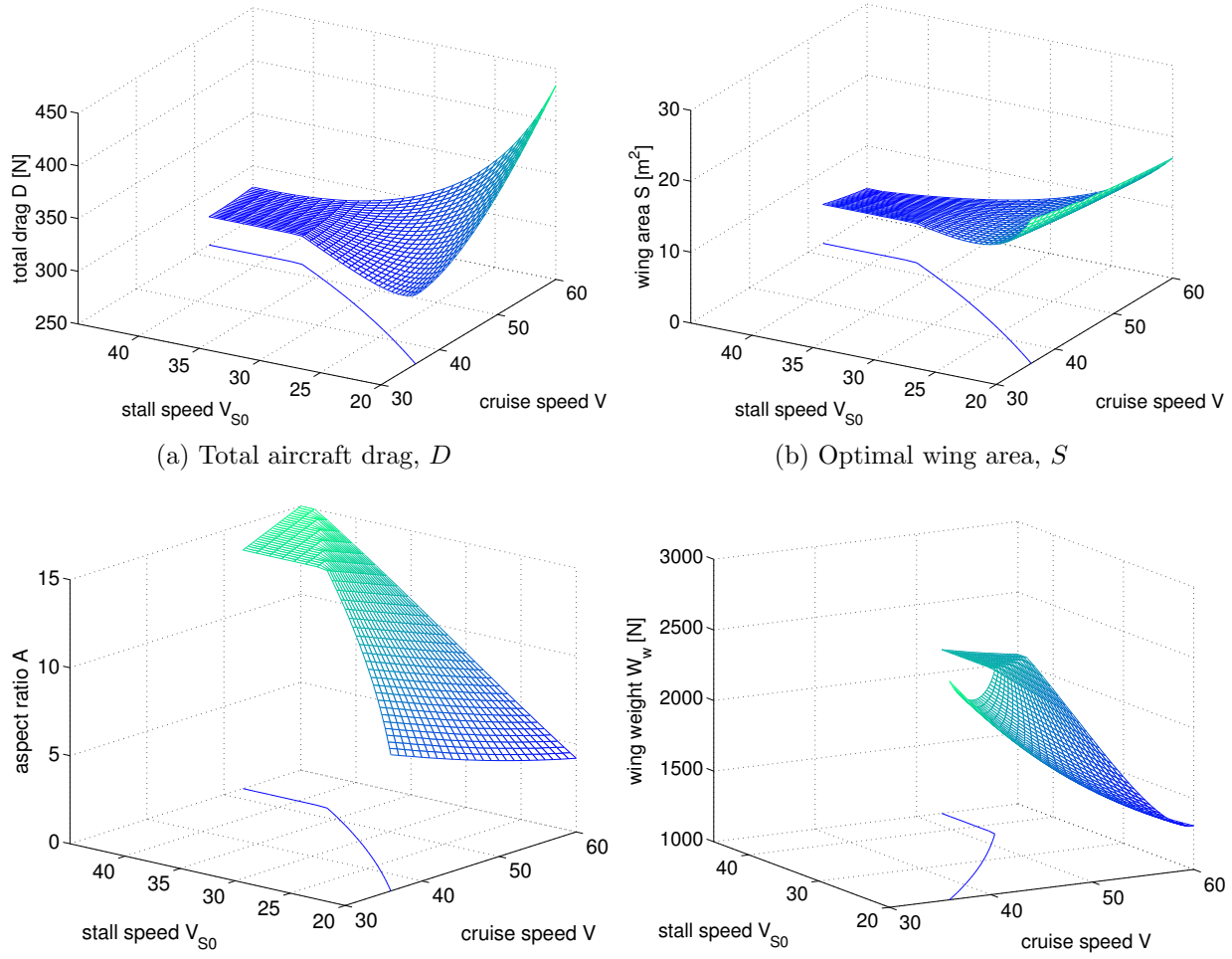
In particular, we assume that C_D does not appear in any monomial equality constraints, and that the objective and inequality constraints (other than (2.19)) are all monotone increasing (or constant) in C_D. Under these conditions, if the equality relation (2.18) did not hold at the optimum, we could clearly decrease C_D until achieving equality, without increasing the objective or moving the solution outside the feasible set.

This type of relaxation is widely applicable, and can also be applied when the direction of the assumed monotonicities are reversed [71, 10]. We will use it extensively in Section 5 without further comment.

2.4 Exploring Tradeoffs

In a design setting, a single point solution is informative, but inadequate. A wise designer or manager considers a range of possible tradeoffs. How would modifying the desired stall speed $V_{S0} = 22\text{m/s}$ affect the drag objective? How expensive would it be to fly at a slightly higher cruise speed V than that which minimizes drag? The answer to these questions lies in a *Pareto frontier*, which quantifies the tradeoffs among the relevant variables.

In Fig. 2.5, we show how GP-based design can be used as a powerful inner loop for quickly exploring Pareto frontiers. For this design example, we re-solved the GP (2.17) across a range of different stall speeds V_{S0}. Then, for each stall speed of interest, we re-solved across a range of different cruise speeds, working up from the drag-optimal V. The resulting tradeoff surface, shown in Fig. 2.5a, represents the design space of aircraft that are Pareto-optimal with respect to drag, cruise speed, and stall speed.



(a) Total aircraft drag, D

(b) Optimal wing area, S

(c) Optimal aspect ratio, A

(d) Optimal wing weight, W_w

Figure 2.5: Tradeoff surfaces for the wing design problem in Section 2.2. Here the GP (2.17) was solved 775 times, across a grid of unique cruise speeds, V, and stall speeds, V_{S0}. This resulted in the Pareto frontier (a), which trades off low cruise drag D, high cruise speed V, and low stall speed V_{S0}. The corresponding optimal design parameters appear in the other figures, where each point on the meshes corresponds to a unique aircraft design. The thin line plotted below each mesh represents the drag-optimal cruise speed as a function of stall speed. On a standard laptop, sweeping out the full Pareto frontier (i.e. solving the GP 775 times) took 3.28 seconds total, or 4.2 milliseconds per solution on average.

Chapter 3

Sensitivity Analysis and the Power of Lagrange Duality

In conceptual aircraft design, the goal is often not just finding an optimal design, but rather understanding the design space – the shape of the Pareto frontier. What would happen if the requirements or specifications were slightly different? What if one or more of the physical models contain errors or uncertainty? A manager who understands how sensitive an optimum is to changes in requirements or specifications is a manager who can better direct engineering effort, better anticipate problems, and better inform decisions. This chapter describes a method for approximating the local shape of tradeoff curves in conceptual aircraft design problems.

One way to investigate these issues is to repeatedly execute the design optimization for many different values of some requirement or model parameter. This *trade study* approach, widely used in practice, results in a Pareto frontier (i.e. a tradeoff curve) for the parameter of interest. Trade studies are useful tools, but a large number of computations (i.e., design optimizations) may be required gain a full understanding of the tradeoff space. For example, the drag vs. cruise speed vs. stall speed Pareto surface swept out in Figure 2.5 consisted of 775 unique GP solutions.

In this chapter, we will consider a more disciplined approach: sensitivity analysis via Lagrange duality. Using this technique, we can solve a GP *once*, and recover partial derivatives of the global optimum with respect to perturbations of each constraint. These *sensitivities*

represent how much the optimum would change if we slightly changed any parameter and re-optimized, but the information can be obtained *without ever re-optimizing*.

Sensitivity analysis for GP is closely related to Lagrange duality. When modern primal-dual interior point methods solve a GP, they determine globally optimal variable values for the original *primal* problem, as well as globally optimal values for the decision variables of a closely related *dual* problem. Importantly, the dual variables are determined for free when a GP is solved. Because they encode sensitivity information, they can help direct engineering or modeling efforts by providing a quantitative comparison among the relative influences of various subsystems, components, or requirements on system-level performance.

3.1 Maximum Entropy Dual of a Geometric Program

A Lagrange dual of the convex form GP (2.10) can be formed by introducing t equality-constrained variables z to represent the affine mappings:

$$\begin{aligned} \text{minimize} \quad & \log \sum_{k=1}^{K_0} \exp z_{0k} \\ \text{subject to} \quad & \log \sum_{k=1}^{K_i} \exp z_{ik} \leq 0, \;\; i = 1, ..., m, \\ & \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i = \boldsymbol{z}_i, \qquad i = 0, ..., m, \end{aligned} \tag{3.1}$$

where $\boldsymbol{A}_i$ and $(\boldsymbol{b}_i \equiv \log \boldsymbol{c}_i)$ contain the exponents and constant coefficients for each of the $m+1$ posynomials. Introducing Lagrange multiplier vectors $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $\boldsymbol{\nu}_i \in \mathbb{R}^{K_i}$, the constraints can be incorporated into the objective to form the Lagrangian:

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \log \sum_{k=1}^{K_0} \exp z_{0k} + \sum_{i=1}^{m} \lambda_i \log \sum_{k=1}^{K_i} \exp z_{ik} + \sum_{i=0}^{m} \boldsymbol{\nu}_i^{\mathrm{T}} (\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i - \boldsymbol{z}_i) \tag{3.2}$$

The dual function is

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}, \boldsymbol{z}} L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{3.3}$$

Taking the infimum over $\boldsymbol{x}$, we see that g tends to $-\infty$ unless $\sum_{i=0}^{m} \boldsymbol{\nu}_i^{\mathrm{T}} \boldsymbol{A}_i = 0$, in which case the terms involving $\boldsymbol{x}$ vanish. Taking the infimum over $\boldsymbol{z}_0$, the infimum is achieved for $\boldsymbol{z}_0$

such that

$$\boldsymbol{\nu}_0 = \frac{\exp \boldsymbol{z}_0}{\sum_{k=1}^{K_0} \exp z_{0k}}. \tag{3.4}$$

This relationship is only possible when $\boldsymbol{\nu}_0$ represents a probability distribution, i.e. $\boldsymbol{\nu}_0 \geq 0$ and $\mathbf{1}^{\mathrm{T}}\boldsymbol{\nu}_0 = 1$. For any $\boldsymbol{\nu}_0$ not satisfying these constraints, g achieves $-\infty$. Finally, taking the infimum over $\boldsymbol{z}_i$, we obtain

$$\boldsymbol{\nu}_i = \frac{\lambda_i \exp \boldsymbol{z}_i}{\sum_{k=1}^{K_i} \exp z_{ik}}, \quad i = 1, ..., m. \tag{3.5}$$

Assuming $\lambda_i \geq 0$, (3.5) is only possible when $\boldsymbol{\nu}_i \geq 0$ and $\mathbf{1}^{\mathrm{T}}\boldsymbol{\nu}_i = \lambda_i$. Otherwise, g achieves $-\infty$.

To form a dual problem, we substitute (3.4) and (3.5) into (3.2), maximize over $\boldsymbol{\nu}$ and $\boldsymbol{\lambda} \geq 0$, make the domain constraints explicit, and eliminate $\boldsymbol{\lambda}$. The resulting dual problem is

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=0}^{m} \left[\boldsymbol{\nu}_i^{\mathrm{T}} \boldsymbol{b}_i - \sum_{k=1}^{K_i} \nu_{ik} \log \frac{\nu_{ik}}{\mathbf{1}^{\mathrm{T}}\boldsymbol{\nu}_i} \right] \\
\text{subject to} \quad & \sum_{i=0}^{m} \boldsymbol{\nu}_i^{\mathrm{T}} \boldsymbol{A}_i = 0 \\
& \boldsymbol{\nu}_i \geq 0, \quad i = 0, ..., m \\
& \mathbf{1}^{\mathrm{T}}\boldsymbol{\nu}_0 = 1.
\end{aligned}$$

Obtaining Dual Variables from Off-The-Shelf Solvers

Most modern GP solvers utilize *primal-dual interior point methods* [52, 55] in their solution architecture. A common feature of these methods is that they determine the optimal values of the primal and dual variables at the same time (so long as both problems are feasible). Due to the convexity of GP and a property called strong duality, the optimal values of the primal and dual problems are *exactly equal* (up to numerical precision).

Off-the-shelf solvers may differ as to exactly which primal and dual variables they return. The commercial solver MOSEK [54], for example, returns the optimal log-transformed primal variables $\boldsymbol{x}$, and the opposites of the dual Lagrange multipliers, $-\lambda_i$, $i = 1, ..., m$. It does not return the dual distributions $\boldsymbol{\nu}_i$, $i = 0, ..., m$. When these variables are needed (they

are used in Section 3.3, for example), they are easily calculated given $\boldsymbol{\lambda}$ and ($\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$) via (3.4) and (3.5).

3.2 Sensitivity Analysis

Perturbed GP – Tradeoff Analysis

Consider the following perturbed version of a GP in standard form:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K_0} c_{0k} \boldsymbol{u}^{\boldsymbol{a}_{0k}} \\ \text{subject to} \quad & \sum_{k=1}^{K_i} c_{ik} \boldsymbol{u}^{\boldsymbol{a}_{ik}} \leq s_i, \;\; i = 1, ..., m. \end{aligned} \tag{3.6}$$

The variables $\boldsymbol{s} \in \mathbb{R}^m$ are perturbations: $\boldsymbol{s} = \boldsymbol{1}$ represents the original unperturbed problem; if $s_i < 1$, then the i$^{\text{th}}$ constraint has been tightened; if $s_i > 1$, then the i$^{\text{th}}$ constraint has been loosened. We denote the optimal objective value of the perturbed problem $p^*(\boldsymbol{s})$. If we sweep one element s_i over a range of values, then we obtain an optimal tradeoff curve, i.e. a Pareto frontier. If we sweep multiple elements of $\boldsymbol{s}$ over a convex set, then we obtain an optimal tradeoff surface. This formulation is the most straightforward way to explore a design space parameterized by multiple objectives. For example, one might minimize some objective (mission fuel burn, say), and explore perturbations in constraints on payload, range, and material cost.

Sensitivity Analysis via Optimal Dual Variables

When a GP is solved, the sensitivity of the objective function with respect to each constraint perturbation is encoded by the optimal dual variables:

$$\left.\frac{\partial \log p^*(\boldsymbol{s})}{\partial \log s_i}\right|_{\boldsymbol{s}=\boldsymbol{1}} = \left.\frac{\partial \left(\frac{p^*(\boldsymbol{s})}{p^*(\boldsymbol{1})}\right)}{\partial \left(\frac{s_i}{1}\right)}\right|_{\boldsymbol{s}=\boldsymbol{1}} = -\lambda_i, \;\; i = 1, ..., m. \tag{3.7}$$

That is, the *percentage* or *fractional* sensitivity of the objective to fractional changes in the i$^{\text{th}}$ constraint is encoded by the i$^{\text{th}}$ Lagrange multiplier. For example, imagine that $\lambda_i = 4.00$

Table 3.1: Optimal dual variables for the simple example (2.17).

i	Constraint Name	λ_i	$\boldsymbol{\nu}_i^{\mathrm{T}}$		
1	`CD breakdown`	1.000	[0.0915	0.4300	0.4785]
2	`Cf definition`	0.4300	[0.4300]		
3	`Re definition`	0.0860	[0.0860]		
4	`CL definition`	0.9570	[0.9570]		
5	`weight breakdown`	1.2867	[0.8655	0.4212]	
6	`wing weight model`	0.4212	[0.1309	0.2903]	
7	`landing stall speed`	0.1845	[0.1845]		

at the unperturbed optimum of some GP. If we were to tighten constraint i by 1% and re-optimize, we would expect the optimal objective value to increase by approximately 4%. Similarly, if we were to loosen constraint i by 1%, we would expect the optimal objective value to decrease by approximately 4%. Because they encode *percentage* sensitivities, optimal dual variables are informative regardless of the units of measurement used in the primal problem. The entire vector of dual sensitivities, $-\boldsymbol{\lambda}$, parameterizes a log-space linearization of the Pareto surface $p^*(\boldsymbol{s})$ around the unperturbed point $p_0 \equiv p^*(\mathbf{1})$:

$$\log p^*(\boldsymbol{s}) \approx \log p_0 - \boldsymbol{\lambda}^{\mathrm{T}}(\log \boldsymbol{s}) \tag{3.8}$$

$$p^*(\boldsymbol{s}) \approx p_0 \boldsymbol{s}^{-\boldsymbol{\lambda}}. \tag{3.9}$$

This approximation is always optimistic, i.e. $p_0 \boldsymbol{s}^{-\boldsymbol{\lambda}} \leq p^*(\boldsymbol{s})$.

Example: Dual Variables from Simple Example

Recall the simple example problem (2.17) from Chapter 2. Consider as an example the posynomial wing weight model. A perturbed version is

$$s_{ww} \geq \frac{45.42S}{W_w} + 8.71 \times 10^{-5} \frac{N_{\mathrm{lift}} A^{3/2} \sqrt{W_0 W S}}{W_w \tau}. \tag{3.10}$$

Compared with the unperturbed model, the perturbed version predicts smaller wing weights when $s_{ww} > 1$, and larger wing weights when $s_{ww} < 1$. Percentage change in wing weight and s_{ww} are directly related: percentage change equals $100(1/s_{ww} - 1)$. For example, $s_{ww} = 1/1.01$

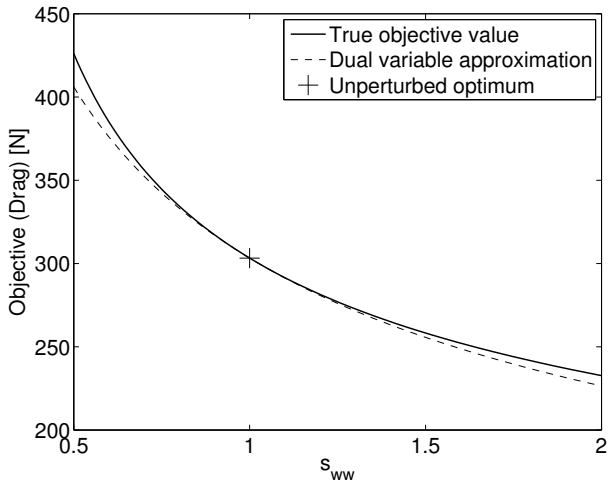


Figure 3.1: Optimal drag as a function of perturbation in wing weight model, with s_{ww} defined as in (3.10). The dual variable approximation is formed by solving the (unperturbed) GP once, and using the wing weight model's dual variable to construct a monomial (dashed line). The solid true objective value curve is produced by re-solving the GP for different model perturbations (in this case, 61 times). The dual variable approximation represents a computationally inexpensive (free, in fact) surrogate. It is always an optimistic approximation, whose error approaches zero for small perturbations.

will cause the model to predict 1% higher wing weights. As s_{ww} shifts, so does the optimal design, resulting in a new *perturbed optimum*.

In the process of solving the *unperturbed* problem, the globally optimal dual variables (listed in Table 3.1) are also determined. These values encode information about the *perturbed* optimum. For example, how would a 10% change in the wing weight model affect the drag objective? According to (3.7), an estimate can be obtained from the corresponding dual variable, $\lambda_{ww} = 0.4212$. This value indicates that increasing (decreasing) the modeled wing weight by 10% and re-optimizing would result in approximately 4.212% more (less) drag. This dual variable approximation can be written as a monomial,

$$p^*(s_{ww}) \approx p^*(1)\, s_{ww}^{-\lambda_{ww}}, \tag{3.11}$$

shown as a dashed line in Figure 3.1. The log-space slope of the tradeoff curve, $-\lambda_{ww}$, directly encodes how sensitive the objective is to small changes in wing weight.

3.3 Fixed Variable Sensitivities

Consider a GP with fixed variables in the form (2.6). To solve this GP, one simply computes the constant terms ($c_{ik}\bar{\boldsymbol{u}}^{\bar{\boldsymbol{a}}_{ik}}$) and treats them as constant values $\boldsymbol{c}$ for a GP in the form (2.5). However, it is often useful to understand how sensitive a solution is to the values of variables that have been fixed to constant values.

The sensitivity of the objective with respect to the j$^{\text{th}}$ fixed variable is

$$\frac{\partial \log p^*}{\partial \log \bar{u}_j} = \frac{\partial \log f_0}{\partial \log \bar{u}_j} + \sum_{i=1}^{m} \frac{\partial \log p^*}{\partial \log u_i} \cdot \frac{\partial \log u_i}{\partial \log \bar{u}_j} \tag{3.12}$$

$$= \sum_{i=0}^{m} \lambda_i \frac{\sum_{k=1}^{K_i} \bar{\boldsymbol{a}}_{ik}^{(j)} \exp z_{ik}}{\sum_{k=1}^{K_i} \exp z_{ik}} \tag{3.13}$$

$$= \sum_{i=0}^{m} \boldsymbol{\nu}_i^{\mathrm{T}} \bar{\boldsymbol{a}}_i^{(j)}. \tag{3.14}$$

That is, the sensitivity of the objective with respect to the j$^{\text{th}}$ fixed variable is encoded by the dot product of dual variables and j$^{\text{th}}$-variable-exponents, summed over all posynomial constraints involving variable j. Thus, when a problem has GP structure but certain variables are to be held constant, we can solve a smaller GP involving only the free decision variables, and then recover the sensitivity with respect to each fixed variable via (3.14).

Example: Fixed Variable Sensitivities from Simple Example

Sensitivities with respect to each fixed variable are listed in Table 3.2. These sensitivities provide actionable intelligence that can redirect modeling and engineering efforts. For example, even after solving only the unperturbed GP, the designer learns that small changes in the fixed weight ($\mathcal{S}_{W_w} = 1.0107$) have 2.35 times more effect on optimal drag than small changes in the form factor k ($\mathcal{S}_k = 0.43$), which in turn have 4.70 times more effect than small changes in fuselage drag area ($\mathcal{S}_{\mathrm{CDA}_0} = 0.0915$). One way to use this information could be to direct engineering effort toward improving the quantities the objective is most sensitive to – W_0, in this case. Or, the information could guide improvements in model fidelity. In this case, an analyst might prioritize refining models for W_0, e, or k, given their relatively high sensitivity to uncertainty, while concluding that a constant model for CDA_0 is reasonable.

Table 3.2: Fixed variable sensitivities for the example problem (2.17). Negative sensitivities indicate that increasing the corresponding variable would improve the objective.

Variable	Value ($\bar{u}$)	$\partial \log p^* / \partial \log \bar{u}$
W_0	4940	1.0107
e	0.95	-0.4785
S_{wet}/S	2.05	0.4300
k	1.2	0.4300
V_{S0}	22	-0.3691
N_{lift}	3.8	0.2903
τ	0.12	-0.2903
ρ	1.23	-0.2275
$C_{L,\text{max}}$	1.5	-0.1845
(CDA_0)	0.031	0.0915
μ	1.78e-5	0.0860

Aside from parameterizing models, fixed variables often parameterize design requirements or specifications. For example, the fixed variable $V_{\text{S0}} = 22\text{m/s}$ reflects a desire for the aircraft to be capable of landing at a safe (slow) speed, independent of its cruise speed V. The associated sensitivity, $\mathcal{S} = -0.3691$, tells us that a 1% increase in stall speed would pay off as approximately a -0.37% decrease in drag. The true Pareto frontier, along with its dual variable approximation, are depicted in Figure 3.2.

3.4 Aside: Recovering known Scaling Laws

As an interesting aside, let us examine an even simpler GP for which we can draw a connection between the dual variables and known scaling laws in aircraft design.

Consider the following GP, which is a simplified version of (2.17) that ignores the effects

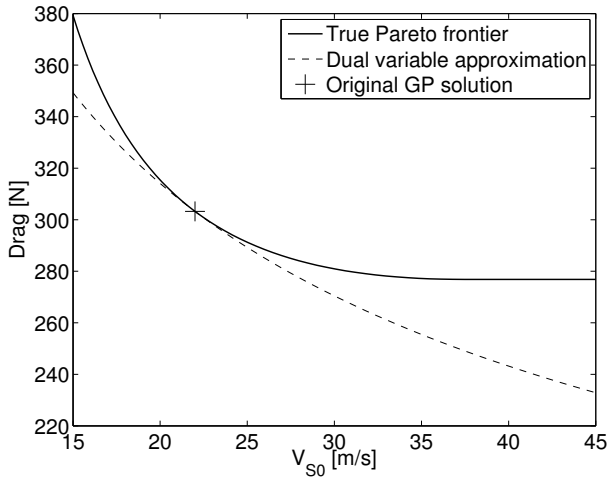


Figure 3.2: Optimal tradeoff curve between drag and stall speed requirement. The dual variable approximation matches exactly for small changes in stall speed. On the far right side of the true Pareto frontier, we observe that the stall speed constraint becomes inactive (ceases to influence the objective) if stall speeds are permitted to be sufficiently large. The dual variable approximation is not particularly close on the extreme left and right ends of the plotted range, but these correspond to -32% and +105% changes in stall speed respectively – well outside the applicable range of "small changes".

of Reynolds number on profile drag.

$$
\begin{aligned}
\underset{A,S,C_D,C_L,W,W_w,V}{\text{minimize}} \quad & f_0(\rho, V, C_D, S) \\
\text{subject to} \quad & \\
\texttt{CD breakdown} \quad & 1 \geq \frac{(\text{CDA}_0)}{C_D S} + \frac{C_{Dp}}{C_D} + \frac{C_L^2}{C_D \pi A e} \\
\texttt{CL definition} \quad & 1 \geq \frac{2W}{\rho V^2 C_L S} \\
\texttt{weight breakdown} \quad & 1 \geq \frac{W_0}{W} + \frac{W_w}{W} \\
\texttt{wing weight model} \quad & 1 \geq \frac{45.42 S}{W_w} + 8.71 \times 10^{-5} \frac{N_{\text{lift}} A^{3/2} \sqrt{W_0 W S}}{W_w \tau} \\
\texttt{stall speed} \quad & 1 \geq \frac{2W}{\rho V_{\text{S0}}^2 S C_{L,\max}}
\end{aligned}
\tag{3.15}
$$

We have introduced the fixed variable $C_{Dp} = 0.0095$ to represent a constant wing profile drag coefficient. We will study this GP for two different objectives: $f_0 = \frac{1}{2}\rho V^2 C_D S$ (drag

Table 3.3: Globally optimal solutions (minimum drag and minimum power) for the simplified GP (3.15).

Variable	$f_0 = \frac{1}{2}\rho V^2 C_D S$	$f_0 = \frac{1}{2}\rho V^3 C_D S$	Units
A	8.792	8.572	
S	16.79	30.55	m
C_D	0.02269	0.04206	
C_L	0.5456	0.8983	
W	7495	8859	N
W_w	2555	3919	N
V	36.48	22.91	m/s

Table 3.4: Optimal dual variables for the drag minimization problem (3.15).

i	Constraint Name	$\boldsymbol{\nu}_i^{\mathrm{T}}$ – minimum drag	$\boldsymbol{\nu}_i^{\mathrm{T}}$ – minimum power
1	`CD breakdown`	[0.0814 0.4186 0.5000]	[0.0241 0.2259 0.7500]
2	`CL definition`	[1.0000]	[1.5000]
3	`weight breakdown`	[0.9186 0.4751]	[0.9759 0.7741]
4	`wing weight model`	[0.1418 0.3333]	[0.2741 0.5000]
5	`landing stall speed`	[0.2271]	[0.0000]

minimization), and $f_0 = \frac{1}{2}\rho V^3 C_D S$ (flight power minimization).

As before, a GP solver finds the globally optimal solutions (listed in Table 3.3) and corresponding dual variables (listed in Table 3.4) in milliseconds.

Several interesting observations can be made.

1. As one would expect, the minimum power design has shifted down the drag-velocity curve. It has a larger wing area, and correspondingly larger wing weight, which enable optimal operation at a significantly lower velocity.

2. In the minimum drag design, the aircraft is flown at a C_L such that induced drag accounts for exactly half of the overall drag ($\nu_{13} = 0.5$). This is a well known result [43] (valid for models that break down C_D into a velocity-independent profile drag term plus a C_L^2-dependent induced drag term).

3. In the minimum power design, induced drag constitutes 75% of the overall drag ($\nu_{13} =$

0.75). This is also a well-known result [21].

4. According to the dual variables, minimum power is 150% sensitive to C_L (the famous 3/2 power scaling law [62]).

5. Finally, we notice that for minimum power design, the stall speed constraint has become inactive – a result of the optimal cruise speed decreasing to a value near the required stall speed. Indeed, the stall speed for the power-optimal design is 17.73 m/s, well under the required 22 m/s.

3.5 Linearized Propagation of Log-Normal Uncertainties

Sensitivity information can help to predict how *small* uncertainties in parameters propagate to the objective. In particular, imagine that our knowledge about one of the fixed variables, $\bar{u}_j$, is described by a log normal distribution:

$$\log \bar{u}_j \sim \mathcal{N}(\mu_j, \sigma_j^2), \tag{3.16}$$

where σ_j is small ($\sigma_j < 0.05$, corresponding to a standard deviation less than 5% error, say). Additionally, assume that we have solved a GP involving $\bar{u}_j$ for the unperturbed setting $\bar{u}_j = e^{\mu_j}$, and recovered the dual sensitivity

$$\mathcal{S}_{\bar{u}_j} \equiv \frac{\partial \log p^*}{\partial \log \bar{u}_j} = \sum_{i=0}^{m} \boldsymbol{\nu}_i^{\mathrm{T}} \bar{\boldsymbol{a}}_i^{(j)},$$

along with the unperturbed objective value p_0^*. In keeping with (3.9), the sensitivity allows us to predict how the objective would change if $\bar{u}_j$ were slightly different:

$$\log p^*(\bar{u}_j) \approx \log p_0^* + \mathcal{S}_{\bar{u}_j}(\log \bar{u}_j - \mu_j).$$

Because normal distributions are closed under affine transformations, this suggests a log normal approximation for our belief about the optimal p^*:

$$\log p^* \sim \mathcal{N}(\log p_0^*, \mathcal{S}_{\bar{u}_j}^2 \sigma_j^2). \tag{3.17}$$

It is very important to understand that this result relies on the log-space linearization of the $\bar{u}_j, p^*$ relationship, and is thus reliable only for small σ_j. Moreover, due to the convexity of the functions involved, this approximation always *underestimates* our uncertainty in p^*. It is therefore not a reliable tool for propagating conservative error bounds or constructing worst-case estimates. Nevertheless, it is extremely useful for making rough comparisons regarding how much uncertainty is introduced by various parameter uncertainties.

Chapter 4

Fitting GP Models to Data

In many engineering systems and disciplines, some of the governing models are encoded in simulations or computational routines, and are therefore not available in analytical forms. This opens a natural question: when can we fit GP-compatible models to data sets that we might collect from black-box computational routines or even from experiments?

As discussed in Chapter 2, monomials and posynomials are *log-convex*. Interestingly, the work presented in this chapter led not just to a new approach for fitting GP-compatible models, but to a new method for fitting convex functions in general. As a result, the right place to begin the discussion is with the general problem of *convex regression*.

4.1 The Convex Regression Problem

In convex regression, we consider the problem of approximating an arbitrary function or data with a *convex* surrogate function. In particular, consider the problem of fitting a multivariate function $f(\boldsymbol{x})$ to a set of m data points

$$(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}, \;\; i = 1 \ldots m.$$

With the restriction that f be convex, we obtain the fitting problem

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & ||Y - f(X)|| \\ \text{subject to} \quad & f \in \mathcal{F}, \end{aligned} \tag{4.1}$$

where $X \in \mathbb{R}^{m \times d}$ is a matrix of the $\boldsymbol{x}_i$ (measurements), $Y \in \mathbb{R}^m$ is a vector of the y_i (responses), f is the function being fitted[1], $\mathcal{F}$ is a set of convex functions, and $||\cdot||$ is some norm. This chapter takes $||\cdot||$ to be the 2-norm, but the work extends to other fitting criteria. Clearly, the norm (residual) cannot be small if the data is not well-approximated by any convex function. Our interest is therefore in data sets that exhibit at least a reasonable amount of underlying convexity .

As with any regression problem, the specific choice of function class $\mathcal{F}$ can dramatically influence the quality of the resulting model with respect to the chosen loss function. One common choice is $\mathcal{F}_{\text{MA}}^K$, the set of *max-affine* functions parameterized by K affine terms:

$$f(\boldsymbol{x}) = \max_{k=1\ldots K} [b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}] \tag{4.2}$$

This choice is motivated by the fact that any convex function can be expressed as the pointwise supremum over a (generally infinite) set of affine functions [11]. That is, $\mathcal{F}_{\text{MA}}^\infty$ can approximate *any* convex function to arbitrary precision. Max-affine functions are also appealing from a practical perspective: they can be converted directly to linear constraint sets (compatible with linear programming), or to monomial constraint sets (compatible with geometric programming).

Methods for fitting max-affine functions are well established. In 2008, Magnani and Boyd [46] proposed a least-squares partition algorithm that works well in practice. They also discussed a more general bi-affine function parameterization. In 2010, Kim et al. [41] used a similar method to fit max-monomial models for circuit design. Hannah and Dunson [31] fit max-affine functions using a statistically consistent adaptive partitioning approach that refines accuracy for increasing K by generating candidate partition splits and selecting splits greedily. They also describe an ensemble version of their method that avoids instability in max-monomial fitting [30].

Although max-affine functions are a natural and popular choice, one drawback is that a large number of affine terms K may be necessary to achieve a desired level of accuracy. In this chapter, we argue for choosing f from a more general class of convex functions, of which $\mathcal{F}_{\text{MA}}^K$ is a subset. Section 4.2 describes two successive generalizations: the *softmax-affine*

[1]The mapping f is overloaded: $f(\boldsymbol{x})$ refers to function evaluation at a single point ($f : \mathbb{R}^d \to \mathbb{R}$), whereas $f(X)$ refers to a vectorized version ($f : \mathbb{R}^{m \times d} \to \mathbb{R}^m$).

class, $\mathcal{F}^K_{\text{SMA}}$, and the *implicit softmax-affine* class, $\mathcal{F}^K_{\text{ISMA}}$. One can show that

$$\mathcal{F}^K_{\text{MA}} \subsetneq \mathcal{F}^K_{\text{SMA}} \subsetneq \mathcal{F}^K_{\text{ISMA}}.$$

This hierarchy implies that for a given data set (X, Y) and number of affine terms K, there always exist SMA and ISMA functions with at least as small a residual as the best max-affine fit. Indeed, significantly improved fits are observed on real data sets (Section 4.5).

The fitting problem (4.1) is formulated as a nonlinear least squares regression, solved locally via sequential convex programming (Section 4.4).

4.2 Some Convex Function Classes

We now explore the general task of fitting some data with a convex function $f \in \mathcal{F}$. This section overviews several candidate function classes $\mathcal{F}$.

Max-affine Functions

The max-affine function class $\mathcal{F}^K_{\text{MA}}$ consists of functions of the form

$$f_{\text{MA}}(\boldsymbol{x}) = \max_{k=1\ldots K} \left[b_k + \boldsymbol{a}_k^{\text{T}}\boldsymbol{x}\right], \tag{4.3}$$

where $b_k \in \mathbb{R}$ and $\boldsymbol{a}_k \in \mathbb{R}^n$ are the model parameters. The total number of model parameters is $n_p = K(n+1)$. It is well known that the supremum over a set of affine functions is a convex function; thus (4.3) is always convex in $\boldsymbol{x}$.

Softmax-affine Functions

The proposed softmax-affine function class $\mathcal{F}^K_{\text{SMA}}$ consists of functions of the form

$$f_{\text{SMA}}(\boldsymbol{x}) = \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp\left(\alpha(b_k + \boldsymbol{a}_k^{\text{T}}\boldsymbol{x})\right). \tag{4.4}$$

The addition of the scalar parameter $\alpha \in \mathbb{R}_{>0} \cup \{+\infty\}$ brings the total number of parameters to $n_p = K(n+1)+1$. Since log-sum-exp functions are convex and convexity is preserved under positive scaling and affine transformations [11], SMA functions are guaranteed to be convex in $\boldsymbol{x}$ for any $\alpha > 0$.

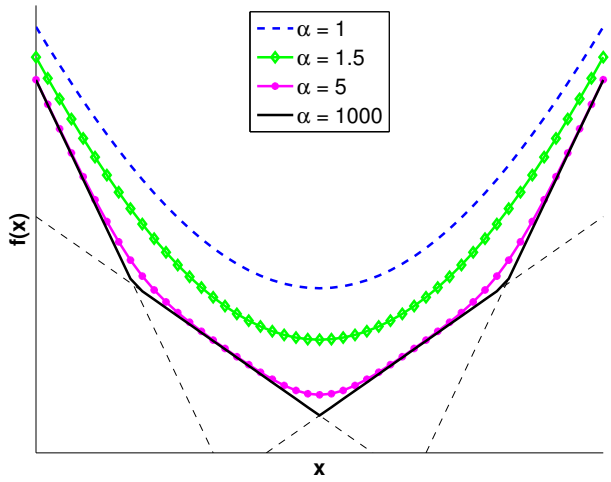


Figure 4.1: Influence of α on softmax-affine functions. Each of the softmax-affine functions plotted above shares the same $K = 4$ affine terms (the thin dashed lines), but has a different α. The solid curve corresponds to a max-affine function; the dashed curve corresponds to a softmax-affine function with $\alpha = 1$, and one can interpolate smoothly among these by varying α. While this figure illustrates the situation in $\mathbb{R}^1$, the limiting cases extend to arbitrary dimensions.

Limiting behavior As depicted in Figure 4.1, one can view α as a smoothing parameter that controls the sharpness of the softmax over the K affine planes. In the limit of infinite sharpness,

$$\lim_{\alpha \to +\infty} \frac{1}{\alpha} \log \sum_{k=1}^{K} \exp\left(\alpha(b_k + \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{x})\right) = \max_{k=1 \ldots K} \left[b_k + \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{x}\right], \tag{4.5}$$

i.e. softmax-affine functions become max-affine functions in the limit $\alpha \to +\infty$. Thus, $\mathcal{F}_{\mathrm{MA}}^{K} \subsetneq \mathcal{F}_{\mathrm{SMA}}^{K}$.

Implicit Softmax-affine Functions

The proposed implicit softmax-affine class, $\mathcal{F}_{\mathrm{ISMA}}^{K}$, expresses the relationship between $\boldsymbol{x}$ and y via an *implicit* function

$$\tilde{f}_{\mathrm{ISMA}}(\boldsymbol{x}, y) = \log \sum_{k=1}^{K} \exp\left(\alpha_k(b_k + \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{x} - y)\right), \tag{4.6}$$

where each $\alpha_k \in \mathbb{R}_{>0} \cup \{+\infty\}$.

Proposition 4.2.1. *For all $\boldsymbol{x}$, there exists a unique y such that $\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, y) = 0$.*

Proof. For all $\boldsymbol{x}$, $y \mapsto \tilde{f}_{\text{ISMA}}(\boldsymbol{x}, y)$ is a continuous monotone strictly decreasing function of y, since increasing y decreases every term in the K-term sum. Moreover, there exists some γ^- such that $\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, \gamma^-) > 0$, and some γ^+ such that $\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, \gamma^+) < 0$. Thus by the intermediate value theorem the function must have a unique zero crossing between γ^- and γ^+. □

Based on Proposition 4.2.1, we define f_{ISMA} such that for all $\boldsymbol{x}$,

$$\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, f_{\text{ISMA}}(\boldsymbol{x})) = 0. \tag{4.7}$$

That is, the predicted value $\hat{y} = f_{\text{ISMA}}(\boldsymbol{x})$ is the unique value $\hat{y}$ such that $\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, \hat{y}) = 0$.

Proposition 4.2.2. *The function $\boldsymbol{x} \mapsto f_{\text{ISMA}}(\boldsymbol{x})$ is convex.*

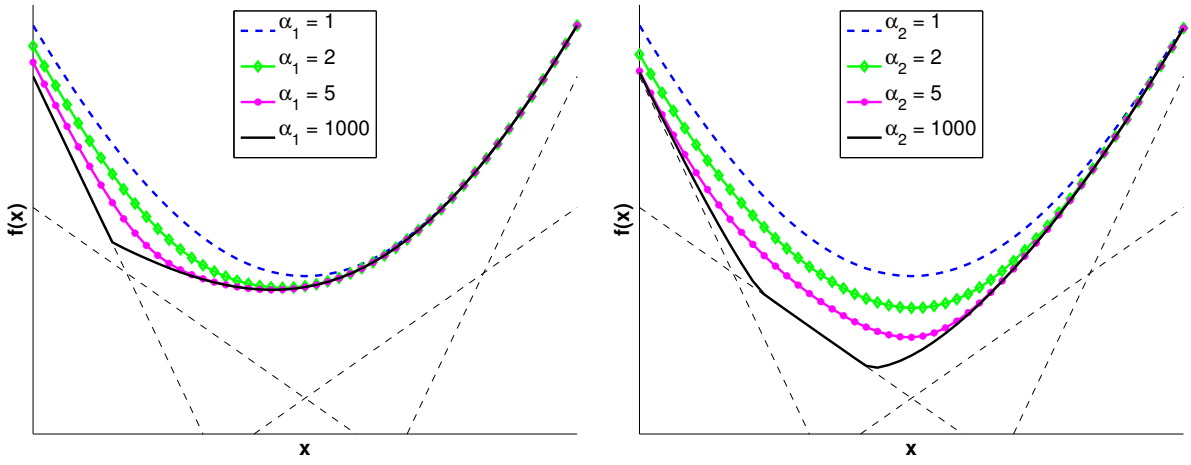
Proof. Consider any two points $(\boldsymbol{x}_1, y_1)$ and $(\boldsymbol{x}_2, y_2)$ that both solve $\tilde{f}_{\text{ISMA}}(\boldsymbol{x}, y) = 0$. By convexity of the log-sum-exp function and preservation of convexity through affine mappings, we must have $\tilde{f}_{\text{ISMA}}(\theta\boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2, \theta y_1 + (1-\theta)y_2) \leq 0 \ \forall \ \theta \in [0, 1]$. But for some $\hat{y}$, $\tilde{f}_{\text{ISMA}}(\theta\boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2, \hat{y}) = 0$. Since $\tilde{f}_{\text{ISMA}}$ is monotone decreasing in y, $\hat{y} \leq \theta y_1 + (1-\theta)y_2$. Thus the function value $\hat{y}$ at any weighted combination of $\boldsymbol{x}$ points is less than or equal to a weighted combination of the y values – exactly the definition of convexity. □

ISMA functions have individual softness parameters $\alpha_k > 0$ for each of the K affine terms in the model, bringing the total number of model parameters to $n_p = K(n+2)$. Figure 4.2 illustrates the influence of these individual softness parameters.

If we set all the α_k parameters to the same value α, then we recover the softmax-affine function class. This implies that the implicit softmax-affine class subsumes the softmax-affine class, and therefore also the max-affine class:

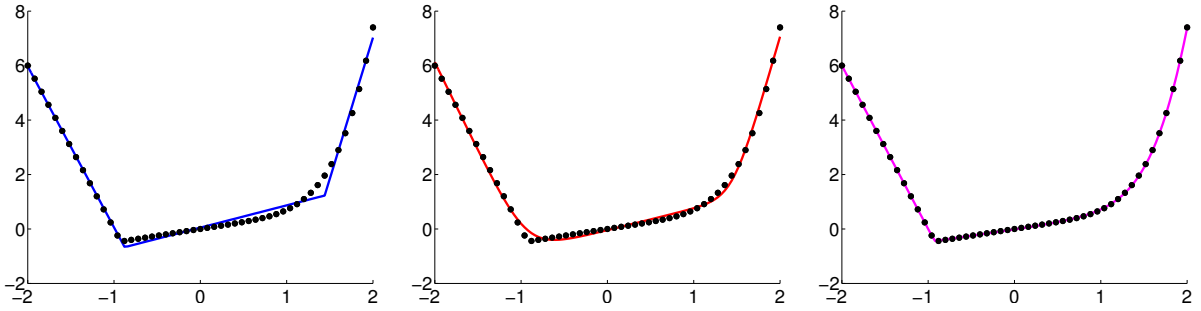
$$\mathcal{F}_{\text{MA}}^K \subsetneq \mathcal{F}_{\text{SMA}}^K \subsetneq \mathcal{F}_{\text{ISMA}}^K \tag{4.8}$$

As a result, there always exists some setting of the α_k parameters for which the ISMA class performs as well as or better than the best model in each of the other function classes.



(a) Varying α_1 (leftmost plane) (b) Varying α_2

Figure 4.2: Influence of individual softness parameters α_k on ISMA functions. Each of the functions above shares the same $K = 4$ affine terms (the thin dashed lines). We set all of the softness parameters α_k to 1, which results in the top curve (dashed line) in each figure. We then varied just one of the four softness parameters to get intuition about its effect. This figure illustrates the situation in $\mathbb{R}^1$, but the qualitative behavior extends to arbitrary dimensions.



(a) MA: RMS = 0.1908 (b) SMA: RMS = 0.1266 (c) ISMA: RMS = 0.0104

Figure 4.3: This toy fitting problem illustrates how ISMA functions can significantly outperform SMA (and therefore also MA) functions. All fits used $K = 3$ affine terms. Here the data are samples of the convex function $y = \max\left(-6x - 6, \frac{1}{2}x, \frac{1}{5}x^5 + \frac{1}{2}x\right)$, which has a sharp kink near $x = -0.8$, but gradual curvature elsewhere. The best fit is an ISMA function, which can conform to the data by adjusting softness locally.

Algorithm 1 Evaluate $y = f_{\text{ISMA}}(\boldsymbol{x})$

$z_k \leftarrow b_k + \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{x},\ \ k = 1 \dots K$
$s \leftarrow \max_k z_k$
$t \leftarrow 0$
repeat
  $f \leftarrow \log \sum_{k=1}^{K} \exp\left(\alpha_k(z_k - s - t)\right)$
  $\partial f/\partial t \leftarrow -\frac{\sum_k \alpha_k \exp(\alpha_k(z_k - s - t))}{\sum_k \exp(\alpha_k(z_k - s - t))}$
  $t \leftarrow t - \frac{f}{\partial f/\partial t}$
until $|f| <$ `machine_precision`
return $y = s + t$

Evaluating ISMA functions No explicit formula is available for evaluating ISMA functions. We therefore give Algorithm 1, a Newton-Raphson [73] algorithm for solving $f_{\text{ISMA}}(\boldsymbol{x}, y) = 0$. The breakdown $y = s + t$ avoids numerical issues associated with computing ratios and logarithms of large exponentials. In practice, Algorithm 1 reliably converges to machine precision in approximately 5-10 Newton iterations.

4.3 Application to Geometric Programming

The GP-compatible Fitting Problem

In *GP-compatible fitting*, we are interested in approximating a set of data points

$$(\boldsymbol{u}_i, w_i) \in \mathbb{R}^d_{>0} \times \mathbb{R}_{>0},\ \ i = 1 \dots m,$$

also represented as $(U, W) \in \mathbb{R}^{m \times d}_{>0} \times \mathbb{R}^m_{>0}$, with monomial or posynomial constraints. The fitting problem variables $(\boldsymbol{u}, w)$ typically represent a subset of the full vector of n GP decision variables; i.e. $d + 1 \leq n$.

Knowing that monomials and posynomials are log-convex, one often applies a log transformation $(X, Y) = (\log U, \log W)$, thereby converting the GP fitting problem into a convex regression problem on the transformed data (X, Y). The most common approach is then to fit y as a max-affine function of $\boldsymbol{x}$, and convert the resulting affine functions back to equivalent monomial constraints on the GP. Since GPs also admit posynomial constraints,

another approach is to directly fit w as a posynomial function of $\boldsymbol{u}$. This approach has been used by a number of authors [16, 45, 57]. Typically mixed results are reported, with max-monomial models performing better on data with sharp kinks, and posynomial models performing better on smoother data [10]. As described in the next section, one benefit of SMA and ISMA functions is the unification of these two approaches.

GP Interpretation of Proposed Convex Function Classes

Each of the convex function classes in this chapter has a parallel interpretation as a GP-compatible function. This makes SMA and ISMA functions natural choices for GP-compatible fitting.

Max-affine Functions as Max-monomial Functions

Recall that under the GP log transformation, monomial functions become affine. Thus, a max-affine model for y,

$$\hat{y} = f_{\text{MA}}(\boldsymbol{x}), \tag{4.9}$$

corresponds exactly to a max-monomial model for w,

$$\hat{w} = \max_{k=1\ldots K} \left[e^{b_k} \prod_{i=1}^{d} u_i^{a_{ik}} \right], \tag{4.10}$$

which is easily converted to a GP-compatible set of K monomial inequality constraints on $(\boldsymbol{u}, w)$,

$$\frac{e^{b_k}}{w} \prod_{i=1}^{d} u_i^{a_{ik}} \leq 1, \;\; k = 1 \ldots K \tag{4.11}$$

Because they can be fit using established max-affine methods, max-monomial functions are currently a common choice for GP modeling [30, 41].

SMA Functions as Posynomial Functions

For the special case $\alpha = 1$, SMA functions reduce to posynomials in convex form (2.9). That is, the model $\hat{y} = f_{\text{SMA}}(\boldsymbol{x}; \alpha = 1)$ corresponds to a posynomial constraint on $(\boldsymbol{u}, w)$,

$$\frac{1}{w} \sum_{k=1}^{K} e^{b_k} \prod_{i=1}^{d} u_i^{a_{ik}} \leq 1. \tag{4.12}$$

More generally, one would allow the fitting process to optimize α, resulting in some SMA model $\hat{y} = f_{\text{SMA}}(\boldsymbol{x})$. Such a model corresponds to a posynomial model for w^α,

$$\frac{1}{w^\alpha} \sum_{k=1}^{K} e^{\alpha b_k} \prod_{i=1}^{d} u_i^{\alpha a_{ik}} \leq 1. \tag{4.13}$$

While the distinction between (4.12) and (4.13) may appear subtle, it has important implications for GP modeling. In particular, in all literature we are aware of, authors who fit posynomial models fit functions of the form $\hat{w} = g(\boldsymbol{u})$, i.e. they search for softmax-affine models, but with the restriction $\alpha = 1$. They typically observe mixed results relative to max-affine functions, meaning the best function class – max-monomial or posynomial – depends on the particular data set [46]. Our work suggests searching for posynomial functions of the form $\hat{w}^\alpha = g(\boldsymbol{u})$.

The general softmax-affine function class includes max-affine functions ($\alpha = +\infty$) and posynomial functions ($\alpha = 1$) as special cases. Softmax-affine functions therefore unify max-monomial and posynomial fitting, eliminating the need to search over multiple model classes.

ISMA Functions as Implicit Posynomial Functions

Consider the GP log transformation applied to (4.6). Evidently, a model $\hat{y} = f_{\text{ISMA}}(\boldsymbol{x})$ corresponds to a posynomial constraint on $(\boldsymbol{u}, w)$,

$$\sum_{k=1}^{K} \frac{e^{\alpha_k b_k}}{w^{\alpha_k}} \prod_{i=1}^{d} u_i^{\alpha_k a_{ik}} \leq 1. \tag{4.14}$$

4.4 Fitting Model Parameters

We now turn to the problem of fitting the proposed function classes to data. We preface this section by noting that data fitting and nonlinear regression are well-established fields supported by an extensive literature. Nevertheless, we expect there is value in describing a practical end-to-end fitting procedure for the specific cases of SMA and ISMA functions.

Fitting Approach Overview

Given m data points $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, a least squares fitting objective is

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \ \sum_{i=1}^{m} \left(f(\boldsymbol{x}_i; \boldsymbol{\beta}) - y_i\right)^2, \tag{4.15}$$

where f is an instance of one of the convex function classes already presented, and $\boldsymbol{\beta} \in \mathbb{R}^{n_p}$ is a vector that contains the parameters a, b, and α for the chosen function class. In general, (4.15) is a *nonlinear least squares* problem. The quantity $\boldsymbol{r}(\boldsymbol{\beta}) = f(X; \boldsymbol{\beta}) - Y$ is the vector of *residuals* at each data point.

Consider some initial set of parameters $\boldsymbol{\beta}_0$ and corresponding residual $\boldsymbol{r}(\boldsymbol{\beta}_0)$. For a small change in parameters $\boldsymbol{\delta}$, the new residual is approximately

$$\boldsymbol{r}(\boldsymbol{\beta}_0 + \boldsymbol{\delta}) \approx \boldsymbol{r}(\boldsymbol{\beta}_0) + \boldsymbol{J}\boldsymbol{\delta}, \tag{4.16}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n_p}$ is $\partial f / \partial \boldsymbol{\beta}$, the *Jacobian* of f. This suggests a first-order approximation of the residual in (4.15), rewritten in terms of the parameter update $\boldsymbol{\delta}$:

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad & \boldsymbol{\delta}^{\mathrm{T}} \boldsymbol{J}^{\mathrm{T}} \boldsymbol{J} \boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}} \boldsymbol{J}^{\mathrm{T}} \boldsymbol{r} + \boldsymbol{r}^{\mathrm{T}} \boldsymbol{r} \\ \text{subject to} \quad & \boldsymbol{\delta} \in \Delta, \end{aligned} \tag{4.17}$$

where Δ represents a *trust region*, imposed to keep the approximation (4.16) valid. Most popular algorithms for nonlinear least squares, including Gauss-Newton methods, Levenberg-Marquardt algorithms [47], and explicit trust region methods, alternate between solving some form of the trust region subproblem (4.17), and updating $\boldsymbol{\beta}$, $\boldsymbol{r}$, and $\boldsymbol{J}$ for those steps that achieve an acceptable improvement in residual. The high-level approach is sketched in Algorithm 2.

Parameter Update

We now describe three options for solving the trust region subproblem (4.17).

Algorithm 2 Nonlinear least squares fitting

$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_0$
$\boldsymbol{\delta} = 0$
$\Delta \leftarrow$ initial trust region parameters
repeat
  $\boldsymbol{r}_{\text{trial}} \leftarrow f(X; \boldsymbol{\beta} + \boldsymbol{\delta}) - Y$
  if trial point acceptable **then**
    $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \boldsymbol{\delta}$
    $\boldsymbol{r} \leftarrow \boldsymbol{r}_{\text{trial}}$
    $\boldsymbol{J} \leftarrow \partial f / \partial \boldsymbol{\beta}$
    $\Delta \leftarrow$ keep or expand trust region
  else
    $\Delta \leftarrow$ constrict trust region
  end if
  $\boldsymbol{\delta} \leftarrow$ `trust-region-subproblem`$(\boldsymbol{J}, \boldsymbol{r}, \Delta)$
until no further improvement
return $\boldsymbol{r}, \boldsymbol{\beta}$

Gauss-Newton Update

Gauss-Newton (GN) methods [56] find the $\boldsymbol{\delta}$ that minimizes the quadratic objective (4.17), with no trust region bounds. The optimal step is the solution to a set of linear equations,

$$\boldsymbol{J}^{\mathrm{T}} \boldsymbol{J} \boldsymbol{\delta} = -\boldsymbol{J}^{\mathrm{T}} \boldsymbol{r}. \tag{4.18}$$

If the computed step does not achieve a satisfactory reduction in residual, a line search is typically used to refine the step length. The least squares partition algorithm due to Magnani and Boyd [46] can be viewed as a Gauss-Newton update for the specific case of max-affine fitting.

Levenberg-Marquardt Algorithms

Levenberg-Marquardt (LM) algorithms [47, 60] are similar to Gauss-Newton methods, but with trust region bounds on the parameter update. Instead of constraining $\boldsymbol{\delta}$ to lie within the bounds of an explicit trust region, LM algorithms construct the trust region implicitly through a quadratic penalty on $\boldsymbol{\delta}$. The advantage of this formulation is that the step is

simply the solution to a linear system,

$$(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda \mathrm{diag}(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}))\,\boldsymbol{\delta} = -\boldsymbol{J}^{\mathrm{T}}\boldsymbol{r}, \tag{4.19}$$

where λ controls the magnitude of the quadratic penalty on $\boldsymbol{\delta}$. Various update schemes for λ exist; in general λ should be increased when the step fails to decrease the residual sufficiently, kept constant when the penalty term is not appreciably affecting the solution, and decreased otherwise.

Explicit Trust Region Methods

A final option, which we use in our implementation, is the direct solution of the trust region subproblem (4.17) via convex programming. For example, with an ellipsoidal trust region shaped by $D = \mathrm{diag}(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J})$, we can find the $\boldsymbol{\delta}$ that solves

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad & \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{r} + \boldsymbol{r}^{\mathrm{T}}\boldsymbol{r} \\ \text{subject to} \quad & \boldsymbol{\delta}^{\mathrm{T}}D\boldsymbol{\delta} \leq \Delta. \end{aligned} \tag{4.20}$$

This is easily transformed to a second order cone program (SOCP), which can be solved extremely efficiently by modern interior point methods. One advantage of this formulation is more direct control over the size of the trust region via Δ instead of λ. Additionally, this formulation readily handles hard constraints and l_1 sparsity penalties on the parameters $\boldsymbol{\beta}$. For example, to locally solve (4.15), but with an added constraint $||\boldsymbol{\beta}||_1 \leq B$, we can formulate the step update as the solution to the SOCP

$$\begin{aligned} \underset{\boldsymbol{\delta},\boldsymbol{\xi},t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & t \geq \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{\delta} + \boldsymbol{\delta}^{\mathrm{T}}\boldsymbol{J}^{\mathrm{T}}\boldsymbol{r} + \boldsymbol{r}^{\mathrm{T}}\boldsymbol{r} \\ & \Delta \geq \boldsymbol{\delta}^{\mathrm{T}}D\boldsymbol{\delta} \\ & \boldsymbol{\xi} \geq \boldsymbol{\beta} + \boldsymbol{\delta} \\ & \boldsymbol{\xi} \geq -\boldsymbol{\beta} - \boldsymbol{\delta} \\ & B \geq \mathbf{1}^{\mathrm{T}}\boldsymbol{\xi}. \end{aligned} \tag{4.21}$$

Model Jacobians

This section lists the partial derivatives needed to construct Jacobians for all the convex function classes presented in Section 4.2. Note that we optimize over $\log \alpha$ instead of α for better numerical conditioning of the Jacobians, and to implicitly constrain $\alpha > 0$.

Max-affine Functions

$$\frac{\partial f_{\text{MA}}}{\partial b_i} = \begin{cases} 1, \; i = \text{argmax}_k \;\; b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x} \\ 0, \text{ otherwise} \end{cases} \tag{4.22}$$

$$\frac{\partial f_{\text{MA}}}{\partial \boldsymbol{a}_i} = \begin{cases} \boldsymbol{x}^{\text{T}}, \; i = \text{argmax}_k \;\; b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x} \\ 0, \text{ otherwise} \end{cases} \tag{4.23}$$

Softmax-affine Functions

$$\frac{\partial f_{\text{SMA}}}{\partial b_i} = \frac{\exp(\alpha(b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}))} \tag{4.24}$$

$$\frac{\partial f_{\text{SMA}}}{\partial \boldsymbol{a}_i} = \frac{\boldsymbol{x}^{\text{T}} \cdot \exp(\alpha(b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}))} \tag{4.25}$$

$$\frac{\partial f_{\text{SMA}}}{\partial (\log \alpha)} = \frac{\sum_{k=1}^{K} (b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}) \exp(\alpha(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}))}{\sum_{k=1}^{K} \exp(\alpha(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x}))} - f_{\text{SMA}} \tag{4.26}$$

Implicit Softmax-affine Functions

$$\frac{\partial f_{\text{ISMA}}}{\partial b_i} = \frac{\alpha_i \exp(\alpha_i(b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))} \tag{4.27}$$

$$\frac{\partial f_{\text{ISMA}}}{\partial \boldsymbol{a}_i} = \frac{\boldsymbol{x}^{\text{T}} \alpha_i \cdot \exp(\alpha_i(b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))} \tag{4.28}$$

$$\frac{\partial f_{\text{ISMA}}}{\partial (\log \alpha_i)} = \frac{\alpha_i (b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}) \exp(\alpha_i(b_i + \boldsymbol{a}_i^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))}{\sum_{k=1}^{K} \alpha_k \exp(\alpha_k(b_k + \boldsymbol{a}_k^{\text{T}} \boldsymbol{x} - f_{\text{ISMA}}))} \tag{4.29}$$

Parameter Initialization

This section describes suitable initial parameter vectors $\boldsymbol{\beta}_0$ for each of the convex function classes.

Max-affine Initialization

To initialize the parameters of a max-affine function, we follow Magnani and Boyd [46]. They pick K data points $\{\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_K\}$ at random without replacement, and partition the data set according to which $\bar{\boldsymbol{x}}$ is closest (in Euclidean distance, for example). Each partition is then fit locally using least squares, thereby forming an initial set of max-affine parameters.

In practice, one or more of the randomly-sampled partitions may contain very few data points, and thus result in a singular local fitting problem. In our implementation, rank-deficient initial partitions are expanded until all the local fits become well-posed.

Softmax-affine Initialization

Since softmax-affine functions represent max-affine functions in the limit $\alpha \to +\infty$, max-affine functions provide a convenient initialization for SMA fitting. In particular, given a max-affine fit $\boldsymbol{\beta}_{\text{MA}} = (\boldsymbol{a}_{\text{MA}}, \boldsymbol{b}_{\text{MA}})$, a natural softmax-affine initialization might be

$$\boldsymbol{\beta}_{\text{SMA}} = (\boldsymbol{a}_{\text{MA}}, \boldsymbol{b}_{\text{MA}}, \alpha_0 = 100)$$

Too large an initial α_0 will cause the α-gradient (4.26) to be extremely small for all data points. We therefore line search over α to find an α_0 that has non-trivial $\boldsymbol{J}_\alpha$ at some data points, but that does not substantially increase the total residual over the max-affine fit.

Implicit Softmax-affine Initialization

Given the parameters $\boldsymbol{\beta}_{\text{SMA}} = (\boldsymbol{a}_{\text{SMA}}, \boldsymbol{b}_{\text{SMA}}, \alpha_{\text{SMA}})$ of a SMA fit, a suitable ISMA initialization is

$$\boldsymbol{\beta}_{\text{ISMA}} = (\boldsymbol{a}_{\text{SMA}}, \boldsymbol{b}_{\text{SMA}}, [\alpha_{\text{SMA}}, \alpha_{\text{SMA}}, \ldots, \alpha_{\text{SMA}}]).$$

Alternatively, one can initialize ISMA fitting directly from a randomly-seeded MA fit:

$$\boldsymbol{\beta}_{\text{ISMA}} = (\boldsymbol{a}_{\text{MA}}, \boldsymbol{b}_{\text{MA}}, [100, 100, ..., 100]).$$

In our implementation, we always try both initializations for every random seed, and select the one with the smaller residual.

Avoiding Numerical Overflow

Many `exp` terms appear in both the convex function definitions and their Jacobians. When exponentiated, a modestly large argument can quickly overwhelm the representation capability of floating point arithmetic. We must therefore use caution when implementing these equations in software. Two common situations occur:

Ratios of sums of exponentials: To handle these, note that

$$\frac{ce^{\alpha p}}{\sum_{i=1}^{N} e^{\alpha p_i}} = \frac{ce^{\alpha(p-s)}}{\sum_{i=1}^{N} e^{\alpha(p_i-s)}}, \tag{4.30}$$

i.e. one can simply subtract some s from all the exponents in the numerator and denominator, such that the largest argument to `exp` is, say, 0.

Log sum exp terms: To handle these, note that

$$\frac{1}{\alpha}\log\sum_{i=1}^{N} e^{\alpha p_i} = s + \frac{1}{\alpha}\log\sum_{i=1}^{N} e^{\alpha(p_i-s)}, \tag{4.31}$$

for some s chosen to keep the maximum exponential small.

4.5 Numerical Examples and Comparisons

Throughout this section, we report *RMS error* on the residuals $\hat{y} - y$, defined as

$$\text{RMS error } = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{x}_i) - y_i)^2}. \tag{4.32}$$

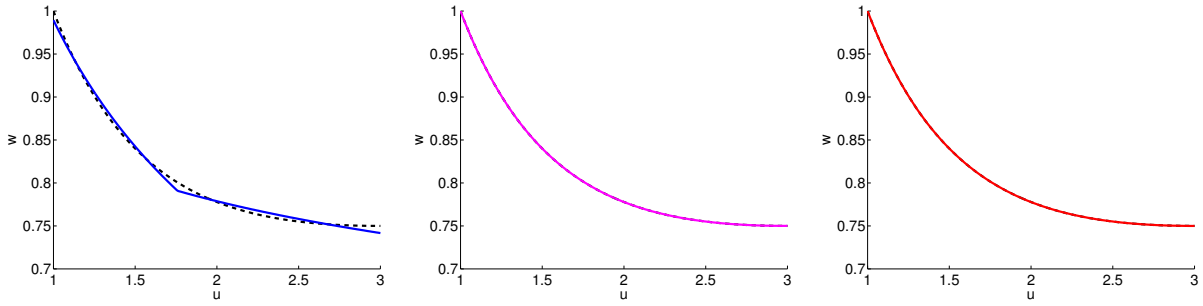
Absolute error on $(\boldsymbol{x}, y)$ is closely related to percentage error on $(\boldsymbol{u}, w)$, since

$$\frac{\hat{w} - w}{w} \approx \log\frac{\hat{w}}{w} = \hat{y} - y. \tag{4.33}$$

Example: Local Convex Fitting of a Scalar Function

Suppose that the scalar relationship

$$w(u) = \frac{u^2 + 3}{(u+1)^2}, \;\; 1 \le u \le 3 \tag{4.34}$$



(a) MA: RMS = 5.24e-3 (b) SMA: RMS = 2.30e-5 (c) ISMA: RMS = 7.48e-6

Figure 4.4: Generalized posynomial fitting of the function $w = (u^2 + 3)/(u + 1)^2$ over $1 \leq u \leq 3$. The function was sampled at 501 logarithmically-spaced points, which were log-transformed and fitted with MA, SMA, and ISMA functions, each with $K = 2$ affine terms. Converting these functions back to to the original (u, w) space turned them into max-monomial and posynomial models.

Table 4.1: Convex fits for the fitting problem in Section 4.5.

Function class	Fitted function	RMS log error
Max-monomial	$\hat{w} = \max\left(0.846u^{-.12}, 0.989u^{-.397}\right)$	5.24×10^{-3}
SMA posynomial	$\hat{w}^{3.44} = 0.154u^{0.584} + 0.847u^{-2.15}$	2.30×10^{-5}
ISMA posynomial	$1 = 0.0658\frac{u^{.958}}{\hat{w}^{3.79}} + 0.934\frac{u^{-1.22}}{\hat{w}^{2.03}}$	7.48×10^{-6}

expresses an important relationship between two variables (u, w) in a GP. Can this relationship be encoded as a GP-compatible constraint?

Importantly, the expression $(u^2 + 3)/(u + 1)^2$ is not log-convex for all $u > 0$ (a simple log-log plot verifies this). Thus there is no hope of algebraically manipulating (4.34) into a posynomial constraint. Nevertheless, (4.34) *is* log-convex over the domain of interest, $1 \leq u \leq 3$. This suggests sampling the function and fitting one of our convex function classes to the 'data'. For this problem we set $K = 2$. Figure 4.4 shows the resulting fits, with SMA and ISMA functions significantly outperforming MA functions. The fitted models are listed in Table 4.1.

Performance on Randomly-generated Convex Functions

To test the proposed function classes and fitting algorithms on a wider range of data sets, we created a convex function generator that produces random instances of convex functions using a stochastic grammar of convexity-preserving operations and composition rules, as follows:

```
function y = rand_cvx_fun(x, step)
dim = size(x, 2);
pbase = step/12; %base case probability increases with depth into recursion
if(rand < pbase)
    %base case: linear function
    c = randn(dim, 1);
    y = x*c;
    if(rand < 0.25)
        %square with some probability
        y = y.^2;
    end
else
    %if not base case, choose sum, max, or softmax
    r = rand;
    if(r < 0.5)  %sum
        y1 = rand_cvx_fun(x, step+1);
        y2 = rand_cvx_fun(x, step+1);
        y = y1+y2;
    elseif(r < 0.75)  %max
        y1 = rand_cvx_fun(x, step+1);
        y2 = rand_cvx_fun(x, step+1);
        y = max([y1, y2], [], 2);
    else  %softmax
        y1 = rand_cvx_fun(x, step+1);
        y2 = rand_cvx_fun(x, step+1);
        y = log(sum([exp(y1), exp(y2)], 2));
    end
end
end
```

We then drew 20 convex functions $\mathbb{R}^2 \to \mathbb{R}$, and 20 more convex functions $\mathbb{R}^5 \to \mathbb{R}$. For example, the first $\mathbb{R}^2 \to \mathbb{R}$ function drawn was

$$\begin{aligned} f_1(x_1,x_2) = \max(&\log(\exp(-0.32724x_1 + 0.89165x_2) + \exp(0.44408x_1 - 0.91182x_2 + ... \\ &\max(-0.10307x_1 - 2.799x_2 + (0.62101x_1 - 1.5075x_2)^2 + 0.2417x_1 + 0.54935x_2 + ... \\ &- 0.2298x_1 - 0.57922x_2 + (0.42162x_1 + 1.0877x_2)^2, \log(\exp((0.17694x_1 + ... \\ &3.4663x_2)^2) + \exp(\max(\log(\exp((-0.4385x_1 + 0.34322x_2)^2) + ... \\ &\exp(\max(-0.83768x_1 - 1.3075x_2, 0.64915x_1 - 0.83147x_2))), 1.5667x_1 + ... \\ &0.8465x_2))) - 0.39754x_1 + 0.25429x_2))), (1.2951x_1 + 2.7681x_2)^2). \end{aligned}$$

Input data $\boldsymbol{x}$ for each function consisted of 1000 points drawn from a multivariate Gaussian with zero mean and identity covariance. For each of the fitting problems, the fitting process was restarted from 10 random initializations, and the best fit was chosen. Tables 4.2 and 4.3 report the average fitting error and time across the 20 randomly-generated fitting problems considered in $\mathbb{R}^2$ and $\mathbb{R}^5$ respectively. These results show that SMA and ISMA functions provide consistent benefits in fitting error, when compared with max-affine functions as a baseline.

Table 4.2: Fitting error comparison for 20 randomly-generated functions $\mathbb{R}^2 \to \mathbb{R}$. Results are reported across the 20 fitting problems considered. The SOCP trust region subproblem (4.20) was solved using MOSEK [54]. The code ran on a quad-core Intel Core i7 CPU @ 2.8GHz with 4GB memory.

	RMS error as percentage of MA error (worst-case, **average**, best-case)			Average fitting time (s) per random restart		
K	MA	SMA	ISMA	MA	SMA	ISMA
2	100.0	(93.4, **74.8**, 56.0)	(93.0, **74.5**, 55.8)	0.18	0.28	0.48
3	100.0	(39.0, **18.7**, 7.1)	(38.6, **19.2**, 9.2)	0.23	0.36	0.90
4	100.0	(43.7, **21.8**, 9.6)	(24.2, **16.0**, 8.0)	0.24	0.55	0.95
5	100.0	(27.5, **18.8**, 9.9)	(22.4, **14.8**, 7.6)	0.26	0.54	1.10
6	100.0	(28.0, **19.9**, 10.4)	(22.0, **15.0**, 7.8)	0.28	0.53	1.56
7	100.0	(32.6, **20.5**, 11.0)	(24.4, **14.2**, 5.8)	0.29	0.92	1.44
8	100.0	(34.6, **20.4**, 10.5)	(24.7, **13.7**, 6.3)	0.32	0.85	1.27
9	100.0	(33.5, **20.6**, 10.2)	(26.2, **14.0**, 6.8)	0.38	1.12	1.55
10	100.0	(34.8, **21.2**, 10.8)	(25.6, **13.8**, 5.8)	0.38	1.22	1.66

Table 4.3: Fitting error comparison for 20 randomly-generated functions $\mathbb{R}^5 \to \mathbb{R}$.

	RMS error as percentage of MA error (worst-case, **average**, best-case)			Average fitting time (s) per random restart		
K	MA	SMA	ISMA	MA	SMA	ISMA
2	100.0	(98.3, **89.4**, 58.6)	(97.6, **88.8**, 58.3)	0.54	0.66	1.40
3	100.0	(90.8, **74.4**, 40.8)	(89.9, **73.3**, 41.0)	0.52	0.74	2.04
4	100.0	(77.9, **59.5**, 39.0)	(73.0, **58.1**, 38.4)	0.62	0.91	2.07
5	100.0	(68.1, **49.1**, 34.2)	(64.2, **47.6**, 30.1)	0.86	0.99	1.98
6	100.0	(68.6, **41.9**, 23.9)	(61.5, **38.1**, 24.3)	0.86	1.43	2.38
7	100.0	(83.9, **47.2**, 25.1)	(73.5, **38.6**, 22.6)	1.15	1.60	2.89
8	100.0	(70.3, **44.1**, 27.0)	(48.3, **33.2**, 20.4)	1.23	1.78	3.00
9	100.0	(76.8, **44.3**, 25.9)	(57.6, **32.5**, 20.7)	1.38	2.39	4.24
10	100.0	(71.0, **43.1**, 25.5)	(46.8, **32.1**, 20.2)	1.56	3.23	5.05

Table 4.4: RMS log errors for the circuit design power model in Section 4.5. For $K = 2$ through $K = 4$, SMA functions outperformed MA functions by a factor of 2.5 to 44.

K	1	2	3	4
MA	0.0904	0.0229	0.01260	0.00760
SMA	”	0.0092	0.00037	0.00017
ISMA	”	0.0091	0.00034	0.00014

Example: Power Modeling for Circuit Design

Here we consider a power dissipation model,

$$P = V_{\text{dd}}^2 + 30 V_{\text{dd}} e^{-(V_{\text{th}} - 0.06 V_{\text{dd}})/0.039}, \tag{4.35}$$

which is relevant for GP-based circuit design. This example comes from [12], and is also studied in [30]. We are interested in modeling the power relationship (4.35), for $1.0 \leq V_{\text{dd}} \leq 2.0$ and $0.2 \leq V_{\text{th}} \leq 0.4$, with a posynomial constraint set.

We proceed by generating 1000 samples of $\boldsymbol{u} = (V_{\text{dd}}, V_{\text{th}})$, uniformly drawn from the region of interest, and associating each sample with the corresponding $w = P(\boldsymbol{u})$. We then apply the log transformation $\boldsymbol{x} = \log \boldsymbol{u}$, $y = \log w$, and fit MA, SMA, and ISMA functions to the resulting data set. The resulting RMS log errors are given in Table 4.4. Of course, any of the resulting models is easily converted to a posynomial constraint on V_{dd}, V_{th}, and

P. For example, the $K = 3$ SMA model corresponds to a posynomial model,

$$P^{2.14} \geq 1.09 V_{\text{dd}}^{4.27} V_{\text{th}}^{0.112} + 7.79 \times 10^{-5} \frac{V_{\text{dd}}^{4.75}}{V_{\text{th}}^{6.44}} + 1.36 \times 10^{-7} \frac{V_{\text{dd}}^{8.94}}{V_{\text{th}}^{8.89}}, \tag{4.36}$$

which is readily substituted directly into any GP corresponding to a practical circuit design problem.

Example: Profile Drag Modeling for Aircraft Design

In aircraft design, *profile drag* on a wing arises due to viscous skin friction in the boundary layer surrounding the airfoil. For a given airfoil cross section shape, the profile drag coefficient (c_d) depends strongly on Reynolds number (Re), airfoil thickness ratio (τ), and lift coefficient (c_l). No analytical expression is available for this relationship, but it can be simulated using the viscous airfoil analysis program XFOIL [19]. We generated 3073 data points $((\text{Re}, \tau, c_l), c_d)_i$, for Re ranging from 10^6 to 10^7, τ ranging from 8% to 16%, and c_l ranging from 0.01 to stall. We are interested in forming a GP-compatible surrogate for the data set, for use in GP-based aircraft design.

As before, we apply a log transformation, $\boldsymbol{x} = \log(\text{Re}, \tau, c_l)$, $y = \log c_d$. We then fit MA, SMA, and ISMA functions to the $(\boldsymbol{x}, y)$ data set for a range of K. As shown in Figure 4.6, an implicit softmax model provides the best fit, followed by softmax-affine, which still outperforms the max-affine fit. Each of the fitted models is directly compatible with GP, and can therefore represent profile drag relations in practical aircraft design problems.

4.6 Conclusions

At the beginning of the chapter, we introduced two new function classes: softmax-affine, and implicit softmax-affine. Section 4.2 showed that these functions form a hierarchy, with the most general ISMA function class able to reproduce the other classes for special parameter settings. This allowed us to conclude that the best ISMA fit is always at least as good (and often much better than) the best fit from the more commonly used max-affine class.

Section 4.3 drew key connections to geometric programming. In particular, both of the proposed function classes can be converted to posynomial constraints, fully compatible with

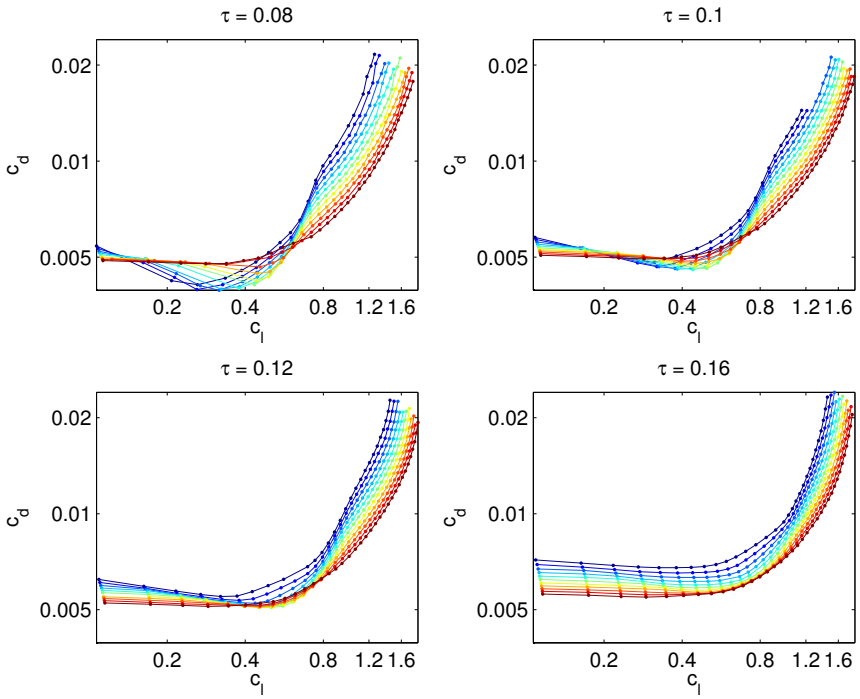


Figure 4.5: Profile drag data as a function of c_l for NACA-24xx airfoils, plotted here on a log scale for slices at four different values of airfoil thickness τ. Colors represent Reynolds number, which ranged from $\mathrm{Re} = 10^6$ (dark blue) to $\mathrm{Re} = 10^7$ (dark red). Each line represents a sweep across angle of attack for a particular airfoil at a constant Re (i.e., a type-1 lift-drag polar). Circles indicate the actual data points, which were generated using XFOIL [19]. An implicit posynomial surrogate model with 8 terms approximates this entire data set with an RMS error of approximately 2%.

GP. Indeed, the proposed approach unifies max-affine fitting, max-monomial fitting, and posynomial fitting as special cases of SMA and ISMA functions. The most general ISMA function class leverages the full expressive power of GP, by using an implicit representation corresponding to the posynomial constraint $g([\boldsymbol{u}, w]) \leq 1$.

Section 4.4 presented the ingredients of a practical fitting algorithm that can be used to fit these functions to real data.

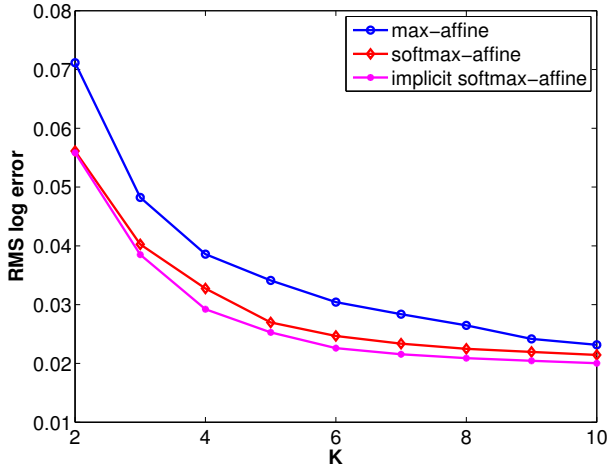


Figure 4.6: Approximating the airfoil profile drag data from Section 5.2. Each model class was fit for a range of K, with implicit softmax-affine functions achieving the best fit. Here the data consists of 3073 samples of profile drag coefficient c_d as a function of lift coefficient (c_l), Reynolds number (Re), and airfoil thickness coefficient (τ). For each K, we did 20 random restarts (initializations), and chose the best fit achieved.

Chapter 5

Selected GP-Compatible Aircraft Design Models

When a GP-compatible formulation of a model is possible, that model can be represented as a constraint inside a GP. When this is possible for all the models in a design problem, the process of *solving the design problem* can be abstracted as a powerful black box that maps a set of design specifications or requirements to a *globally optimal design* (or certificate of infeasibility, if there is no design $\boldsymbol{u}$ that satisfies all the constraints).

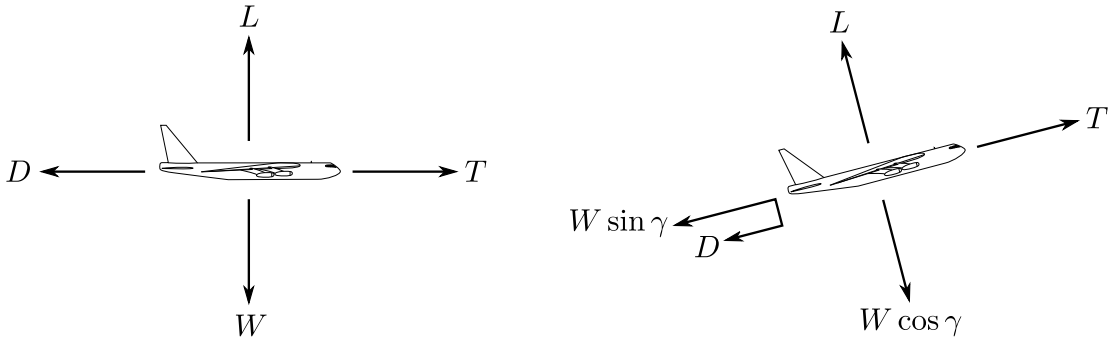
In this section, we give examples of GP-compatible models for the aircraft design domain. Here *GP-compatible* refers to models of two possible forms:

1. $f(\boldsymbol{u}) \leq 1$, where f is a posynomial (or monomial)
2. $g(\boldsymbol{u}) = 1$, where g is a monomial

Note that monomials and posynomials are closed under monomial division. This implies that models of the following forms are also GP-compatible:

3. $f(\boldsymbol{u}) \leq g(\boldsymbol{u})$, where f is a posynomial (or monomial), and g is a monomial
4. $g_1(\boldsymbol{u}) = g_2(\boldsymbol{u})$, where g_1 and g_2 are both monomials

Finally, observe throughout this section that each model involves only a subset of the full decision variable vector $\boldsymbol{u}$. This implies that a GP consisting of these models is *sparse*.



(a) Level flight (b) Climbing flight

Figure 5.1: Force balances in steady flight.

5.1 Steady Flight Relations

Level Flight

The steady level flight relations are perhaps the most basic relationships in aircraft design. In steady level flight, a force balance dictates that lift equals total aircraft weight, and thrust equals drag. Expanding lift and drag into standard expressions involving lift and drag coefficients gives:

$$\frac{1}{2}\rho V^2 C_L S \geq W \tag{5.1}$$

$$D \geq \frac{1}{2}\rho V^2 C_D S \tag{5.2}$$

$$T \geq D. \tag{5.3}$$

Each of these equations are monomial constraints, and therefore GP-compatible.

Climbing Flight

It is straightforward to extend the level flight relations to climbing flight. There are two cases to consider: when the climb angle γ is a decision variable, and when γ is known.

If γ is a decision variable, then we require γ to be small, so that we can approximate $\cos\gamma \approx 1$. We then work with the decision variable $s_\gamma = \sin\gamma$ instead of γ, and obtain the following constraints:

$$W \leq \frac{1}{2}\rho V^2 C_L S \tag{5.4}$$

$$T \geq \frac{1}{2}\rho V^2 C_D S + W s_\gamma \tag{5.5}$$

If, on the other hand, γ is known in advance, then the small angle restriction is removed. The values of $\sin\gamma$ and $\cos\gamma$ become constants known in advance, and we obtain the following constraints:

$$W\cos\gamma \leq \frac{1}{2}\rho V^2 C_L S \tag{5.6}$$

$$T \geq \frac{1}{2}\rho V^2 C_D S + W\sin\gamma \tag{5.7}$$

This model reduces to the level flight model in Section 5.1 for $\gamma = 0$. If γ is known in advance, it can be used to model steep climbs, all the way up to vertical.

Unpowered Gliding Flight

We can also model steady unpowered descents for small descent angles. Because decision variables in a GP must be positive quantities, we replace the climb angle γ with a descent angle $\bar{\gamma} \equiv -\gamma$. For small $\bar{\gamma}$, we approximate $\cos\bar{\gamma} = 1$, and work with the decision variable $s_{\bar{\gamma}} = \sin(\bar{\gamma})$ instead of γ. We obtain the following monomial equality constraints:

$$W \leq \frac{1}{2}\rho V^2 C_L S \tag{5.8}$$

$$W s_{\bar{\gamma}} \geq \frac{1}{2}\rho V^2 C_D S \tag{5.9}$$

In cases where a descent angle of interest is known in advance, we can remove the small angle restriction, as in climbing flight.

5.2 Weight, Drag, and Efficiency Breakdowns

The models in this section capture high-level relationships among lift, drag, weight, and efficiency. Additional constraints in other sections will capture their dependency on more

detailed design parameters.

Drag Breakdown

The total aircraft drag, $D = 1/2\rho V^2 C_D S$ depends on the drag coefficient C_D, which can in turn be modeled as a sum of contributions from different sources. For example, in subsonic flight regimes, a model might break down C_D into three components:

$$C_D \geq \underbrace{\frac{C_L^2}{\pi e A}}_{\substack{\text{induced} \\ \text{drag}}} + \underbrace{C_{Dp}}_{\substack{\text{profile} \\ \text{drag}}} + \underbrace{\frac{(\text{CDA})_0}{S}}_{\substack{\text{non-wing} \\ \text{form drag}}}. \tag{5.10}$$

The *induced drag* term comes from lifting line theory [5], which predicts a vortex-induced downwash distribution over the wing that effectively reduces the angle of attack.

Profile drag is a result of viscous skin friction on the wing. The profile drag coefficient C_{Dp} is in general a complicated function of airfoil shape and flow conditions. Assuming a constant value for C_{Dp} provides a crude but simple model. Far more refined GP-compatible models are also possible. For example, in Chapter 4 the function $C_{Dp}(C_L, \text{Re}, \tau)$ was sampled offline (Figure 4.5) and approximated by an implicit softmax posynomial model.

The final term in the drag breakdown corresponds to *form drag* on the fuselage and other components. For a detailed treatment, it can be further broken up into a posynomial model for component drag areas:

$$(\text{CDA})_0 \geq (\text{CDA})_{\text{tail}} + (\text{CDA})_{\text{fuse}} + (\text{CDA})_{\text{gear}} + ..., \tag{5.11}$$

Weight Breakdown

The total operational weight W breaks down into a sum of component weights, for instance

$$W_{\text{zfw}} \geq W_{\text{fixed}} + W_{\text{payload}} + W_{\text{wing}} + W_{\text{engine}} + W_{\text{tail}} + \ldots, \tag{5.12}$$

$$W \geq W_{\text{zfw}}(1 + \theta_{\text{fuel}}), \tag{5.13}$$

where $\theta_{\text{fuel}} \equiv \frac{W_{\text{fuel}}}{W_{\text{zfw}}}$ is the usable fuel mass fraction. Each of the individual weight terms can either be modeled as a constant, modeled as a GP-compatible function of other design

variables, or further broken down into a (posynomial) sum of component weights. Time varying weights, such as θ_{fuel} and W_{payload}, can be treated as vector variables with one entry per flight condition to be analyzed.

Efficiency Breakdown

Another important steady flight relation is the chain of efficiencies η that relate cruise thrust power to fuel power. A simple version is the monomial

$$TV \leq \dot{m}_{\text{fuel}} h_{\text{fuel}} \eta_{\text{eng}} \eta_{\text{prop}}, \tag{5.14}$$

where TV is thrust times velocity; $\dot{m}_{\text{fuel}}$ is the fuel flow rate; h_{fuel} is the fuel heating value in [J/kg]; η_{eng} is the engine's fuel power to shaft power conversion efficiency, and η_{prop} is the propulsive shaft power to thrust power conversion efficiency. Both η_{eng} and η_{prop} can be broken down further and modeled as posynomial functions of design variables.

5.3 Performance Metrics

The GP framework provides a straightforward interface for trading off competing goals: we optimize or constrain multiple *performance metrics* of interest.

When a performance metric is also a decision variable, (cruise speed, payload capacity, or fuel burn rate, for example), it can be inserted directly into the objective function or constraints. Other metrics are more complicated summary statistics, whose relationships to other variables must themselves be modeled. In this section, we give two examples: range and takeoff distance.

Range

Breguet Range Equation

One common model for the range of a fuel-burning aircraft is the Breguet range equation [21], which assumes a constant lift to drag ratio L/D and overall efficiency η_0, resulting in the

expression

$$R = \frac{h_{\text{fuel}}}{g}\eta_0 \frac{L}{D}\log(1+\theta_{\text{fuel}}). \tag{5.15}$$

This is a fairly crude model. In particular, the assumption that both η_0 and L/D remain constant could be replaced with models for their variation with aircraft velocity and mass. Nevertheless, the equation is common in the field, so we will express it within the GP framework. Equation (5.15) is not allowed in GP due to the log term, but if we rewrite it as

$$1+\theta_{\text{fuel}} = \exp\left(\frac{gRD}{h_{\text{fuel}}\eta_0 L}\right), \tag{5.16}$$

and note that the Taylor expansion of the exp function has a posynomial structure [1] , then we obtain a GP-compatible model:

$$z \geq \frac{gRT}{h_{\text{fuel}}\eta_0 W} \tag{5.17}$$

$$\theta_{\text{fuel}} \geq z + \frac{z^2}{2!} + \frac{z^3}{3!} + \dots \tag{5.18}$$

Recall that the Breguet model assumed that the lift to drag ratio L/D and overall efficiency η_0 remained constant over the entire mission. For real missions, these quantities vary slightly with changes in wing loading, speed, and density altitude. To model these effects more accurately, one can break down long or complex missions into shorter segments of length R_i. One would then constrain each segment according to the Breguet range equation, but allow each segment to take on a different lift to drag ratio W_i/T_i, overall efficiency $\eta_{0,i}$, and fuel fraction $\theta_{\text{fuel},i}$. This approach enables accurate modeling of very long or complex missions, and has the added benefit of reducing the fuel fraction of each segment, thereby improving the accuracy of the Taylor approximation shown in Fig. 5.2.

[1] We could alternatively treat the exp function directly instead of Taylor-expanding it, but at the expense of requiring more specialized convex programming software instead of a standard GP solver.

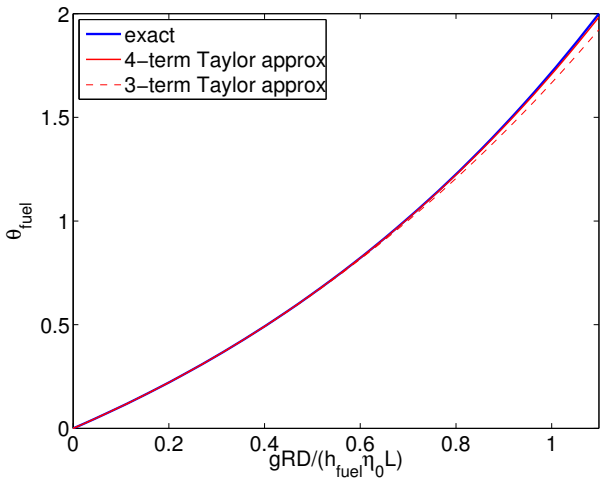


Figure 5.2: GP-compatible approximation of the Breguet range equation, via posynomial structure in the Taylor expansion of exp. A 3-term expansion is more than 99% accurate for fuel fractions less than 0.95, and a 4-term expansion is more than 99% accurate for fuel fractions less than 2.0.

Electric Aircraft Range

For an *electric* aircraft in steady level flight, the range R is limited by the total usable battery energy, E_{bat}, which is expended at a rate $P_{\text{elec}} = TV/\eta_0$. Assuming that the overall efficiency η_0 remains constant, the range prediction is governed by the monomial constraint

$$R \leq \frac{E_{\text{bat}}\eta_0}{T}. \tag{5.19}$$

Endurance

Endurance N is the maximum amount of time an aircraft can stay aloft.

Fuel-Burning Aircraft

An expression similar to the Breguet range equation can be derived for fuel-burning aircraft that are maximizing endurance. The model assumes that as fuel is burned, induced drag decreases accordingly, while $C_{D_0} = C_{Dp} + \text{CDA}_0/S$ remains constant. Under this model, the

aircraft varies its velocity throughout the mission to minimize the drag D. The resulting expression for endurance is

$$N \leq \frac{\eta_0 h_f}{Vg}\left[\frac{16C_{D_0}S}{3\pi e b^2}\right]^{-\frac{1}{2}} \log\left(1+\theta_{\text{fuel}}\right). \tag{5.20}$$

This equation works well for jet-powered aircraft, for which the thrust specific fuel consumption TSFC $\propto \eta_0/V$ stays roughly constant. It is less applicable, however, to propeller aircraft, for which η_0 is roughly constant, but V varies significantly. In any case, it can be converted to GP using the same technique we used for the range equation, where we move the log term to the other side of the equation and closely approximate it.

Electric Aircraft

Like range, the endurance for an electric aircraft is governed by the total available battery energy. Assuming that η and V remain constant, we obtain the monomial constraint

$$N \leq \frac{E_{\text{bat}}\eta_0}{TV}. \tag{5.21}$$

Takeoff Distance

To model takeoff distance x_{TO}, we define a wheels-up flight condition immediately after rotation, where the aircraft first achieves lift $L_{\text{TO}} \geq W$, and is still accelerating under thrust $T_{\text{TO}} > D_{\text{TO}}$. Prior to this instant, a net force $T - D$ accelerated the aircraft from speed 0 to speed V_{TO}. Technically both T and D are functions of V, but let us assume that the thrust variation is small, taking the conservative approximation $T(V) = T(V_{\text{TO}}) = T_{\text{TO}}$. Under this assumption, we have the differential relation

$$gdx = \frac{WVdV}{T_{\text{TO}} - \frac{1}{2}\rho V^2 C_D S} \tag{5.22}$$

from basic mechanics. If we additionally assume that C_D stays constant ($C_D(V) = C_D^{\text{TO}}$), then we can analytically integrate (5.22) along the takeoff run, which results in the expression

$$x_{\text{TO}} = \frac{WV_{\text{TO}}^2}{2gD_{\text{TO}}} \log\left(\frac{T_{\text{TO}}}{T_{\text{TO}} - D_{\text{TO}}}\right). \tag{5.23}$$

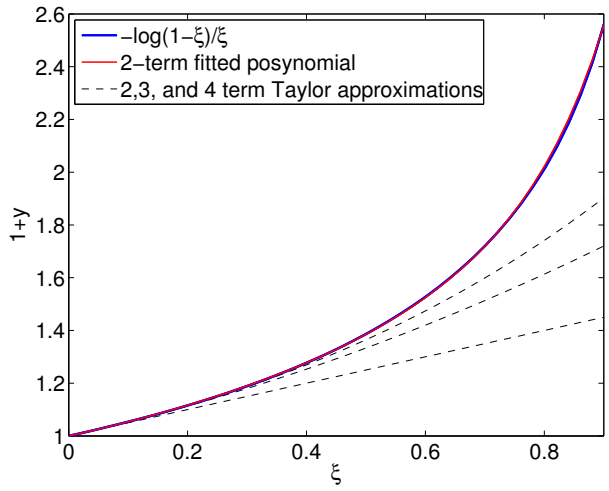


Figure 5.3: GP-compatible posynomial approximation of the function $-\log(1-\xi)/\xi$, for use in modeling takeoff distance. Taylor expansions (dashed green lines) are one option, but would require many terms to achieve a reasonable fit. In contrast, a fitted implicit posynomial (solid red curve) achieves a near-perfect fit with only two terms.

To clarify the limiting behavior as $D_{\text{TO}}/T_{\text{TO}} \to 0$, we rewrite (5.23) as

$$\frac{2gx_{\text{TO}}T_{\text{TO}}}{WV_{\text{TO}}^2} = \frac{-\log(1-\xi)}{\xi}, \tag{5.24}$$

where $\xi \equiv D_{\text{TO}}/T_{\text{TO}}$. This expression is not compatible with GP, but we can proceed by modeling the $-\log(1-\xi)/\xi$ term with a posynomial, as shown in Fig. 5.3. This results in a set of GP-compatible implicit posynomial constraints for takeoff distance:

$$\xi \geq \frac{D_{\text{TO}}}{T_{\text{TO}}} \tag{5.25}$$

$$\frac{2gx_{\text{TO}}T_{\text{TO}}}{WV_{\text{TO}}^2} \geq 1 + \upsilon \tag{5.26}$$

$$1 \geq \frac{0.0464\xi^{\,2.73}}{y^{2.88}} + \frac{1.044\xi^{\,0.296}}{y^{0.049}} \tag{5.27}$$

As suggested for the Breguet range model, one can optionally refine the accuracy of the takeoff distance model by dividing the takeoff run into multiple individually-modeled segments. We can also include 50' obstacle clearance distance using the GP-compatible models for climbing flight from Section 5.1.

Fuel Burn

Fuel burn (and fuel costs) are directly proportional to the mission fuel fraction θ_{fuel}:

$$W_{\text{fuel}} = W_{\text{zfw}}\theta_{\text{fuel}}. \tag{5.28}$$

The mission fuel fractioin θ_{fuel} includes all fuel burn predicted by e.g. the Breguet range equation. It does not include unusable or reserve fuel; these are included in W_{zfw}.

Climb rate

The instantaneous climb rate $\dot{h}$ is governed by the monomial constraint

$$\dot{h} \leq V\gamma. \tag{5.29}$$

This can be used to set constraints on the best climb speed V_{Y}, or on time-to-climb to a specific altitude. If the altitude change is large, then variation of atmospheric properties as well as weight change with fuel burn must be taken into account.

Velocity and payload

Velocity V and payload W_{pay} are metrics that can be optimized directly, since they are decision variables.

5.4 Propulsive Efficiency

A propeller converting mechanical shaft power P_{shaft} into propulsive power TV experiences losses that vary significantly with both thrust and velocity. Following Drela [20], we model propulsive efficiency as the product of a viscous loss term η_v, and an inviscid term η_i that accounts for kinetic energy lost in the high-velocity prop-wash:

$$\eta_{\text{prop}} = \eta_i\eta_v. \tag{5.30}$$

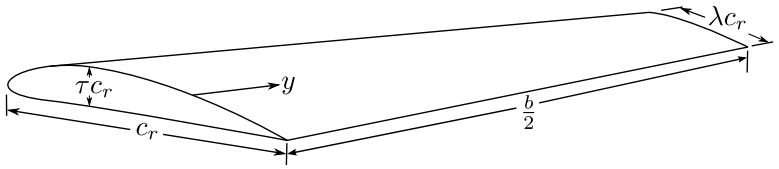


Figure 5.4: Wing design variables τ (airfoil thickness to chord ratio) and λ (taper ratio), for a single-taper wing or wing section.

Actuator disk theory [28] gives us the following approximations for the inviscid efficiency:

$$\dot{m} = \frac{1}{2}\rho A_{\text{prop}}(V_e + V) \tag{5.31}$$

$$T = \dot{m}(V_e - V) = \frac{1}{2}\rho A_{\text{prop}}(V_e^2 - V^2) \tag{5.32}$$

$$\eta_i = \frac{TV}{\frac{1}{2}\dot{m}(V_e^2 - V^2)} = \frac{2}{1 + \frac{V_e}{V}} \tag{5.33}$$

Combining the above, we obtain

$$\eta_i \leq \frac{2}{1 + \sqrt{1 + \frac{T}{\frac{1}{2}\rho V^2 A_{\text{prop}}}}}. \tag{5.34}$$

The quantity $T/(\frac{1}{2}\rho A_{\text{prop}} V^2)$ is recognized as C_T/λ_a^2, where C_T is the propeller thrust coefficient, and λ_a is the advance ratio. The constraint (5.34) is not allowed by the GP framework, but we can algebraically manipulate it into an equivalent posynomial constraint,

$$4\eta_i + \frac{T\eta_i^2}{\frac{1}{2}\rho V^2 A_{\text{prop}}} \leq 4 \tag{5.35}$$

This GP-compatible model captures the strong variation of propulsive efficiency with thrust and velocity.

5.5 Lifting Surface Structural Models

In this section we consider the structural design of an unswept single-taper wing (or tail), as depicted in Fig. 5.4. The high-level stress constraint we will impose is

$$S_r \sigma_{\max} \geq N_{\text{lift}} M_r \tag{5.36}$$

where S_r is the root *section modulus*, $\sigma_{\max}$ is the material-specific allowable stress, N_{lift} is the design G loading or turbulence loading including safety factor, and M_r is the applied moment at the root [2].

We may also wish to impose a deflection limit, e.g.

$$\frac{\delta}{b} \leq 0.05 \tag{5.37}$$

We must now model the spanwise lift distribution, applied root moment, and bending stiffness.

Coordinate Definitions

The wing sizing variables are related by:

$$b = \sqrt{SA} \tag{5.38}$$

$$c_r = \frac{2}{1+\lambda}\sqrt{\frac{S}{A}} \tag{5.39}$$

We also define a wing spanwise coordinate $2y/b = \eta \in [0, 1]$. The chord as a function of span is then

$$\frac{c(\eta)}{c_r} = 1 + \eta(\lambda - 1). \tag{5.40}$$

Root Moment

The moment applied to the wing root depends on the spanwise lift distribution, as shown in Fig. 5.5. The wing must support its own weight (as well as any fuel contained in it), in addition to the weight of the fuselage and payload. Let $\tilde{W} = \tilde{L}$ represent the weight of the aircraft excluding the wing. A simple conservative assumption assumes that the local net-upward-force-per-unit-span, $\tilde{L}'$, is proportional to the local chord [21]. Under this assumption, the differential loading per unit span is

[2] We actually impose the constraint (5.36) for two different values of S_0: one for tensile (bottom skin) loading, and one for compressive (top skin) loading.

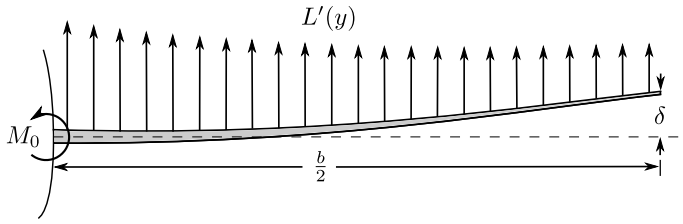


Figure 5.5: The spanwise lift distribution $L'(y)$ creates a root moment M_0 and tip deflection δ.

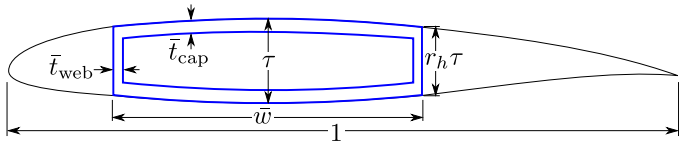


Figure 5.6: Structural cross section geometry for unit-chord airfoil, from Drela [22]. The structural wing box has the same maximum thickness as the airfoil, τ, and tapers quadratically to a fraction r_h at the webs.

$$d\tilde{L} = \frac{\tilde{L}}{1+\lambda}\left[1+\eta(\lambda-1)\right]d\eta \tag{5.41}$$

To find the root moment, we twice-integrate Equation (5.41) with appropriate boundary conditions, which results in

$$M_r = \frac{\tilde{L}b}{12}\left[\frac{1+2\lambda}{1+\lambda}\right] = \frac{\tilde{L}Ac_r}{24}\left(1+2\lambda\right) \tag{5.42}$$

Root Stiffness

The wing root's ability to resist applied moments is governed by two important quantities: the root area moment of inertia, I_r, and the root section modulus, S_r. S_r relates applied moments to maximum bending *stress*, whereas I_r relates applied moments to curvature (and therefore, deflection). For a symmetric structural cross section, the two quantities are related by the monomial

$$S_r = \frac{I_r}{\frac{1}{2}\tau c_r}. \tag{5.43}$$

It is generally possible to fit a posynomial model for the area moment of inertia per chord to the fourth, $\bar{I} = I/c^4$, for an airfoil family and structural geometry of choice. For example, here we will use a wing box geometry defined by Drela [22] and shown in Fig. 5.6. The spar cap $\bar{I}$ is related to material thickness by

$$\bar{I}_{\text{cap}} = \frac{\bar{w}}{12}\left(\bar{h}_{\text{rms}}^3 - (\bar{h}_{\text{rms}} - 2\bar{t}_{\text{cap}})^3\right) \approx \frac{\bar{w}}{2}(\bar{h}_{\text{rms}}^2\bar{t}_{\text{cap}} - 2\bar{h}_{\text{rms}}\bar{t}_{\text{cap}}^2), \tag{5.44}$$

where $\bar{h}_{\text{rms}}$ is the root mean square box height. If we assume that $r_h = 0.75$, then $\bar{h}_{\text{rms}} \approx 0.92\tau$, and a posynomial model for $\bar{I}_{\text{cap}}$ is

$$0.92\bar{w}\tau\bar{t}_{\text{cap}}^2 + \bar{I}_{\text{cap}} \leq \frac{0.92^2}{2}\bar{w}\tau^2\bar{t}_{\text{cap}} \tag{5.45}$$

Making the conservative assumption that the bending stress is carried by the caps only ($\bar{I}_{\text{cap}} >> \bar{I}_{\text{web}}$), the stress limit (5.36) becomes

$$\frac{2\bar{I}_{\text{cap}}c_r^3\sigma_{\max}}{\tau} \geq N_{\text{lift}}M_r. \tag{5.46}$$

Shear Web Sizing

Assuming that all shear loads are carried by the web, the root shear stress is

$$\sigma_{\text{shear}} = \frac{\tilde{L}}{4c_r^2\bar{t}_{\text{web}}r_h\tau}. \tag{5.47}$$

Letting $r_h = 3/4$ and substituting (5.39) for c_r, we obtain the shear web sizing relation:

$$\frac{12\tau S\bar{t}_{\text{web}}\sigma_{\max,\text{shear}}}{A\tilde{L}N_{\text{lift}}} \geq 1 + 2\lambda + \lambda^2 \tag{5.48}$$

Wing System Component Masses

To determine the weight of the spar caps and shear webs, we must integrate their spanwise area distribution. A wing structural component $(\cdot)_c$, with area per chord squared $\bar{A}_c = A_c/c^2$, has a total weight of

$$W_c = \rho_c g 2 \int_0^{b/2} \frac{A_c(y)}{c(y)^2} \frac{c(y)^2}{c_r^2} c_r^2 dy = \rho_c g \bar{A}_c c_r^2 b \int_0^1 \frac{c(\eta)^2}{c_r^2} d\eta \tag{5.49}$$

The spar cap and shear web areas are

$$\bar{A}_{\text{cap}} = 2\bar{w}\bar{t}_{\text{cap}} \tag{5.50}$$

$$\bar{A}_{\text{web}} = 2r_h\tau\bar{t}_{\text{web}} \tag{5.51}$$

Evaluating the integral (5.49), we obtain weight equations for the spar caps and shear webs:

$$W_{\text{cap}} = \frac{8\rho_{\text{cap}} g \bar{w}\bar{t}_{\text{cap}} S^{3/2}}{3A^{1/2}} \left[\frac{\lambda^2+\lambda+1}{(1+\lambda)^2}\right] \tag{5.52}$$

$$W_{\text{web}} = \frac{8\rho_{\text{web}} g r_h \tau\bar{t}_{\text{web}} S^{3/2}}{3A^{1/2}} \left[\frac{\lambda^2+\lambda+1}{(1+\lambda)^2}\right] \tag{5.53}$$

Wing Tip Deflection

Under Euler-Bernoulli bending theory, we have the relationship

$$\frac{d^2\delta}{dy^2} = \frac{M(y)}{EI_{xx}(y)} \tag{5.54}$$

Because both M and I_{xx} vary with y, integrating (5.54) can introduce significant complication. One conservative simplifying assumption is that the curvature is constant along the span and equal to the root curvature. This leads to the relationship

$$\delta = \frac{1}{2}\frac{M_r}{EI_r}\left(\frac{b}{2}\right)^2, \tag{5.55}$$

where E is the Young's modulus of the spar cap.

GP-Compatible Formulation

The key relations for modeling are (5.42), (5.45), (5.46), (5.48), (5.52), (5.53), and (5.55). At first glance, we note that these equations are not GP-compatible, since they are not all

posynomial in λ. To make the wing structural relations GP-compatible, we introduce the change of variables

$$p = 1 + 2\lambda \tag{5.56}$$

We also define $q = 1 + \lambda$. p and q are related by the posynomial constraint

$$2q \geq 1 + p \tag{5.57}$$

We now proceed by expressing the governing relations in terms of p and q instead of λ [3]

.

We define the root moment per root chord, $\bar{M}_r = M_r/c_r$, and replace (5.42) with the equivalent monomial constraint

$$\bar{M}_r \geq \frac{\tilde{L}Ap}{24}. \tag{5.58}$$

The area moment of inertia model (5.45) is already a posynomial constraint on $\bar{I}_{\text{cap}}$, and does not require further modification. The stress limit (5.46) becomes a monomial constraint,

$$8 \geq \frac{N_{\text{lift}}\bar{M}_r A q^2 \tau}{S\bar{I}_{\text{cap}}\sigma_{\max}}. \tag{5.59}$$

The shear web sizing equation (5.48) also becomes a monomial,

$$12 \geq \frac{A\tilde{L}N_{\text{lift}}q^2}{\tau S\bar{t}_{\text{web}}\sigma_{\max,\text{shear}}}. \tag{5.60}$$

To handle the weight equations (5.52) and (5.53), we introduce the function $\nu(\lambda) = (\lambda^2 + \lambda + 1)/(1 + \lambda)^2$, and note that ν is log-convex in p. We can approximate $\nu(p)$ via the posynomial constraint

$$\nu^{3.94} \geq 0.86p^{-2.38} + 0.14p^{0.56}. \tag{5.61}$$

[3] In order for our posynomial equality relaxations to hold with equality, we must ensure that with the exception of (5.57), all constraints involving q are monotone increasing in q. That is, for all constraints $1 \geq f(q)$ involving q, we need $\partial f/\partial q \geq 0$, which holds for all models presented herein.

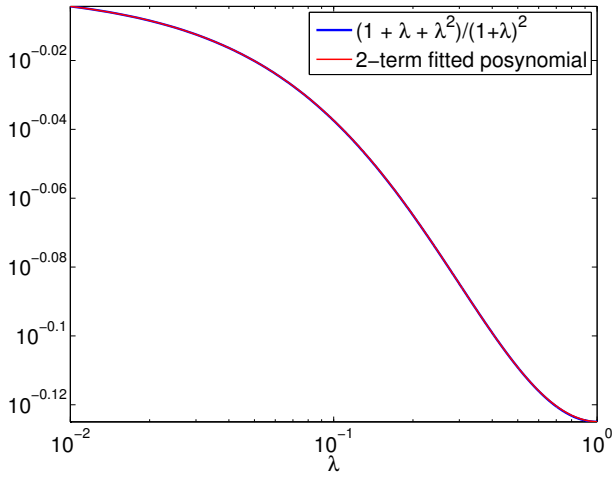


Figure 5.7: The function $\nu(\lambda) = (1+\lambda+\lambda^2)/(1+\lambda)^2$ is clearly not log-convex, and is therefore not GP-compatible. Nevertheless, the function is *quasi-convex*. This allows us to apply a change of variables (5.56) that causes ν to become log-convex in p over the range $1 \leq p \leq 3$, which corresponds to $\lambda \in [0, 1]$. We can then model $\nu(p)$ with a posynomial (5.61). The approximation error is less than 0.03%.

The approximation error is very close to 0, as shown in Fig. 5.7. The weight equations then become monomials:

$$W_{\text{cap}} \geq \frac{8\rho_{\text{cap}} g \bar{w} \bar{t}_{\text{cap}} S^{3/2} \nu}{3A^{1/2}} \tag{5.62}$$

$$W_{\text{web}} \geq \frac{8\rho_{\text{web}} g r_h \tau \bar{t}_{\text{web}} S^{3/2} \nu}{3A^{1/2}} \tag{5.63}$$

Finally, under the change of variables, the wing deflection equation (5.55) is also equivalent to a monomial:

$$\delta \geq \frac{A^{5/2} \bar{M}_r q^3}{64 S^{1/2} E_{\text{cap}} \bar{I}_{\text{cap}}} \tag{5.64}$$

The models we have presented so far will favor small values of λ, since tapering the wing has significant structural benefits in the form of weight savings. However, too small a λ can be dangerous, since it can overload the outboard sections of the wing, leading to risk of tip stall. It is therefore prudent to set a lower limit on λ, e.g.

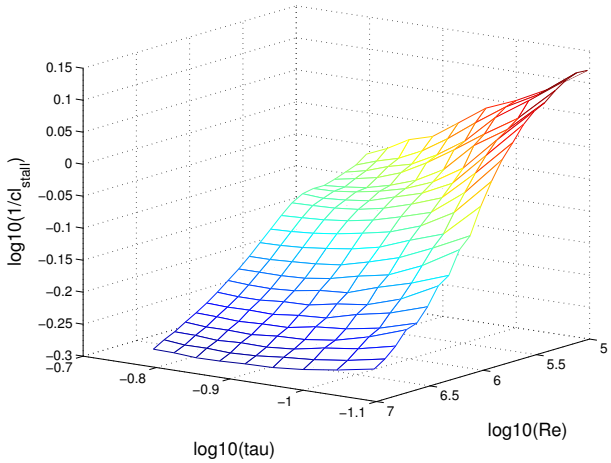


Figure 5.8: NACA-00xx series stall point data from XFOIL, plotted on a log scale. The data is not log-convex, but a single monomial plane provides a reasonable approximation. Here the z-axis is $1/C_{L,\max}$. Unlike drag, high values of $C_{L,\max}$ are actually desirable, so it is $1/C_{L,\max}$ that we seek a log-convex (or log-affine, in this case) model for.

$$p \geq 1.9 \tag{5.65}$$

5.6 Stall

Stall occurs when the aircraft exceeds a critical angle of attack, above which the flow separates from the wing, causing a reduction in lift accompanied by an increase in drag. Stall is an important consideration for slow speed, heavily loaded flight conditions, especially takeoff and landing. Therefore, for flight conditions where stall is a consideration, we impose the constraint

$$C_L \leq (1-\zeta) C_{L,\max}(\tau, \mathrm{Re}), \tag{5.66}$$

where $(1-\zeta)$ is a predetermined safety factor, for instance $(1-\zeta) = 0.9$. In combination with the steady flight relations, this constraint makes it impossible to fly arbitrarily slow.

The functional dependence $C_{L,\max}(\tau, \mathrm{Re})$ can be fit from XFOIL data. For a grid of relevant τ and Re values, we sweep angle of attack from 0 degrees upwards, until the lift

coefficient peaks and begins to decrease. We define $C_{L,\max}$ as the maximum point in this curve, or as the highest lift coefficient observed in the few cases where XFOIL convergence fails prior to stall.

Figure 5.8 shows the data for the NACA-00xx series of airfoils for thickness ranging from 8% to 16%, and Re ranging from 10^5 to 10^7. The data is not perfectly log-convex, but the monomial

$$C_{L,\max} = 0.3528\tau^{0.3535}\mathrm{Re}^{0.1517} \tag{5.67}$$

provides an approximation with an average error of 4.69%, and max error of 18.45%.

Chapter 6

Aircraft Design Example

Here we solve an example design problem via GP-compatible modeling. Our task is the design of a new UAV, which will fly an out-and-back reconnaissance mission.

Objective

The objective is to minimize the total out-and-back fuel burn.

Requirements

The high-level vehicle requirements are

1. Specified range for out-and-back mission, $R \geq 5000$km
2. Specified payload for out-and-back mission, $m_{\text{pay}} \geq 500$kg
3. Sprint speed requirement, separate from design mission, $V_{\text{sprint}} \geq 150$m/s
4. Stall speed requirement for safe landing after aborted takeoff at MTOW, $V_{\text{S0}} \leq 38$m/s

Propulsion

The vehicle will be powered by a single turboprop engine, with propulsive efficiency governed by (5.35). We assume a power-law scaling for engine weight as a function of installed power [62].

Table 6.1: Fixed constant parameters for the design problem in Section 6.

Quantity	Value	Units	Description
N_{lift}	6.0		wing loading multiplier
$\sigma_{\max}$	250×10^6	Pa	allowable stress, 6061-T6
$\sigma_{\max,\text{shear}}$	167×10^6	Pa	allowable shear stress
ρ_{alum}	2700	kg/m^3	aluminum density
g	9.8	m/s^2	gravitational constant
$\bar{w}$	0.5		wing box width/chord
r_h	0.75		see Fig. 5.6
f_{wadd}	2.0		wing added weight fraction
W_{fixed}	14700	N	fixed weight
$C_{L,\max}$	1.5		max C_L, flaps down
ρ	0.91	kg/m^3	air density, 3000m
ρ_{sl}	1.23	kg/m^3	air density, sea level
μ	1.69×10^{-5}	$kg/(sm)$	dynamic viscosity, 3000m
e	0.95		wing spanwise efficiency
A_{prop}	0.785	m^2	propeller disk area
η_v	0.85		propeller viscous efficiency
η_{eng}	0.35		engine efficiency
h_{fuel}	46×10^6	J/kg	fuel heating value

Design Mission

The vehicle must fly out and back some specified distance, at a cruise altitude of 3000m. For GP modeling we split this mission into two legs, outbound and return, with different flight conditions (velocity, lift and drag coefficients, efficiency, etc) along each leg. The fuel burn along each leg is governed by the Breguet range equation. Climb and descent are ignored for simplicity, although we note that this framework is entirely capable of far more detailed climb, cruise, and descent analysis.

Vector Variables

Because we need to analyze the vehicle in three different flight conditions (`outbound`, `return`, `sprint`), the following decision variables are 3-vectors instead of scalars:

$$V,\ C_L,\ C_D,\ C_{D\text{fuse}},\ C_{Dp},\ C_{Di},\ T,\ W,\ \text{Re},\ \eta_i,\ \eta_{\text{prop}},\eta_0$$

When any of these variables appears in a constraint, that constraint is enforced for each element of the vector.

GP Formulation of Example Design Problem

The design problem is given by the following GP:

minimize $W_{\text{fuel,out}} + W_{\text{fuel,ret}}$

subject to

Steady Level Flight Relations

$$W = \frac{1}{2}\rho V^2 C_L S$$

$$T \geq \frac{1}{2}\rho V^2 C_D S$$

$$\text{Re} = \frac{\rho V S^{1/2}}{A^{1/2}\mu}$$

Landing Flight Condition

$$W_{\text{MTO}} \leq \frac{1}{2}\rho_{\text{sl}} V_{\text{S0}}^2 C_{L,\max} S$$

$$V_{\text{S0}} \leq 38$$

Sprint Flight Condition

$$P_{\max} \geq \frac{T_{\text{sprint}} V_{\text{sprint}}}{\eta_{0,\text{sprint}}}$$

$$V_{\text{sprint}} \geq 150$$

Drag Model

$$C_D \geq \frac{0.05}{S} + C_{Dp} + \frac{C_L^2}{\pi e A}$$

$$1 \geq \frac{2.56 C_L^{5.88}}{\text{Re}^{1.54}\tau^{3.32} C_{Dp}^{2.62}} + \frac{3.8\times 10^{-9}\tau^{6.23}}{C_L^{.92}\text{Re}^{1.38} C_{Dp}^{9.57}} + 0.0022\frac{\text{Re}^{0.14}\tau^{0.033}}{C_L^{0.01} C_{Dp}^{0.73}} + \ldots$$

$$\ldots 1.19\times 10^4 \frac{C_L^{9.78}\tau^{1.76}}{\text{Re} C_{Dp}^{0.91}} + \frac{6.14\times 10^{-6} C_L^{6.53}}{\text{Re}^{0.99}\tau^{0.52} C_{Dp}^{5.19}}$$

Propulsive Efficiency

$$\eta_0 \leq \eta_{\text{eng}}\eta_{\text{prop}}$$

$$\eta_{\text{prop}} \leq \eta_i\eta_v$$

$$4\eta_i + \frac{T\eta_i^2}{\frac{1}{2}\rho V^2 A_{\text{prop}}} \leq 4$$

Range Constraints

$$R \geq 5000 \times 10^3$$

$$z_{\text{bre}} \geq \frac{gRT}{h_{\text{fuel}}\eta_0 W}$$

$$\frac{W_{\text{fuel}}}{W} \geq z_{bre} + \frac{z_{bre}^2}{2} + \frac{z_{bre}^3}{6} + \frac{z_{bre}^4}{24}$$

Weight relations

$$W_{\text{pay}} > 500g$$

$$\tilde{W} \geq W_{\text{fixed}} + W_{\text{pay}} + W_{\text{eng}}$$

$$W_{\text{zfw}} \geq \tilde{W} + W_{\text{w}}$$

$$W_{\text{eng}} \geq 0.0372 P_{\text{max}}^{0.803}$$

$$\frac{W_{\text{w}}}{f_{wadd}} \geq W_{\text{web}} + W_{\text{cap}}$$

$$W_{\text{outbound}} \geq W_{\text{zfw}} + W_{\text{fuel,ret}}$$

$$W_{\text{MTO}} \geq W_{\text{outbound}} + W_{\text{fuel,out}}$$

$$W_{\text{sprint}} = W_{\text{outbound}}$$

Wing Structural Models

$$2q \geq 1 + p$$

$$p \geq 1.9$$

$$\tau \leq 0.15$$

$$\bar{M}_r \geq \frac{\tilde{W}Ap}{24}$$

$$0.92\bar{w}\tau\bar{t}_{\text{cap}}^2 + \bar{I}_{\text{cap}} \leq \frac{0.92^2}{2}\bar{w}\tau^2\bar{t}_{\text{cap}}$$

$$8 \geq \frac{N_{\text{lift}} \bar{M}_r A q^2 \tau}{S \bar{I}_{\text{cap}} \sigma_{\max}}$$

$$12 \geq \frac{A \tilde{W} N_{\text{lift}} q^2}{\tau S \bar{t}_{\text{web}} \sigma_{\max,\text{shear}}}$$

$$\nu^{3.94} \geq 0.86 p^{-2.38} + 0.14 p^{0.56}$$

$$W_{\text{cap}} \geq \frac{8 \rho_{\text{cap}} g \bar{w} \bar{t}_{\text{cap}} S^{3/2} \nu}{3 A^{1/2}}$$

$$W_{\text{web}} \geq \frac{8 \rho_{\text{web}} g r_h \tau \bar{t}_{\text{web}} S^{3/2} \nu}{3 A^{1/2}}$$

Optimal Solution of Example Design Problem

With the GP defined, we turn to a commercially-available solver, MOSEK, which finds the globally optimal values for all unknowns in less than 0.01 seconds on a standard desktop computer:

	outbound	return	sprint
V	69.2	65.78	150
CL	0.5523	0.5521	0.1175
CD	0.01292	0.01293	0.007883
CDfuse	0.001725	0.001725	0.001725
CDp	0.005546	0.005559	0.005903
CDi	0.005647	0.005644	0.0002558
T	816	737.8	2340
W	3.489e+04	3.151e+04	3.489e+04
Re	4.716e+06	4.483e+06	1.022e+07
eta_i	0.9028	0.9027	0.9362
eta_prop	0.7674	0.7673	0.7958
eta0	0.2686	0.2686	0.2785

| outbound | return

```
------|-----------------------
Wfuel | 3731       3374
z_bre | 0.1016     0.1017

DESIGN VARIABLES
A           18.1
Ibar        1.908e-05
Mr/cr       3.231e+04
Pmax        1.26e+06
R           5e+06
S           28.99
Vstall      38
nu          0.786
p           1.9
q           1.45
tau         0.15
tcap        0.004273
tweb        0.0005907

WEIGHTS
Wcap        4347
Wzfw        3.151e+04
Weng        2949
Wmto        3.862e+04
Wpay        4900
Wtilde      2.255e+04
Wweb        135.2
Wwing       8965
```

Chapter 7

Current Limitations and Future Perspectives

What about models that are not GP compatible?

The primary limitation of the GP design paradigm is that it only applies to 1) analytical models that can be expressed in terms of posynomials, and 2) data that is well-approximated by log-convex functions. This thesis argues that these conditions are met in a surprisingly wide range of aircraft design relations. Nevertheless, we anticipate three distinct types of GP-incompatibility that will occur:

Discrete Decisions such as number of engines, elevator vs. canard, choice of material, etc., can be modeled by *integer* variables that correspond to the discrete possibilities for each decision. Including these configuration choices in the design optimization results in a feasible set (i.e., design space) that is not convex.

When the integer variables represent a count, such as the number of seating rows in a fuselage, one option is to relax the problem by treating the integer variable(s) as continuous, and then to round the solution to the nearest integer.

A more general technique, which is well suited to any type of discrete decision variable, is to model the design problem as a *mixed-integer GP*. With the exception of extremely small problems (where the choices can simply be enumerated), mixed integer optimiza-

tion problems are significantly more difficult to solve than convex programs. Most solution methods make some sacrifices (such as no longer guaranteeing global optimality) in return for finding an acceptable solution in a reasonable amount of time. Solution methods for mixed-integer GPs are an active area of research [10, 14].

Quasi-convex functions are those functions for which every *level set* is a convex set [11]. Put informally, quasi-convexity extends the concept of unimodality to higher dimensional functions. When a model or physical relationship is described by a quasi-convex (or log-quasi-convex) function, it *may* be possible to find a nonlinear change of variables under which the relevant functions become convex (or log-convex). A perfect example is posynomial functions. Written in standard form, a GP is a set of (quasi-convex, but not necessarily convex) posynomial functions. Under the variable change $\boldsymbol{x} = \log \boldsymbol{u}$, however, all of the posynomials become convex. Another example is the change of variables (5.56) used in Chapter 5 to form a set of GP-compatible wing structural properties.

The art of GP modeling would benefit greatly from a theory of how to automatically search for a change of variables that converts a quasi-convex constraint set to a convex constraint set. If such a change of variables is possible in general, the modeling process would require less effort, and convex optimization could be applied to an expanded set of engineering models.

Even when a change of variables is not possible, there are many techniques available for handling more general (not necessarily log-convex) continuous models within the GP framework. All such methods sacrifice guarantees of global optimality, and are therefore qualitatively similar to solving general nonlinear programs. They differ from NLP methods, however, in their treatment of large subsets of the problem in a convex form. Notable methods of this type include *signomial programming* and the *convex-concave procedure* [11].

Multi-modal functions are neither convex nor quasi-convex. If we encounter a truly multi-modal relationship in our modeling efforts, we have uncovered something very interesting. With multiple locally optimal points, perhaps the underlying relationship

can be modeled as a discrete decision among a number of 'basins', with a locally convex model corresponding to each basin? Even this simple knowledge can be informative, by helping to disect sources of non-convexity in the design space.

In each of the three situations listed above, the knowledge that a large portion of the problem is GP-compatible allows much of the computational work can be offloaded as a GP, with only a few troublesome relations remaining to iterate over.

How does the GP approach relate to Multidisciplinary Design Optimization (MDO) architectures?

In their 1993 paper, Cramer et al. outlined standard formulations for MDO [15]. These include all-at-once (AAO, sometimes called SAND), individual discipline feasible (IDF), and multidisciplinary feasible (MDF). The key differences among the approaches are degree to which optimization is centralized or decentralized, and what kind of feasibility is maintained during each optimization iteration. GP formulations are AAO approaches, characterized by a centralized solver and lack of disciplinary separation.

Historically, AAO approaches have performed very well in benchmarking tests against other MDO algorithms [68]. Some believe they are the most computationally tractable of all MDO approaches [15], but they are often written off because they tend to create extremely large problems with many constraints. We hold hope for the GP version of AAO, because the restriction to convex constraint sets enables efficient solutions that scale to problems with hundreds of thousands of constraints [10].

Can black-box analysis routines be called by a GP solver?

No. GP solvers accept as input a standard parameterization (posynomial coefficients and exponents) of the GP that is to be solved. There is no way for a GP solver to interface with a disciplinary solver. Such an interface would void all guarantees of global optimality and efficient optimization that are provided by GP.

As detailed in Chapter 4, however, one promising approach is to sample disciplinary solvers offline, and incorporate the results by fitting GP-compatible surrogate functions to

the resulting data. A relevant research direction along these lines is automating the discovery of convexity in arbitrary data sets, and identifying regions within a data set that exhibit log-convexity.

How does the GP approach handle the organizational and communication challenges typical in MDO?

One powerful organizational strength of the GP approach derives from centralizing the optimization, but *decentralizing* the modeling of physics. A well known challenge for MDO is coordinating communication and data transfer, especially when subsystem teams or optimizer subroutines depend on other analyses to determine the values of shared variables.

The GP paradigm takes a different approach. Instead of sharing the results of analysis runs, subsystem teams share GP-compatible *models*. Coordination problems are limited to agreeing on a common modeling language (e.g. the variable names). Each subsystem expert's input into the optimization problem is a set of mathematical models for the physical relationships they so deeply understand.

One strength of GP is the ability to add constraints and refine models with zero overhead. When teams want to improve a model, capture a new effect, or model a new tradeoff, they simply add or update the corresponding constraint. There is no need to update communication protocols or change the order of computations, since these are all handled externally by the GP solver at run time.

Does this approach maintain 'feasibility' at every iteration?

In the standard MDO context, *feasibility* usually refers to an equilibrium condition where the inputs and outputs of the various analysis equations agree with equality. Many formulations impose constraints that drive the residual of these quantities to zero. MDO formulations vary as to when they enforce these feasibility equality constraints.

In geometric programming, and more generally in convex optimization, *feasibility* refers to a condition where all equality and inequality constraints are satisfied. In this paper, we used the posynomial equality relaxation described in Section 2.3 to expand the feasible set of

the optimization problem. Interior point methods for solving GPs stay inside this expanded feasible set during optimization.

Does this approach support multiobjective optimization?

Yes. One of the strongest benefits of the GP approach is complete flexibility with regard to the objective function. Concretely, the objective may be any monomial or posynomial function of the decision variables. In practice, one chooses to maximize (or minimize) some key criteria of interest, such as range, fuel consumption, payload, or cruise speed. The dependence of each of these quantities on other decision variables is modeled as part of the GP-compatible constraint set.

In multiobjective optimization, we are interested in sweeping out a Pareto frontier corresponding to the tradeoff surface among a number of variables. Again, this is one of the strongest capabilities of the GP approach. There are two ways to sweep out such a Pareto frontier:

1. We can formulate a posynomial objective function corresponding to a weighted combination of some criteria of interest (a weighted combination of range and payload, for example). We then sweep the weights through a convex set, solving the GP at each point.

2. We can pick one variable as the objective (range, for example), and constrain other variables of interest (payload, for example) to be greater than (or less than) some value s. We then sweep s over a range of interest, solving the GP at each point.

In both cases, the speed of GP solution methods allows each point to be calculated extremely quickly, freeing up the decision maker to consider a larger number of possible Pareto-optimal designs. Moreover, the sensitivity analysis methods of Chapter 3 provide a local approximation of the Pareto frontier from just one GP solution.

Bibliography

[1] Ira H Abbott and AE Von Doenhoff. *Theory of wing sections: including a summary of airfoil data*. Dover Publications, 1959.

[2] Charles N Adkins and Robert H Liebeck. Design of optimum propellers. *Journal of Propulsion and Power*, 10(5):676–682, 1994.

[3] Natalia M Alexandrov and Robert Michael Lewis. Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA Journal*, 40(2):301–309, 2002.

[4] Juan J. Alonso. Aa222: Introduction to multidisciplinary design optimization. Lecture Notes, 2010.

[5] John D. Anderson. *Fundamentals of Aerodynamics*. McGraw-Hill, third edition, 2001.

[6] Holt Ashley. *Aerodynamics of wings and bodies*. Courier Dover Publications, 1965.

[7] A. Babakhani, J. Lavaei, J. Doyle, and A. Hajimiri. Finding globally optimum solutions in antenna optimization problems. *IEEE International Symposium on Antennas and Propagation*, 2010.

[8] Oktay Baysal and Mohamed E Eleshaky. Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA journal*, 30(3):718–725, 1992.

[9] Charles S Beightler and Don T Phillips. *Applied geometric programming*, volume 150. Wiley New York, 1976.

[10] Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 2007.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[12] Stephen P. Boyd, Seung-Jean Kim, Dinesh D. Patil, and Mark A. Horowitz. Digital circuit optimization via geometric programming. *Operations Research*, 53:899–932, 2005.

[13] T. Bui-Thanh, K. Willcox, and O. Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 2008.

[14] Mung Chiang. Geometric programming for communication systems. *Commun. Inf. Theory*, 2:1–154, July 2005.

[15] Evin J Cramer, JE Dennis, Jr, Paul D Frank, Robert Michael Lewis, and Gregory R Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4(4):754–776, 1994.

[16] Walter Daems, Georges Gielen, and Willy Sansen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(5):517–534, 2003.

[17] Mark Drela. *Two-dimensional transonic aerodynamic design and analysis using the Euler equations*. PhD thesis, Massachusetts Institute of Technology, 1985.

[18] Mark Drela. Pros and cons of airfoil optimization. *Frontiers of computational fluid dynamics*, 1998, 1998.

[19] Mark Drela. Xfoil subsonic airfoil development system. Open source software available at `http://web.mit.edu/drela/Public/web/xfoil/`, 2000.

[20] Mark Drela. Qprop formulation - theory document. MIT Aero & Astro, 2006.

[21] Mark Drela. Course notes. MIT Unified Engineering, 2009.

[22] Mark Drela. Tasopt 2.00 – transport aircraft system optimization. *MIT N+3 Final Report*, March 2010.

[23] Mark Drela and Michael B Giles. Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA journal*, 25(10):1347–1355, 1987.

[24] Richard James Duffin, Elmor L Peterson, and Clarence Zener. *Geometric programming: theory and application*. Wiley New York, 1967.

[25] Jonathan Elliott and Jaume Peraire. Practical three-dimensional aerodynamic design and optimization using unstructured meshes. *AIAA journal*, 35(9):1479–1485, 1997.

[26] Bernard Etkin and Lloyd Duff Reid. *Dynamics of flight: stability and control*. Wiley New York, 1982.

[27] E Eugene Larrabee and Susan E French. Minimum induced loss windmills and propellers. *Journal of Wind Engineering and Industrial Aerodynamics*, 15(1):317–327, 1983.

[28] R.E. Froude. On the part played in propulsion by differences of fluid pressure. *Trans. Inst. Naval Architects*, 30:390, 1889.

[29] Sydney Goldstein. On the vortex theory of screw propellers. *Proceedings of the Royal Society of London. Series A*, 123(792):440–465, 1929.

[30] Lauren Hannah and David Dunson. Ensemble methods for convex regression with applications to geometric programming based circuit design. *arXiv preprint arXiv:1206.4645*, 2012.

[31] Lauren A Hannah and David B Dunson. Multivariate convex regression with adaptive partitioning. *arXiv preprint arXiv:1105.1924*, 2011.

[32] Warren Hoburg and Pieter Abbeel. Geometric programming for aircraft design optimization. *In Review*, 2013.

[33] R.T. Haftka J. Sobieszczanski-Sobieski. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14:1–23, 1997.

[34] Antony Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3(3):233–260, 1988.

[35] Antony Jameson. Computational aerodynamics for aircraft design. *Science(Washington)*, 245(4916):361–371, 1989.

[36] Antony Jameson. Re-engineering the design process through computation. *Journal of Aircraft*, 36(1):36–50, 1999.

[37] Antony Jameson and JJ Alonso. Automatic aerodynamic optimization on distributed memory architectures. *AIAA paper*, pages 96–0409, 1996.

[38] Antony Jameson, L Martinelli, and NA Pierce. Optimum aerodynamic design using the navier–stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1-4):213–237, 1998.

[39] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics: from wing theory to panel methods*, volume 1110. McGraw-Hill Signapore, 1991.

[40] Hyung Min Kim, Nestor F Michelena, Panos Y Papalambros, and Tao Jiang. Target cascading in optimal system design. *Journal of Mechanical Design*, 125:474, 2003.

[41] Jintae Kim, L. Vandenberghe, and Chih-Kong Ken Yang. Convex piecewise-linear modeling method for circuit optimization via geometric programming. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(11):1823 –1827, nov. 2010.

[42] Srinivas Kodiyalam and Jaroslaw Sobieszczanski-Sobieski. Bilevel integrated system synthesis with response surfaces. *AIAA Journal*, 38(8):1479–1485, 2000.

[43] Ilan Kroo and Richard Shevell. Aircraft design: Synthesis and analysis. Online Textbook, Version 0.9.

[44] Ilan M. Kroo. MDO for large-scale design. In N. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State-of-the-Art*, pages 22–44. SIAM, 1997.

[45] Xin Li, Padmini Gopalakrishnan, Yang Xu, and T Pileggi. Robust analog/rf circuit design with projection-based posynomial modeling. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 855–862. IEEE Computer Society, 2004.

[46] Alessandro Magnani and Stephen P. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10:1–17, 2009.

[47] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):pp. 431–441, 1963.

[48] Jay D Martin and Timothy W Simpson. Use of kriging models to approximate deterministic computer models. *AIAA journal*, 43(4):853–863, 2005.

[49] Joaquim RRA Martins, Juan J Alonso, and James J Reuther. A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. *Optimization and Engineering*, 6(1):33–62, 2005.

[50] Joaquim RRA Martins, Peter Sturdza, and Juan J Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[51] JRRA Martins and Andrew B Lambe. Multidisciplinary design optimization: Survey of architectures. *AIAA Journal*, 2012.

[52] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on optimization*, 2(4):575–601, 1992.

[53] AJ Morris. Structural optimization by geometric programming. *International Journal of Solids and Structures*, 8(7):847–864, 1972.

[54] Mosek-ApS. Mosek version 6.0.0.148. Free academic license available at `http://mosek.com/resources/academic-license/personal-license/`, 2012.

[55] Yu Nesterov and A Nemirovsky. Interior-point polynomial methods in convex programming, volume 13 of studies in applied mathematics. *SIAM, Philadelphia, PA*, 1994.

[56] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer Science+ Business Media, 2006.

[57] Jorge Oliveros, Dwight Cabrera, Elkim Roa, and Wilhelmus Van Noije. An improved and automated design tool for the optimization of cmos otas using geometric programming. In *Proceedings of the 21st annual symposium on Integrated circuits and system design*, pages 146–151. ACM, 2008.

[58] Elmor L Peterson. Geometric programming. In *Advances in Geometric Programming*, pages 31–94. Springer, 1980.

[59] Olivier Pironneau. *Optimal shape design for elliptic systems*, volume 2983. Springer-Verlag New York, 1984.

[60] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[61] Joaquin R RA Martins, Juan J Alonso, and James J Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3):523–530, 2004.

[62] Daniel P Raymer et al. *Aircraft design: a conceptual approach*. American Institute of Aeronautics and Astronautics Reston, 2006.

[63] TD Robinson, MS Eldred, KE Willcox, and R Haimes. Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. *Aiaa Journal*, 46(11):2814–2822, 2008.

[64] William P Rodden and Erwin H Johnson. *MSC/NASTRAN Aeroelastic Analysis: User's Guide, Version 68*. MacNeal-Schwendler Corporation, 1994.

[65] IP Sobieski and IM Kroo. Collaborative optimization using response surface estimation. *AIAA journal*, 38(10):1931–1938, 2000.

[66] Jaroslaw Sobieszczanski-Sobieski, Troy D Altus, Matthew Phillips, and Robert Sandusky. Bilevel integrated system synthesis for concurrent and distributed processing. *AIAA Journal*, 41(10):1996–2003, 2003.

[67] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 546 pp, 1986.

[68] Nathan P. Tedford and Joaquim R.R.A. Martins. Benchmarking multidisciplinary design optimization algorithms. *Optim Eng*, February 2009.

[69] Theodore Theodorsen. *Theory of wing sections of arbitrary shape*. US Government Printing Office, 1932.

[70] Theodore Theodorsen. *Theory of propellers*, volume 9. McGraw-Hill Book Company, 1948.

[71] D.J. Wilde. *Globally optimal design*. Wiley interscience publication. Wiley, 1978.

[72] Karen Willcox and Jaime Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[73] Tjalling J Ypma. Historical development of the newton-raphson method. *SIAM review*, 37(4):531–551, 1995.

Appendix A

Airfoil Area and Inertia Calculations

The shape of a 2D airfoil section is often specified using a polygon with unit chord, points (x_i, y_i), leading edge at (0, 0), and trailing edge at (1, 0). Area integrals over the polygon can be converted to line integrals around the polygon using Green's Theorem:

$$\iint \left(\frac{dQ}{dx} - \frac{dP}{dy} \right) dA = \oint P dx + Q dy \tag{A.1}$$

With appropriate choice of P and Q, the unit airfoil area, neutral axis, and second moments are:

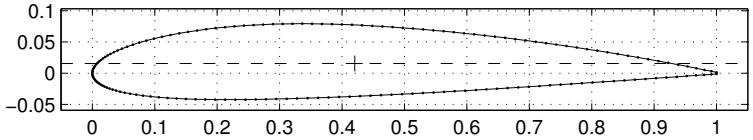


Figure A.1: Unit-chord polygon representation of a NACA2412 airfoil, using 200 panels (polygon edges).

$$A_{2D} = \iint dA = \oint x dy \qquad (P = 0,\ Q = x) \tag{A.2}$$

$$\bar{x} = \frac{1}{A_{2D}} \iint x dA = \frac{1}{A_{2D}} \oint \frac{x^2}{2} dy \qquad (P = 0,\ Q = \frac{x^2}{2}) \tag{A.3}$$

$$\bar{y} = \frac{1}{A_{2D}} \iint y dA = \frac{-1}{A_{2D}} \oint \frac{1}{2} y^2 dx \qquad (P = -\frac{y^2}{2},\ Q = 0) \tag{A.4}$$

$$I_{xx} + A_{2D}\bar{y}^2 = \iint y^2 dA = -\oint \frac{y^3}{3} dx \qquad (P = -\frac{y^3}{3},\ Q = 0) \tag{A.5}$$

$$I_{yy} + A_{2D}\bar{x}^2 = \iint x^2 dA = \oint \frac{1}{3} x^3 dy \qquad (P = 0,\ Q = \frac{x^3}{3}) \tag{A.6}$$

$$I_{xy} + A_{2D}\bar{x}\bar{y} = \iint xy dA = \oint \frac{1}{2} x^2 y dy \qquad (P = 0,\ Q = \frac{x^2}{2} y) \tag{A.7}$$

Assuming that the polygon points are simply connected and ordered counterclockwise, the line integrals reduce to sums of integrals over each polygon edge. Noting that $dy = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} dx$, the above line integrals evaluate to:

$$A_{2D} = \frac{1}{2} \sum_i (x_i y_{i+1} - y_i x_{i+1}) \tag{A.8}$$

$$\bar{x} = \frac{1}{6A_{2D}} \sum_i (x_i + x_{i+1})(x_i y_{i+1} - y_i x_{i+1}) \tag{A.9}$$

$$\bar{y} = \frac{1}{6A_{2D}} \sum_i (y_i + y_{i+1})(x_i y_{i+1} - y_i x_{i+1}) \tag{A.10}$$

$$I_{xx} = \frac{1}{12} \sum_i (y_i^2 + y_i y_{i+1} + y_{i+1}^2)(x_i y_{i+1} - y_i x_{i+1}) - A_{2D}\bar{y}^2 \tag{A.11}$$

$$I_{yy} = \frac{1}{12} \sum_i (x_i^2 + x_i x_{i+1} + x_{i+1}^2)(x_i y_{i+1} - y_i x_{i+1}) - A_{2D}\bar{x}^2 \tag{A.12}$$

$$I_{xy} = \frac{1}{24} \sum_i (2x_i y_i + 2x_{i+1} y_{i+1} + x_i y_{i+1} + y_i x_{i+1})(x_i y_{i+1} - y_i x_{i+1}) - A_{2D}\bar{x}\bar{y} \tag{A.13}$$

Where cancellations have been made by noting that sums around the polygon of terms with a constant index cancel. For example, $\sum_i (x_{i+1} y_{i+1} - x_i y_i) = 0$.

For a thin skin around the airfoil, the area integrals reduce directly to line integrals (without invoking Green's theorem). Letting $ds^2 = dx^2 + dy^2$, the perimeter, neutral point, and second moments for a unit airfoil skin of thickness $\bar{t}$ are:

$$l_{\text{perim}} = \oint ds = \oint \sqrt{dy^2 + dx^2} \tag{A.14}$$

$$\bar{x} = \frac{1}{l_{\text{perim}}} \oint x ds \tag{A.15}$$

$$\bar{y} = \frac{1}{l_{\text{perim}}} \oint y ds \tag{A.16}$$

$$\frac{I_{xx}}{\bar{t}} = \oint y^2 \bar{t} ds \tag{A.17}$$

$$\frac{I_{yy}}{\bar{t}} = \oint x^2 \bar{t} ds \tag{A.18}$$

These integrals can be solved by noting that $ds = dy\sqrt{1 + (dx/dy)^2} = dx\sqrt{1 + (dy/dx)^2}$, where dx/dy and dy/dx are constants along any polygon edge. Cancellations eliminate any problems with infinite terms for vertical or horizontal edges, and result in finite sums as expected:

$$l_{\text{perim}} = \sum_i \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{A.19}$$

$$\bar{x} = \frac{1}{2l_{\text{perim}}} \sum_i (y_i + y_{i+1}) \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{A.20}$$

$$\bar{y} = \frac{1}{2l_{\text{perim}}} \sum_i (x_i + x_{i+1}) \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{A.21}$$

$$\frac{I_{xx}}{bart} = \frac{1}{3} \sum_i (y_i^2 + y_i y_{i+1} + y_{i+1}^2) \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{A.22}$$

$$\frac{I_{yy}}{\bar{t}} = \frac{1}{3} \sum_i (x_i^2 + x_i x_{i+1} + x_{i+1}^2) \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{A.23}$$