

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Simulation-Informed Optimization and Machine Learning for Advanced Manufacturing

Permalink

<https://escholarship.org/uc/item/8p61g06t>

Author

Howell, Brian M

Publication Date

2024

Peer reviewed|Thesis/dissertation

Simulation-Informed Optimization and Machine Learning for Advanced Manufacturing

by

Brian Matthew Howell

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering Science-Mechanical Engineering

and the Designated Emphasis

in

Computational Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Tarek Zohdi, Chair

Associate Professor Grace Gu

Professor Jon Wilkening

Summer 2024

SIMULATION-INFORMED OPTIMIZATION AND MACHINE LEARNING FOR
ADVANCED MANUFACTURING

COPYRIGHT 2024
BY
BRIAN MATTHEW HOWELL

Simulation-Informed Optimization and Machine Learning for Advanced Manufacturing ©
2024 by Brian M Howell is licensed under Creative Commons Attribution 4.0 International.
To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

All Rights Reserved

No part of this work may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Abstract

Simulation-Informed Optimization and Machine Learning for Advanced Manufacturing

by

Brian Matthew Howell

Doctor of Philosophy in Engineering Science-Mechanical Engineering

and the Designated Emphasis in

Computational Data Science and Engineering

University of California, Berkeley

Professor Tarek Zohdi, Chair

In modern manufacturing, optimizing chemical properties, material composition, and processing parameters is essential for achieving desired performance benchmarks given manufacturing and design constraints. Traditional methods often rely on iterative trial-and-error or brute-force design of experiments (DOE), where materials and operating parameters are selected based on intuition and experience. This process is typically repeated until critical benchmarks are met or resources are depleted. Recent advancements in computing are beginning to transform this approach, enabling rapid multi-physics simulations and efficient machine learning/optimization algorithms that offer significant advantages over traditional DOE methods. These simulations are faster, more cost-effective, and environmentally friendly, reducing engineering time and manufacturing resources while minimizing overall development risk.

This work presents an integrated approach that combines experimentation, multi-physics modeling/simulation, numerical optimization, and machine learning techniques. These components are integrated into a cohesive, simulation-informed optimization framework for designing materials in advanced manufacturing applications. This dissertation demonstrates how these components interact and inform each other in the context of designing acrylate-based UV-curable inks for additive manufacturing processes. Specifically, it illustrates how multi-physics modeling provides a virtual environment, and how Evolutionary Strategies and Bayesian Optimization accelerate the search for optimal input parameters within experimentally determined constraints. This comprehensive approach not only offers a more efficient method for addressing formulation strategies in additive manufacturing but also paves the way for general material development across various industrial applications.

To the memory of my sister, Tiffany Howell.

Contents

Contents	ii
List of Figures	iv
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Simulation-Informed Optimization Framework	1
1.3 Outline of Work	2
2 Background and Mathematical Tools	4
2.1 Additive Manufacturing	4
2.2 Numerical Methods for Partial Differential Equations	8
2.3 Machine Learning and Surrogate Modeling	23
2.4 Numerical Optimization	33
3 UV-curable Inks in Additive Manufacturing	43
3.1 Introduction	43
3.2 Experimental Methods & Materials	44
3.3 Results & Discussion	47
3.4 Applications	60
3.5 Conclusion	62
4 Modeling and Simulation of UV-Curable Materials	63
4.1 Introduction	63
4.2 Thermo-Chemical Continuum Formulation	65
4.3 Numerical Example: UV kinetics for UGAP	85
4.4 Summary and Extensions	100
5 Computational Optimization for Material Design	102
5.1 Introduction	102
5.2 Materials Optimization Strategy	103

5.3	No Free Lunch	105
5.4	Applications to Advanced Manufacturing	106
5.5	Designing the Multi-Objective Function	107
5.6	Bayesian Optimization	111
5.7	Covariance Matrix Adaptation - Evolutionary Strategy	125
5.8	Genetic Algorithms	132
5.9	Discussion and Comparison	139
5.10	Conclusions	143
6	Summary and Future Work	145
6.1	Summary	145
6.2	Future Work	146
7	Appendix	147
7.1	Step-Wise Reactions	147
7.2	Numerical Stability of Fixed-Point Iteration	148
7.3	Numerical Examples	150
7.4	Effects of Variable Mesh Size on Optimization for Bayesian Optimization . .	158
7.5	Graveyard of Unfinished Modeling and Simulation	172
	References	185

List of Figures

2.1	General categories of AM [9].	4
2.2	Schematic illustration of continuous direct-ink writing of a filament [20].	6
2.3	Linear approximation of a shear-thinning fluid.	7
2.4	An example 3D finite difference grid where each node is equally spaced from neighboring nodes by some amount $\Delta x = \frac{L}{M-1}$, where L is the length of the domain across a single axis, and M are the total number of nodes along that same axis.	9
2.5	Method of lines interpretation. $U_i(t)$ is the solution along the line forward in time at the grid point x_i [21].	19
2.6	A comparison of the stability regions for 1) Forward Euler, 2) Backward Euler and 3) Trapezoidal where the white area corresponds to stability (the absolute stability region) and the gray area to instability [22]	22
2.7	A typical visual representation of fully-connected 4-layer neural network or multi-layer perception (MLP) where $\mathbf{x}_i$ is input vector to the next layer, $\mathbf{W}_i$ are the matrix weights, and n_i are the number of neurons in a layer.	23
2.8	Comparison of different hyper-parameter values of l where the shaded blue areas represent the uncertainty bands on the scale of standard deviation.	29
2.9	Comparison of different hyper-parameter values of l where the shaded blue areas represent the uncertainty bands on the scale of standard deviation.	31
3.1	Custom on-demand UV curing mount with three light guides focused to a single spot size under the printing nozzle.	46
3.2	Spanning tests of 70wt% UGAP-NaCl with variable amounts of acrylate and photoinitiator (PI) overlaying a model of a spanning block (varied wt% with respect to resin without solids). Diagram at the top of the graph represents the spanning test block with increasing void gaps in mm.	49
3.3	Tensile test of Type IV dog bones printed with different toolpaths. A & B) The stress-strain curves and corresponding tensile moduli of specimens printed with 0-, 45-, and 90-degree orientations. Tensile samples failed slowly in multiple areas, as indicated by the step-wise failure in (A).	50
3.4	Printing Al_2O_3 filled UGAP. A) Printing a lattice with UGAP- Al_2O_3 without UV light. B) Printing a lattice with UGAP- Al_2O_3 with UV light and subsequently thermally removing UGAP and sintering Al_2O_3 particles.	50

3.5	Schematic of possible printing combinations with UGAP DIW printing. A) Layer-by-layer complexity via intermediate UV-cure, B) Millimeter scale helix, C) 3D Benchie, D) Gyroid printed using UGAP inks filled with fumed silica (UGAP-FS), E) Porosity on variable length scales, F) Designed macro-porosity, G) SEM images of micro-porosity produced by leaching NaCl particles, H) Hierarchical combination of macro and micro-porosity, I) multi-material printing, J) a cylinder co-printed, UGAP-NaCl and 10wt% UGAP-FS, and K) UGAP-FS support structure for silicone ink. L) Infill fabrication for: M) printing an outer mold and filling to full density, and N) Multiple methods of printing an outer shell with various internal structure.	51
3.6	X-ray CT of printed samples leached of salt particles with lattice densities of A) 40%, B) 60%, and C) 80% indicating full removal of salt particles and the resulting hierarchical porosity.	52
3.7	A) Yield stress comparison of UGAP resins filled with 70wt% NaCl and 15wt% fumed silica. B) 1 cm x 1cm towers printed as tall as possible with UGAP-NaCl and UGAP-FS with and without UV light. C) UV-rheology storage modulus for each UGAP ink with increasing light intensity. D) Hollow cones with UGAP-NaCl and 15wt% UGAP-FS demonstrating overhangs. E) Spanning tests measuring sag distance for UGAP inks. F) Inverted pyramids with 15° and 45° side angles printed with UGAP-NaCl and UGAP-FS. G) Printed lattices withstanding 25% strain. H) Static 5 N loading on UGAP-NaCl and UGAP-pore lattices with same filament spacing, SEM images of UGAP-NaCl and UGAP-pore samples.	54
3.8	Direct visual comparison of 1 cm x 1 cm towers printed to a height of 8.7 cm with 15wt% UGAP-FS and UGAP-NaCl.	55
3.9	Time to gelation at a consistent $0.6mW/cm^2$ for varying parallel plate gap thicknesses where gelation is defined when $G' = G''$	56
3.10	Gelation with time for a $140\mu m$ sample continuously exposed to $0.6mW/cm^2$ UV light vs samples exposed to only 10 seconds of UV light, indicating continued dark curing.	56
3.11	Fourier Transform (FT) IR of UGAP resin samples (i) uncured, (ii) UV-only cured, (iii) heat-only cured at 80 °C, and (iv) UV, then heat cure. The acrylate subplot indicates a conversion of 81% (as measured from the maximum points within the bounds of 797 and 823 of the uncured and UV+80 °C data.	57
3.12	A) Just the urethane components without the minor acrylate cross-linker is cured with heat to confirm a single elastomeric glass transition temperature. B) Adding in the minor acrylate component and dual curing with UV and heat maintains a single glass transition temperature while maintaining a dominant elastomeric property.	58
3.13	Iterative compression cycles of 1 cm x 1 cm x 0.5 cm 60% lattice density structures printed with A) 70wt% UGAP-NaCl and B) leached NaCl pores.	59

3.14	A) Printed gripper holding ball in “programmed” configuration and releasing to the original configuration with heating. B) Compressive mechanical properties of inks in multimaterial cylinder UGAP-pore with lattice densities 40, 60 & 80% and UGAP-FS segments. C) Multimaterial cylinder in the programmed state actuating to the original state. D) Printed porous structures facilitating the growth of human lung fibroblasts. Fluorescence microscope image showing the microporous background (blue), fibroblast cell membrane stain (pink), and the actin stain (green).	61
4.1	The step-wise reaction mechanism for radical initiated acrylate reactions: A) Dissociation, B) Initiation, C) Propagation, D) Termination.	68
4.2	Taki et al. reports measurement techniques to obtain kinetic constants required for multi-physics modeling and simulation (figure used from [64]).	85
4.3	Comparison of reaction coefficients: (a) Propagation reaction coefficient k_p , and (b) Termination reaction coefficient k_t .	89
4.4	The rate of diffusion of monomers over time.	90
4.5	Temperature and concentration evolution over 30s simulations.	92
4.6	Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations.	94
4.7	Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations for a sweep of ambient temperatures .	95
4.8	Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations for a sweep of UV intensities .	96
4.9	Numerical stability for temperature and concentration profiles over a temporal/spatial discretization refinement.	97
4.10	Dynamics of the first half second: Columns 1-5 show the time evolution of the concentration of PI, the primary radical ($PI\cdot$), the propagating radical ($M\cdot$), and the monomer (M), each with units mol/m^3 . The dynamics of the first 0.5 seconds are dominated by the rapid conversion of PI (column 1) to the primary radical (column 2). During this rapid reaction, heat is generated from within and is convected from the surface to the environment, as indicated by the temperature distribution in column 5.	98
4.11	Dynamics of next 29s: Columns 1-5 show the time evolution of the concentration of PI, the primary radical ($PI\cdot$), the propagating radical ($M\cdot$), and the monomer (M), each with units mol/m^3 . The dynamics from 1-30s are dominated by the slower conversion of primary radical (column 2) to the propagating radical (column 3) to consumption of monomer (column 4). During this slower reaction phase, the convection to the surrounding environment provides a more stable temperature distribution.	99
5.1	Example space of potential algorithms parameterized by the assumptions that can be made about the underlying objective and the computational cost.	106

5.2	Pairwise Pareto fronts for objective pair in which the relative weights for each pairwise objectives are varied.	109
5.3	Maximization of the negative log likelihood over 1000 generations.	115
5.4	Convergence of squared exponential kernel hyperparameters over 1000 generations.	116
5.5	Hyper-parameter Convergence of the SEK.	118
5.6	Hyper-parameter Convergence of the RQK.	119
5.7	Hyper-parameter Convergence of the LPK.	119
5.8	Material parameter convergence of for SEK, RQK, LPK functions.	120
5.9	Process parameter convergence of for SEK, RQK, LPK functions.	121
5.10	Overall convergence of BO for SEK, RQK and LPK.	123
5.11	Convergence of individual objectives.	124
5.12	Material parameter convergence for CMA-ES.	128
5.13	Process parameter convergence for CMA-ES.	129
5.14	Optimization convergence plots for CMA-ES: The candidate solution performance over each optimization iteration for the best parent, average parents, and overall average entire population.	131
5.15	Optimization convergence plots for CMA-ES: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI $\cdot$), propagating radical (M $\cdot$) and starting monomer (M) concentrations.	132
5.16	Material parameter convergence for GA.	135
5.17	Process parameter convergence:	136
5.18	Optimization convergence plots for GA: The candidate solution performance over each optimization iteration for the best parent, average parents, and overall average entire population.	138
5.19	Optimization convergence plots: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI $\cdot$), propagating radical (M $\cdot$) and starting monomer (M) concentrations.	139
5.20	Performance comparison between all optimization approaches.	141
5.21	Time complexity comparison of optimization approaches over increasingly fine meshes: BOpt, CMA-ES, GA.	142
7.1	The step-wise reaction mechanism for radical initiated acrylate reactions: A) Dissociation, B) Initiation, C) Propagation, D) Termination.	147
7.2	The step-wise reaction mechanism for heat initiated urethane reactions: A) hydroxyl group attacks carbon. electrons in double bond of carbon moves to nitrogen, B) hydrogen on oxygen moves to free electrons on nitrogen group, C) urethane segment complete.	148
7.3	A benchmark comparison between gradient descent, gradient descent + momentum, and ADaptive Momentum (ADAM) gradient descent on the non-convex Beale Function	158

7.4	Optimization convergence plots of the SEK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.	159
7.5	Optimization convergence plots of the SEK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI $\cdot$), propagating radical (M $\cdot$) and starting monomer (M) concentrations. . . .	160
7.6	Process parameter convergence of the SEK:	161
7.7	Material parameter convergence of the SEK:	162
7.8	Hyper-parameter Convergence of the RQK: As more data is collected after each optimization iteration, the hyperparameters selection begins to stabilize for each mesh as the Gaussian Process <i>learns</i> the parameter space.	163
7.9	Optimization convergence plots for RQK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.	164
7.10	Optimization convergence plots for RQK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI $\cdot$), propagating radical (M $\cdot$) and starting monomer (M) concentrations. . . .	165
7.11	Process parameter convergence for RQK.	166
7.12	Material parameter convergence for RQK.	167
7.13	Hyper-parameter Convergence of the LPK: As more data is collected after each optimization iteration, the hyperparameters selection begins to stabilize for each mesh as the Gaussian Process <i>learns</i> the parameter space.	168
7.14	Optimization convergence plots for LPK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.	169
7.15	Optimization convergence plots for LPK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI $\cdot$), propagating radical (M $\cdot$) and starting monomer (M) concentrations. . . .	170
7.16	Process parameter convergence for LPK.	171
7.17	Material parameter convergence.	172
7.18	Typical DIW system	173
7.19	Initial results for discrete element modeling of a complex fluid being dispensed from a nozzle on to a build plate.	184

List of Tables

3.1	Explicit formulation for urethane grafted acrylate polymer (UGAP) resin in fractional wt%.	48
4.1	Summary of multi-physics model for transient chemical-thermal coupled fields.	75
4.2	Reaction-diffusion parameters	86
4.3	Photoinitiator properties	86
4.4	Thermal properties of the material	87
4.5	Parameter values	87
4.6	Physical parameters of constituents	87
5.1	Optimization parameters in materials development	104
5.2	Performance metrics in materials optimization	104

Acknowledgments

First and foremost, I would like to thank my beloved life partner, Yoonjoo Lee, for her constant love and encouragement. You made the highs of this journey greater than they ever could have been on my own, and you helped me survive the darkest lows. You brought balance to my imbalance and stability to my turbulence. Thank you for your companionship during the most fulfilling and beautiful period of my life. I will forever cherish the memories from this chapter of our lives.

I feel privileged to have studied under the direction and advisement of Professor Tarek Zohdi and I will always be grateful for the years of his mentorship and collaboration. His philosophy, drive, intellectual curiosity, vision, efficiency, time management and sincerity will continue to inspire me throughout the rest of my life as I strive to emulate his exemplary attributes. Thank you for instilling an appreciation and love for physics, simulation, computing and teaching.

Furthermore, I would like to extend my appreciation to my professional colleagues at Lawrence Livermore National Laboratory, Google X, and Apple: Eric Bukovsky, Max Muri-
aldo, Antonio Papania-Davis, Jonas Einarsson, Ray Nagatani, Jiaqi Jiang, and Ta Tang, each of which provided their unique expertise, insights and mentorship and shaped the way I operate in research, engineering, and business during this transformative period of my life.

I am also grateful to the members of the Multi-physics Simulation and Optimization Laboratory, Donghoon Kim, Youngkyu Kim, Payton Goodrich, Zachary Yun, Nicolas Cast-
trillon, Roger Isied, Avery Rock, Omar Betancourt, Emre Mengi, Maya Horii, Shao-Yi Yu, Aidan Gould, Abdullah Alrashdan, Tommy Hosmer, and Mostafa Sedkey, each of whom has generously offered their time and expertise to discuss and explore my research ideas and questions.

As an undergraduate, Professor John Hedengren's teaching and influence provided a career-altering path and a new-found love for applied math and optimization. Your courses on control theory changed my life and helped me become the engineer that I am today.

To Taylor Howell, thank you for being a once-in-a-lifetime, world-class friend.

Finally, I would like to acknowledge my parents, Matt and Hillary Howell, for their unconditional love and support throughout my academic journey. Their unwavering commitment to providing a challenging environment to explore practical and abstract ideas has been instrumental in shaping my intellectual curiosity and propensity for critical analysis. Without their influence throughout my younger years, I likely would have left with my Master's degree, or worse, pursued an MBA.

“If there ever was a period during which Robert was happy, those were his California years in the 1930s, I would think”

— Abraham Pais on J. R. Oppenheimer's time at U.C. Berkeley

Chapter 1

Introduction

“Civilization advances by extending the number of important operations which we can perform without thinking of them.”

- Alfred North Whitehead

1.1 Motivation

Modern materials development and manufacturing often require meticulous design and formulation strategies for chemical-based products and functional materials [1, 2]. Industrial standards for formulation typically involve brute-force grid search or design of experiments (DOE) as the guiding principle for determining the design of specific products. These approaches are often costly in terms of engineering time, required materials, and environmental impact. Moreover, they may fail to fully explore the complex parameter spaces inherent in advanced materials design.

The advent of high-performance computing and advanced simulation techniques offers an opportunity to revolutionize this process [3, 4, 5, 6, 7]. By leveraging computational models and optimization algorithms, we can potentially accelerate the materials discovery process, reduce resource consumption, and uncover novel solutions that might be overlooked by traditional methods. This dissertation presents a comprehensive, simulation-informed framework for materials optimization, with a specific focus on UV-curable inks used in additive manufacturing. By integrating experimental data, multi-physics modeling, and advanced optimization techniques, I aim to demonstrate a more efficient and effective approach to materials design that can be applied across various industrial applications.

1.2 Simulation-Informed Optimization Framework

The framework proposed in this dissertation consists of three main components:

Constraint Identification via Experimentation

Experimental work forms the foundation of this approach, providing crucial data for model validation and establishing practical constraints for the optimization process. Chapter 3 details our experimental methods, focusing on UV-curable ink systems. These experiments help identify key parameters such as particle volume fraction, particle diameter, UV exposure time, UV intensity, and ambient temperature, which significantly impact material performance. The experimental results not only inform our modeling efforts but also set realistic bounds for the optimization algorithms.

Multi-physics Modeling and Simulation

Building on the experimental foundation, Chapter 4 presents a detailed multi-physics model of the UV-curing process. This model incorporates various physical phenomena, including reaction kinetics, heat transfer, and light propagation. By developing a high-fidelity simulation environment, I create a virtual laboratory for exploring material behavior under various conditions. This approach allows for rapid iteration and testing of hypotheses without the resource constraints of physical experiments.

Machine Learning and Optimization

Chapter 5 explores the application of advanced optimization techniques to navigate the complex parameter space defined by our multi-physics model. I investigate three primary optimization methods: Bayesian Optimization (BO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Genetic Algorithms (GA). These methods are compared in terms of their effectiveness in finding optimal material formulations, computational efficiency, and ease of implementation. These results demonstrate significant improvements over traditional DOE approaches, highlighting the potential of these computational methods in materials design.

1.3 Outline of Work

This dissertation is structured as follows:

- Chapter 2 provides a comprehensive literature review, situating our work within the broader context of materials science, additive manufacturing, and computational optimization.
- Chapter 3 details our experimental methods and results, establishing the practical constraints and validation data for our computational approach. This section is adapted from my work performed with my colleagues at Lawrence Livermore National Laboratory [8].

- Chapter 4 presents the development and implementation of our multi-physics model, creating a virtual environment for materials exploration.
- Chapter 5 explores the application of various optimization techniques to our model, comparing their performance and discussing their relative merits.
- Chapter 6 provides the overarching commentary, discussing the implications of this work for industrial applications and identifying areas for future research.

By integrating these components, I aim to demonstrate a new approach to materials design that balances theoretical rigor with practical applicability. This work not only advances the specific field of UV-curable ink formulation but also provides a template for applying similar methods across a wide range of materials science and engineering challenges.

Chapter 2

Background and Mathematical Tools

“I was born not knowing and have had only a little time to change that here and there.”

- Richard P. Feynman

2.1 Additive Manufacturing

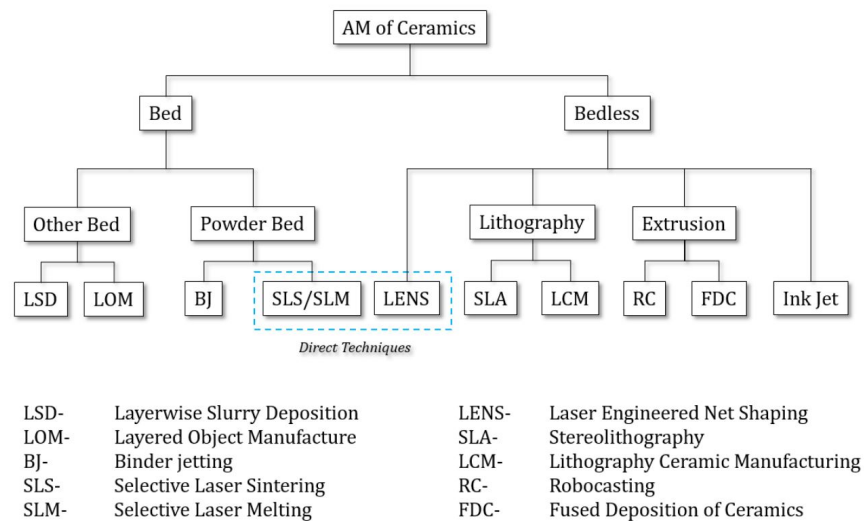


Figure 2.1: General categories of AM [9].

In recent years, significant advances in additive manufacturing (AM) have expanded the opportunities for industry, research & development, and even consumers to digitally design and fabricate complex objects from a variety of different materials [10, 11, 12]. This

emergent technology has the potential to be a disruptive player in traditional industrial settings by enabling cheaper and safer methods for large and small-scale manufacturing. However, advances in hardware, printing quality, and overall throughput are required for AM to be used at industrial scales [13, 14]. Within AM, there are various techniques that enable different types of materials to be built in layer-by-layer fashion. Although there are various ways to categorize these techniques, one such way has been presented by Feilden [9] and is shown below in Figure 2.1. The objectives of this dissertation will address common manufacturing issues that arise in the category of *extrusion-based printing*.

Extrusion-based printing

Extrusion-based printing is an attractive method for users due to its relatively inexpensive processes, minimal moving parts, and easy cleanup. Fused-deposition modeling (FDM) is by far the most common method of AM amongst hobbyist and consumers. These processes utilize filaments made from thermoplastics and are extruded at temperatures just above the melting point. Once the filament is deposited onto the substrate, rapid heat transfer allows the thermoplastic to return to its solid state and retain its deposited shape. This particular field has been well studied and impressively complex objects can be printed utilizing this method. One drawback of FDM printing is the availability of a wide range of printable materials. In this case, Direct-Ink-Write (DIW) printing has large potential since it facilitates a wider range of polymeric materials to be extruded from a nozzle onto substrate and subsequently cured. Over the years, the general form of DIW has been called many things including, but not limited to, direct-write fabrication, robot assisted deposition, robocasting (RC) [9], rapid prototyping (RP) [15], freeform fabrication [16], freeze-form extrusion fabrication (FEF) [17], ceramic on-demand extrusion (CODE) [18], and solid freeform (SFF) [19].

DIW printing has gained substantial attention in recent years due to its potential for diverse applications. In DIW, shear-thinning ‘inks’ are extruded from a reservoir through a nozzle onto a substrate, forming various shapes over successive layers (refer to Figure 2.2). For optimal printing, the ink requires a sufficiently high yield stress enabling extrusion under an applied force and rapid recovery to a solid state once dispensed. This method integrates a computer-controlled three-axis robotic printer, with configurations varying from X-Y translation build stages to ink dispensers translating in the Z-axis. Notably, recent advancements in computer-controlled stages and extrusion systems have shifted the limitations of DIW printing to the properties of the extruded ink.

Although DIW printing greatly expands the materials that can be used for fabricating 3D objects, the various combinations of prepolymers and solid filler interact in nonlinear ways, common with most non-Newtonian fluids. This requires meticulous formulation to achieve specific rheological properties conducive to extrusion, and manual estimation of printing parameters (e.g., extrusion rate, stage speed, z-height). Typically these parameters are chosen by the chemists, materials scientists and engineers and are modified via design of experiments. Formulation chemistry in particular is an expensive process in which many

iterations of a material are made in a lab and subsequently tested to obtain desired performance properties. The material designer must balance these properties to achieve the desired performance requirements of the final object, while constrained to the rheological requirements of the DIW method.

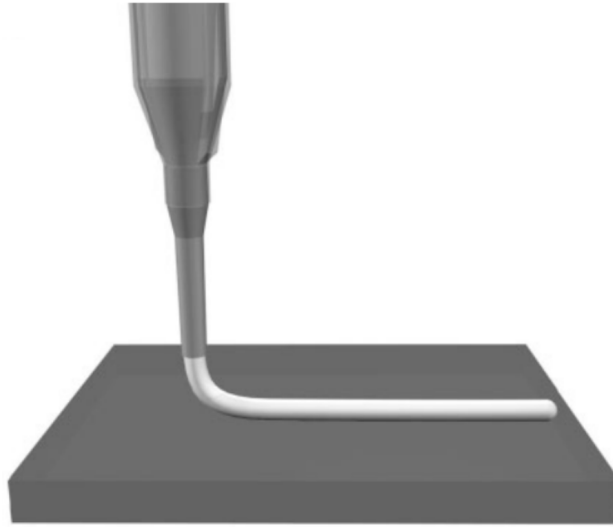


Figure 2.2: Schematic illustration of continuous direct-ink writing of a filament [20].

Rheology

Each component of the ink interacts with other components to produce the overall flow properties. When a shear force is applied to a fluid in a pipe, internal resistance to flow between the constituents of the ink results in a velocity profile, where the maximum velocity of fluid is at the center of the pipe ($r = 0$) and the velocity of the fluid at the wall ($r = R$) is zero. This velocity profile is considered as the shear rate within an annular structure and is indicated in Figure 2.3 by the blue arrows. For Newtonian fluids, this shear rate is linearly related to shear stress (see Equation 2.1). This constant of proportionality is the viscosity of the fluid and dictates the magnitude of the internal stress based on the velocity profile.

$$\tau_{shear} = \mu \dot{\gamma} = \mu \frac{du}{dr} \quad (2.1)$$

For non-Newtonian fluids, the viscosity is a function of the shear rate and is not constant. For shear-thinning fluids, the viscosity nonlinearly decreases with increasing shear rate. Although an exact model describing the relationship between shear rate and shear stress for any formulation is not tractable, shear-thinning fluids often do contain a linear segment within an applicable range of the shear rates (see Figure 2.3). These linear approximations give rise to the power law approaches such as the Herschel-Bulkley shown in Equation 2.2:

$$\dot{\gamma} = 0, \quad \text{if } \tau < \tau_0 \quad (2.2)$$

$$\tau = \tau_0 + k\dot{\gamma}^n \quad (2.3)$$

where τ is the shear stress, τ_0 is the yield stress, k is the consistency index, $\dot{\gamma}$ is the shear rate and n the flow index.

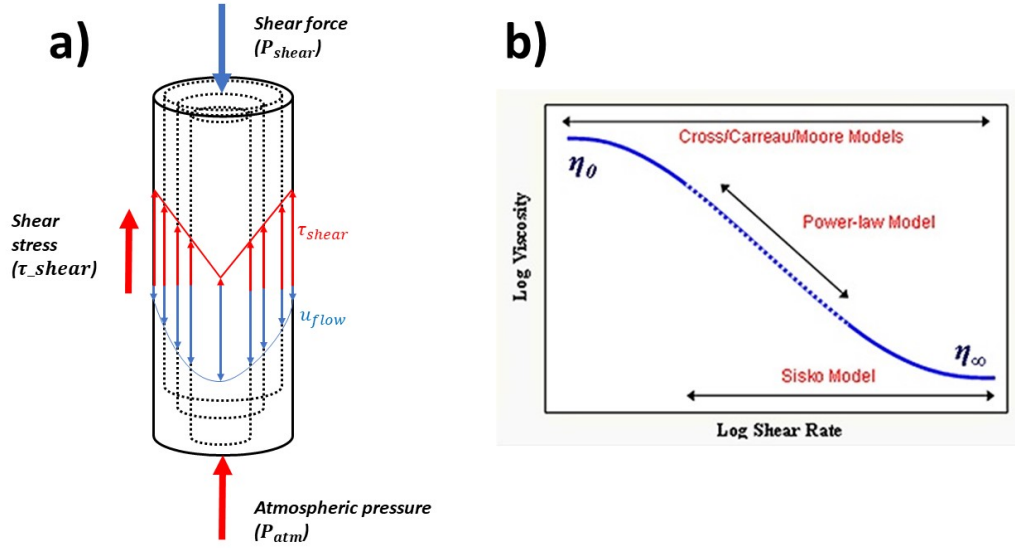


Figure 2.3: Linear approximation of a shear-thinning fluid.

Although flow rates can be estimated or measured for these dispensing techniques, variation in ink preparation and the resulting complex rheological properties make precision fluid control difficult to manage. The problem is exacerbated when attempting print a large variety of inks. The molecular weight of the polymers, particle loading, particle morphology and size distribution, particle-polymer interaction, intermolecular interactions, geometry of system, temperature, imperfect processing, phase separation, and gas entrainment all contribute to the non-Newtonian flow, often resulting unpredictable properties pertaining to printing. Over the years, typical rheological measurements – yield stress, complex viscosity, storage and loss modulus, and tan delta to name a few – have been reported in the literature as indicators of what constitutes a “good ink”.

For most applications, the fluid and the boundary of system is considered not immobile (also referred to as the no-slip condition), and fluid layers that are progressively closer to the center of the configuration can be considered as concentric circles with thickness dx , move at an increasing viscosity resulting in a velocity profile (see Figure 2.3a). The resistance to flow between each of these concentric layers is determined by the constituents of fluid, which

maybe complex due to the interactions of many components at many different scales. For non-Newtonian flow through a nozzle, the force being applied to the fluid is called the shear force and is in the direction of flow. The resulting deformation within the fluid is called the shear strain and is given by $e = \frac{L_f - L_i}{L_i}$. On the molecular level, intermolecular forces and entanglement of polymer resist flow. As the material strains, the intermolecular forces are broken allowing the polymers and filler to flow past each other. Fluids that exhibit this phenomena are referred to "shear-thinning fluids".

In general, the quality of a DIW printed object is determined by how well the fluid behaves during the build. In Chapter 3, I present an approach to relax the harsh rheological constraints required for extrusion-based printing processes.

2.2 Numerical Methods for Partial Differential Equations

Parabolic partial differential equations

Parabolic partial differential equations (PDEs) play a fundamental role in modeling various phenomena, ranging *from black holes to Black-Scholes*. More specifically, the form of parabolic PDEs elegantly captures the dynamics of a large range processes such as heat and chemical diffusion, Schrödinger's equation, and option pricing in financial mathematics. This class of differential equations differ from ordinary differential equations (ODEs) in that the modeled quantities vary in space and time. One of the most commonly used numerical methods for solving such equations is the finite difference method (FDM). FDM is a simple, efficient, and robust technique that discretizes the PDE on a grid and approximates the derivatives using finite differences [21]. This method has been extensively studied and applied in various fields of science and engineering for over a century. In this section of the dissertation, I discuss the theoretical foundations and practical implementation of FDM for parabolic PDEs.

$$\underbrace{u_t}_{\text{time differential}} - \underbrace{\nabla_x \cdot (\kappa \nabla_x u)}_{\text{diffusion of quantity } u} = \underbrace{f_m}_{\text{source terms}} \quad (2.4)$$

The general form of the classic heat equation is shown in [Eqn. 2.4](#) where u models temperature or chemical concentration at a given point in space and time, κ is the heat capacity or diffusion coefficient (possibly temporally and/or spatially dependent), and f_m , $\forall m = 1, \dots, M$ nodes are the source term (also possibly temporally and/or spatially dependent). To solve the heat equation numerically, it is necessary to specify both initial conditions and boundary conditions. The initial conditions specify the temperature distribution within the domain at time zero, while the boundary conditions specify the behavior of the solution at the boundaries of the domain. Dirichlet boundary conditions set the temperature on the boundary of the domain, while Neumann boundary conditions define the flux through that boundary.

Without these boundary conditions, the heat equation is ill-defined with an infinite number of possible solutions that satisfy the PDE. Therefore, the appropriate choice of boundary conditions is crucial for obtaining physically meaningful solutions to the heat equation.

Concretely, a well-defined parabolic PDE consists of:

1. PDE: $u_t - \nabla_x \cdot (\kappa \nabla_x u) = f, \quad x \in \Omega, \quad t > 0$
2. Initial condition: $u(x, t = 0) = \eta(x), \quad x \in \Omega$
3. Boundary conditions: $\Gamma = \partial\Omega$
 - Dirichlet boundary condition: $u = u_D$ on $\Gamma \times (0, T)$ which directly prescribes the temperature or chemical concentration.
 - Neumann boundary condition: $(\kappa \nabla_x u) \cdot \mathbf{n} = g_M$ on $\Gamma \times (0, T)$ which prescribes the heat or diffusion flux across the boundary.

Finite difference grid

For simple geometries, the finite difference grid provides a straight-forward approach to discretizing the domain into a finite number of points or nodes. Each node represents a discrete location where the solution is evaluated, and the solution at each node is approximated using a finite difference approximation (example in Fig. 2.4).

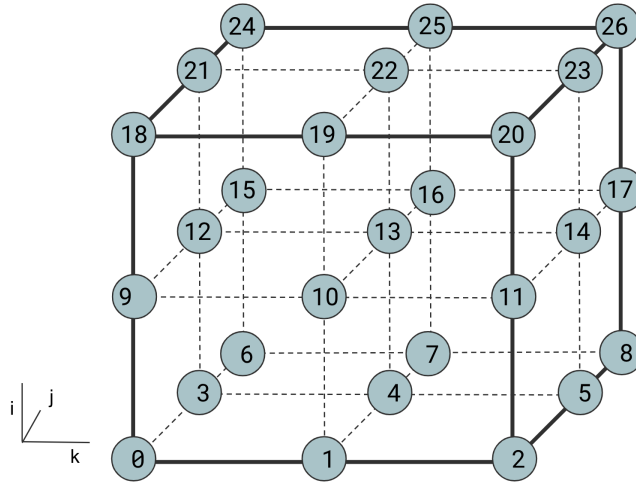


Figure 2.4: An example 3D finite difference grid where each node is equally spaced from neighboring nodes by some amount $\Delta x = \frac{L}{M-1}$, where L is the length of the domain across a single axis, and M are the total number of nodes along that same axis.

The accuracy of the FDM depends on the *size of the grid* and the *order of the finite difference approximation* used. In this way, the spatial domain of the PDE is transformed into a discrete set of equations that can be solved numerically using linear algebra techniques. The choice of grid size and finite difference approximation is critical in achieving a stable and accurate numerical solution.

Finite difference methods

Forward Difference in Time, Centered in Space (FTCS)

We seek to find an approximate solution $U_m^n \approx u(x_m, t_n)$ by approximating the diffusion term in Eqn. 2.4 which will enable us to “march forward in time”, computing $U_m^{n+1} \forall m = 1, \dots, M$ as a function of U_m^n at the previous time step.

FTCS seeks to find an approximate solution $U_m^n \approx u(x_m, t_n)$ by spatially discretizing the domain into a finite number of equally spaced grid points and approximates the spatial derivative at each point using a centered difference formula. The temporal derivative is approximated using a forward difference formula, and the solution at each grid point is updated at each time step based on the values at its neighboring points in space and the previous time step.

To obtain the FTCS approximation for $x \in \mathbb{R}^1$, we first consider a second-order center-difference approximation [21] of the Laplacian of the flux:

$$\nabla_x \cdot \mathbf{q} = \frac{\partial q}{\partial x} \approx \frac{q(x_1 + h/2) - q(x_1 - h/2)}{h} \quad (2.5)$$

where $\mathbf{q} = \kappa \nabla_x u$ and $h = \Delta x$. Assuming κ is constant in space, each partial flux can be further approximated by applying a successive first-order one-sided approximation:

$$q(x + h/2) \approx \kappa \frac{u(x + h) - u(x)}{h} \quad (2.6)$$

$$q(x - h/2) \approx \kappa \frac{u(x) - u(x - h)}{h} \quad (2.7)$$

yielding the centered-in-space approximation:

$$\frac{\partial}{\partial x} \left(\kappa \frac{\partial u}{\partial x} \right) = \kappa u_{xx} \approx \kappa \frac{u(x - h) - 2u(x) + u(x + h)}{h^2}. \quad (2.8)$$

Finally, the time derivative on the left-hand side of Eqn. 2.4 is approximated using a first-order one-sided approximation

$$U_m^{n+1} = U_m^n + \frac{k\kappa}{h^2} (U_{m-1}^n - 2U_m^n + U_{m+1}^n) + \frac{k}{h^2} f(U_m^n, t_n) \quad (2.9)$$

where $k = \Delta t$. This method is more commonly referred to as the *Forward Euler* time stepping scheme. The FTCS method is considered to be a first-order accurate in time, and

second-order accurate in space. Since the right-hand side of Eqn. 2.9 is only dependent on U_m^n at the previous time step (all of which are known), this scheme is considered to be in an *explicit method* since U_m^{n+1} can be explicitly computed.

Trapezoidal Method

Like the FTCS method, the trapezoidal method also employs a second-order center-difference approximation for the spatial discretization of the diffusion term in Eqn. 2.4. However, rather than using an exclusive forward difference scheme, the trapezoidal method employs a weighted average of backward and forward differences in time parameterized by $\phi \in [0, 1]$:

$$\begin{aligned} U_m^{n+1} = & U_m^n \\ & + \phi \frac{k}{h^2} \left(\kappa(U_{m-1}^n - 2U_m^n + U_{m+1}^n) + f(U_m^n, t_n) \right) \\ & + (1 - \phi) \frac{k}{h^2} \left(\underbrace{\kappa(U_{m-1}^{n+1} - 2U_m^{n+1} + U_{m+1}^{n+1})}_{U_m^{n+1} \text{ is unknown}} + f(U_m^{n+1}, t_n) \right). \end{aligned} \quad (2.10)$$

When $\phi = 1/2$, the trapezoidal method reduces to the *Crank-Nicholson* method. The trapezoidal/Crank-Nicholson method is considered to be a second-order accurate in time, and second-order accurate in space. Since the right-hand side of Eqn. 2.9 is only dependent on both U_m^n at the previous time step (all of which are known) and U_m^{n+1} at the next time step (all of which are unknown), this scheme is considered to be in an *implicit method* since U_m^{n+1} must be implicitly computed.

Example: Crank-Nicholson for linear PDEs

Consider the simple parabolic PDE:

$$u_t = u_{xx}. \quad (2.11)$$

The Crank-Nicholson method discretizing Eqn. 2.11 yields:

$$U_m^{n+1} = U_m^n + \frac{k}{2h^2} \left(U_{m-1}^n - 2U_m^n + U_{m+1}^n + U_{m-1}^{n+1} - 2U_m^{n+1} + U_{m+1}^{n+1} \right). \quad (2.12)$$

Example: Crank-Nicholson for linear PDEs (continued...)

In the case in which the PDE is linear, we can move all terms containing the unknown solutions U_m^{n+1} at the text time step to the left-hand side and leave all of the known solutions U_m^n at the previous time step on the right-hand side:

$$-\frac{k}{2h^2}U_{m-1}^{n+1} + (1 + 2\frac{k}{2h^2})U_m^{n+1} - \frac{k}{2h^2}U_{m+1}^{n+1} = \underbrace{\frac{k}{2h^2}U_{m-1}^n + (1 - 2\frac{k}{2h^2})U_m^n - \frac{k}{2h^2}U_{m+1}^n}_{\text{known } \mathbf{b}}. \quad (2.13)$$

Our known solutions at the previous and next time steps can be organized as vectors $\mathbf{U}_m^n, \mathbf{U}_m^{n+1} \in \mathbb{R}^{m_{\text{nodes}}}$ respectively. This means we can re-write the left-hand side of Eqn. 2.13 as system of M equations, and the right-hand side as a single column vector:

$$\underbrace{\begin{bmatrix} (1 + 2\frac{k}{2h^2}) & -\frac{k}{2h^2} & 0 & \dots & 0 \\ -\frac{k}{2h^2} & (1 + 2\frac{k}{2h^2}) & -\frac{k}{2h^2} & \dots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & -\frac{k}{2h^2} & (1 + 2\frac{k}{2h^2}) \\ 0 & \dots & & -\frac{k}{2h^2} & (1 + 2\frac{k}{2h^2}) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ \vdots \\ U_{M-1}^{n+1} \\ U_M^{n+1} \end{bmatrix}}_{\mathbf{x}} = \mathbf{b} \quad (2.14)$$

where

$$\mathbf{b} = \begin{bmatrix} \frac{k}{2h^2}(g_0(t_n) + g_0(t_{n+1})) + (1 - \frac{k}{2h^2})U_1^n + \frac{k}{2h^2}U_2^n \\ \frac{k}{2h^2}U_1^n + (1 - 2\frac{k}{2h^2})U_2^n + \frac{k}{2h^2}U_3^n \\ \vdots \\ \frac{k}{2h^2}U_{M-2}^n + (1 - 2\frac{k}{2h^2})U_{M-1}^n + \frac{k}{2h^2}U_M^n \\ \frac{k}{2h^2}U_{M-1}^n + (1 - 2\frac{k}{2h^2})U_M^n + \frac{k}{2h^2}(g_1(t_n) + g_1(t_{n+1})) \end{bmatrix} \quad (2.15)$$

which reduces to solving $\mathbf{A}\mathbf{x} = \mathbf{b}$. Since $\mathbf{A}$ is tri-diagonal, this problem can be solved as efficiently as an explicit method. A simple approach to solving such equations can be found in Appendix Example section 7.3.

Analysis of Numerical Methods

Local Truncation Error and Order of Accuracy

Numerical methods provide approximate solutions for ODEs/PDEs, and the accuracy of the solutions are dependent on the time discretization $k = \Delta t$ and spatial discretization $h = \Delta x$. After taking a single time step, some amount of error is introduced to the solution,

also referred to as the local truncation error (LTE). This provides a method for analyzing the order of accuracy of a numerical method and how the quickly the error decreases as $h, k \rightarrow 0$. The LTE for the simple ODE $u_t = \lambda u + f(t)$ with an initial value $u(0) = u_0$ for the Backward Euler time discretization method is:

$$\tau_n = \frac{u(t_n) - u(t_{n-1})}{k} - \lambda u(t_n) - f(t_n), \quad \forall n = 1, \dots, N \quad (2.16)$$

Order of accuracy is a measure of how well a finite difference method approximates a solution to a differential equation and describes how quickly the error between the approximate solution and the true solution decreases as $k, h \rightarrow 0$. Intuitively, a method will converge to the true solution at rate proportional to the order of accuracy. For example, a method with second order of accuracy in space will have an error that decreases proportional to the square of the grid spacing. Although we *expect* the approximate solution to approach the true solution such that the error of the method $\|\tau(x, t)\| \rightarrow 0$, for convergence to the true solution to occur, the temporal and spatial discretization must satisfy a the two properties: consistency and stability.

Consistency

As both $h, k \rightarrow 0$, the we *expect* the approximate solution to approach the true solution such that the LTE of the method $\|\tau\| \rightarrow 0$. If this is indeed true, the discretization method is said to be *consistent*, meaning that the discrete equation approximates the same process as the underlying PDE as the temporal and spatial grids become finer at a rate that is at least as fast as $h, k \rightarrow 0$.

More concretely, *consistency* requires:

$$\|\tau\|_\infty = \max_{n=1, \dots, N} |\tau_n| \rightarrow 0 \text{ as } \Delta t \rightarrow 0. \quad (2.17)$$

In the *LTE and order of accuracy for the Crank-Nicolson Method* example, the LTE is shown to be $\mathcal{O}(k^2 + h^2)$. A method is said to be *consistent* since the global order of accuracy will agree with the order of the LTE:

$$U_m^n - u(X, T) = \mathcal{O}(k^2 + h^2). \quad (2.18)$$

To determine whether a method is consistent, we need to analyze LTE. The following are two examples for deriving the LTE and order of accuracy for the *Forward in Time, Centered in Space* and *Crank-Nicolson* methods, and how the error corresponds to k, h .

Example: LTE and order of accuracy for FTCS method

Consider the simple parabolic PDE $u_t = u_{xx}$. The resulting error for a chosen time step k and for a chosen spatial step h for Eqn. 2.9 is defined as:

$$\tau(x, t) = \frac{U_m^{n+1} - U_m^n}{k} - \frac{\kappa}{h^2} (U_{m-1}^n - 2U_m^n + U_{m+1}^n). \quad (2.19)$$

To obtain the LTE, insert the true solution:

$$\tau(x, t) = \frac{u(x, t+k) - u(x, t)}{k} - \frac{\kappa}{h^2} (u(x-h, t) - 2u(x, t) + u(x+h, t)) \quad (2.20)$$

For the first term on the right-hand side, we care about the error relative to k , and for the second term we care about the error relative to h . If we make the assumption that u is smooth, we can expand each term around k , h respectively:

Assuming that u is sufficiently smooth and differentiable, each u term containing increment in k and h can be expanded in a Taylor series:

$$u(x, t+k) = u + ku_t + \frac{1}{2}k^2u_{tt} + \frac{1}{6}k^3u_{ttt} + \mathcal{O}(k^4) \quad (2.21)$$

$$u(x-h, t) = u - hu_x + \frac{1}{2}h^2u_{xx} - \frac{1}{6}h^3u_{xxx} + \mathcal{O}(h^4) \quad (2.22)$$

$$u(x+h, t) = u + hu_x + \frac{1}{2}h^2u_{xx} + \frac{1}{6}h^3u_{xxx} + \mathcal{O}(h^4). \quad (2.23)$$

Inserting each Taylor series expansion from Eqn. 2.21, Eqn. 2.22, and Eqn. 2.23 into Eqn. 2.20 yields:

$$\tau(x, t) = \left(\underbrace{u_t}_{= u_{xx}} + \underbrace{\frac{1}{2}ku_{tt}}_{= u_{txx} = u_{xxx}} + \mathcal{O}(k^2) \right) - \left(u_{xx} + \frac{1}{12}h^2u_{xxxx} + \mathcal{O}(h^3) \right) \quad (2.24)$$

Recall that $u_t = u_{xx}$ and $u_{tt} = u_{xxx}$ allowing for additional terms to simplify:

$$\tau(x, t) = \left(\frac{1}{2}k - \frac{1}{12}h^2 \right) u_{xxxx} = \mathcal{O}(k + h^2). \quad (2.25)$$

This derivation shows that the error for the FTCS method is expected to increase linearly with k and to increase with the power of two with h . This is more commonly stated to be *first-order accurate in time, and second-order accurate in space*.

Example: LTE and order of accuracy for the Crank-Nicolson method

Once again, consider the simple parabolic PDE $u_t = u_{xx}$. The resulting error for a chosen time step k and for a chosen spatial step h is defined as:

$$\tau(x, t) = \frac{U_m^{n+1} - U_m^n}{k} + \frac{1}{2h^2} \left(U_{m-1}^n - 2U_m^n + U_{m+1}^n + U_{m-1}^{n+1} - 2U_m^{n+1} + U_{m+1}^{n+1} \right). \quad (2.26)$$

To obtain the LTE, insert the true solution:

$$\begin{aligned} \tau(x, t) = & \frac{u(x, t+k) - u(x, t)}{k} \\ & + \frac{1}{2h^2} \left(u(x-h, t) - 2u(x, t) + u(x+h, t) \right. \\ & \left. + u(x-h, t+k) - 2u(x, t+k) + u(x+h, t+k) \right). \end{aligned} \quad (2.27)$$

Once again assuming the true solution u is sufficiently smooth and differentiable, the Taylor series expansion for $u(x, t+k)$, $u(x-h, t)$ and $u(x+h, t)$ can be recycled from [Eqn. 2.21](#), [Eqn. 2.22](#), [Eqn. 2.23](#). For $u(x-h, t+k)$ and $u(x+h, t+k)$, a multi-variable Taylor series expansion is required:

$$u(x-h, t+k) = u - hu_x + ku_t + \frac{1}{2}h^2u_{xx} - hku_{tx} + \frac{1}{2}k^2u_{tt} + \mathcal{O}(h^3, k^3, h^2k, hk^2) \quad (2.28)$$

$$u(x+h, t+k) = u + hu_x + ku_t + \frac{1}{2}h^2u_{xx} + hku_{tx} + \frac{1}{2}k^2u_{tt} + \mathcal{O}(h^3, k^3, h^2k, hk^2). \quad (2.29)$$

Inserting each Taylor series expansion from [Equations 2.21-2.23](#) and [Equations 2.28-2.29](#) into [Eqn. 2.27](#) and a lot of tedious cancellation of terms yields:

$$\tau(x, t) = \mathcal{O}(k^2, h^2). \quad (2.30)$$

This derivation shows that the error for the Crank-Nicolson method is expected to increase with the power of two with k and to increase with the power of two with h . This is more commonly stated to be *second-order accurate in time, and second-order accurate in space*.

Stability

Small variations in the initial and boundary conditions may cause sensitive numerical methods to be unstable. These small LTEs may cause the approximate solution to diverge from the true solution over time since each error may compound on itself. For time-dependent ODEs, the concept of *absolute stability* allows us to analyze terms that recursively amplify or dampen the global error after a time step.

Example: Absolute stability of Forward Euler

The notion of *absolute stability* states that an ordinary differential equation (ODE) is absolutely stable under a numerical method if the numerical solution remains bounded as the step size or time step approaches zero, regardless of initial conditions. Consider the following classic ODE:

$$u_t = \lambda u(t) + g(t), \quad u(t_0) = \eta \quad (2.31)$$

and its discretization using Forward Euler time stepping

$$U^{n+1} = (1 + k\lambda)U^n + kg(t) \quad (2.32)$$

where the error

$$k\tau^n = u(t_{n+1}) - u(t_n) - k(\lambda u(t_n) + g(t_n)) \quad (2.33)$$

and its LTE:

$$\tau^n = \frac{1}{2}ku_{tt}(t_n) + \mathcal{O}(k^2). \quad (2.34)$$

To obtain the global error, subtract the following:

$$u(t_{n+1}) = (1 + k\lambda)u(t_n) + kg(t_n) + k\tau^n \quad (2.35)$$

$$U^{n+1} = (1 + k\lambda)U^n + kg(t_n) \quad (2.36)$$

to get

$$E^{n+1} = (1 + k\lambda)E^n - k\tau^n. \quad (2.37)$$

As the numerical method marches in time, the error is recursively scaled by $(1 + k\lambda)$.

Example: Absolute stability of Forward Euler (continued...) For $N+1$ total time steps, the final global error is:

$$E^{N+1} = (1 + k\lambda)^N E^N - k\tau^N. \quad (2.38)$$

where the superscript N above E represents the second to last time step and the superscript N above $(1 + k\lambda)$ obviously represents a power. We expect to see an exponential growth factor any k for which $|1 + k\lambda| > 1$. Therefore, there exists a critical requirement $|1 + k\lambda| \leq 1$ for the Forward Euler method to be absolutely stable. More generally, since λ maybe complex, the region of absolute stability is a disk of radius 1, centered at the point -1 (see Fig. 2.6).

Example: Absolute stability of Trapezoidal

Once again, consider the following classic ODE:

$$u_t = \lambda u(t) + g(t), \quad u(t_0) = \eta \quad (2.39)$$

and its discretization using Trapezoidal time stepping

$$U^{n+1} = U^n + \frac{k}{2}(\lambda U^n + g(t)) + \frac{k}{2}(\lambda U^{n+1} + g(t_{n+1})) \quad (2.40)$$

and solving for U^{n+1} and inserting exact solution yields:

$$U^{n+1} = \frac{(1 + \frac{k\lambda}{2})}{(1 - \frac{k\lambda}{2})} U^n + \frac{\frac{k}{2}(g^n + g^{n+1})}{(1 - \frac{k\lambda}{2})} \quad (2.41)$$

$$U(t_{n+1}) = \frac{(1 + \frac{k\lambda}{2})}{(1 - \frac{k\lambda}{2})} U(t_n) + \frac{\frac{k}{2}(g^n + g^{n+1})}{(1 - \frac{k\lambda}{2})} + k\tau^n. \quad (2.42)$$

Subtracting the two and analyzing for the global error:

$$E^{N+1} = \rho^N E^N - k\tau^N, \quad \text{where } \rho = \frac{(1 + \frac{k\lambda}{2})}{(1 - \frac{k\lambda}{2})} \quad (2.43)$$

Example: Absolute stability of Trapezoidal (continued...)

where the superscript N above E represents the second to last time step and the superscript N above ρ obviously represents a power. For absolute stability, we require $|\rho| < 1$. By induction, it is simple to see that $|\rho| < 1 \forall \lambda k < 0$ since the absolute value of the denominator will always be greater than the absolute value of the numerator. This means that the trapezoidal method's stability region is the entire left-hand plane in Fig. 2.6 since the recursive scaling $\rho^N \forall N$ will drive the global error $E^N \rightarrow 0$.

The stability theory for time-dependent PDEs can be viewed as an extension to the stability theory for time-dependent ODEs through two different approaches: the method of lines (MOL) (see Fig. 2.5) and Von Neumann Analysis.

The MOL approach first discretizes in space alone, resulting in a system of ODEs that can be solved using traditional time-stepping methods. This system of ODEs is known as a semi-discrete method since it is discretized in space but not yet in time. Once system of semi-discrete ODEs has been obtained from the MOL, we can apply our chosen time stepping scheme to obtain fully discrete method in which we can isolate the amplifying or dampening factor that affects stability. For ODE absolute stability, this factor is a scalar value (see *Absolute Stability* example below). However, once the MOL has been applied to a PDE, we obtain a system of equations that can be represented as matrices and vectors (if linear). This means the amplifying/dampening factor can be described by analyzing eigenvalues of matrices that recursively amplify/dampen the solution $\mathbf{u} \in \mathbb{R}^m$.

In general, the MOL approach to stability analysis can be written as the analysis of the equation with the following form:

$$\mathbf{U}^{n+1} = \mathbf{B}(k)\mathbf{U}^n + \mathbf{b}^n(k) \quad (2.44)$$

where $\mathbf{B}(k) \in \mathbb{R}^{M \times M}$, $\mathbf{b}(k) \in \mathbb{R}^M$, on a grid $h = \frac{1}{1+M}$. We seek to analyze the eigenvalues of $\mathbf{B}(k)$ to determine the stability of the chosen numerical methods of the PDE. For the FTCS method $\mathbf{B}(k)$ and $\mathbf{b}(k)$ can be determined by the following:

$$\begin{aligned} \mathbf{U}^{n+1} &= \mathbf{U}^n + k\mathbf{A}(k)\mathbf{U}^n + \mathbf{b}^n(k) \\ &= \underbrace{(\mathbf{I} + k\mathbf{A}(k))}_{\mathbf{B}(k)} \mathbf{U}^n + \mathbf{b}^n(k) \end{aligned} \quad (2.45)$$

where we need to ensure that the eigenvalues $\lambda_i \in \mathcal{S} \forall i = 1, \dots, m$ of $(\mathbf{I} + k\mathbf{A}(k))$ where the stability region $\mathcal{S}$ can be obtained by the *Absolute Stability Analysis* from ODE theory. Since $k\mathbf{A}$ is the term that will dominate in recursive scaling, it is sufficient to only analyze its eigenvalues. Likewise, the Crank-Nicolson method requires $\mathbf{B}(k)$ and $\mathbf{b}(k)$ to take the form:

$$\begin{aligned}
\left(I - \frac{k}{2}\mathbf{A}\right)U^{n+1} &= \left(I - \frac{k}{2}\mathbf{A}\right)U^n \\
U^{n+1} &= \underbrace{\left(I - \frac{k}{2}\mathbf{A}\right)^{-1} \left(I - \frac{k}{2}\mathbf{A}\right)}_{\mathbf{B}(k)} U^n
\end{aligned} \tag{2.46}$$

where a similar analysis of the eigenvalues can provide a insight on stability. The following provides explicit examples for *Stability Analysis of FTCS*.

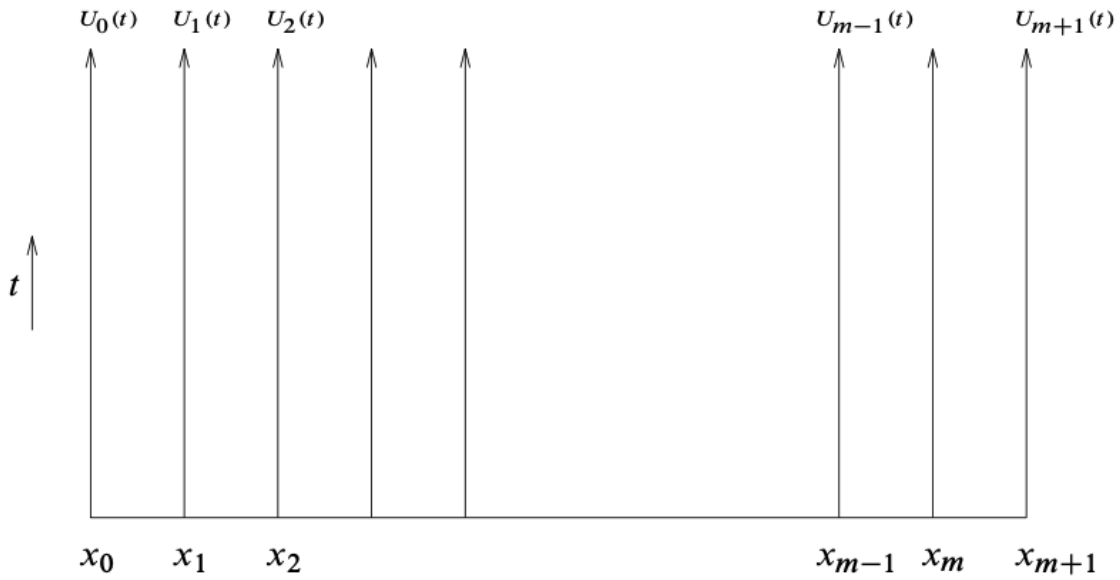


Figure 2.5: Method of lines interpretation. $U_i(t)$ is the solution along the line forward in time at the grid point x_i [21].

Example: Stability analysis of FTCS

As with the analysis of *absolute stability* for an ODE, analysis can be performed on the FTCS method by using first spatially discretizing in space using *Method of Lines* (MOL) to obtain a system of ODEs to interpret the method as a time integration methods of ODEs.

Example: Stability analysis of FTCS continued...

For the simple heat equation $u_t = u_{xx}$, $u(t = 0) = u_0$, the MOL for FTCS yields a semi-discrete ODE:

$$u_t = \frac{1}{h^2}(U_{m-1}^n - 2U_m^n + U_{m+1}^n), \quad m = 1, \dots, M. \quad (2.47)$$

After temporally discretizing, the boundary conditions can be explicitly encoded in the matrix form:

$$\mathbf{U}^{n+1} = (\mathbf{I} + k\mathbf{A})\mathbf{U}^n + k\mathbf{g} \quad (2.48)$$

where

$$\mathbf{A} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}, \quad \mathbf{g} = \frac{1}{h^2} \begin{bmatrix} g_0(t) = U_0(t) \\ 0 \\ \vdots \\ 0 \\ g_1(t) = U_M(t) \end{bmatrix}. \quad (2.49)$$

As in the *absolute stability* example, we can compute the final global error:

$$\mathbf{E}^{N+1} = (\mathbf{I} + k\mathbf{A})^N \mathbf{E}^N - k\boldsymbol{\tau}^N \quad (2.50)$$

where the superscript N above $\mathbf{E}$ represents the second to last time step and the superscript N above $(\mathbf{I} + k\mathbf{A})$ obviously represents a power. More importantly than the inconsistent notation are the eigenvalues of $\mathbf{A}$ which determine whether the global error converges or diverges. For the FTCS method to be stable, we require that $k\lambda_m \in \mathcal{S}$.

For this simple heat equation, the eigenvalues of $\mathbf{A}$ are:

$$\lambda_m = \frac{2}{h^2}(\cos(m\pi h) - 1), \quad \forall m = 1, \dots, M. \quad (2.51)$$

where $h = \frac{1}{m+1}$. The magnitude of $|\lambda_m|$ is largest when $m = M$ and resulting in the eigenvalue:

$$\lambda_M \approx \frac{2}{h^2}(\cos(M\pi h) - 1) = \frac{-4}{h^2}. \quad (2.52)$$

Therefore we require that $-k\frac{4}{h^2} \in \mathcal{S}$, which for Forward Euler (derived in the *Absolute Stability example* corresponds to $-2 \leq -k\frac{4}{h^2} \leq 0$, or $\frac{k}{h^2} \leq \frac{1}{2}$.

Example: Stability analysis of Crank-Nicolson

As with the analysis of *absolute stability* for an ODE, analysis can be performed on the FTCS method by using first spatially discretizing in space using *Method of Lines* (MOL) to obtain a system of ODEs to interpret the method as a time integration methods of ODEs.

For the simple heat equation $u_t = u_{xx}$, $u(t = 0) = u_0$, the MOL for FTCS yields a semi-discrete ODE:

$$u_t = \frac{1}{h^2}(U_{m-1}^n - 2U_m^n + U_{m+1}^n), \quad m = 1, \dots, M. \quad (2.53)$$

An alternative (and simpler) method to analyzing the stability of the Crank-Nicolson method for Eqn. 2.11 is to use the *Von Neumann* approach. This analysis involves making an ansatz (assumption) that the numerical solution has a certain form, and then deriving a condition on the time step that ensures that the numerical solution does not grow without bound.

First, assume that the numerical solution has the form:

$$u_m^n = \xi^n e^{i\omega m h}, \quad (2.54)$$

where h is the spatial step size, k is the time step size, ω is the wave number, m is the spatial index, and n is the time index. Substituting this form into the Crank-Nicolson method and solving for ξ^{n+1} , we obtain:

$$\xi^{n+1} = \frac{1 + \frac{\theta}{2}i\omega h^2}{1 - \frac{\theta}{2}i\omega h^2} \xi^n \quad (2.55)$$

where θ is a parameter that controls the weighting between the backward and forward differences in time, and is given by $\theta = \frac{k}{h^2}$.

The von Neumann stability analysis requires that the amplification factor $G(\omega, k) = \frac{\xi^{n+1}}{\xi^n}$ satisfies $|G(\omega, k)| \leq 1$ for all ω and k , in order to ensure that the numerical solution does not grow without bound.

Solving for the absolute value of the amplification factor, we obtain:

$$|G(\omega, k)| = \left| \frac{1 + \frac{\theta}{2}i\omega h^2}{1 - \frac{\theta}{2}i\omega h^2} \right|, \quad (2.56)$$

which simplifies to:

$$|G(\omega, k)| = \frac{1 + \frac{1}{2}\theta(\omega h)^2}{1 - \frac{1}{2}\theta(\omega h)^2}. \quad (2.57)$$

To ensure that $|G(\omega, k)| \leq 1$ for all ω and k , we require that $\theta \leq 1$. This condition is the stability criterion for the Crank-Nicolson method for the heat equation, and it ensures that the numerical solution does not grow without bound, regardless of the time step size.

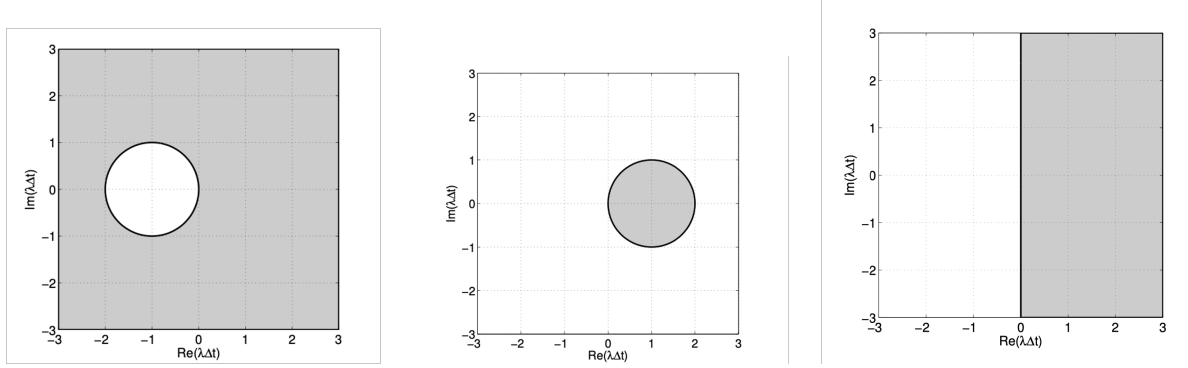


Figure 2.6: A comparison of the stability regions for 1) Forward Euler, 2) Backward Euler and 3) Trapezoidal where the white area corresponds to stability (the absolute stability region) and the gray area to instability [22]

Convergence

The Dahlquist Equivalence Theorem is the primary tool for assessing whether or not a numerical scheme is convergent. Using the concepts of consistency and zero stability alone, we can draw a conclusion about the convergence. To summarize from above, we have the following concepts:

- Consistency: In the limit $k \rightarrow 0$, the method gives a consistent discretization of the ordinary differential equation.
- Zero stability: In the limit $k \rightarrow 0$, the method has no solutions that grow unbounded as $N = T/k \rightarrow \infty$.

The Dahlquist Equivalence Theorem guarantees that a method is consistent and stable is convergent, and also that convergent method is consistent and stable.

Proving consistency and stability directly can be challenging for complex PDEs that reflect coarseness of reality. As such, the numerical approaches presented in this dissertation reflect highly-coupled, nonlinear physical systems that do not lend well to direct analysis. However, the Dahlquist Equivalence Theorem will be used to show overall convergence of the methods used in this work in Chapter 4.

2.3 Machine Learning and Surrogate Modeling

Physical simulations can be costly in terms computational space and time. To address this challenge, researchers often turn to surrogate modeling, also known as reduced-order modeling. This methodology involves employing alternative, lower-fidelity mathematical techniques that significantly reduce computational expenses, albeit at the expense of a controlled decrease in accuracy. Particularly in optimization tasks, where numerous simulations are necessary to determine optimal input parameters maximizing a specified objective, the use of surrogate models becomes invaluable. By constructing a surrogate model capable of handling many efficient queries, the computational time required to converge on new designs for the physical system can be drastically accelerated. In this section, I explore two prevalent approaches to surrogate modeling: deep neural networks and Gaussian processes. It's worth noting that this discussion excludes alternative methods such as low-rank approximation through singular value decomposition and auto-encoding, which may find applicability in related problems not covered in this work.

Multi-Layer Perceptron

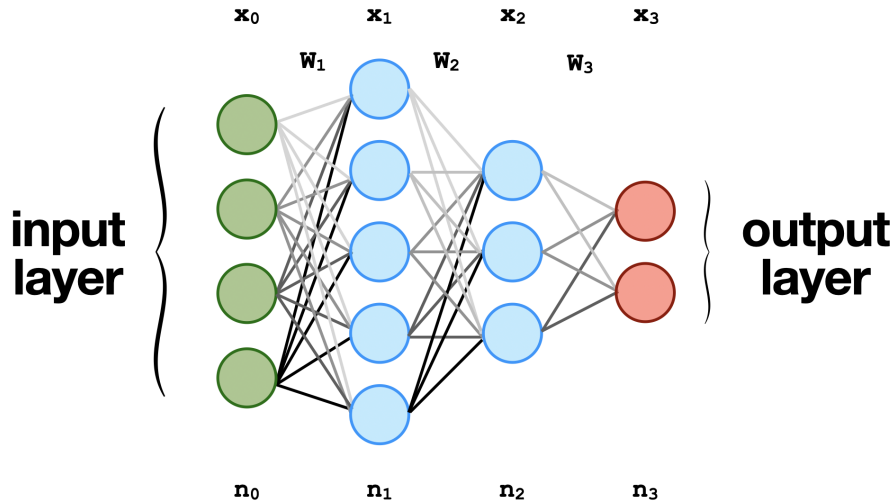


Figure 2.7: A typical visual representation of fully-connected 4-layer neural network or multi-layer perception (MLP) where x_i is input vector to the next layer, W_i are the matrix weights, and n_i are the number of neurons in a layer.

Deep neural networks are a subset of machine learning algorithms that are designed to model the behavior of the human brain. They are composed of interconnected nodes or neurons, which are organized into layers and process information in a parallel and distributed

manner. Neural networks have gained significant attention in recent years due to their ability to solve complex problems in various domains, including image and speech recognition, natural language processing, and predictive analytics. One of the key strengths of neural networks is their ability to approximate any function to arbitrary accuracy, given a sufficiently large number of neurons in the hidden layers. This property, known as universal approximation, has made neural networks an essential tool for data scientists and researchers seeking to develop intelligent systems capable of learning and adapting to changing environments. In this background section, we will provide a comprehensive overview of the fundamentals of neural networks, including their architecture, training algorithms, and applications.

Multi-layer perceptions (MLPs) are a class of feed-forward neural networks that consist of one or more hidden layers of fully connected neurons (see Fig. 2.7). In an MLP, each neuron receives inputs from all the neurons in the previous layer and computes a weighted sum of these inputs with additional bias term b . The weighted sum is then passed through an activation function, which introduces non-linearity into the model. The most commonly used activation functions include `sigmoid`, `ReLU`, and `tanh`. The output of each neuron is then passed on to the neurons in the next layer until the output layer is reached, which produces the final output of the network. This sequence of operations is typically referred to as the *forward pass*. The neural network *learns* by iteratively updating itself using optimization methods that minimize the error between output of the neural network and the output from the training data, also known as the *backward pass* or *backward propagation*.

Forward Pass

The computations performed by the neurons in an MLP can be mathematically represented as a sequence of matrix-vector multiplication, where the weights of each neuron in a layer are represented as a row in a weight matrix. Consider the MLP represented in Fig. 2.7. The first layer contains four input nodes, which can be represented as an input vector $\mathbf{x}_0 \in \mathbb{R}^{n_0+1}$. The additional dimension is a place holder for the bias term b . The non-linear mapping between each layer begins with a linear matrix-vector operation between the preceding layer and its corresponding weight matrix $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_0+1}$. The output of this matrix is passed element-wise through a non-linear activation function of the user's choice. The output of this operation has a dimension equal to the size of the subsequent layer. Explicitly, this operation is

$$\phi(\mathbf{W}_1 \mathbf{x}_0) = \phi \left(\begin{bmatrix} W_{0,0}^1 & W_{0,1}^1 & W_{0,2}^1 & W_{0,3}^1 & b_0 \\ W_{1,0}^1 & W_{1,1}^1 & W_{1,2}^1 & W_{1,3}^1 & b_1 \\ W_{2,0}^1 & W_{2,1}^1 & W_{2,2}^1 & W_{2,3}^1 & b_2 \\ W_{3,0}^1 & W_{3,1}^1 & W_{3,2}^1 & W_{3,3}^1 & b_3 \\ W_{4,0}^1 & W_{4,1}^1 & W_{4,2}^1 & W_{4,3}^1 & b_4 \end{bmatrix} \begin{bmatrix} x_0^0 \\ x_1^0 \\ x_2^0 \\ x_3^0 \\ 1 \end{bmatrix} \right) = \mathbf{x}_1 \quad (2.58)$$

where $W_{i,j}^1$ represents the element in the $(i+1)$ -th row and $(j+1)$ -th column of the weight matrix $\mathbf{W}_1$, using zero-indexing, the superscript 1 indicates that $\mathbf{W}_1$ is weight matrix mapping to the first layer, and ϕ represents the activation function of choice.

Iterating this non-linear mapping procedure through each subsequent layer in Fig. 2.7 yields the following result

$$\psi\left(\underbrace{\mathbf{W}_3 \phi\left(\underbrace{\mathbf{W}_2 \phi(W_1 \mathbf{x}_0)}_{\mathbf{x}_1}\right)}_{\mathbf{x}_2}\right) = \mathbf{x}_3. \quad (2.59)$$

where the outer activation function ψ is the soft max function used to produce a probability distribution over the possible output classes.

Backward Pass

When training a model, inputs from a training data set are passed through a neural network following the matrix-vector multiplications provided above and return some output depending on the initial guess values in the weight matrices $\mathbf{W}_i$. The error between the outputs of the neural network and the training data can be mathematically defined as the norm of the differences between both outputs:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{x}_3 - \mathbf{t}\|_2 \quad (2.60)$$

where $\mathbf{x}_3$ is the output of the neural network, $\mathbf{t}$ is the ground truth from the training data, and $\mathcal{L}$ is the *loss*. Although there are a variety of optimization methods (more methods are discussed in later in this chapter) for determining the weights, the widely accepted approach is a first order gradient descent method [23]:

$$\mathbf{W}_i^{n+1} = \underbrace{\mathbf{W}_i^n}_{\in \mathbb{R}^{n_i \times n_{i-1}+1}} - \underbrace{\alpha \nabla_{\mathbf{W}_i} \mathcal{L}}_{\in \mathbb{R}^{n_i \times n_{i-1}+1}} \quad (2.61)$$

where $i = 1, \dots, m$ represents the total number of transitions between layers. Note that in practice this first order gradient descent method is modified with additional “momentum” terms. For the four-layer network in Fig. 2.7, there are three transitions meaning back propagation of this network requires differentiating $\mathcal{L}$ with respect to three weight matrices, $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$.

Back propagation of a four-layer fully connected neural network

The loss function we want to minimize is the L_2 norm of the difference between the output of the network and the training data:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{x}_3 - \mathbf{t}\|_2^2 = \frac{1}{2} (\mathbf{x}_3 - \mathbf{t})^\top (\mathbf{x}_3 - \mathbf{t}) \quad (2.62)$$

where the explicit equation for the neural network is:

$$\mathbf{x}_3 = \psi \left(\mathbf{W}_3 \phi \left(\mathbf{W}_2 \phi (\mathbf{W}_1 \mathbf{x}_0) \right) \right). \quad (2.63)$$

To update the next guess matrix for $\mathbf{W}_i$, we need to compute the gradients $\nabla \mathcal{L} = [\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_2}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_3}]$. The derivatives are obtained by using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} = (\mathbf{x}_3 - \mathbf{t}) \odot \psi'(\mathbf{W}_3 \mathbf{x}_2) \odot \mathbf{W}_3 \phi'(\mathbf{W}_2 \mathbf{x}_1) \odot \mathbf{W}_2 \phi'(\mathbf{W}_1 \mathbf{x}_0) \odot \mathbf{x}_0^\top \quad (2.64)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} = (\mathbf{x}_3 - \mathbf{t}) \odot \psi'(\mathbf{W}_3 \mathbf{x}_2) \odot \mathbf{W}_3 \phi'(\mathbf{W}_2 \mathbf{x}_1) \mathbf{x}_1^\top \quad (2.65)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_3} = \underbrace{(\mathbf{x}_3 - \mathbf{t}) \odot \psi'(\mathbf{W}_3 \mathbf{x}_2)}_{\substack{\in \mathbb{R}^{n_3} \\ \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}}} \underbrace{\mathbf{x}_2^\top}_{\substack{\in \mathbb{R}^{1 \times n_2 + 1} \\ \begin{bmatrix} x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & 1 \end{bmatrix}}} \quad (2.66)$$

where $\odot$ is the Hadamard product, and the final operation for each partial derivative is an outer product.

Inference from neural networks refers to the process of using a trained model to make predictions or decisions on new input data. During inference, the input data is passed through the network, which performs the sequential matrix-vector operations with the weights $\mathbf{W}_i$ obtained from training on the input to generate an output. The output is then used to make a prediction or decision based on the task the network was trained to perform, such as image classification or natural language processing. Inference is an important step in using neural networks for real-world applications, as it allows the model to be applied to new data and provide useful insights or actions.

Gaussian Processes

An alternative surrogate modeling technique to neural networks for are Gaussian Processes, a data efficient non-parametric approach to modeling complex phenomena. In this section, I will cover the core ideas behind the Gaussian process, which will heavily reference Carl Rasmussen's text *Gaussian Processes for Machine Learning* (referred to as GPML). Beyond the mathematical derivations, I will also provided cover a comparison of Gaussian Processes to Neural Networks as surrogate models in the context of optimization.

Gaussian Processes are fundamentally challenging, and the ideas often appear abstract. This section builds up from basic probability theory to the computational implementation of Gaussian Process. The sequence will provide a generalized abstract representation of the math, and slowly become more explicit, providing detailed matrix representations and computational algorithms for implementation. For reference, the following GitHub repository

provides my in-house implementation for Gaussian Process machine learning that I used throughout my PhD, specifically for the results in Chapter 5:

<https://github.com/bmhowell/GaussianProcess>.

Multivariate Gaussian Distributions

Recall that a univariate Gaussian distribution of a random variable x_1 can be represented as a probability density function (PDF):

$$\mathbf{prob}(x_1|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-1}{2\sigma^2}(x_1 - \mu)^2\right) \quad (2.67)$$

where $x_1 \in \mathbb{R}$ is a single measurement from a set of m measurements $\mathbf{x}_1 \in \mathbb{R}^m$, $\mu \in \mathbb{R}$ is the mean of those measurements, and $\sigma^2 \in \mathbb{R}$ is the variance. We write this as a conditional probability since the distribution is dependent on the parameters μ and σ . These can be computed as follows:

$$\mu = \frac{\mathbf{1}^\top \mathbf{x}_1}{n-1} \quad (2.68)$$

$$\sigma^2 = \frac{(\mathbf{x}_1 - \mu)^\top (\mathbf{x}_1 - \mu)}{n}. \quad (2.69)$$

Now consider an additional univariate probability distribution for variable x_2 . The mean, variance, and probability density function can be constructed in the same way as x_1 .

$$\mathbf{prob}(x_2|\mu_2, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-1}{2\sigma^2}(x_2 - \mu_2)^2\right) \quad (2.70)$$

$$\mu_2 = \frac{\mathbf{1}^\top \mathbf{x}_2}{n-1} \quad (2.71)$$

$$\sigma^2 = \frac{(\mathbf{x}_2 - \mu_2)^\top (\mathbf{x}_2 - \mu_2)}{n}. \quad (2.72)$$

The intersection of these two distributions can be encoded in the *multi-variate Gaussian distribution*, or also known as the *joint probability distribution*, which is also fully parameterized as given a known mean μ and covariance Σ^2 . In matrix form:

$$\mathbf{prob}(\mathbf{x} | \mu, \Sigma) = \frac{1}{\sqrt{2\pi^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{X} - \mu)^\top \Sigma^{-1}(\mathbf{X} - \mu)\right). \quad (2.73)$$

Now $\mathbf{x}, \mu \in \mathbb{R}^n$, $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\Sigma \in \mathbb{R}^{n \times n}$ where n is number of variables in the multivariate Gaussian distribution and m is the number of data points.

In the traditional formulation of the Gaussian Process (GP), the PDF of the joint probability distribution is not used directly, rather the covariance matrix and mean vector will

be used to manually sample from the normal distribution. The mean of a collection of data points $\mathbf{X}$ is defined as:

$$\mu(\mathbf{X}) = \mathbb{E}[f(\mathbf{X})] \in \mathbb{R}^n \quad (2.74)$$

and the covariance between two mean-centered data points $\mathbf{x}$, $\mathbf{x}'$ is defined as:

$$\text{cov}(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))^\top] \quad (2.75)$$

where $f(\mathbf{x}) \in \mathbb{R}^n$ is defined as “an infinite collection of random variables, indexed by x , any finite number of which have a joint Gaussian distribution with mean and covariance” [24], and the covariance function measures the covariance between the function values $f(\mathbf{x})$ and $f(\mathbf{x}')$. Explicitly, $\mathbf{x}$ is the input parameters to the data, and $f(\mathbf{x})$ are the observed measurements. When sampling from the joint probability distribution parameterized by μ and $\text{cov}(\mathbf{x}, \mathbf{x}')$, which is commonly denoted as:

$$f \sim \mathcal{GP}(\mu, \text{cov}(\mathbf{x}, \mathbf{x}')) \rightarrow f | \mathcal{D} \sim \mathcal{GP}(\mu', k') \quad (2.76)$$

where $\mathcal{D} = \{(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \dots, N\}$ is the data.

The core assumption underlying a Gaussian Process (GP) is that the output variables are correlated based on their input variables. For instance, two data points, $\mathbf{x}_1$ and $\mathbf{x}_2$, will be more correlated than another pair, $\mathbf{x}_1$ and $\mathbf{x}_3$, if the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ is smaller than the distance between $\mathbf{x}_1$ and $\mathbf{x}_3$, i.e., $\|\mathbf{x}_1 - \mathbf{x}_2\| < \|\mathbf{x}_1 - \mathbf{x}_3\|$. The mechanism by which correlation is measured dictates way the GP models the data, and is typically referred to as the kernel function $k(\mathbf{x}, \mathbf{x}')$

Unconditioned GPs

Consider an unconditioned GP parameterized with zero mean and the classic radial basis function (RBF) as the measure of correlation:

$$\mu = \mathbb{E}[f(\mathbf{x})] = 0 \quad (2.77)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right). \quad (2.78)$$

Sampling from this GP using 1D linear spaced variables can produce a variety of different models depending on the choice of l . Consider the models sampled in Fig. 2.8:

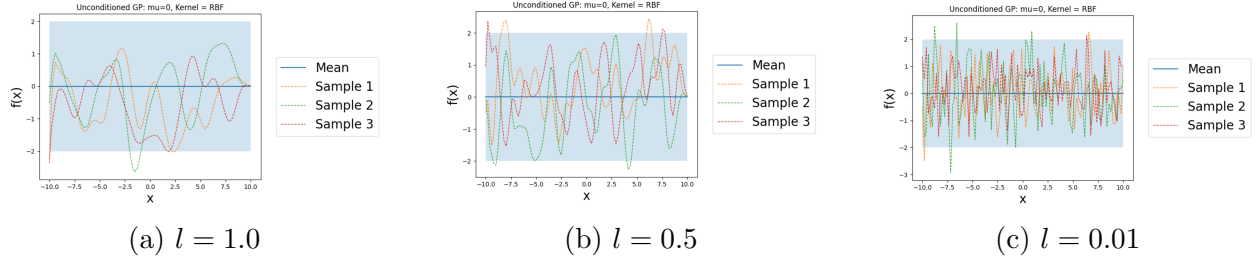


Figure 2.8: Comparison of different hyper-parameter values of l where the shaded blue areas represent the uncertainty bands on the scale of standard deviation.

In essence, the hyper parameter l captures the length scale in which two data points are correlated. With larger values of l , the two points may have higher correlated outputs over a longer distances and effectively smoothing out interpolation between data points. For lower values of l , the outputs of two variables may be less correlated, enabling the kernel function to capture sharper jumps in the underlying data.

Sampling from a GP

Once the mean $\boldsymbol{\mu}$ and a covariance matrix $\mathbf{K}$ are defined, sampling from the GP is analogous to sampling from the multi-variate normal distribution:

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \rightarrow \begin{bmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_n) \end{bmatrix} \sim \mathcal{GP} \left(\begin{bmatrix} \mu(x_1) \\ \mu(x_2) \\ \dots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_m) \\ k(x_2, x_1) & \dots & k(x_2, x_m) \\ \dots & \dots & \dots \\ k(x_n, x_1) & \dots & k(x_n, x_m) \end{bmatrix} \right) \quad (2.79)$$

where each element of covariance matrix is computed by using kernel function and the two corresponding points:

$$K_{ij} = k(x_i, x_j). \quad (2.80)$$

Computational sampling can be performed by:

$$\mathbf{f} = \boldsymbol{\mu} + \mathbf{L}\mathbf{z} \quad (2.81)$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{L}\mathbf{L}^\top = \mathbf{K}$. It is here we see one of the critical bottlenecks for GPs in computation, since $\mathbf{L}$ requires a Cholesky decomposition of the covariance matrix, which is $\in \mathbb{R}^{m \times m}$ where m is the number of data points. As the data set size increases, the cost of the Cholesky decomposition increases on the $\mathcal{O}(\frac{m^3}{3})$.

Conditioning the GP

Consider a dataset $\mathcal{D}$ that $\mathbf{x}_i \rightarrow y = f(\mathbf{x}_i)$ where there might be some uncertainty or noise ϵ :

$$y = f(\mathbf{x}) + \epsilon, \quad (2.82)$$

the mean and covariance matrix for inference is updated as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{GP} \left(\begin{bmatrix} \boldsymbol{\mu}(\mathbf{x}) \\ \boldsymbol{\mu}_*(\mathbf{x}) \end{bmatrix}, \mathbf{cov} = \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \quad (2.83)$$

where each sub-block of the covariance matrix is application of the kernel function with both measured data points and predicted points points.

Explicitly, the elements of the mean vector and covariance matrix are:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} = [y_1 \ y_2 \ \dots \ y_N \ f_*(x_1) \ f_*(x_2) \ \dots \ f_*(x_N)]^\top, \quad (2.84)$$

$$\begin{bmatrix} \boldsymbol{\mu}(\mathbf{x}) \\ \boldsymbol{\mu}_*(\mathbf{x}) \end{bmatrix} = [\mu(x_1) \ \mu(x_2) \ \dots \ \mu(x_N) \ \mu_*(x_1) \ \mu_*(x_2) \ \dots \ \mu_*(x_M)]^\top, \quad (2.85)$$

$$\mathbf{cov} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_N) & k(x_1, x_1^*) & \dots & k(x_1, x_M^*) \\ k(x_2, x_1) & \dots & k(x_2, x_N) & k(x_2, x_1^*) & \dots & k(x_2, x_M^*) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & \dots & k(x_N, x_N) & k(x_N, x_1^*) & \dots & k(x_N, x_M^*) \\ k(x_1^*, x_1) & \dots & k(x_1^*, x_N) & k(x_1^*, x_1^*) & \dots & k(x_1^*, x_M^*) \\ k(x_2^*, x_1) & \dots & k(x_2^*, x_N) & k(x_2^*, x_1^*) & \dots & k(x_2^*, x_M^*) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k(x_M^*, x_1) & \dots & k(x_M^*, x_N) & k(x_M^*, x_1^*) & \dots & k(x_M^*, x_M^*) \end{bmatrix}. \quad (2.86)$$

Each addition of measured data points decrease the uncertainty bounds. This means that when sampling from the distribution, the function space becomes more certain. Fig. 2.8 visually demonstrates the collapse of function space as incremental points are added.

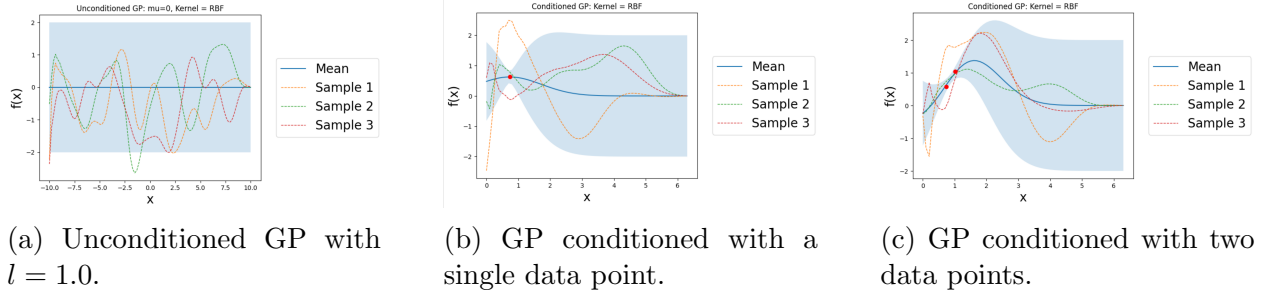


Figure 2.9: Comparison of different hyper-parameter values of l where the shaded blue areas represent the uncertainty bands on the scale of standard deviation.

GPML [24] provides the derivation leading to the critical equations required for GP regression:

$$\bar{\mathbf{f}}_* = \boldsymbol{\mu}_*(\mathbf{x}) = \mathbf{K}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \quad (2.87)$$

$$\text{cov}(\mathbf{f}_*(\mathbf{x})) = \mathbf{K}_{**} - \mathbf{K}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}_*. \quad (2.88)$$

Once again, we see a matrix inversion of a symmetric, positive-definite Hermitian matrix $(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}$, which can be computed using Cholesky decomposition.

Automatic bias-variance trade-off and model selection

As indicated in Fig. 2.8, the hyperparameters of the kernel function play a critical role in the quality of model. Tuning these hyperparameters by hand often introduces bias to the model and may produce overfitting. Fortunately, the formulation of the GP provides a method that inherently balances model complexity and degree of fit leading to the “automatic bias-variance trade-off”. We can do this by performing a meta optimization step by maximizing the models probability of a certain set of hyperparameters $\boldsymbol{\theta}$, given the data $\mathcal{D}$:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \propto p(\mathcal{D}|\boldsymbol{\theta}). \quad (2.89)$$

Note that if we make the basic assumption of having a flat prior $p(\boldsymbol{\theta}) = 1$ and recalling that the location of the maximum of a function is independent of its scaling, we see that the only part of Eqn. 2.89 that requires attention is marginal likelihood $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})$. The form of $p(\mathcal{D}|\boldsymbol{\theta})$ is explicitly the joint probability distribution. For computational and distributional robustness, the log of the marginal likelihood given the data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ can be expressed as:

$$\log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2} \mathbf{y}^\top \mathbf{K}_\theta^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_\theta| - \frac{n}{2} \log 2\pi, \quad (2.90)$$

where $\mathbf{K}_\theta$ is the covariance matrix parameterized by the hyperparameters θ , and n is the number of data points.

Maximizing the marginal log likelihood is done in different ways in different implementations. From my experience, the two approaches most common approaches are gradient based methods and Markov-Chain Monte-Carlo methods. For gradient based methods, the gradient of the marginal log likelihood with respect to the hyperparameters θ_i is given by:

$$\frac{\partial \log p(\mathbf{y} \mid \mathbf{X}, \theta)}{\partial \theta_i} = \frac{1}{2} \mathbf{y}^\top \mathbf{K}_\theta^{-1} \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \mathbf{K}_\theta^{-1} \mathbf{y} - \frac{1}{2} \text{tr} \left(\mathbf{K}_\theta^{-1} \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \right). \quad (2.91)$$

The computational complexity of evaluating the marginal log likelihood and its gradients is still dominated by the inversion of the covariance matrix $\mathbf{K}_\theta$, which is $\mathcal{O}(n^3)$. Various methods, such as sparse approximations and inducing points, can be used to reduce this complexity.

2.4 Numerical Optimization

Optimization, a concept widely applied across various disciplines, holds profound significance in mathematics and physics. The observed tendency of physical systems towards states of stable equilibrium serves as a fundamental principle underlying numerous phenomena. In physics and chemistry, dynamical systems evolve along paths that minimize their energy state, ultimately reaching an equilibrium point. This paradigm extends to diverse fields such as finance, engineering, operations research, and parameter estimation, where the objective is to maximize or minimize a specific quantity by manipulating decision variables (also referred to as optimization variables or manipulated variables).

The optimization process can manifest in different forms. In some instances, systems converge to a local minimum energy state following a descent path, analogous to a ball rolling down a hill to settle in a valley. In other cases, optimization emerges from seemingly random processes, as exemplified by the evolutionary principle of survival of the fittest in Darwinian theory.

Mathematically, we can formulate these problem statements as a mapping between inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$, represented by a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The objective of mathematical optimization is to identify an input $\mathbf{x} \in \mathbb{R}^n$ that minimizes or maximizes the output $y \in \mathbb{R}$. The standard form of such an optimization problem can be expressed as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq b_i, \quad i = 1, \dots, m. \end{aligned}$$

which reads as "minimize $f_0(\mathbf{x})$ subject to $f_i(\mathbf{x}) \leq b_i$ for all $i = 1, \dots, m$," where $f_0(\mathbf{x})$ is the objective function $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$ (also referred to as the cost or evaluation function), $\mathbf{x} \in \mathbb{R}^n$ is the vector of decision variables, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$ are the inequality constraints, and $b_1, \dots, b_m$ are the constraint bounds. The optimal solution is denoted as $\mathbf{x}^*$.

It is important to note that while the concept of optimization is broadly applicable, not all optimization problems are solvable. The tractability of an optimization problem primarily depends on the structure of the objective function $f_0(\mathbf{x})$. In some cases, $f_0(\mathbf{x})$ is an explicit function amenable to analytical operations. For instance, the univariate objective function $f_0(x) = (x + 2)^2 + 1$ can be minimized using elementary calculus techniques, yielding an analytical solution of $x^* = -2$. However, analytical solutions are often unattainable for most practical problems.

For more complex optimization problems, numerical methods provide a powerful toolset. These problems can generally be categorized into two main classes: convex optimization and stochastic (blackbox) optimization. The following sections will delve into various numerical techniques for solving tractable optimization problems within these categories, exploring their principles, applications, and limitations.

Gradient-based methods

Newton's method:

Let us first consider an convex objective $f_0(x)$ and method in which we can approach solving the minimization. Convexity enables us to makes certain assumptions about optimality conditions. One optimality condition being

$$\nabla_{\mathbf{x}} f_0(\mathbf{x}) = 0, \quad (2.92)$$

or more simply, we would like to find the root of the gradient of $f_0(\mathbf{x})$.

Newton's method is a simple numerical approach used for find roots of an equation if an initial candidate solution is close the exact solution. Here I derive Newton's method using a Taylor Series expansion.

Suppose $f_0(\mathbf{x})$ is a continuous function on a closed interval $[a, b]$ and has $n + 1$ continuous derivatives on the open interval (a, b) . If $x, c \in (a, b)$, then the expansion of $f_0(x)$ about c is:

$$f(c) + f'(c)(x - c) + \frac{1}{2!}f''(c)(x - c)^2 + \frac{1}{3!}f'''(c)(x - c)^3 + \dots + \frac{1}{n!}f^n(c)(x - c)^n \quad (2.93)$$

or more concisely:

$$f(x) \approx \sum_{k=0}^{\infty} \frac{1}{k!} f^{(k)}(c)(x - c)^k \quad (2.94)$$

By approximating $f_0(\mathbf{x})$ using this expansion, we can find the optimal x^* by expanding $\nabla_x f_0(x)$ around a initial guess value x^j , which at optimum should be equal to zero:

$$\nabla_x f_0(x^{j+1}) = \nabla_x f_0(x^j) + \nabla_x(\nabla_x f_0(x^j))(x^{j+1} - x^j) + \mathcal{O}(x^{j+1} - x^j)^2 = 0|_{x^{j+1}=x^*}. \quad (2.95)$$

The resulting error $\mathcal{O}(x^{j+1} - x^j)^2$ indicates that this optimization method is a second-order gradient-based approach. Rearranging the above equation yields:

$$-\nabla_x f_0(x^j) = \nabla_x(\nabla_x f_0(x^j))(x^{j+1} - x^j) \quad (2.96)$$

where we seek to solve for x^{j+1} , where $\nabla_x(\nabla_x f_0(x^j))$ is the Hessian. If x is a vector, than a matrix inversion is required to solve for x^{j+1} .

$$\mathbf{x}^{j+1} = \mathbf{x}^j - [\nabla_x(\nabla_x f_0(\mathbf{x}^j))]^{-1} \nabla_x f_0(\mathbf{x}^j). \quad (2.97)$$

Explicitly, the Hessian is defined as:

$$H = \nabla_x(\nabla_x f_0(\mathbf{x}_i)) = \begin{bmatrix} \frac{\partial^2 f_0}{\partial x_1^2} & \frac{\partial^2 f_0}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f_0}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f_0}{\partial x_2 \partial x_1} & \frac{\partial^2 f_0}{\partial x_2^2} & \cdots & \frac{\partial^2 f_0}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f_0}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f_0}{\partial x_n^2} \end{bmatrix} \quad (2.98)$$

This means that use of Newton's methods for gradient descent requires an inversion of a $n \times n$ matrix (a computational cost $\mathcal{O}(n^3)$) to compute the next candidate solution. The update equation can be explicitly referred to as:

$$\begin{bmatrix} x_1^{j+1} \\ \vdots \\ x_n^{j+1} \end{bmatrix} = \begin{bmatrix} x_1^j \\ \vdots \\ x_n^j \end{bmatrix} - \begin{bmatrix} \frac{\partial f_0^2}{\partial x_1^2} & \cdots & \frac{\partial f_0^2}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_0^2}{\partial x_n \partial x_1} & \cdots & \frac{\partial f_0^2}{\partial x_n^2} \end{bmatrix} \begin{bmatrix} \frac{\partial f_0}{\partial x_1} \\ \vdots \\ \frac{\partial f_0}{\partial x_n} \end{bmatrix}. \quad (2.99)$$

Sometime, we do not have an explicit form of $f_0(\mathbf{x})$ and therefore do not have access to the gradients directly. We can approximate these gradients numerically (see Section on Numerical Methods), however Newton's method quickly deteriorates as a viable option as $n \rightarrow \infty$ since matrix inversion has computational complexity on the order of $\mathcal{O}(n^3)$.

First-order methods

Gradient descent optimizes a function by using first-order gradient information, while Newton's method optimizes a function utilizing second-order gradient information. The most basic implementation of gradient descent computes the gradient of the objective function and slightly perturbs the current position. Although first-order methods are considerably faster, these algorithms can be unstable and perform poorly on non-convex functions. Consider the example in which parameters of a model are being tuned to fit a data set:

Let θ be the parameters for a function approximator $f_0(x)$. Gradient descent can be computed iteratively as

$$\theta^{i+1} = \theta^i - \gamma \nabla_{\theta} f_0(x), \quad x \in \mathcal{D} \quad (2.100)$$

where γ is a hyper parameter chosen to determine the incremental step size performed by optimizer.

Extensions of First-Order Methods

First-order methods can be improved by incorporating additional information about past gradients or modifying the learning rate dynamically. Here I discuss a few popular extensions: Gradient Descent with Momentum, Nesterov Accelerated Gradient, and Adaptive Gradient Algorithms (such as AdaGrad, RMSprop, and Adam). See section 7.3 for basic python implementations and examples.

Gradient descent with momentum

Momentum is a method that helps accelerate gradients vectors in the right directions, thus leading to faster converging. The momentum method introduces a velocity vector that

accumulates past gradients to smooth out the update process.

The update rule for gradient descent with momentum is:

$$\begin{aligned}\mathbf{v}^i &= \beta \mathbf{v}^{i-1} + (1 - \beta) \nabla_{\theta} f_0(\theta^i) \\ \theta^{i+1} &= \theta^i - \gamma \mathbf{v}^i\end{aligned}\tag{2.101}$$

where β is the momentum term (typically set between 0.8 and 0.9) and $\mathbf{v}$ is the velocity vector.

Nesterov Accelerated Gradient

Nesterov Accelerated Gradient (NAG) is a modification of the momentum method that improves the convergence rate by making a correction before the gradient is evaluated.

The update rule for NAG is:

$$\begin{aligned}\mathbf{v}^i &= \beta \mathbf{v}^{i-1} + (1 - \beta) \nabla_{\theta} f_0(\theta^i - \gamma \beta \mathbf{v}^{i-1}) \\ \theta^{i+1} &= \theta^i - \gamma \mathbf{v}^i\end{aligned}\tag{2.102}$$

Adaptive Gradient Algorithms

Adaptive gradient algorithms adjust the learning rate based on the past gradients' magnitudes, allowing for more fine-tuned updates. Here we introduce a few popular adaptive methods:

AdaGrad: AdaGrad adapts the learning rate for each parameter based on the past squared gradients, allowing for larger updates for infrequent parameters and smaller updates for frequent ones.

The update rule for AdaGrad is:

$$\begin{aligned}\mathbf{G}^i &= \mathbf{G}^{i-1} + \nabla_{\theta} f_0(\theta^i) \odot \nabla_{\theta} f_0(\theta^i) \\ \theta^{i+1} &= \theta^i - \frac{\gamma}{\sqrt{\mathbf{G}^i + \epsilon}} \odot \nabla_{\theta} f_0(\theta^i)\end{aligned}\tag{2.103}$$

where $\mathbf{G}$ is the accumulation of past squared gradients and ϵ is a small constant to avoid division by zero.

RMSprop: RMSprop addresses AdaGrad's issue of rapidly diminishing learning rates by introducing a decay factor.

The update rule for RMSprop is:

$$\begin{aligned}\mathbf{G}^i &= \beta \mathbf{G}^{i-1} + (1 - \beta) \nabla_{\theta} f_0(\theta^i) \odot \nabla_{\theta} f_0(\theta^i) \\ \theta^{i+1} &= \theta^i - \frac{\gamma}{\sqrt{\mathbf{G}^i + \epsilon}} \odot \nabla_{\theta} f_0(\theta^i)\end{aligned}\tag{2.104}$$

Adam: Adam (Adaptive Moment Estimation) combines the ideas of momentum and RMSprop by maintaining both first and second moment estimates of the gradients.

The update rule for Adam is:

$$\begin{aligned}
\mathbf{m}^i &= \beta_1 \mathbf{m}^{i-1} + (1 - \beta_1) \nabla_{\theta} f_0(\theta^i) \\
\mathbf{v}^i &= \beta_2 \mathbf{v}^{i-1} + (1 - \beta_2) \nabla_{\theta} f_0(\theta^i) \odot \nabla_{\theta} f_0(\theta^i) \\
\hat{\mathbf{m}}^i &= \frac{\mathbf{m}^i}{1 - \beta_1^i} \\
\hat{\mathbf{v}}^i &= \frac{\mathbf{v}^i}{1 - \beta_2^i} \\
\theta^{i+1} &= \theta^i - \frac{\gamma}{\sqrt{\hat{\mathbf{v}}^i} + \epsilon} \hat{\mathbf{m}}^i
\end{aligned} \tag{2.105}$$

where β_1 and β_2 are hyperparameters typically set to 0.9 and 0.999, respectively, and ϵ is a small constant.

Genetic Algorithms

Genetic Algorithms (GAs) are a subset of Machine Learning Algorithms (MLAs) that are particularly well-suited for solving nonconvex, nonsmooth, multicomponent, and multistage optimization problems. These algorithms draw inspiration from the principles of natural selection and evolution, making them robust tools for complex system optimization where traditional gradient-based methods may fail.

Algorithm Structure

Following the work of Zohdi [25, 26, 27, 28, 29, 30], the basic structure of a genetic algorithm can be described as follows:

Algorithm 1 Genetic Algorithm

- 1: Generate an initial population of S genetic strings
 - 2: **while** termination condition not met **do**
 - 3: Compute fitness of each string
 - 4: Rank genetic strings
 - 5: Mate nearest pairs to produce offspring
 - 6: Eliminate bottom M strings
 - 7: Keep top K parents and their K offspring
 - 8: Introduce M new random strings
 - 9: **end while**
 - 10: **return** best solution found
-

Genetic String Representation

Each candidate solution in the population is represented as a genetic string λ_i , defined as:

$$\lambda_i \stackrel{\text{def}}{=} [\lambda_1^i, \lambda_2^i, \lambda_3^i, \dots, \lambda_N^i] \quad (2.106)$$

where each λ_j^i represents a parameter in the solution, constrained within specific ranges: $\lambda_j^{(-)} \leq \lambda_j^i \leq \lambda_j^{(+)}$.

Mating and Offspring Generation

The algorithm generates offspring through a mating process between pairs of parent strings. For a pair of parent strings λ_i and λ_{i+1} , two offspring are produced:

$$\lambda_i^{\text{new}} = \phi \odot \lambda_i + (1 - \phi) \odot \lambda_{i+1} \quad (2.107)$$

$$\lambda_{i+1}^{\text{new}} = \psi \odot \lambda_i + (1 - \psi) \odot \lambda_{i+1} \quad (2.108)$$

where ϕ and ψ are vectors of random numbers between 0 and 1, and $\odot$ denotes the Hadamard (element-wise) product.

Key Observations and Best Practices

Hybrid Approach: It's often effective to use a GA to isolate multiple local minima, then apply gradient-based methods in these locally convex regions or reset the GA to focus its search around the best-performing parameter sets.

Mutation: While it's possible to induce mutations by selecting mating parameters outside the [0,1] range, this is somewhat redundant with the introduction of new random members in each generation.

Parent Retention: Keeping top-performing parents in the next generation ensures monotonic reduction of the cost function and reduces computational cost, as retained parents don't need re-evaluation.

Population Size: Sufficiently large population sizes help mitigate potential issues with inbreeding when retaining parents.

Bayesian Optimization (BO)

The following are my personal notes on the topic of BO.

Bayesian optimization is an efficient strategy for the optimization of expensive black-box functions. It is particularly useful when function evaluations are costly, time-consuming, or have limited availability. The core principle of BO is to construct a probabilistic model (surrogate model) of and leverage this model to intelligently select the most promising points for

subsequent evaluation. This section gives a broad overview of core concepts in BO, specifically the various approaches to surrogate modeling and manage the Exploration/Exploitation Dilemma.

Surrogate Models

Surrogate models are probabilistic approximations of the objective function that enable efficient exploration of the parameter space. While Gaussian Processes (GPs) are commonly used, other methods have gained popularity due to their flexibility and scalability. Here, I mention several types of surrogate models that are covered in the literature [31, 32, 33, 34]:

Gaussian Processes (GPs)

Gaussian Processes (discussed thoroughly in section 2.3) provide a probabilistic representation of the objective function. For a given input $\mathbf{x}$, the GP provides a predictive distribution characterized by its mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (2.109)$$

where $k(\mathbf{x}, \mathbf{x}')$ is the kernel function that defines the covariance between any two points in the input space. GPs offer a natural way to model uncertainty and are particularly effective for low-dimensional problems.

Neural Networks

Neural Networks (NNs) have emerged as powerful surrogate models, especially for high-dimensional and large-scale optimization problems. They can capture complex, non-linear relationships in the data. In the context of BO, we often use:

- **Bayesian Neural Networks (BNNs):** These assign a prior distribution to the network weights, allowing for uncertainty quantification. The predictive distribution for an input $\mathbf{x}$ is given by:

$$p(f(\mathbf{x})|\mathcal{D}) = \int p(f(\mathbf{x})|\mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \quad (2.110)$$

where $\mathbf{w}$ are the network weights and $\mathcal{D}$ is the observed data.

- **Ensembles of Neural Networks:** Multiple networks are trained on bootstrap samples of the data. The predictive mean and variance are estimated from the ensemble predictions:

$$\mu(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}), \quad \sigma^2(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^M (f_i(\mathbf{x}) - \mu(\mathbf{x}))^2 \quad (2.111)$$

where M is the number of ensemble members and $f_i(\mathbf{x})$ is the prediction of the i -th network.

Random Forests

Random Forests offer another alternative for surrogate modeling, particularly useful for categorical variables and mixed integer-continuous optimization problems. They provide uncertainty estimates through the variance of predictions across trees:

$$\mu(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T f_i(\mathbf{x}), \quad \sigma^2(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T (f_i(\mathbf{x}) - \mu(\mathbf{x}))^2 \quad (2.112)$$

where T is the number of trees and $f_i(\mathbf{x})$ is the prediction of the i -th tree.

Gaussian Process and Neural Network Hybrids

Recent research has explored hybrid models that combine the strengths of GPs and NNs. For instance, Deep Kernel Learning uses a neural network to learn features that are then fed into a GP:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\phi(\mathbf{x})), k(\phi(\mathbf{x}), \phi(\mathbf{x}')))) \quad (2.113)$$

where $\phi(\cdot)$ is a neural network feature extractor.

The choice of surrogate model depends on factors such as the dimensionality of the problem, the presence of categorical variables, the available computational resources, and the specific characteristics of the objective function. Each model offers different trade-offs between expressiveness, uncertainty quantification, and computational efficiency.

Exploration and Exploitation

A core concept in machine learning (specifically in decision making under uncertainty) is the trade-off between two competing strategies:

- **Exploration:** Selecting points in regions where the model uncertainty is high to improve the model's accuracy and coverage of the search space.
- **Exploitation:** Selecting points where the model predicts high values of the objective function to rapidly converge on the optimum.

Excessive exploration can lead to unnecessary evaluations in suboptimal areas, while over-exploitation might result in premature convergence to local optima, potentially missing the global optimum. Decision making on how to balance these two strategies is commonly referred to as the “Exploration/Exploitation Dilemma”.

Multi-Armed Bandit Problem

The Exploration/Exploitation Dilemma in BO is closely related to the multi-armed bandit problem, a classic problem in decision theory and reinforcement learning. In the multi-armed bandit scenario, a gambler must decide which arm of several slot machines (bandits) to pull to maximize their cumulative reward. Each arm has an unknown probability distribution of rewards, and the gambler must balance the exploration of less-played arms and the exploitation of arms known to yield higher rewards.

Formally, the multi-armed bandit problem can be described as follows:

Let $\{a_1, \dots, a_K\}$ be a set of K arms, each with an unknown reward distribution. At each time step t , the agent selects an arm a_t and receives a reward r_t . The objective is to maximize the cumulative reward over T time steps:

$$\max_{\{a_t\}_{t=1}^T} \mathbb{E} \left[\sum_{t=1}^T r_t \right] \quad (2.114)$$

This formulation closely mirrors the iterative nature of BO, where each "arm pull" corresponds to an evaluation of the objective function at a chosen point.

Acquisition Functions

Acquisition functions are the cornerstone of BO, guiding the selection of the next evaluation point by quantifying the exploration-exploitation trade-off. These functions utilize the predictive distribution provided by the surrogate model to assess the potential value of evaluating the objective function at a given point. Commonly used acquisition functions include:

Probability of Improvement (PI)

The Probability of Improvement acquisition function calculates the probability that a new point will yield an improvement over the current best observed value:

$$\alpha_{PI}(\mathbf{x}) = \Phi \left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \right) \quad (2.115)$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and standard deviation predicted by the surrogate model at point $\mathbf{x}$, $f(\mathbf{x}^+)$ is the current best observed value, ξ is a small positive constant that controls the exploration-exploitation trade-off, and Φ is the cumulative distribution function of the standard normal distribution.

Expected Improvement (EI)

The Expected Improvement acquisition function calculates the expected value of the improvement over the current best observed value:

$$\alpha_{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi) \Phi \left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \right) + \sigma(\mathbf{x}) \phi \left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} \right) \quad (2.116)$$

where ϕ is the probability density function of the standard normal distribution.

Upper Confidence Bound (UCB)

The Upper Confidence Bound acquisition function provides a weighted sum of the predicted mean and standard deviation:

$$\alpha_{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa \sigma(\mathbf{x}) \quad (2.117)$$

where κ is a parameter that controls the balance between exploration and exploitation. Higher κ values favor exploration by placing more emphasis on the uncertainty term.

Entropy Search and Predictive Entropy Search

These more advanced acquisition functions aim to maximize the information gain about the location of the global optimum. They are based on the principle of reducing the entropy of the posterior distribution over the location of the optimum.

For Entropy Search:

$$\alpha_{ES}(\mathbf{x}) = H(p(\mathbf{x}^*|D)) - \mathbb{E}_y [H(p(\mathbf{x}^*|D \cup \{(\mathbf{x}, y)\}))] \quad (2.118)$$

where $H(\cdot)$ denotes the differential entropy, $p(\mathbf{x}^*|D)$ is the posterior distribution over the location of the optimum given the current data D , and the expectation is taken over the predictive distribution of y at $\mathbf{x}$.

Optimization Process

The BO process can be summarized as follows:

1. Initialize the surrogate model with a small number of initial observations.
2. For $t = 1, 2, \dots, T$:
 - a) Optimize the acquisition function to select the next point: $\mathbf{x}_t = \arg \max_{\mathbf{x}} \alpha(\mathbf{x})$
 - b) Evaluate the objective function at $\mathbf{x}_t$: $y_t = f(\mathbf{x}_t)$
 - c) Update the surrogate model with the new observation $(\mathbf{x}_t, y_t)$
3. Return the best observed solution

This iterative process continues until a stopping criterion is met, such as a maximum number of iterations or a satisfactory improvement threshold.

Chapter 3

UV-curable Inks in Additive Manufacturing

“It doesn’t matter how beautiful your theory is, it doesn’t matter how smart you are. If it doesn’t agree with experiment, it’s wrong.”

- Richard P. Feynman

3.1 Introduction

Direct-Ink-Write Printing

Additive manufacturing (AM) processes are pushing design into previously impossible geometries with advanced and multi-functional materials triggering disruptive (albeit positive) shifts in the techno-socio-economic landscape [35, 36, 37, 38, 39]. The ever-increasing 3D complexity of objects is driving development of new AM techniques [10, 11, 12], increasing spatial and compositional control across multiple orders of length scale [13, 14]. Direct ink writing (DIW) extends this feat by extruding shear thinning viscoelastic fluids that exhibit shape retention when the shear-strain is dramatically reduced after deposition from the nozzle, thus allowing for ink composition flexibility to encompass a wide range of mechanical properties [40, 41]. However, materials with poor hysteretic recovery or low yield stresses can limit spatial precision for complex print patterns (e.g., spanning gaps).

Prior Work

To address these difficulties, recent published methods utilized various mechanisms to aid in filament shape retention while maintaining desired material properties. For example, embedded printing methods enables freeform printing within support-bath material that provides buoyant forces to stabilize the deposited filaments [14]. Other approaches include the use of multimaterial 3D printing for fine compositional tuning within printed materials

in which scaffolding material can be precisely deposited, or manipulating an electromagnetic field to control the placement of inks doped with ferromagnetic particles [42, 43]. On-demand UV curing has further improved spatial resolution of DIW by rapidly modifying ink rheology, typically through the addition of acrylate functional groups and a photoinitiator [44, 45, 46]. Past attempts with rapid UV curing, however, have resulted in brittle and rigid mechanical properties due to high concentrations of low molecular weight monomers in the ink formulations [44]. Additionally, on-demand UV curing has reduced interfacial strength between filaments resulting in anisotropic mechanical properties depending on print orientation [44, 47]. This occurs as a result of uncontrolled rapid UV kinetics that gel too rapidly to enable subsequent interfilament bonding or are too sensitive to UV light upon exiting the nozzle. Dual curing efforts have shown promise in controlling the extent of UV curable components to allow sufficient stiffening when printing while accessing mechanical properties inherent to tough epoxies [44, 48, 49, 50].

Chapter Scope

Here we expand this dual curing approach to a urethane-grafted acrylate polymer (UGAP) in order to exploit the rheological flexibility provided by on-demand curing. Elastomeric mechanical properties provided by combinations of urethane and acrylate components and leaching of solid filler [51, 52, 53] produces a material that accesses previously unattainable structures often required for application spaces such as soft robotics, functional materials and biomedical devices [54, 55, 56, 57].

3.2 Experimental Methods & Materials

Materials

Polycaprolactone diol (8025D) and triol (8025E) (Perstorp Chemical, MW = 2000 *g/mol*, OH equiv = 56 *mg/g* and 84 *mg/g* respectively), polyisocyanates/acrylate (EBECRYL[®] 4396, Allnex USA Inc., NCO = 7.5wt%, acrylate = 1 equiv), ethylene glycol phenyl ethyl acrylate (EGPEA), 1,6-hexanediol diacrylate (HDDA), diphenyl-(2,4,6-trimethylbenzoyl)-phosphine oxide (Irgacure[®] 819) (Sigma Aldrich), fumed silica (Aerosil R202, Evonik) were used as received. NaCl (Sigma Aldrich) was ground in a Retsch RM 200 automatic mortar and pestle and sieved to course (75-106 μm) fine ($\leq 45 \mu\text{m}$). NaCl powders dried at 100 °C for 1 *h* and were stored in a desiccator until used.

Ink Formulation

UGAP components (40.84wt% 8025D, 35.07wt% EBECRYL[®] 4396, 15wt% EGPEA, 4.08wt% 8025E, 3.33wt% Irgacure[®] 819, and 1.67wt% HDDA) were mixed in a Flacktek

DAC 150.1 FV-K SpeedMixer™ for 60 second intervals at 2000 rpm, ensuring that the temperature did not exceed 35 °C. Refer to Tab. 3.1 for explicit formulations. a

Ink Rheology and UV Sensitivity

Oscillatory rheology was performed on a TA Instruments AR2000Ex rheometer. A 25 mm Peltier parallel plate, sample thickness of 1.5 mm and constant frequency of 1 Hz was used for stress sweeps. For UV rheology experiments, 0.840mm thick samples were exposed (S2000, Omnicure, 400-500 nm filter) between a 20 mm aluminum upper plate and 20 mm acrylic lower plate. A linear viscoelastic regime was found using a 0.5% strain and a 1 rad/s oscillation frequency.

3D Printing

Prepared inks were loaded into Nordson Optimum UV-blocking 30 cc syringes with an Optimum syringe piston and 0.840 mm tapered nozzle wrapped in Kapton tape. Samples were degassed in the Flacktek for 2 minutes at 3500 rpm. An air-powered fluid dispenser (Ultimus V, Nordson EFD) was used to extrude the ink. UV light was generated by an Omnicure lamp through 3 probes attached to a custom-designed mount (see Fig. 3.1). An Aerotech ANT 130-XY and 130-L-Z5 axis stage was controlled via an A3200 controller through an Aerotech A3200 CNC operator interface (v.5.05.000). Gravimetric removal of sodium chloride (solids leaching) was performed by stirring parts in a covered container of distilled water at 80 °C until NaCl was removed.

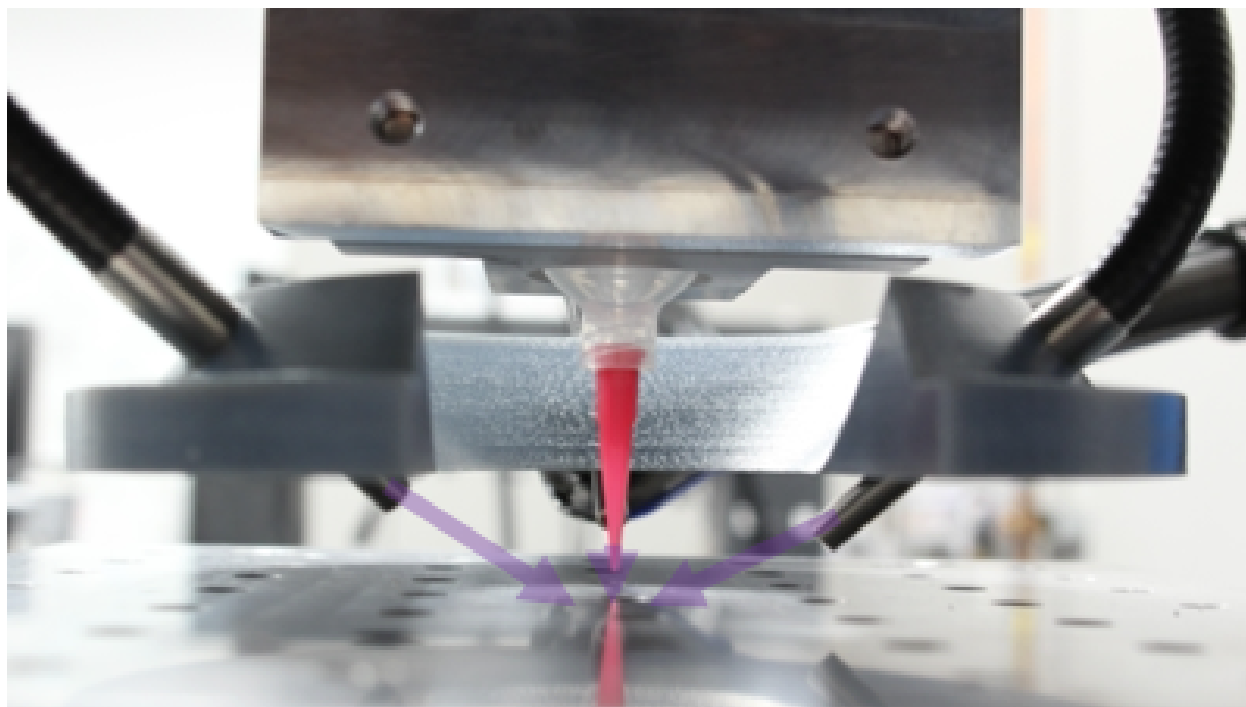


Figure 3.1: Custom on-demand UV curing mount with three light guides focused to a single spot size under the printing nozzle.

Mechanical Testing

Spanning tests were performed on custom blocks with increasing voids based on a procedure from Smay et. al. [58]. Sag distance was measured with a Keyence One-shot 3D Measurement Macroscopic VR-3200. The surface of each sample was sprayed with zinc stearate to dull the reflection. Compression specimens were tested on an Instron 5943 equipped with a 1- kN load cell. Samples were tested under uniaxial tensile loading crosshead speed of 12 mm/min .

X-ray Computed Tomography

(NSI) X25 computed tomography (CT) system. The Hamamatsu L12161-07 x-ray source was operated with a high voltage of 70 kV , a current of 140 μA , and the “small” focal spot selected. No beam filter was used. The Varian panel detector was operated in its “01.1x10.pFVG1” mode, at a 1.5 frame-per-second integration rate, with a 1920-pixel x 1536-pixel region of interest, with its native pixel size of 127 μm , and with 16-bit data values. The system geometry consisted of source-to-object and source-to-detector distances of 80 and 640 mm , respectively. The corresponding magnification factor was 8X and effective pixel size at the sample was 15.88 μm . The estimated source focal spot blur was 0.54 pixel.

The sample fixturing consisted of a custom additively-manufactured plastic component – a base (“C-2”), a Teflon cylindrical post, and a capped plastic vial. A helical CT scan was executed and consisted of 11,520 projections over a height of 87 *mm* with an effective angular sampling rate of 1/4 degree. Frame averaging of 2 was used for the projections. The data sets were reconstructed using the NSI-provided commercial software package. No beam-hardening correction was used, the range of gray levels in the reconstructed data was selected to provide good image contrast for the sample material, and the reconstructed data was exported in “stacks” of 8-bit vertical slices and both 8- and 16-bit horizontal slices.

Dynamic Mechanical Analysis

DMTA measurements were performed on a TA Instruments Ares G2 rheometer in tensile torsion mode at a frequency of 1*Hz* and a strain of 0.1%. Heating was applied at a 5°C/*min* ramp rate when ramping from -110°C to the end temperature of 50°C in order to determine the glass transition temperature (T_g) of the elastomeric samples with and without the minor acrylate component crosslinking. Sample dimensions and procedure for DMTA followed ASTM D7028.

Tensile Testing

Tensile specimens (ASTM D-638 Type IV) were tested on Instron 5943 equipped with a 1-*kN* load cell. Samples were tested under uniaxial tensile loading crosshead speed of 1 *mm/min*.

Cell Growth and Fluorescence Microscope

Extracellular matrix (ECM) protein and plasma treatment was performed by heat treating the porous UGAP material with heat at 200 °C for 2 hours, followed by 150 W for 3 minutes of O₂ plasma in Nordson Asher chamber. Once completed, samples were immediately submerged in sterile ECM protein solution (0.1wt% Fibronectin from human plasma, Sigma, F0895) and incubated at 4°C overnight. Samples were dried and seeded with normal human lung fibroblasts in fibroblast growth media-2 (FGM-2, Lonza CC-3132). Seeded samples were cultured in a humidified incubator maintained at 37°C with 5% CO₂ for one week.

3.3 Results & Discussion

Chemical Framework

We demonstrate the UGAP ink’s versatility in a combination of complex printing motifs to manufacture objects exhibiting a range of mechanical properties while maintaining

spatial print control. On-demand UV exposure chemically bonds the minor concentration of low molecular acrylate monomer, ethylene glycol phenyl ethyl acrylate (EGPEA) and 1,6-hexanediol diacrylate (HDDA), to the backbone of an isocyanate functionalized UGAP prepolymer (EBECRYL[®] 4396) to provide sufficient shape retention through stiffening while maintaining interfilament adhesion. A subsequent post-process heat cure establishes the dominant urethane network as polycaprolactone diol (8025D) and triol (8025E) bond to isocyanates without disrupting the printed shape.

Formulation

The optimized formulation using 16.67wt% low molecular weight acrylate and 3.33wt% Irgacure 819 was formulated based on sag testing presented below in Fig. 3.2, where increasing the acrylate diluents decreased ink yield stress and increased sagging. Polyol components and corresponding ratios were chosen to maximize the cross-link density while forming a stoichiometric ratio with the isocyanates in the UGAP pre-polymer. Rheology of UGAP formulations (Tab. 3.1) was varied by concentrations of either fumed silica (UGAP-FS) or NaCl (UGAP-NaCl) fillers. Mechanical testing was performed on UGAP-FS Type IV dog bones with directionally dependent toolpaths (Fig. 3.3A). The resulting stress-strain curves and the tensile modulus indicate that in-situ curing does not inhibit interfilament adhesion and that printing orientation does not result in anisotropic material properties (Fig. 3.3B) as indicated in past work [44]. Alternative fillers such as alumina, were easily incorporated into the UGAP base resin due to the use of acrylate diluents (Fig. 3.4).

8025D	8025E	HDDA	EGPEA	Irgacure 819	EBECRYL 4396
40.94%	4.08%	1.68%	15%	3.33%	35.07%

Table 3.1: Explicit formulation for urethane grafted acrylate polymer (UGAP) resin in fractional wt%.

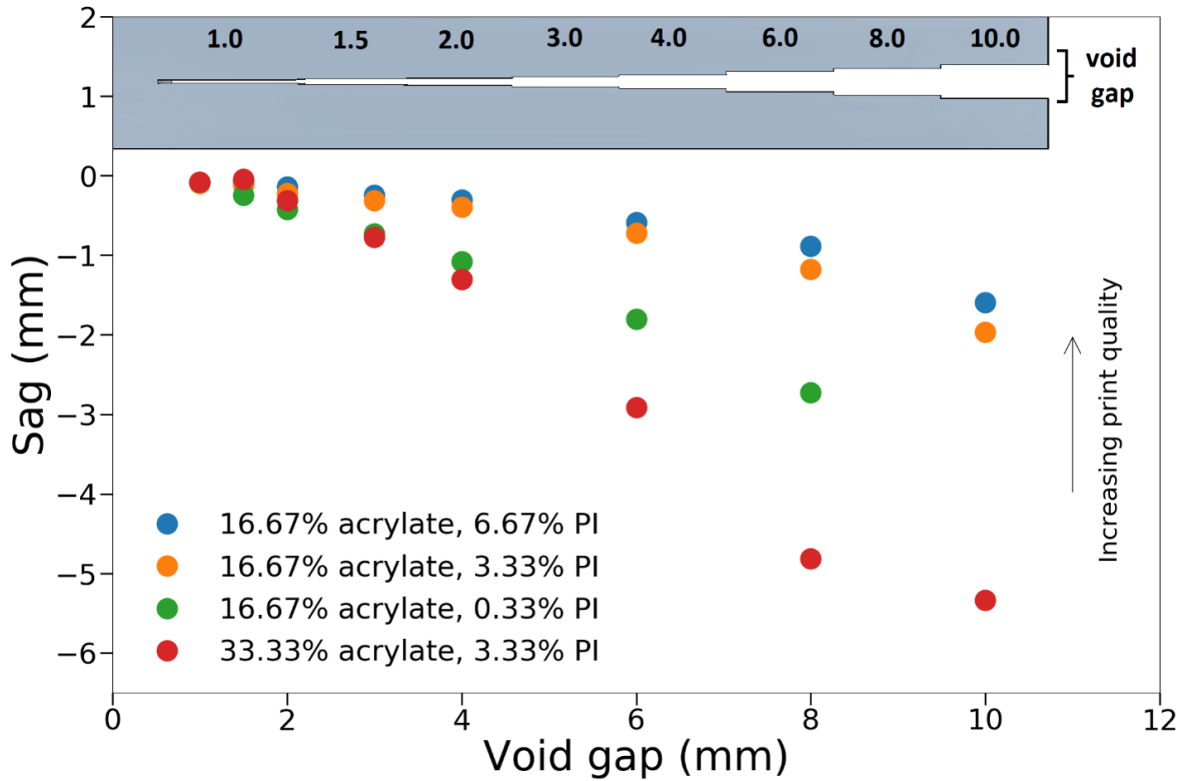


Figure 3.2: Spanning tests of 70wt% UGAP-NaCl with variable amounts of acrylate and photoinitiator (PI) overlaying a model of a spanning block (varied wt% with respect to resin without solids). Diagram at the top of the graph represents the spanning test block with increasing void gaps in mm.

Geometrically Complex Design Space

The UV curing relaxes the rheological and dimensional constraints for geometrically complex objects (Fig. 3.5A). By increasing filament stiffness during printing, shape retention is maintained allowing for unsupported printing in free space of a millimeter-scale helix (Fig. 3.5B). Rapid stiffening also extended printing scales permitting large-format and high-aspect ratio designs like the #3DBenchy (Fig. 3.5C), a notoriously difficult-to-print benchmark for fused-deposition modelling as well as the steep overhangs required for a gyroid (Fig. 3.5D). Structural composition control (Fig. 3.5E) was achieved through macroscopic voids from toolpath variations (Fig. 3.5F) and microscopic voids through post-process leaching of NaCl resulting in microscopic pores patterned by NaCl loading and particle size distribution (Fig. 3.5G and Fig. 3.6) [59]. The combination of spatial print control and solids leaching methodologies allowed structures exhibiting hierarchical porosity (Fig. 3.5H) forming porous UGAP (UGAP-pore). Utilizing multi-material printing [42, 60] (two different UGAP based inks) significantly expanded the application space allowing for property local-

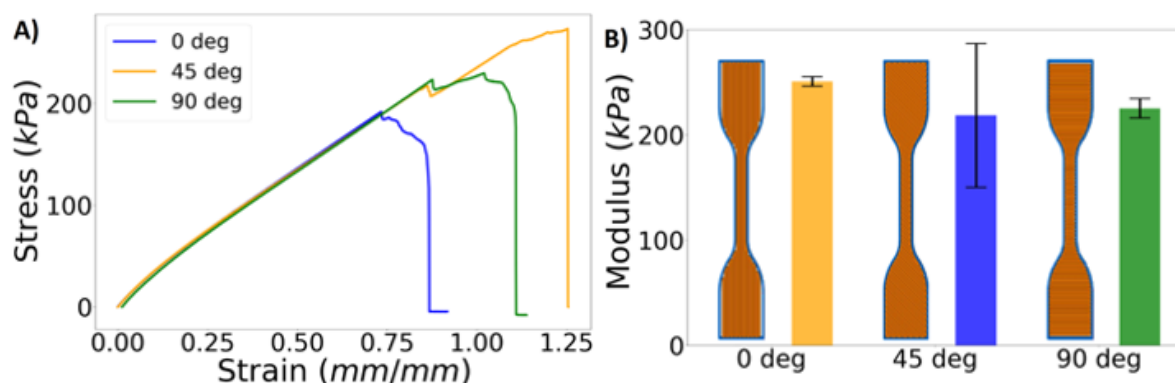


Figure 3.3: Tensile test of Type IV dog bones printed with different toolpaths. A & B) The stress-strain curves and corresponding tensile moduli of specimens printed with 0-, 45-, and 90-degree orientations. Tensile samples failed slowly in multiple areas, as indicated by the step-wise failure in (A).

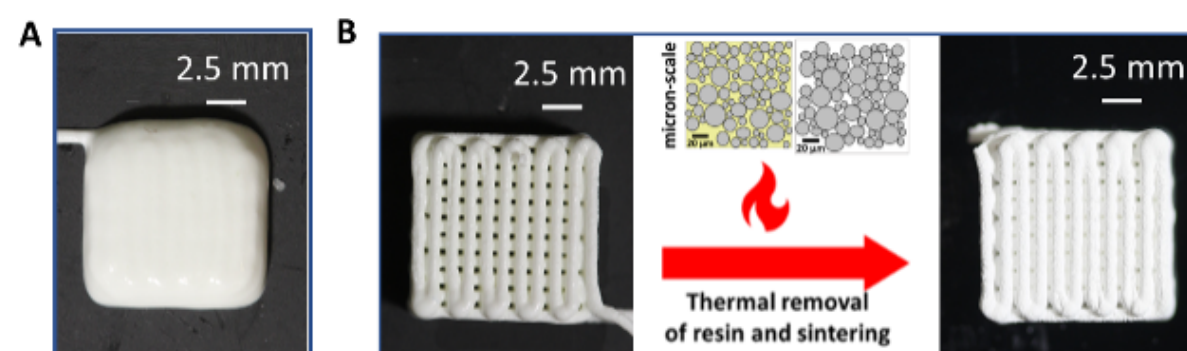


Figure 3.4: Printing Al₂O₃ filled UGAP. A) Printing a lattice with UGAP-Al₂O₃ without UV light. B) Printing a lattice with UGAP-Al₂O₃ with UV light and subsequently thermally removing UGAP and sintering Al₂O₃ particles.

ization throughout the part (Fig. 3.5I). Concurrent printing of UGAP-FS or UGAP-NaCl inks produced structures with spatially resolved mechanical responses that were chemically bonded forming a resilient composite (Fig. 3.5J). UGAP-based inks further expanded the fabrication space as support material for inks ill-suited for DIW techniques (Fig. 3.5K). Modulating the UV light intensity and exposure allows spatio-temporal control of printed filament stiffness throughout the toolpath such that shape retention and volume filling behavior can be utilized in the same print (Figures Fig. 3.5L, Fig. 3.5M). Such range of utility permits a variety of printing motifs (Fig. 3.5N) and complex internal architectures.

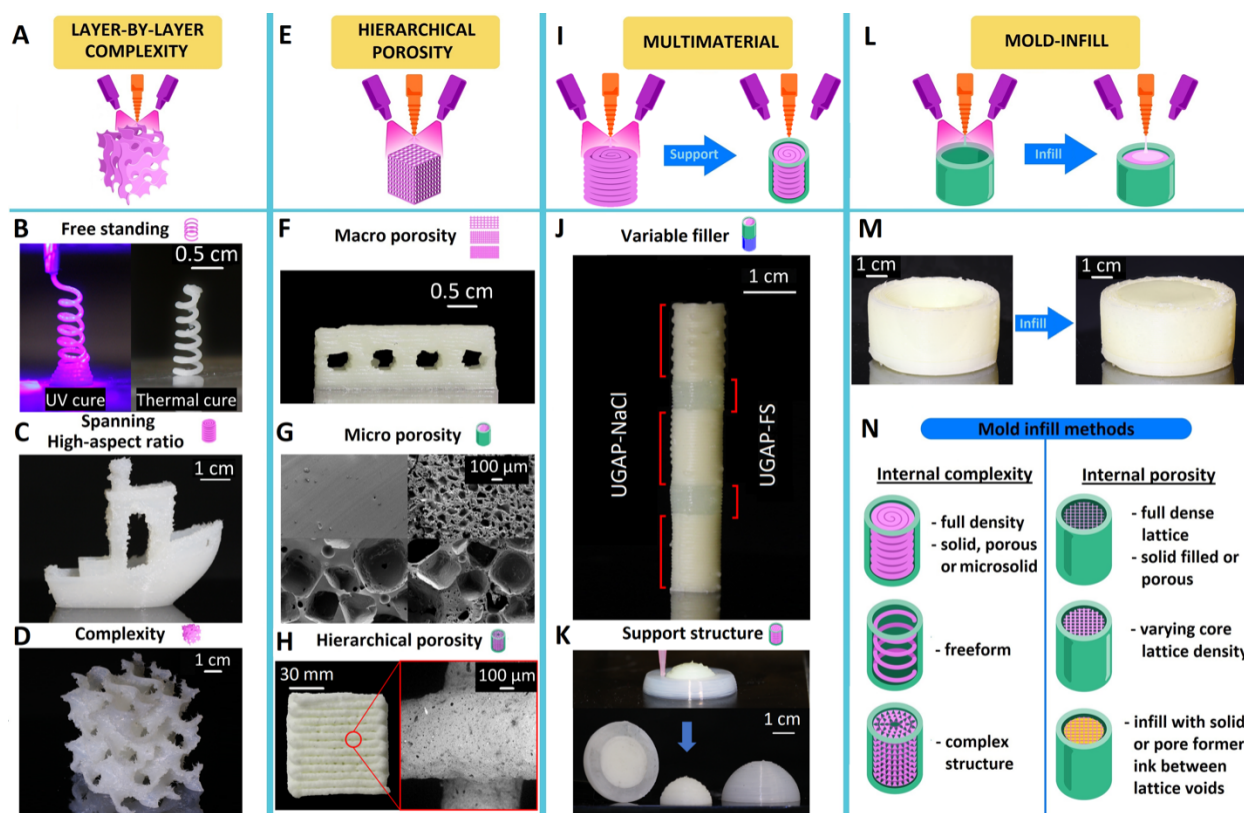


Figure 3.5: Schematic of possible printing combinations with UGAP DIW printing. A) Layer-by-layer complexity via intermediate UV-cure, B) Millimeter scale helix, C) 3D Benchie, D) Gyroid printed using UGAP inks filled with fumed silica (UGAP-FS), E) Porosity on variable length scales, F) Designed macro-porosity, G) SEM images of micro-porosity produced by leaching NaCl particles, H) Hierarchical combination of macro and micro-porosity, I) multi-material printing, J) a cylinder co-printed, UGAP-NaCl and 10wt% UGAP-FS, and K) UGAP-FS support structure for silicone ink. L) Infill fabrication for: M) printing an outer mold and filling to full density, and N) Multiple methods of printing an outer shell with various internal structure.

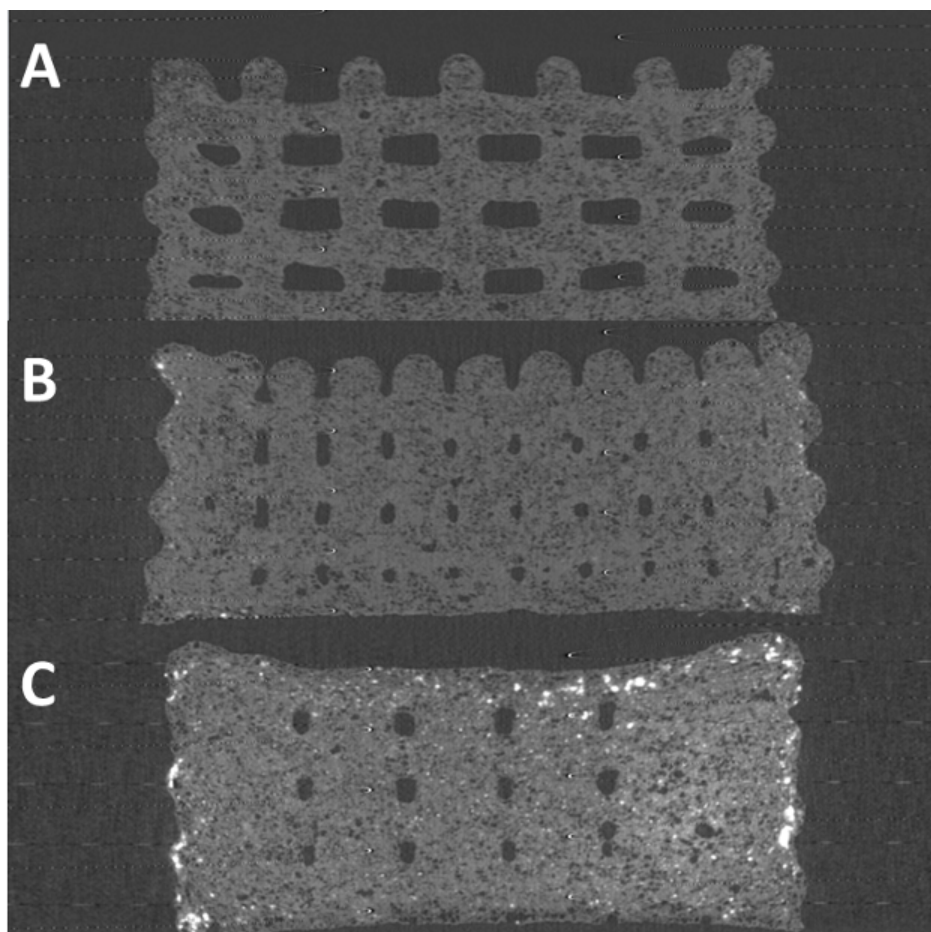


Figure 3.6: X-ray CT of printed samples leached of salt particles with lattice densities of A) 40%, B) 60%, and C) 80% indicating full removal of salt particles and the resulting hierarchical porosity.

Rheology

Incorporating the minor acrylate components EGPEA, HDDA, and including the acrylate groups on the EBECRYL[®] 4396 not only assists in on-demand UV curing of complex shapes but also enables printing of a wide range of ink yield stresses. The 70wt% micron-scale bimodal distribution of NaCl in UGAP-NaCl and increasing nano scale silica content from 5 to 15wt% in UGAP-FS ranged in yield stress from 11.5 and 8042 *Pa* (Fig. 3.7A). When printed without UV light, filaments of the lower yield stress ink UGAP-NaCl (11.5 *Pa*) collapsed along the central axis due to the weight of successive layers (Fig. 3.7B). Conversely, the higher yield stress of 15wt% UGAP-FS (8042 *Pa*) allowed taller lattices without UV light. However, even the 15wt% UGAP-FS high-aspect-ratio tower collapsed mid print due to lateral printing loads. With on-demand UV exposure, the same tower was printed with both

UGAP-NaCl and 15wt% UGAP-FS to the physical limits of our printer (8.7 cm in height) with minimal dimensional differences between the two parts (dimensional comparison shown Fig. 3.8). As shown, NaCl and silica UGAP inks all demonstrated successful DIW printing despite the rheological differences.

UV Kinetics

On-demand UV curing during printing enabled rapid stiffening of the minor acrylate component through characterization of UGAP curing kinetics. UGAP-NaCl UV-rheology results demonstrate the rapid kinetics with increasing light intensity (Fig. 3.7C), attaining a full-depth cure within 60 seconds with 0.6 mW/cm^2 flood exposure. For printing speeds of 15 mm/s , sufficient stiffening was observed within approximately 10 seconds of UV exposure creating optimum filament-filament adhesion and shape retention. This observation indicates a lack of full filament depth cure caused by solid-filler-induced light scattering. Full acrylate cure originates from surface cure (Fig. 3.9) relying on chain-growth radical-based kinetics to propagate towards the filament core where the UV light could not penetrate (Fig. 3.10). The rapid filament stiffening allowed parts with filament overhangs (i.e., hollow cone with each subsequent layer off-set by half filament) to be printed with UGAP-NaCl and 15wt% UGAP-FS (Fig. 3.7D). Filament gap spanning tests (Fig. 3.7E) further confirmed the stiffening rate due to UV exposure at the 15 mm/s print speed was sufficient as the low yield stress UGAP-NaCl formulation was able to span the largest 10 mm gap with minor sag only when exposed to UV light. Conversely, the UGAP-FS inks were able to span the increasing gaps with and without the aid of the UV light. The UGAP-NaCl and 15wt% UGAP-FS performed similarly when printing overhangs with UV light. Fig. 3.7F shows an inverted pyramid with each side having different angles to demonstrate sag resistance in an overhang.

Mechanical Properties

After thermally curing the major urethane component at 80°C objects did not exhibit dimensional changes indicating a suitably crosslinked matrix created from UV-induced curing was stable enough to resist thermal induced dimensional change. Fourier Transform IR (FTIR) characterization indicates complete conversion of acrylate and urethane components (Fig. 3.11), while dynamical mechanical analysis (DMA) reveals a single but somewhat broad glass transition temperature (T_g) peak centered at -25°C to signify a fully covalently linked UGAP elastomer network (Fig. 3.12).

While the elastomeric mechanical properties are dictated by the thermal curing of the major urethane pre-polymer component, modifying filler and filler concentrations provided tunable stiffness control. Comparing compression moduli of 15wt% UGAP-FS to 70wt% UGAP-NaCl printed lattices showed a 1.5 times lower modulus in UGAP-FS filled parts to that of UGAP-NaCl, going from 1.24 MPa to 0.80 MPa , respectively (Fig. 3.7G). Since the polycaprolactone urethane pre-polymer used in UGAP allowed for the cured network to be water permeable, the NaCl filler was gravimetrically extractable with water post printing and

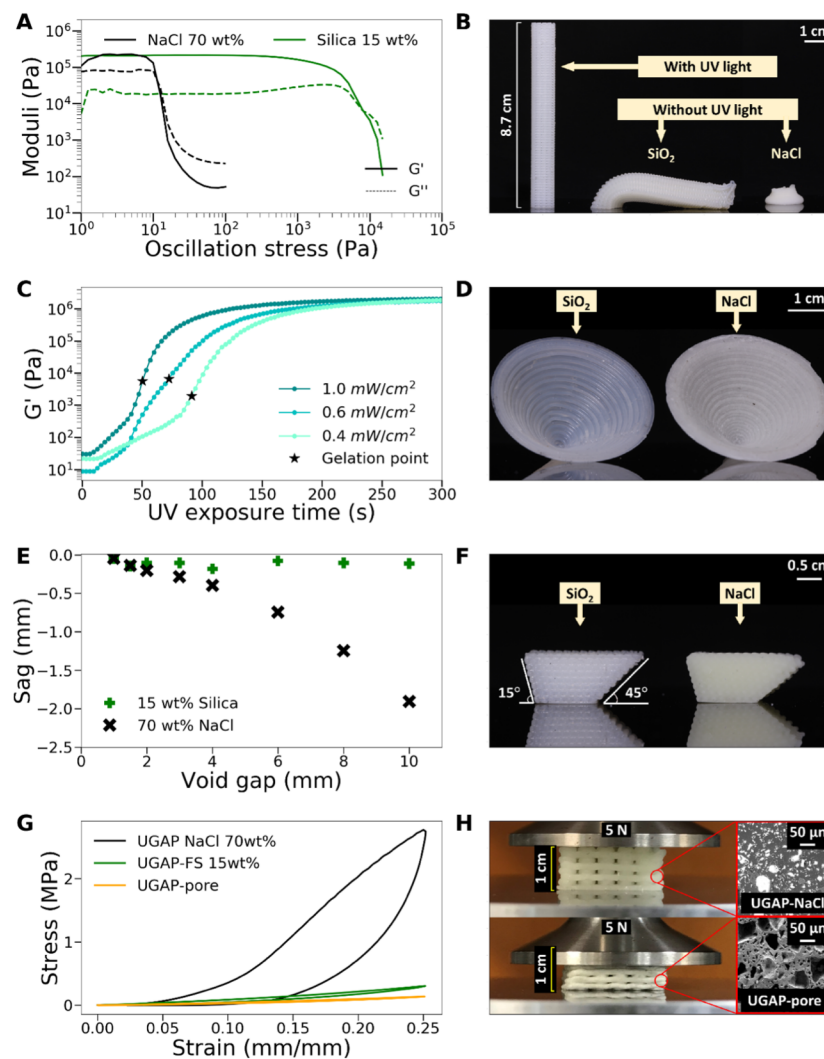


Figure 3.7: A) Yield stress comparison of UGAP resins filled with 70wt% NaCl and 15wt% fumed silica. B) 1 cm x 1 cm towers printed as tall as possible with UGAP-NaCl and UGAP-FS with and without UV light. C) UV-rheology storage modulus for each UGAP ink with increasing light intensity. D) Hollow cones with UGAP-NaCl and 15wt% UGAP-FS demonstrating overhangs. E) Spanning tests measuring sag distance for UGAP inks. F) Inverted pyramids with 15° and 45° side angles printed with UGAP-NaCl and UGAP-FS. G) Printed lattices withstanding 25% strain. H) Static 5 N loading on UGAP-NaCl and UGAP-pore lattices with same filament spacing, SEM images of UGAP-NaCl and UGAP-pore samples.

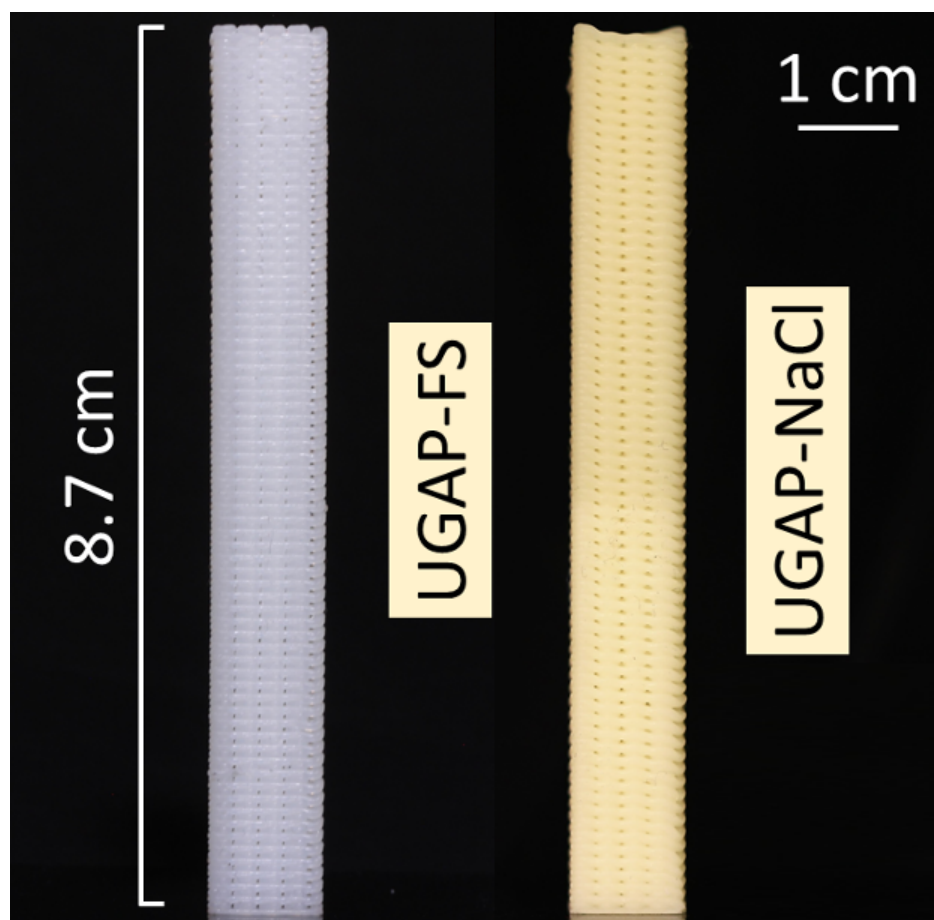


Figure 3.8: Direct visual comparison of 1 cm x 1 cm towers printed to a height of 8.7 cm with 15wt% UGAP-FS and UGAP-NaCl.

curing. UGAP-pore lattice compression modulus was 3.4 times lower than the NaCl counterpart with a modulus of 0.37 MPa due to the resultant porous microstructure. UGAP-NaCl and UGAP-pore lattices are shown in Fig. 3.7H withstanding a 5 N load demonstrating increased compressibility due to the porous microstructure, while UGAP-pore lattices exhibited repeated compression without permanent deformation (Fig. 3.13). The large range in compressive moduli within the same polymeric network provides opportunities for localized control of mechanical responses required in functional materials not possible with traditional DIW.

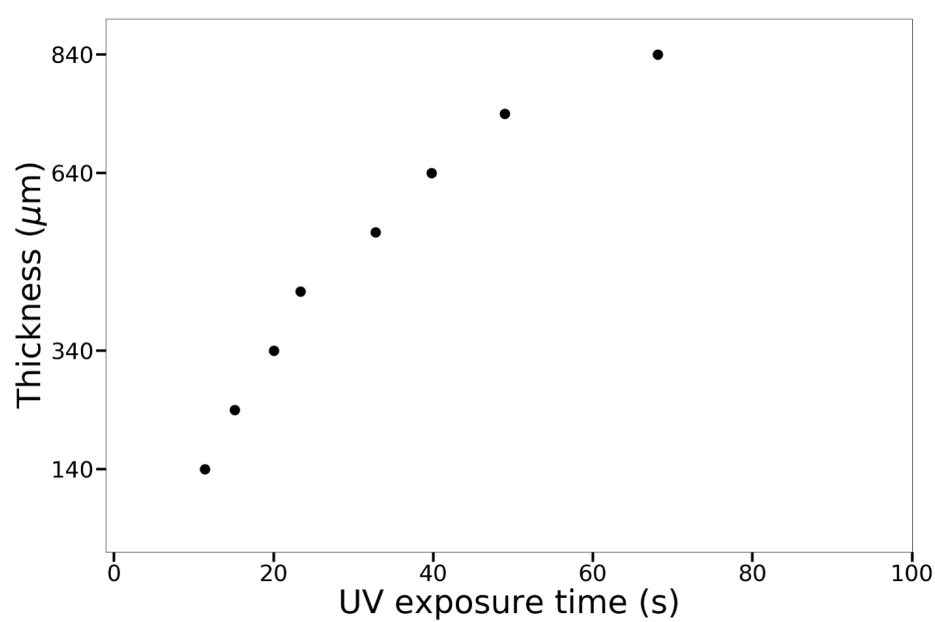


Figure 3.9: Time to gelation at a consistent $0.6mW/cm^2$ for varying parallel plate gap thicknesses where gelation is defined when $G' = G''$

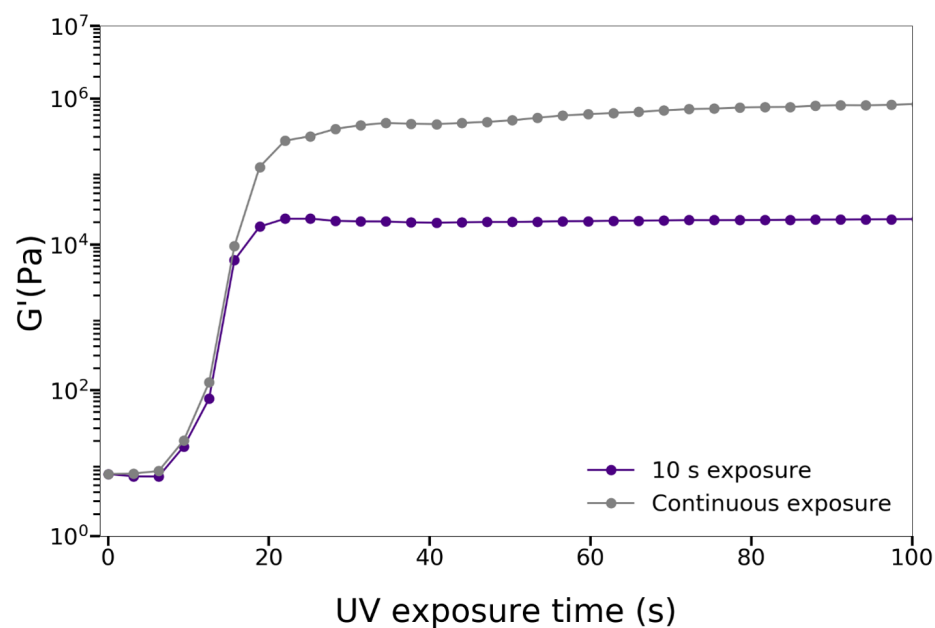


Figure 3.10: Gelation with time for a $140\mu m$ sample continuously exposed to $0.6mW/cm^2$ UV light vs samples exposed to only 10 seconds of UV light, indicating continued dark curing.

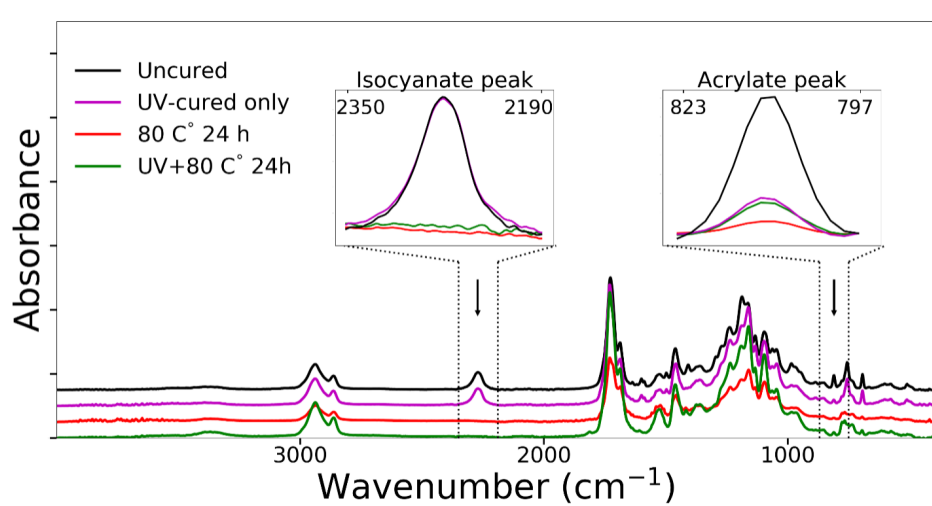


Figure 3.11: Fourier Transform (FT) IR of UGAP resin samples (i) uncured, (ii) UV-only cured, (iii) heat-only cured at 80 °C, and (iv) UV, then heat cure. The acrylate subplot indicates a conversion of 81% (as measured from the maximum points within the bounds of 797 and 823 of the uncured and UV+80 °C data).

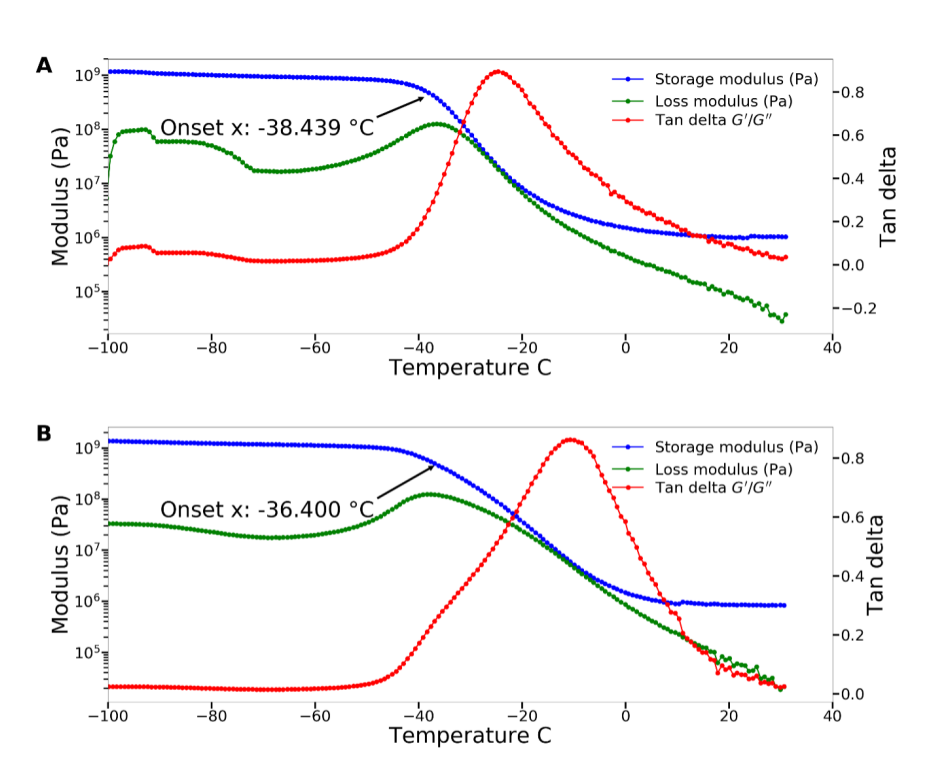


Figure 3.12: A) Just the urethane components without the minor acrylate cross-linker is cured with heat to confirm a single elastomeric glass transition temperature. B) Adding in the minor acrylate component and dual curing with UV and heat maintains a single glass transition temperature while maintaining a dominant elastomeric property.

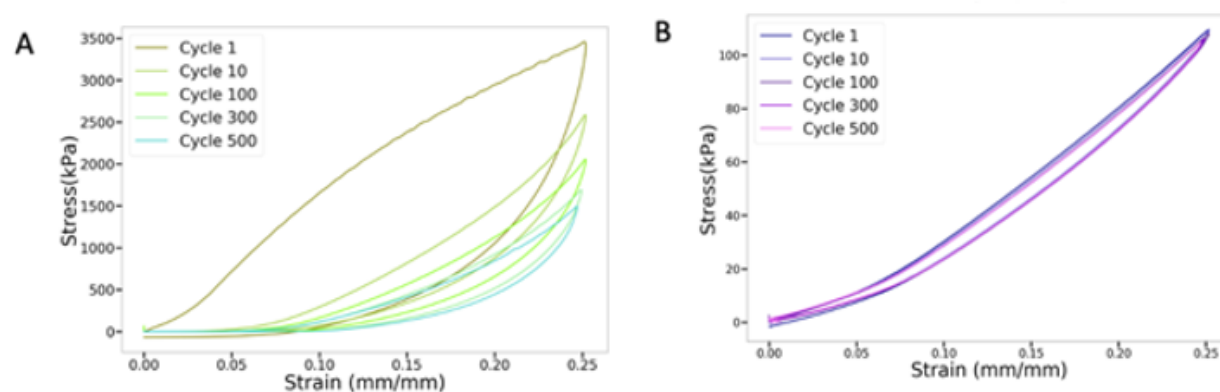


Figure 3.13: Iterative compression cycles of 1 cm x 1 cm x 0.5 cm 60% lattice density structures printed with A) 70wt% UGAP-NaCl and B) leached NaCl pores.

3.4 Applications

Soft Robotics

Elastomeric and functional materials are showing greater promise for soft-robotic applications. Shape memory motion of elastomeric materials is triggered by heating past the T_g of the elastomer[61]. At temperatures above the T_g , a material can be placed into a desired configuration and “programmed” by cooling below the T_g . The material will remain in this programmed state until heated through its T_g , at which time it will return to its original shape. Demonstrating this phenomenon with UGAP-FS, a printed gripper (Fig. 3.14A) is shown actuating from the programmed state (holding a ball) to the original shape, releasing the ball. To demonstrate more complex print structures, a soft-hard segmented finger was designed containing in-fill motifs outlined in Fig. 3.14J. To simulate the motion of an opening finger, three sections of the finger were printed with varying lattice densities with UGAP-pore to modulate flexibility separated by two fully dense joints using UGAP-FS. The hierarchical structure of the pores facilitated the variation of compressibility (Fig. 3.14B) allowing for segmented flexibility during actuation. Actuation was achieved by heating the material from the programmed state (curled) to the original state (straight). As shown in Fig. 3.14C, the degree of flex between each joint was limited by the macro porosity dictated by the infilled lattice density.

Cell Scaffolding

In a final dimension of this work, it was found that the UGAP-pore microstructure facilitated growth of human lung fibroblasts (Fig. 3.14D) indicating that UGAP could potentially be used as an environment for cell growth in future work and serve as a framework for actuating scaffolds for biological applications.

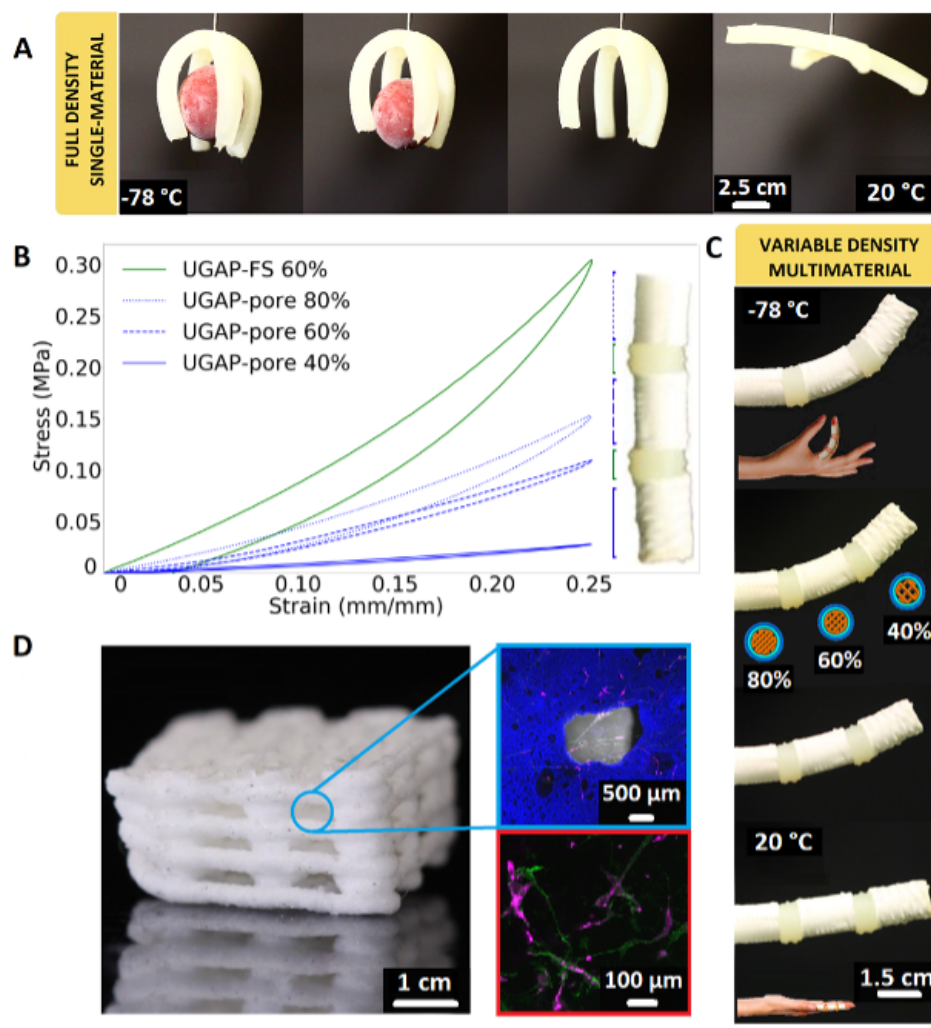


Figure 3.14: A) Printed gripper holding ball in “programmed” configuration and releasing to the original configuration with heating. B) Compressive mechanical properties of inks in multimaterial cylinder UGAP-pore with lattice densities 40, 60 & 80% and UGAP-FS segments. C) Multimaterial cylinder in the programmed state actuating to the original state. D) Printed porous structures facilitating the growth of human lung fibroblasts. Fluorescence microscope image showing the microporous background (blue), fibroblast cell membrane stain (pink), and the actin stain (green).

3.5 Conclusion

Introducing spatial control using a dual-cure low-molecular weight acrylate component UGAP ink is a significant advancement for DIW, demonstrating a previously unattainable broad range of printing motifs with tunable elastomeric properties. As the acrylate diluent afforded lower initial ink viscosities, a wide range of rheological yield stress from 11.5 Pa and 8042 Pa was printed with the incorporation of dissimilar 70wt% NaCl and 15wt% fumed silica solid fillers, respectively, and with the assistance of on-demand UV curing. Rapid stiffening with UV light exposure allowed for near instantaneous rheology modification and shape retention in this range of printable inks, maintaining dimensional print accuracy for complex prints such as free form helices, gyroids and lattice towers reaching the z-height limit of the printer. With the assistance of UV curing kinetics characterization, finer tuning of spatial print control with on-demand printing was achieved by tuning the light exposure intensity and understanding the extent of dark-cure polymerization. Since previous acrylate-based UV curable DIW inks exhibited brittle mechanical properties or poor interfilament adhesion, developing the UGAP ink with a minor concentration of acrylate diluent to form an interpenetrating network with the major urethane prepolymers provided parts with elastomeric properties and robust interfilament bonding. The resultant compression moduli of filled printed lattice parts spanned an order of magnitude, from 1.24 MPa in UGAP-NaCl parts to 0.80 MPa in UGAP-FS and further tuning to 0.37 MPa of UGAP-pore by leaching out NaCl. This mechanical property range within the same base UGAP ink in conjunction with the low elastomeric T_g provides shape memory material properties, setting the stage for applications in soft robotics, which was also successfully demonstrated. Achieving spatial print control in DIW inks with superior and tunable elastomeric properties raises the benchmark for future UV curable ink formulations and opens the door to increasing the functionality in printed applications.

Chapter 4

Modeling and Simulation of UV-Curable Materials

“All stable processes we shall predict. All unstable processes we shall control.”

- John Von Neumann

The source code for simulations and experiments in this chapter can be found in the following GitHub repository:

https://github.com/bmhowell/ugap_simulation

4.1 Introduction

In the field of modern manufacturing processes, the UV curing of particle-laden materials has emerged as a vital and versatile field with a broad range of applications across industries such as coatings, dental fillers and implants, electronics, and additive manufacturing. The ability to rapidly solidify UV-curable materials embedded with particles offers the advantage of precise control over material deposition leading to material waste reduction and improved production speed. Advancements in computing power have enabled rapid simulation of reaction-dependent processes providing the capability to comprehensively analyze and effectively manage complex phenomena such as heat generation and propagation of chain polymerization. The intricate kinetics of the curing process are influenced by a myriad of material properties and processing parameters, and can be further understood through numerical modeling and simulation techniques. This paper aims to provide a computational framework allowing engineers and scientists to quickly formulate and analyze UV-curing dependent manufacturing processes to achieve a reduction in material waste and overall operating costs.

Traditionally, various applications of polymer manufacturing require formulation engineers to rapidly iterate over different combinations of materials and experimentally test

for desired outcomes. Often, formulation engineers must combine a mixture of solid-filler particles that provide structural integrity and material strength, alongside a binder serving as the polymer matrix holding the composite material together. Solid-filler design choices include particle type, size and volume fraction that dictate final mechanical, thermal and electrical properties. The binder is composed of viscous pre-polymers that can be synthetically designed for reactivity and final material properties. One common binder choice for UV-curable applications are acrylate pre-polymers (e.g., monomers, acrylate-functionalized polymers, etc.) with a small concentration (0.5-5 wt.%) of photo-initiator (PI). When exposed to a specific wavelength of UV light, the chosen PI will dissociate into unstable free radicals which subsequently react with the acrylates, initializing the polymerization process. The free radical from the PI is transferred to the acrylate group, which is now free to form a chain reaction with other available monomers growing the acrylate polymer chain until a termination reaction. The relative rate at which either of these reactions progress dictates the final properties of the polymer such as chain length, entanglement and crosslinking, as well as total degree of cure.

The final material is greatly determined by the quality of binder cure, which in turn is intricately influenced by a variety of process parameters such as ambient temperature, UV intensity, and exposure time, as well as material properties such as binder viscosity, heat capacity, diffusivity, particle size and volume fraction. The ambient temperature plays a critical role by affecting reaction rates, diffusivity, and overall degree and quality of cure. While higher temperatures can accelerate the curing process, increased temperatures can also lead to early termination in polymerization resulting in shorter-chained polymers that can lead to accelerated aging. UV intensity acts as the initiator of the curing reaction, triggering PIs that initiate polymerization of starting pre-polymer acrylates. Modulating UV-intensity controls the rate at which initiation occurs, but can simultaneously cause unwanted effects such as increased temperatures or hazardous working conditions. Material properties, such as the viscosity of the uncured mixture, the reactivity of the binder polymer, and the size and distribution of the filler particles, also significantly impact the curing kinetics. As curing progresses, the viscosity increases and limits the mobility (diffusion) of the polymer chains, affecting chain growth and termination leaving radical byproducts that can serve as initiation sites for degradation. At a constant volume fraction of the total material, a smaller particle size distribution yields a higher surface area exposure to the binder affecting the internal temperature distribution from heat of reaction, UV exposure, and chemical diffusion. The resulting reactivity of the binder polymer influences the rate at which each reaction in the step-wise polymerization occurs.

The highly-coupled physics and chemistry in both space and time necessitates many experiments, a large amount of material, and time to effectively optimize the formulation for a desired material performance. Currently, experimental design approaches such as grid, random, factorial or statistical design of experiments are the de facto methods for testing and determining best combination of materials. The utilization of numerical modeling and simulation techniques provides a cost-effective method to study of the effects of ambient heat, UV intensity, and material properties on the curing kinetics as an aid to direct experimental

efforts. This computational framework aims to facilitate waste reduction, improve operating efficiency, increase understanding of manufacturing dynamics and empowers the development of advanced manufacturing processes with enhanced material performance and reliability.

Example Application and Objectives of this Chapter

An emerging area of additive manufacturing (AM) is the use of UV-curable “inks” for extrusion-based methods such as direct-ink-writing (DIW). These ink formulations contain acrylate monomers and PIs at relatively low concentrations that undergo photo-polymerization upon exposure to UV light. This intermediate cure establishes a network of cured acrylate components that partially solidifies the filament upon extrusion and improves spatial resolution and repeatability in printed objects.

However, one concern of manufacturers in such applications is the incomplete conversion of starting monomers to final desired polymers attributed to initial conditions, material properties, process parameters and object toolpath. The presence of unreacted monomers and polymers can significantly impact the robustness of the final product, rendering it susceptible to environmental conditions. Specifically, UV-curable acrylates are especially susceptible to aging due to the complexity of their polymerization mechanism, which generates radical byproducts that may serve as initiation sites for degradation. The conversion of starting monomers and radicals to the final material can be influenced by various processing parameters, such as ambient temperature, UV intensity, and exposure time. Additionally material properties including the volume fraction and size of particles, which can affect the spatial reaction profile and overall heat distribution. To fully leverage the potential of UV-assisted DIW in AM, it is necessary to leverage numerical modeling and simulation techniques to develop design rules for upscaling to industrial manufacturing levels.

The objective of this chapter is to develop a numerical model that captures the chemical kinetics of UV-curing acrylate reactions in particle-laden inks. The model will enable the investigation of diverse processing parameters and material formulations, specifically focusing on their influence on the conversion of acrylate monomers to polymers. The foundation of our numerical model relies on a finite difference discretization method of a system of coupled reaction-diffusion equations, a modified Beer-Lambert law to describe UV light penetration and provides a computational framework that can rapidly and inexpensively explore material formulations and process parameters.

4.2 Thermo-Chemical Continuum Formulation

This model formulation addresses the thermal and chemical behaviors observed during the UV-curing of particulate-laden acrylate-based resins. During the manufacturing process, dual-curing inks (see Chapter 3) are dispensed onto a build plate following a toolpath that layer-by-layer builds a desired object. As the material is extruded, UV light directed near the tip of the nozzle initiates *in-situ* UV curing. As the light penetrates the filament, the

components of the ink absorb energy from the irradiation, and the intensity of light decays exponentially with depth. This energy activates PIs, which in turn activate acrylate monomers, which go on to react with other monomers. These chains can continue to propagate or terminate depending on the depletion of monomers or solidification of the polymer inhibiting mobility of reactants. The presence of solid additives with random size and location, as well as the decay of light intensity throughout the material, causes concentration and temperature gradients that also affect the spatial/temporal evolution of the polymer.

The primary aim of this study is to develop a user-friendly computational framework that researchers can readily implement as an efficient design tool; Specifically, I employ a Finite Difference Time Domain (FDTD) approach [3, 6] to model the material's internal thermal and chemical fields. Temporal discretization uses the trapezoidal method for second-order time accuracy, while spatial discretization employs a center-difference approximation for second-order spatial accuracy. To prioritize computational efficiency for high performance computing on local devices such as laptops or desktops, a uniform and dense spatial grid representing microstructure of material is stored within an array to avoid memory-intensive matrix operations. The presentation is divided into four main sections: (1) characterization of UV input, (2) model formulations for each chemical and thermal field, (3) spatial and temporal discretization for numerical solution, and (4) computation examples demonstrating model application.

The following section seeks to develop kinetic models for each reaction step in the photopolymerization process, as well as the coupled heat and diffusion equations that describe the temperature and concentration gradients.

Attenuation of UV Light

In this approach, an external UV source induces the dissociation of PIs as the material is extruded. The irradiation intensity can be controlled in real time to obtain precise curing rates, which is particularly pertinent for complex areas in a printed object that require extrusion over overhangs (see Fig. 3.7F). However, for the purposes of this initial work, UV intensity is held constant in time. The rate at which the energy is absorbed is determined by the intensity of the light source, which decreases exponentially with depth. The decay of intensity with depth is commonly modeled using the Beer-Lambert law:

$$I(z) = I_0 e^{-\epsilon z} \quad (4.1)$$

where I_0 is the intensity of the UV light at the surface, ϵ is the material absorptivity, and z is a positive value that represents depth of penetration below the surface. The first law of thermodynamics implies that the rate at which the intensity decays with depth must be equivalent to the rate at which energy is absorbed by the surrounding material at that depth:

$$I_{abs}(z) = -\frac{dI}{dz} = \epsilon I_0 e^{-\epsilon z}. \quad (4.2)$$

The overall objective is to identify reaction rate trends given environmental and processing parameters on bulk materials. The above conclusion assumes that the effects of light scattering and refraction are negligible relative to energy absorption. These assumptions become critical for molecular- to atomistic-scale simulations such as molecular dynamics or density functional theory and should be considered when examining the physics on a finer scale.

Photo-Polymerization of Acrylates

As light diffuses throughout the unreacted material, UV light activates the polymerization of monomers, leading to the formation of a polymeric matrix. Acrylates are a common class of chemicals for this process because they polymerize readily upon exposure to free radicals. The step-wise reactions for these acrylate-based polymerization schemes that are applied to the work in the section can be summarized in four basic steps:

1. Dissociation: The PI is activated when exposed to a UV light source with a specific wavelength, resulting in two radicalized components (Fig. 4.1A). This species is referred to as the “primary radical” and labeled as $PI\cdot$ throughout this work.
2. Initiation: The radicalized PI components come in contact with monomers M and attack the double bond between the methylene and ester functional groups, leaving a radicalized electron on the carbon (Fig. 4.1B). This species is referred to as the “primary chain radical” and labeled as $M\cdot$ throughout this work.
3. Propagation: The radicalized acrylate group now performs the same attack on other acrylate groups that it comes in contact with, propagating the chain and polymerizing the monomers (Fig. 4.1C). This species is referred to as the “active chain radical” and labeled as $M_n\cdot$ throughout this work.
4. Termination: When two radicalized acrylate groups come in contact with each other, the radicals form a bond and terminate the polymerization, fixing the chain length (Fig. 4.1D).

A summary of the simplifying assumptions used for this model is listed here:

- PI dissociates into two primary radicals with equivalent mobility and reactivity;
- reactivity of each term assumes constant and equal chain length (simplifying assumption for macroscopic trends);
- bimolecular termination dominates the effects of PI radical termination;
- UV light source is monochromatic (as used in the experimental section in Chapter 3);

- temperature fluctuations are low enough to not affect the density of any constituents within the composite material;
- each reaction is irreversible.

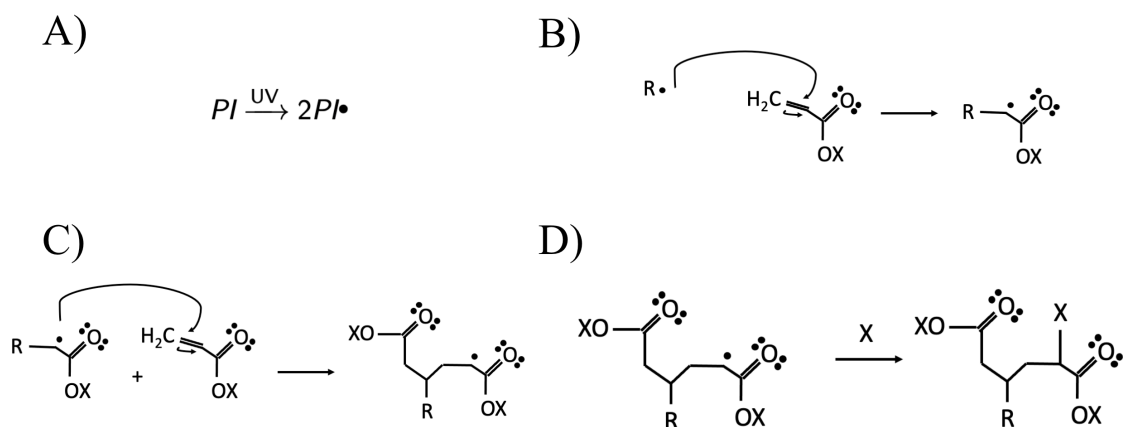


Figure 4.1: The step-wise reaction mechanism for radical initiated acrylate reactions: A) Dissociation, B) Initiation, C) Propagation, D) Termination.

The propagation and evolution of these reactions can vary substantially between experiments and even more so in industrial applications. At the molecular level, concentration gradients can create unequal probability distributions of reactivity during polymerization since monomers must come in contact with radicalized groups for reactions to occur. This leads to slight variations in the polymeric network in different batches of solution. On larger scales, these differences may average out to batch properties. For these applications, it is common to assume that a solution is well-mixed and to focus on bulk reaction properties rather than molecular-scale variations. Both of these paradigms require a different approach to modeling: molecular dynamics for the former and continuum-based rate equations for the latter.

On industrial scales, the primary concern is to estimate bulk (average) properties throughout the material at a given time and location. In these cases, continuum-based approaches are useful in understanding how concentrations of specific molecules are expected to evolve in time and space. The following subsections introduce the ordinary and partial differential equations that describe these dynamics with the purpose of providing a framework for engi-

neers to rapidly iterate on input parameters and estimate the resulting expected effects for the material as a whole.

Absorption of Light for Composite Materials

As light diffuses throughout the composite material, different components of the material will absorb the energy from the UV source. A slight modification to the Beer-Lambert law can be made to account for the light absorbed by a component Y :

$$\frac{dI}{dz} = -\epsilon_Y[Y]I, \quad (4.3)$$

where ϵ_Y and $[Y]$ are the absorptivity and concentration ($\frac{\text{mol}}{\text{m}^3}$) of material Y . During the printing process, the wavelength of UV light is chosen to match a specific PI, and it is assumed that the light is purely absorbed by the PI at the locations where the resin/binder is present. However, due to the high concentration of solid additives A , absorption of light by the particles cannot be neglected. The simulations performed in this work utilize [Eqn. 4.3](#) where Y is either the photoinitiator PI or solid additive A (note that future work may also consider the effects of scattering, diffraction, and reflection in either material, or at the boundaries). For the PI:

$$\frac{dI}{dz} = -\epsilon_{PI}[PI]I, \quad (4.4)$$

where $[PI]$ is the PI concentration, z is the distance from the exposed surface, and ϵ_{PI} is the absorptivity constant of the PI. An analytical solution for this relationship can be easily derived by integrating [Eqn. 4.4](#):

$$I_{PIabs}(z) = \epsilon_{PI}I_0[PI]e^{-\epsilon_{PI}z}. \quad (4.5)$$

Dissociation of the PI occurs when a specific wavelength of light breaks a double bond, dividing the molecule into two distinct free radicals [\[62\]](#) (see [Fig. 4.1 \(A\)](#)). This process is sometimes referred to as the rate of initiation [\[63, 64, 62\]](#):

$$R_i = \phi I_{PIabs} = \phi \epsilon_{PI}I_0[PI] e^{-\epsilon_{PI}[PI]z} \quad (4.6)$$

where ϕ is the quantum yield of initiation [\[63\]](#).

Rate of Dissociation of PI by UV Irradiation

The time rate of change of particular monomer (chemical) is typically modeled as the sum of all sources of rates of generation (positive terms) and rates of consumption (negative terms). An additional term that accounts for diffusion of material Y can be described by the divergence of the spatial gradient (Laplacian) of that material $\nabla_x \cdot (\kappa \nabla[Y])$. The Laplacian can either be positive or negative at any given point in time and is dictated by a diffusion coefficient κ which scales the rate of diffusion. When the rate of diffusion is $\ll$ than the

rate of generation/consumption of other terms, the diffusion term can be assumed to be negligible.

In the case of PI, the initial concentration is determined by the formulation and is purely consumed. Since dissociation of the PI yields two free radicals each, the time rate of change of the concentration of PI is half of Eqn. 4.5:

$$\frac{d[PI]}{dt} = -\frac{1}{2}R_i, \quad (4.7)$$

where it is assumed that the rate of consumption is much greater than the rate of diffusion.

Note: It should be clear from Eqn. 4.5 and Eqn. 4.7 that the dynamics of light attenuation directly impact the dynamics of the rate of consumption of PI. As we will see in the upcoming reactions, there is a similar cause-effect relationship that is built on the core assumption that these reactions are irreversible and move only forward. Thus, the state of each chemical is dependent on the previous state of the preceding chemical in the step-wise reaction mechanism. This in turn will define how these coupled equations are solved numerically in section 4.2.

Primary Radical Formation

The time rate of change of the primary radical $PI\bullet$ changes based on its generation and consumption over time. The generation of $PI\bullet$ is directly determined by its rate of initiation, as previously stated as Eqn. 4.5. The rate of consumption is determined by the reaction of $PI\bullet$ with a M :

$$R_a = k_i[PI\bullet][M] \quad (4.8)$$

where k_i is the primary radical kinetic coefficient, and $[M]$ is the concentration of the monomer. The difference between the rates of generation and consumption give the rate equation for the primary radical:

$$\frac{d[PI\bullet]}{dt} = \phi\epsilon_{PI}I_0[PI]e^{-\epsilon_{PI}[PI]z} - k_i[PI\bullet][M], \quad (4.9)$$

where once again, where it is assumed that the rate of diffusion is much slower than the rates of generation or consumption.

Active Chain Initiation & Termination

The rate of polymerization of acrylate monomers is dictated by the rate of initiation (generation source) and the termination of active chains $M_n\bullet$ (consumption source). An active chain can form via a reaction between a primary free-radical and a monomer, or terminate via a reaction with another active chain or primary radical. Since the initial concentration of PI is $\ll$ the concentration of available monomer, *this work assumes that termination of active chains occurs only with other active chains*. For steps 1 and 2, it is

assumed that the rate of initiation R_i and activation R_a are $\gg$ the rate of diffusion. However, this assumption cannot be made for the termination active chains $M_n\bullet$. Thus, reaction rate of active chains is modeled as:

$$\frac{d[M_n\bullet]}{dt} = k_i[PI\bullet][M] + \nabla_x \cdot (\kappa_m \nabla_x [M_n\bullet]) - k_t[M_n\bullet]^2 \quad (4.10)$$

where κ_m is the diffusion coefficient of the monomer, k_t is the active chain termination kinetic coefficient, $[M_n\bullet]$ is the concentration of active chain monomers, and the κ_m is the diffusion coefficient. In this case, it is no longer assumed that the reaction rates are $\gg$ the rate of diffusion.

Propagation Reaction

The concentration of acrylate monomers $[M]$ decreases as the rates of reaction for primary free radicals $[PI\bullet]$ and active chains $[M_n\bullet]$ increases. Since the UV intensity varies with depth, a concentration gradient occurs, driving diffusion of the inactive monomer. The change in concentration of the inactive monomer can be described as:

$$\frac{d[M]}{dt} = \nabla_x \cdot (\kappa_m \nabla_x [M]) - k_i[PI\bullet][M] - k_p[M][M_n\bullet] \quad (4.11)$$

where k_p is the propagation kinetic coefficient between an active chain and a monomer.

Degree of Cure and Diffusion Coefficients

The basic structure of the reaction-diffusion/heat equation elegantly captures the dynamics of dispersing gradients, whether it be specific chemical species or energy in the form of heat. However, the rate at which these dispersions happen is scaled by physical coefficients, which in turn may be subject to environmental variables. Reaction and diffusion coefficients κ_i , κ_t , and κ_m are particularly sensitive to temperature fluctuations, but even more critically to the degree of cure (which is also dependent on these coefficients). Modeling how these coefficients evolve in time and space is critical for building valid simulations that provide value to engineering applications.

The degree of cure can be simply defined as the percent conversion of starting material to final material, where 100% conversion indicates the ideal (but unrealistic) scenario in which all starting monomers are depleted due to polymerization. Although this simplified metric does not capture the length or quality of the resulting polymeric network, it does capture macroscopic trends of the consumption of small molecules that may be the source of undesired reactions or leaching from the final cured materials, each of which may cause sites for degradation as the material ages. For the purposes of this work, I use the common metric for estimating the degree of cure:

$$C = \left(1 - \frac{[M]}{[M]_0}\right) * 100 \quad (4.12)$$

where C is the degree of cure (measured in %), $[M]$ is the concentration of monomer, and $[M]_0$ is initial concentration of monomer. Assuming that the initial solution is well mixed, concentration gradients begin to form as the light penetrates the uncured material, in which case diffusion begins to occur for mobile molecules. In this work, I use a modified Doolittle free volume equation [65, 66, 63]:

$$D_m = D_{m,0} e^{\frac{-A_m}{f}} \quad (4.13)$$

where $D_{m,0}$ is the pre-exponential factor, A_m captures the rate at which the diffusion rate decreases, and f is the fractional free volume. The free volume within a material is the space not occupied by its constituent atoms or molecules, which enables movement and influences reaction kinetics. The fractional free volume is defined as the proportion of the free volume compared to the total volume of the material. For diffusion-controlled kinetics and termination by reaction diffusion, the fractional free volume of a polymerizing mixture is also dependent on temperature and the current concentration of the remaining monomer. For applicable systems, Goodner et al. [66] developed the following model for the fractional free volume parameter f :

$$v_T = \frac{[M]mw_M}{\rho_M} + \frac{([M]_0 - [M])mw_M}{\rho_P} \quad (4.14)$$

$$\phi_M = \frac{[M]mw_M}{\rho_M v_T} \quad (4.15)$$

$$\phi_P = \frac{([M]_0 - [M])mw_M}{\rho_P v_T} \quad (4.16)$$

$$f = 0.025 + \alpha_M \phi_M (\theta - \theta_{gM}) + \alpha_P \phi_P (\theta - \theta_{gP}). \quad (4.17)$$

In these equations, mw_i are the molecular weights (let $i = M, P$ for the monomer and polymer respectively), ϕ_i represents the volume fraction, ρ_i is the density, $[i]_0$ is the initial concentration, θ_{gi} is the glass transition temperature, and v_T is a normalizing factor to ensure the volume fractions sum to one. This decision was made by Goodner et al. to manage uncertainties in manufacturer reported values such as density and molecular weight.

Kinetic Rate Coefficients

In a series of papers, Goodner et al. [67, 68, 66] published an evolving series of empirically validated models that capture the temperature dependency for the diffusion coefficient of the monomer, and the kinetic rate coefficients for initiation κ_i and termination κ_t . In the third paper [66], Goodner et al. takes into consideration the temperature dependence of the chemical reaction kinetic coefficients (constants):

$$k_p = \frac{k_{p0}e^{-E_p/RT}}{1 + e^{A_p(1/f-1/f_{cp})}} \quad (4.18)$$

$$k_t = \frac{k_{t0}e^{-E_t/RT}}{\left(1 + \frac{1}{\left(\frac{R_{td}k_p[M]}{k_{t0}e^{-E_t/RT}} + e^{-A_t(1/f-1/f_{ct})}\right)}\right)} \quad (4.19)$$

where k_{i0} are the pre-exponential factors for the true reaction kinetic constants – *assuming* temperature is constant and consistent with the experimental conditions in which these were measured, E_i are the activation energies, and f_{ci} represents the critical fractional free volumes (let $i = p, t$ for propagation and termination respectively). The primary purpose of including f_{cp} is to capture the effects of when the propagation reaction or diffusion of the monomer is the rate limiting step in the process, with a similar logic for f_{ct} and termination.

Additional assumption: It is common to treat the initiation reaction rate coefficient as constant since the rate of initiation occurs rapidly.

Heat Generation

For UV-curable materials, the dynamics of heat and temperature fluctuate due to the following physical phenomena:

- UV absorption: as light dissipates through the composite, energy from the source is absorbed by the materials;
- Heat of reactions: after the PI dissociates into radicals, subsequent exothermic reactions release heat;
- Diffusion: as heat is absorbed and generated, different constituents of the composite diffuse the heat at different rates;
- Boundary conditions: it is common in additive manufacturing environments to control the external temperatures with fans to keep a constant print environment.

Since the reaction-diffusion model is temperature dependent, it is critical to include heat transfer as part of the multi-physics model:

$$\rho C \frac{\partial \theta}{\partial t} = \nabla_x \cdot (\kappa \nabla_x \theta) \quad (4.20a)$$

$$+ (k_i[PI\bullet][M] + k_p[M][M_n\bullet] + k_t[M_n\bullet]^2) \Delta H \quad (4.20b)$$

$$+ (\epsilon_{PI}[PI]I_0e^{-\epsilon_{PI}[PI]z} \parallel \epsilon_{NaCl}I_0e^{-\epsilon_{NaCl}z}) \quad (4.20c)$$

where $\parallel$ represent “or”, the terms on the right hand side represent Fourier’s law and the diffusion of heat in Eqn. 4.20a, the heats of reaction in Eqn. 4.20b, and the absorption of UV radiation in Eqn. 4.20c. In addition to the assumptions made in the kinetics modeling, I make the following two additional assumptions:

- dissociation of PI is activated purely from the UV source and does not take an energy from the surrounding material;
- the heats of reaction are the same for initiation, propagation and termination;
- photo-polymerization of intended starting materials are considered (e.g., potential degradation or decomposition is ignored).

Multi-Physical Coupled Fields

The overall multi-physics model captures the highly coupled and dynamic interplay between multiple physical phenomena occurring during the curing stages of an advanced manufacturing process. This complexity and non-linearity arise from the interdependencies between heat and mass transfer and photo-polymerization. These effects, facilitated by changing material properties, make it challenging to intuit a priori the impact of small changes to input parameters or inherent process noise. For instance, variations in temperature or particle distribution influence absorption and diffusion, which in turn affect the reactions, subsequently impacting temperature and diffusion. The model summarized in Tab. 4.1 integrates equations representing the conservation of mass and energy, along with state equations describing light attenuation and material properties. This comprehensive framework captures the feedback loops and interactions governing the curing process.

By incorporating these coupled equations, the model provides a robust tool for understanding how minor adjustments in process parameters can propagate through the system, potentially leading to significant changes in the final outcome. The non-linear nature of these interactions means that seemingly negligible fluctuations in one area can have cascading effects, underscoring the importance of a holistic approach to modeling. This model enables engineers to rapidly iterate over different input parameters to understand the system’s sensitivity or to optimize for quality control.

Description	Reaction/Physics Scheme	Kinetic/Physics Model
<i>Kinetic coeff.</i>	Propagation	$k_p = \frac{k_{p0}e^{-E_p/RT}}{1+e^{A_p(1/f-1/f_{cp})}}$
<i>Kinetic coeff.</i>	Termination	$k_t = \frac{k_{t0}e^{-E_t/RT}}{\left(1 + \frac{1}{\left(\frac{R_t d k_p [M]}{k_{t0}e^{-E_t/RT}} + e^{-A_t(1/f-1/f_{ct})}\right)}\right)}$
<i>Intensity</i>	Beer-Lambert Law	$I(z) = I_0 e^{-\epsilon z}$
<i>Attenuation</i>	w.r.t. z , where i = material type	$\frac{dI}{dz} = -\epsilon_i[i]I_0 e^{-\epsilon_i z}$
<i>Dissociation</i>	$PI \xrightarrow{UV} 2PI\bullet$	$\frac{d[PI]}{dt} = -\frac{\phi}{2}\epsilon_{PI}[PI]I$
<i>Initiation</i>	$PI\bullet + M \rightarrow PI-M_n\bullet$	$\frac{d[PI\bullet]}{dt} = \phi\epsilon_{PI}[PI]I - k_i[PI\bullet][M]$
<i>Termination</i>	$PI-M_n\bullet + PI-M_n\bullet \xrightarrow{k_t} PI-M_nM_n-PI$	$\frac{d[M_n\bullet]}{dt} = k_i[PI\bullet][M]$ $+ \nabla_x \cdot (\kappa_m \nabla_x [M_n\bullet])$ $- k_t[M_n\bullet]^2$
<i>Propagation</i>	$PI-M_n\bullet + M \xrightarrow{k_p} PI-M_{n+1}\bullet$	$\frac{d[M]}{dt} = \nabla_x \cdot (\kappa_m \nabla_x [M])$ $- k_i[PI\bullet][M] - k_p[M][M_n\bullet]$
<i>Temperature</i>	Heat equation	$\rho C \frac{d\theta}{dt} = \nabla_x \cdot (K \nabla_x \theta) + \Delta H_{rxn} + I_{abs}$
<i>Heats of rxn</i>	Initiation, termination, propagation	$\Delta H_{rxn} = (k_i[PI\bullet][M]$ $+ k_p[M][M_n\bullet]$ $+ k_t[M_n\bullet]^2)\Delta H$
<i>Absorption</i>	Polymer or particle	$I_{abs} = \begin{cases} \epsilon_{PI}[PI]I_0 e^{-\epsilon_{PI}[PI]z} \\ \epsilon_{NaCl}I_0 e^{-\epsilon_{NaCl}z} \end{cases}$

Table 4.1: Summary of multi-physics model for transient chemical-thermal coupled fields.

Initial and Boundary Conditions

To obtain a unique solution, the differential equations in Tab. 4.1 require temporal and spatial information at the boundaries of each domain. The initial conditions of the curing system are relatively straight forward as the concentrations and UV intensity are defined by the user (e.g., see formulation provided in Chapter 3). The conditions at the boundaries may require a more strategic approach and may change depending on the environment. However, for many advanced manufacturing setups it is common to have temperature controlled environments (e.g., fans for cooling/heating). Given this common scenario, I make the following assumptions to apply the most applicable boundary conditions:

1. Temperature on the surface of the film or filament remains constant;
2. mass transfer of the material at the surface of the film or filament is negligible on the time scale applicable for the manufacturing process.

From these assumptions, the numerical solver implements the following boundary conditions (provided in 1D):

$$\theta(x=0) = \theta(x=L) = \theta_{amb} \quad (4.21)$$

$$\mathcal{F} = \kappa \frac{\partial u}{\partial x} \Big|_{x=0} = \kappa \frac{\partial u}{\partial x} \Big|_{x=L} = 0 \quad (4.22)$$

$$(4.23)$$

where $0 \leq x \leq L$, θ_{amb} is the ambient temperature, $\mathcal{F}$ is the flux across the boundary. The physical meaning behind Eqn. 4.21 and Eqn. 4.22 can be interpreted as set temperature and zero concentration flux (no evaporation) on the boundaries.

Spatial Discretization of Fields

Each partial differential equation in Tab. 4.1 includes a Laplacian term:

$$\nabla_x \cdot (\kappa \nabla_x u) \quad (4.24)$$

where u represents the quantity of interest, such as the concentration of chemical species $[B]$ or temperature θ , and κ represents the diffusion coefficient for the thermal or chemical species components of the model. A practical and effective approach to numerically solving the system of equations in Tab. 4.1 is to apply finite difference schemes to the spatial components. This method allows for the rapid development of models while maintaining a straightforward solution scheme.

Using finite difference methods also eliminates the need for memory-intensive methods like finite element methods, which can require the storage of large matrices. Furthermore, finite difference schemes are well-suited for parallelization on GPUs, enabling large-scale

simulations that capture variable-scale phenomena efficiently. However, discretization of composite material requires decision on how to handle the representation of materials embedded in the grid, particularly at the boundaries. The following sections cover the basic approach to discretizing Eqn. 4.24, and how material properties are computed in subsequent simulations.

1D discretization

In this section, I explore the process of discretizing the partial differential equations in one dimension. Assume $x \in \mathbb{R}^1$ and let $h = \Delta x$ represent the spatial step size, while $q = \kappa \nabla_x u$ is the flux term. To approximate the gradient of q , I utilize a central difference scheme, which provides a straightforward and effective method for numerical differentiation.

Using the central difference approximation, the gradient of q can be expressed as:

$$\nabla_x q \approx \frac{q(x + \frac{h}{2}) - q(x - \frac{h}{2})}{h}. \quad (4.25)$$

Next, I apply the same finite difference scheme to capture the second derivatives of u . This involves evaluating the flux at points $x + \frac{h}{2}$ and $x - \frac{h}{2}$, as shown below:

$$q\left(x + \frac{h}{2}\right) \approx -\kappa\left(x + \frac{h}{2}\right) \frac{u(x + h) - u(x)}{h} \quad (4.26)$$

$$q\left(x - \frac{h}{2}\right) \approx -\kappa\left(x - \frac{h}{2}\right) \frac{u(x) - u(x - h)}{h} \quad (4.27)$$

By combining these forward and backward differences, I derive a total approximation for the Laplacian term:

$$\nabla_x(\kappa \nabla_x u) \approx \frac{\bar{\kappa}_{x+} u(x + h) - (\bar{\kappa}_{x+} + \bar{\kappa}_{x-}) u(x) + \bar{\kappa}_{x-} u(x - h)}{h^2}. \quad (4.28)$$

To ensure accuracy in our numerical model, I must also approximate the diffusion coefficient κ . Using the same difference scheme, I can calculate the average values of κ at the intermediate points:

$$\kappa\left(x + \frac{h}{2}\right) \approx \bar{\kappa}_{x+} = \frac{1}{2}(\kappa(x + h) + \kappa(x)) \quad (4.29)$$

$$\kappa\left(x - \frac{h}{2}\right) \approx \bar{\kappa}_{x-} = \frac{1}{2}(\kappa(x) + \kappa(x - h)) \quad (4.30)$$

$$(4.31)$$

To match the second-order accurate central difference scheme, I also apply a second-order accurate discretization of the zero-flux conditions at the boundaries:

$$\kappa \frac{\partial u}{\partial x} \Big|_{x=0} \approx \frac{\bar{\kappa}}{h} \left(\frac{3}{2}u(x) - 2u(x+h) + \frac{1}{2}u(x+2h) \right) = 0 \quad (4.32)$$

where

$$\bar{\kappa} = \frac{3}{2}\kappa(x) - 2\kappa(x+h) + \frac{1}{2}\kappa(x+2h). \quad (4.33)$$

Solving for $u(x)$ at the boundaries:

$$u(x) \Big|_{x=0} = \frac{2}{3} \left(2u(x+h) - \frac{1}{2}u(x+2h) \right) \quad (4.34)$$

$$u(x) \Big|_{x=L} = \frac{2}{3} \left(2u(x-h) - \frac{1}{2}u(x-2h) \right). \quad (4.35)$$

3D discretization

In this section, I now explore the process of discretizing the partial differential equations in three dimensions. Now $\mathbf{x} \in \mathbb{R}^3$ and assume $h = \Delta x$ represents the spatial step size *that is equal in all directions*, while $\mathbf{q} = \kappa \nabla_{\mathbf{x}} u$ is the flux term. To approximate the gradient of $\mathbf{q}$, I utilize successive first-order derivatives to obtain the central difference scheme:

$$\nabla_{\mathbf{x}} \cdot \mathbf{q} \approx \frac{q(x_1 + \frac{\Delta x}{2}, x_2, x_3) - q(x_1 - \frac{\Delta x}{2}, x_2, x_3)}{\Delta x} \quad (4.36)$$

$$+ \frac{q(x_1, x_2 + \frac{\Delta x}{2}, x_3) - q(x_1, x_2 - \frac{\Delta x}{2}, x_3)}{\Delta x} \quad (4.37)$$

$$+ \frac{q(x_1, x_2, x_3 + \frac{\Delta x}{2}) - q(x_1, x_2, x_3 - \frac{\Delta x}{2})}{\Delta x} \quad (4.38)$$

Each derivative of the flux q can be further approximated using first-order derivatives:

$$q\left(x_1 + \frac{\Delta x}{2}, x_2, x_3\right) \approx -\kappa\left(x_1 + \frac{\Delta x}{2}, x_2, x_3\right) \frac{u(x_1 + \Delta x, x_2, x_3) - u(x_1, x_2, x_3)}{\Delta x} \quad (4.39)$$

$$q\left(x_1 - \frac{\Delta x}{2}, x_2, x_3\right) \approx -\kappa\left(x_1 - \frac{\Delta x}{2}, x_2, x_3\right) \frac{u(x_1, x_2, x_3) - u(x_1 - \Delta x, x_2, x_3)}{\Delta x} \quad (4.40)$$

$$q\left(x_1, x_2 + \frac{\Delta x}{2}, x_3\right) \approx -\kappa\left(x_1, x_2 + \frac{\Delta x}{2}, x_3\right) \frac{u(x_1, x_2 + \Delta x, x_3) - u(x_1, x_2, x_3)}{\Delta x} \quad (4.41)$$

$$q\left(x_1, x_2 - \frac{\Delta x}{2}, x_3\right) \approx -\kappa\left(x_1, x_2 - \frac{\Delta x}{2}, x_3\right) \frac{u(x_1, x_2, x_3) - u(x_1, x_2 - \Delta x, x_3)}{\Delta x} \quad (4.42)$$

$$q\left(x_1, x_2, x_3 + \frac{\Delta x}{2}\right) \approx -\kappa\left(x_1, x_2, x_3 + \frac{\Delta x}{2}\right) \frac{u(x_1, x_2, x_3 + \Delta x) - u(x_1, x_2, x_3)}{\Delta x} \quad (4.43)$$

$$q\left(x_1, x_2, x_3 - \frac{\Delta x}{2}\right) \approx -\kappa\left(x_1, x_2, x_3 - \frac{\Delta x}{2}\right) \frac{u(x_1, x_2, x_3) - u(x_1, x_2, x_3 - \Delta x)}{\Delta x}. \quad (4.44)$$

The averaged diffusivity coefficients becomes a difference approximation between component stencil surrounding the voxel:

$$\kappa_{x_1^+} = \kappa\left(x_1 + \frac{\Delta x}{2}, x_2, x_3\right) \approx \frac{1}{2} (\kappa(x_1 + \Delta x, x_2, x_3) + \kappa(x_1, x_2, x_3)) \quad (4.45)$$

$$\kappa_{x_1^-} = \kappa\left(x_1 - \frac{\Delta x}{2}, x_2, x_3\right) \approx \frac{1}{2} (\kappa(x_1, x_2, x_3) + \kappa(x_1 - \Delta x, x_2, x_3)) \quad (4.46)$$

$$\kappa_{x_2^+} = \kappa\left(x_1, x_2 + \frac{\Delta x}{2}, x_3\right) \approx \frac{1}{2} (\kappa(x_1, x_2 + \Delta x, x_3) + \kappa(x_1, x_2, x_3)) \quad (4.47)$$

$$\kappa_{x_2^-} = \kappa\left(x_1, x_2 - \frac{\Delta x}{2}, x_3\right) \approx \frac{1}{2} (\kappa(x_1, x_2, x_3) + \kappa(x_1, x_2 - \Delta x, x_3)) \quad (4.48)$$

$$\kappa_{x_3^+} = \kappa\left(x_1, x_2, x_3 + \frac{\Delta x}{2}\right) \approx \frac{1}{2} (\kappa(x_1, x_2, x_3 + \Delta x) + \kappa(x_1, x_2, x_3)) \quad (4.49)$$

$$\kappa_{x_3^-} = \kappa\left(x_1, x_2, x_3 - \frac{\Delta x}{2}\right) \approx \frac{1}{2} (\kappa(x_1, x_2, x_3) + \kappa(x_1, x_2, x_3 - \Delta x)) \quad (4.50)$$

The overall diffusion term in 3D, $\nabla_{\mathbf{x}} \cdot (-\kappa \nabla_{\mathbf{x}} u)$, can be approximated as:

$$\begin{aligned} \nabla_{\mathbf{x}} \cdot (\kappa \nabla_{\mathbf{x}} u) &\approx \frac{\bar{\kappa}_{x_1^+} (u(x_1 + h, x_2, x_3) - u(x_1, x_2, x_3)) - \bar{\kappa}_{x_1^-} (u(x_1, x_2, x_3) - u(x_1 - h, x_2, x_3))}{h^2} \\ &+ \frac{\bar{\kappa}_{x_2^+} (u(x_1, x_2 + h, x_3) - u(x_1, x_2, x_3)) - \bar{\kappa}_{x_2^-} (u(x_1, x_2, x_3) - u(x_1, x_2 - h, x_3))}{h^2} \\ &+ \frac{\bar{\kappa}_{x_3^+} (u(x_1, x_2, x_3 + h) - u(x_1, x_2, x_3)) - \bar{\kappa}_{x_3^-} (u(x_1, x_2, x_3) - u(x_1, x_2, x_3 - h))}{h^2} \end{aligned} \quad (4.51)$$

Temporal Discretization of Fields

Once spatial derivatives have an explicit numeric approximation, the simulation then requires a temporal discretization strategy. Here I employ the stable trapezoidal method (section 2.2) for a second order accuracy in time:

$$u_t = \underbrace{\nabla_x \cdot (\kappa \nabla_x u)}_{\mathcal{F}} + \mathcal{S} \quad (4.52)$$

where u is once again an arbitrary quantity of interest, such as temperature or concentration of a chemical species, and $\mathcal{S}$ represents a collection of source/sink terms. Here, $\mathcal{F}$ is used to denote the right-hand side of the equation.

To discretize this in time, I use the trapezoidal rule, which is a combination of the explicit and implicit methods. This approach can be represented as:

$$u(t + \Delta t) = u(t) + \Delta t \left(\phi \mathcal{F}(u(t)) + (1 - \phi) \mathcal{F}(u(t + \Delta t)) \right) \quad (4.53)$$

where ϕ is a parameter that controls the weighting between the explicit and implicit parts of the method. Typically, $\phi = 0.5$ for the trapezoidal rule, ensuring a balanced contribution from both the current and next time steps (and is commonly referred to as Crank-Nicolson). Since the term $\mathcal{F}(u(t + \Delta t))$ involves the unknown future state $u(t + \Delta t)$, I need an iterative approach to solve for it. The fixed-point iteration method is employed to handle this implicit nature (see Alg. 2).

It's important to note that the choice of fixed-point iteration in this implementation was primarily driven by its simplicity and ease of implementation. However, this decision has implications for the overall stability of the algorithm. While the trapezoidal method itself is unconditionally stable due to its implicit nature, the introduction of the fixed-point iteration method imposes conditional stability on the entire scheme. This conditional stability is analogous to the constraints found in explicit methods like forward Euler. In the case for fixed-point iteration and the trapezoidal method, the solution converges if and only if $\frac{\alpha k}{h^2} < \frac{1}{2}$ (see section 7.2). Specifically, the stability of the fixed-point iteration is governed by a pre-factor that places a restriction on the allowable time step size. This constraint is similar to the Courant-Friedrichs-Lewy (CFL) condition often encountered in explicit schemes. While this method benefits from the simplicity of fixed-point iteration, it requires careful consideration of the time step size to ensure numerical stability (further discussion in section 4.3). Future implementations might consider alternative approaches, such as multigrid, which could potentially allow for larger time steps at the cost of increased computational complexity per iteration.

Overall Solution Scheme

The overall solution scheme for the numerical solver encompasses the initialization, iterative solution of discretized equations, and enforcing of boundary conditions throughout the

Algorithm 2 Fixed Point Iteration for coupled fields

```

1: procedure FP_SOLVER
2:                                     ▷ Initialize variables
3:    $n \leftarrow 0, \quad err\_total \leftarrow 100, \quad err\_step \leftarrow 0$ 
4:    $PI_0^{j+1} \leftarrow PI^j \quad PIDot_0^{j+1} \leftarrow PIDot^j \quad Mdot_0^{j+1} \leftarrow Mdot^j \quad M_0^{j+1} \leftarrow M^j \quad \theta_0^{j+1} \leftarrow \theta^j$ 
5:   while  $err < TOL$  do                                     ▷ while less than tolerance
6:      $PI_{n+1}^{j+1} \leftarrow PI^j + \Delta t [\phi \mathcal{F}(PI^j) + (1 - \phi) \mathcal{F}(PI_n^{j+1})]$ 
7:      $PIDot_{n+1}^{j+1} \leftarrow PIDot^j + \Delta t [\phi \mathcal{F}(PIDot^j) + (1 - \phi) \mathcal{F}(PIDot_n^{j+1})]$ 
8:      $Mdot_{n+1}^{j+1} \leftarrow Mdot^j + \Delta t [\phi \mathcal{F}(Mdot^j) + (1 - \phi) \mathcal{F}(Mdot_n^{j+1})]$ 
9:      $M_{n+1}^{j+1} \leftarrow M^j + \Delta t [\phi \mathcal{F}(M^j) + (1 - \phi) \mathcal{F}(M_n^{j+1})]$ 
10:     $\theta_{n+1}^{j+1} \leftarrow \theta^j + \Delta t [\phi \mathcal{F}(\theta^j) + (1 - \phi) \mathcal{F}(\theta_n^{j+1})]$ 
11:                                     ▷ Compute step errors
12:     $err\_step \leftarrow err\_step + ||PI_{n+1}^{j+1} - PI_n^{j+1}||$ 
13:     $err\_step \leftarrow err\_step + ||PIDot_{n+1}^{j+1} - PIDot_n^{j+1}||$ 
14:     $err\_step \leftarrow err\_step + ||Mdot_{n+1}^{j+1} - Mdot_n^{j+1}||$ 
15:     $err\_step \leftarrow err\_step + ||M_{n+1}^{j+1} - M_n^{j+1}||$ 
16:     $err\_step \leftarrow err\_step + ||\theta_{n+1}^{j+1} - \theta_n^{j+1}||$ 
17:                                     ▷ Compute total error
18:     $err\_total \leftarrow \sqrt{err\_step}$ 
19:     $n \leftarrow n + 1$ 
20:                                     ▷ Set new initial values
21:     $PI_n^{j+1} \leftarrow PI_{n+1}^{j+1}$ 
22:     $PIDot_n^{j+1} \leftarrow PIDot_{n+1}^{j+1}$ 
23:     $Mdot_n^{j+1} \leftarrow Mdot_{n+1}^{j+1}$ 
24:     $\theta_n^{j+1} \leftarrow \theta_{n+1}^{j+1}$ 
25:                                     ▷ Update variables for the next iteration
26:     $u_n^{j+1} \leftarrow u_{n+1}^{j+1}, \quad n = n + 1$ 
27:  end while
28: end procedure

```

simulation. This section details the step-by-step process employed to achieve the numerical solution of the multi-physics model. A summary of the solution scheme can be found in Alg. 3.

Initialization

The initialization phase involves setting up the computational grid (material initialization), defining initial conditions, and specifying material properties. A broad overview of the main steps in the solver include:

Algorithm 3 Overall solution algorithm

```

1: procedure SOLVE SYSTEM
2:    $temp \leftarrow 300.$ ;  $rp \leftarrow 7.27612e - 05$ ;  $vp \leftarrow 0.5$ ;  $uvi \leftarrow 32.7876$ ;  $uvt \leftarrow 6.03151$ 
3:    $voxel \leftarrow Voxel(temp, uvi, uvt)$  ▷ Initialize field
4:    $voxel.init\_particles(rp, vp)$  ▷ Initialize particles
5:   while  $time < time\_total$  do
6:      $PI^{j+1}, PIdot^{j+1}, Mdot^{j+1}, M^{j+1}, \theta^{j+1} \leftarrow SOLVER(PI^j, PIdot^j, Mdot^j, M^j, \theta^j)$ 
7:   end while
8: end procedure

```

1. **Grid Generation:** Create a computational grid that discretizes the spatial domain into nodes or elements. The grid used in this work is structured (e.g., Cartesian grid), but more complex domains can be captured using unstructured grids (e.g., triangular or tetrahedral elements).
2. **Particle Initialization:** Distribute particles within the computational domain based on the desired initial particle distribution. This includes defining the initial positions, size (drawn from a Gaussian distribution), and solids loading (%vol. of representative volume element). This work uses a random sequential algorithm (Alg. 4) to add particles to the grid.
3. **Material Generation:** For voxels that do not take on properties of a particle, material properties of the formulated polymer such as thermal conductivity, diffusivity, and reaction rate constants to each node or element. For composite materials, use effective properties are computed using a weighted average homogenization scheme.
4. **Initial Conditions:** Initialize the fields for temperature θ , species concentrations $[PI]$, $[PI\bullet]$, $[M_n\bullet]$, and $[M]$. The initial conditions are based on initial formulations provided in Chapter 3.

Time Stepping

The time-stepping phase involves iteratively solving the discretized equations for each time step. The process is repeated until the desired simulation end time is reached. The following steps outline the time-stepping procedure:

1. **Time Step Selection:** As discussed in section 4.2, the time step must satisfy $k < \frac{\alpha h^2}{2}$.
2. **Field Updates:** For each time step t^n to t^f :
 - a) **Material Properties Update:** Recompute the material properties at each node based on current time step values;

- b) **Reaction and Source Terms:** Include the energy source (UV light);
- c) **Temperature and Species Concentration Update:** Solve the discretized species concentration equations using the trapezoidal method to update the next values for temperature and concentrations: θ^{n+1} , $[PI]^{n+1}$, $[PI\bullet]^{n+1}$, $[M]^{n+1}$, $[M_n\bullet]^{n+1}$ using Alg. 2;
- d) **Boundary Conditions:** Enforce the boundary conditions specified in Eqn. 4.21 and Eqn. 4.22 to update the fields at the boundaries.

Determining Material Parameters

Critical to the multi-physics model described above is the set of parameters associated with the material properties. These parameters represent measurable quantities that can be obtained via traditional analytical chemistry techniques. Taki et al. reports a comprehensive flowchart for the equipment and tests required to obtain many of the material properties required for the multi-physics model indicated in Tab. 4.1 (see Fig. 4.2). These values may be challenging to obtain without a full-scale chemical laboratory since specific components may be synthesized in-house or whose details are proprietary trade secrets and not readily available. Since it is challenging to directly determine every parameter experimentally, there are a variety of approaches in collecting the required parameters from different approaches. The following papers use the following strategies [67, 68, 66, 64]:

- Direct measurements when equipment is available;
- indirect methods (e.g., least squares) using measured data for other parameters;
- reported values from the literature;
- approximating using values from similar materials;
- trial and error.

Algorithm 4 Random Sequential Addition (RSA) of particles in a 3D mesh

```

1:                                     ▷ Initialize mesh and parameters
2:  $mesh \leftarrow$  empty 3D mesh of size  $L \times W \times H$ 
3:  $num\_particles \leftarrow 0$ 
4:  $max\_particles \leftarrow$  desired number of particles
5:  $max\_attempts \leftarrow$  maximum number of attempts to place a particle
6:  $radius \leftarrow$  radius of the particles
7: function ISVALIDPOSITION( $mesh, x, y, z, radius$ )
8:   for all  $particle \in mesh$  do
9:     if  $\sqrt{(x - particle.x)^2 + (y - particle.y)^2 + (z - particle.z)^2} < 2 \times radius$  then
10:      return false
11:    end if
12:  end for
13:  return true
14: end function
15: while  $num\_particles < max\_particles$  do
16:    $attempts \leftarrow 0$ 
17:   while  $attempts < max\_attempts$  do
18:     ▷ Generate random coordinates within the mesh boundaries
19:      $x \leftarrow$  random float between  $radius$  and  $L - radius$ 
20:      $y \leftarrow$  random float between  $radius$  and  $W - radius$ 
21:      $z \leftarrow$  random float between  $radius$  and  $H - radius$ 
22:     if ISVALIDPOSITION( $mesh, x, y, z, radius$ ) then
23:       ▷ Place particle and increment count
24:       add particle at  $(x, y, z)$  to  $mesh$ 
25:        $num\_particles \leftarrow num\_particles + 1$ 
26:       break
27:     else
28:       ▷ Increment attempts
29:        $attempts \leftarrow attempts + 1$ 
30:     end if
31:   end while
32:   if  $attempts = max\_attempts$  then
33:     ▷ Termination condition if unable to place particle
34:     break
35:   end if
36: end while

```

4.3 Numerical Example: UV kinetics for UGAP

To demonstrate the application of the multi-physics model developed in this chapter, I use the formulations reported in Chapter 3 as a case study. This example examines a single representative volume element (RVE) of a filament being extruded onto a build plate. The RVE is modeled using a rectilinear domain with dimensions $D \times D \times D$, where $D = 840\mu m$ corresponds to the diameter of the filament. This case study simulates the temporal/spatial evolution of the temperature and concentration profiles over 30 seconds as a UGAP-based ink is exposed to a UV light source with an intensity of $10W/m^2$ and a wavelength range of $400 - 500nm$.

This rectilinear approximation represents the top and bottom surfaces of the extruded filament. The bottom surface conforms to the flat build plate (or preceding layer), while the top surface is smoothed by the motion of the nozzle. Although the sides of an isolated filament would typically have curvature, I approximate them as flat for simplicity. This simplification is justified because the primary dynamics of interest occur due to the attenuation of light in the z-direction, perpendicular to the build plate. This allows the applications of Eqn. 4.21 and Eqn. 4.22 as boundary conditions for the temperature and concentration fields respectively.

Within this domain are NaCl particles with an average radius of $84\mu m$ and a particle volume fraction of 70%. The grid was generated using RSA Alg. 4, and remaining elements within the grid were assigned values and material properties of the constituents described in

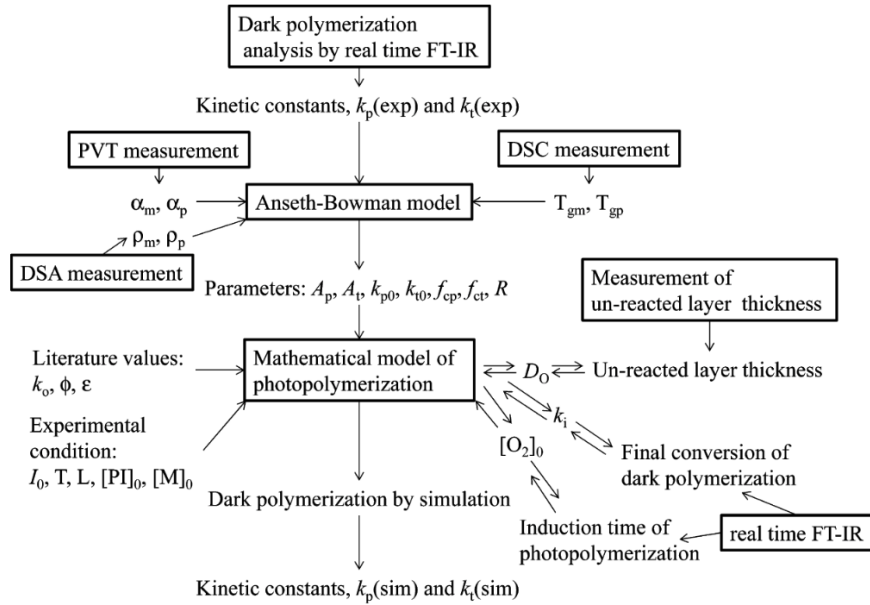


Figure 4.2: Taki et al. reports measurement techniques to obtain kinetic constants required for multi-physics modeling and simulation (figure used from [64]).

the UGAP formulation. However, since not all specific material parameters were available from the experiments designed in Chapter 3 and access to lab equipment was no longer available, values from similar reactive materials reported from the literature are used. The comprehensive list of material parameters and their sources are defined in Tab. 4.2, Tab. 4.3, Tab. 4.4, Tab. 4.5, and Tab. 4.6.

Table 4.2: Reaction-diffusion parameters

Parameter	Value	Unit	Description
Dm0	2.36e-6	m^2/s	Diffusion pre-exponential for monomer [63]
A_m	0.66	-	Diffusion constant parameter for monomer [66]
R_g	8.3145	$J/molK$	Universal gas constant
α_P	0.000075	$1/K$	Coef. of therm. exp. for polymerization [64, 66]
α_M	0.0005	$1/K$	Coef. of therm. exp. for monomer [64, 66]
θ_{gP}	236.75	K	Glass transition temperature for polymer [8]
θ_{gM}	213.0	K	Glass transition temperature for monomer [66]
k_{P0}	1.6e3	$m^3/mols$	True kinetic constant for polymerization [66]
E_P	18.23e3	J/mol	Activation energy for polymerization [66]
A_{Dp}	0.66	-	Diffusion parameter for polymerization [66]
f_{cp}	4.2e-2	-	Critical free volume for polymerization [63]
k_{T0}	3.6e3	$m^3/mols$	True kinetic constant for termination [63]
E_T	2.94e3	J/mol	Activation energy for termination [66]
A_{Dt}	1.2	-	Activation energy for termination [66]
f_{ct}	6.0e-2	-	Critical free volume for termination [66]
R_{rd}	11.	m^3/mol	Reaction diffusion parameter [64]
k_{I0}	5.1e-4	$m^3/mols$	Primary radical rate constant (trial-and-error)
E_I	18.23e3	J/mol	Activation energy for initiation [66]
A_I	0.66	-	Diffusion parameter for initiation [66]
f_{ci}	0.042	-	Critical free volume for initiation [66]

Table 4.3: Photoinitiator properties

Property	Value	Unit	Description	Source
ϵ	9.66e-1	$m^3/molm$	Initiator absorbtivity	spec sheet
ϵ_{NaCl}	7e-4	$1/m$	NaCl absorbtivity	NIST
ϕ	0.6	-	Quantum yield initiation	[63, 66]

Table 4.4: Thermal properties of the material

Parameter	Value	Unit	Description
θ_0	303.15	K	Initial and ambient temperature
ΔH_p	5.48e4	J/mol	Heat of polymerization (polymer database)
$C_{p,NaCl}$	880	$J/kg/K$	Heat capacity of NaCl (NIST database)
$C_{p,PEA}$	180.3	$J/mol/K$	Heat capacity of PEA at 298K (polymer database)
$C_{p,HDDA}$	202.9	$J/mol/K$	Solid heat capacity of HDDA (NIST database)
$K_{th,NaCl}$	0.069	$W/m/K$	Thermal conductivity of NaCl (NIST database)
$C_{p,resin}$	1700	$J/kg/K$	Heat capacity of the resin [63]
$K_{th,resin}$	0.2	$W/m/K$	Thermal conductivity of the resin [63]

Table 4.5: Parameter values

Parameter	Value	Unit	Description
$nodes$	51	-	Number of nodes
dt	5×10^{-5}	s	Time step
t_f	30	s	Final time
$nodes_{int}$	1	-	Interfacial thickness parameter
len	8.4×10^{-4}	m	Sample length
h	7.29×10^{-6}	m	Spatial discretization

Table 4.6: Physical parameters of constituents

Property	Symbol	Units	Value	Name
Density	ρ_{HDDA}	kg/m^3	1020	HDDA
Density	ρ_{PEA}	kg/m^3	1020	PEA
Density	ρ_M	kg/m^3	1020.0	Monomer (HDDA + PEA)
Density	ρ_P	kg/m^3	1074.4	Polymer (HDDA + PEA + E4396)
Density	ρ_{UGAP}	kg/m^3	1840	UGAP + NaCl filler
Molecular weight	mw_{PEA}	kg/mol	0.19221	PEA
Molecular weight	mw_{HDDA}	kg/mol	0.226	HDDA
Molecular weight	mw_{PI}	kg/mol	0.4185	Irgacure 819
Molecular weight	mw_M	kg/mol	0.19221	Monomer (HDDA + PEA)

Time step selection

As discussed in section 4.2, the time step must satisfy $k < \frac{\alpha h^2}{2}$, where α is lumped-parameter as defined by the numerically limiting differential equation. In this case, the set of differential equations in Tab. 4.1 is numerically limited by the diffusion equations for which $\alpha = 1/Dm_0$. For this work, simulations were performed using the following number of nodes:

$$nodes = [11, 15, 21, 25, 31, 41, 51] \quad (4.54)$$

with the corresponding spatial discretizations with respect the RVE dimension D :

$$h = [8.4 \times 10^{-5}, 6 \times 10^{-5}, 4.2 \times 10^{-5}, 3.5 \times 10^{-5}, 2.8 \times 10^{-5}, 2.1 \times 10^{-5}, 1.68 \times 10^{-5}] \quad (4.55)$$

and the resulting time steps that satisfy $k < \frac{h^2}{2Dm_0}$:

$$dt = [1.45 \times 10^{-3}, 7.5 \times 10^{-4}, 3.5 \times 10^{-4}, 2.5 \times 10^{-4}, 1.65 \times 10^{-4}, 9.3 \times 10^{-5}, 5.5 \times 10^{-5}]. \quad (4.56)$$

Results and Discussion

The following section reports the complex and coupled dynamics of a UV-curing system in an advanced manufacturing setting. Simulations were performed using the input and material parameters reported above, and the resulting dynamics of the reaction rate coefficients, fractional free volume, concentrations, and temperature are reported below. In this section, I also report the effects resulting from the impact of solid particles, ambient temperature, and UV intensity on the reaction profiles. From these results, I discuss the input parameters that determine the evolution of polymerization in this system and suggest parameters that have the highest impact on the quality of manufacturing. The time evolution of each field can be observed in Fig. 4.5, and the accompanying animated results are can be found in Fig. 4.10 for the first half second and Fig. 4.11 for the last 29 seconds.

Evolution of Reaction Rate and Diffusion Coefficients

The reaction rate and diffusion coefficients are the primary control of the evolution of each field. Over time, each of these dynamically affect each other and drive the reaction faster or slower. The results shown in Fig. 4.3 and Fig. 4.4 indicate that within the first 5s of UV light exposure, there is a rapid decrease in reaction potential. In particular, the termination rate coefficient decreases more rapidly than the propagation rate coefficient. This may have implications on the quality of the final polymer since the termination reaction caps the radical on an active polymer chain. When termination is low, this indicates that radicalized

polymers are still active within the system and may serve as initiation sites for degradation, thus affecting the aging of the material a longer time horizon.

Additionally, the average free volume (ability for monomer movement), and by extension, the diffusion coefficient, also exponentially drop within the first 5s of exposure to UV light (Fig. 4.4a and Fig. 4.4). While this greatly affects the reactivity of the system, it is challenging to avoid this aspect since the primary goal of the manufacturing process is rapid polymerization solidification of the system.

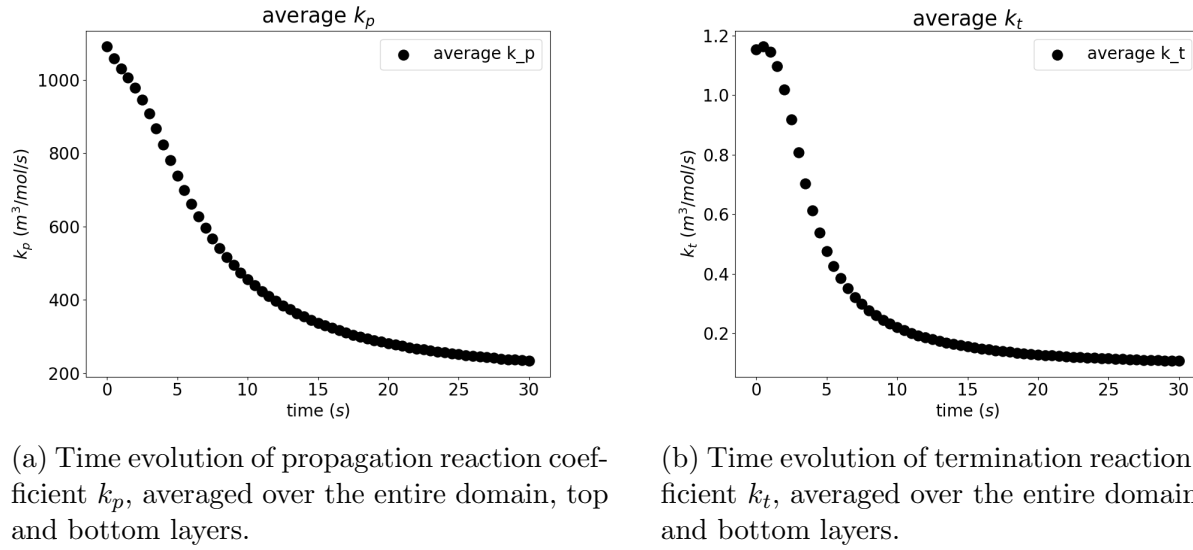


Figure 4.3: Comparison of reaction coefficients: (a) Propagation reaction coefficient k_p , and (b) Termination reaction coefficient k_t .

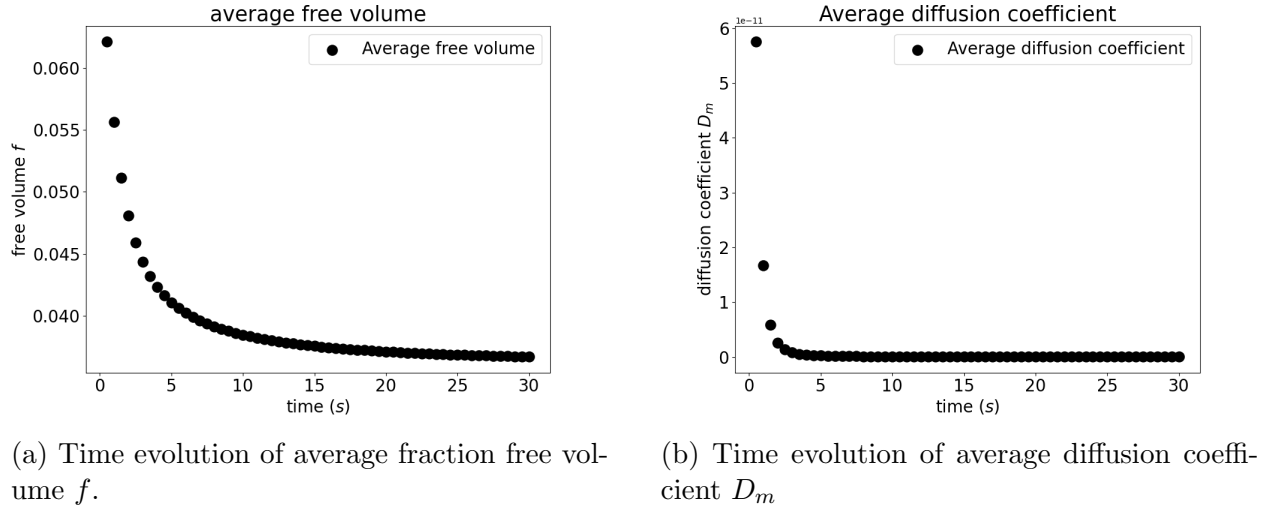


Figure 4.4: The rate of diffusion of monomers over time.

Evolution of Temperature and Concentration Fields

The temperature and concentration profiles are shown in Fig. 4.5, where the reported values are average fields across the entire domain, the top layer, and bottom layer over 30 seconds. For the concentration fields, the averages only include voxels that represent polymers, whereas temperature fields take into account voxels representing polymers and particles.

The dissociation of PI is well understood and is designed to quickly reach completion for materials that do not interfere with the required wavelengths (400-500 nm). As expected, the concentration profile for PI decreases rapidly to zero within the first few seconds (Fig. 4.5a) of the simulation. Minimal spatial gradients form between the top and bottom layers, indicating that the UV light source efficiently initiates the radicalization of the PI.

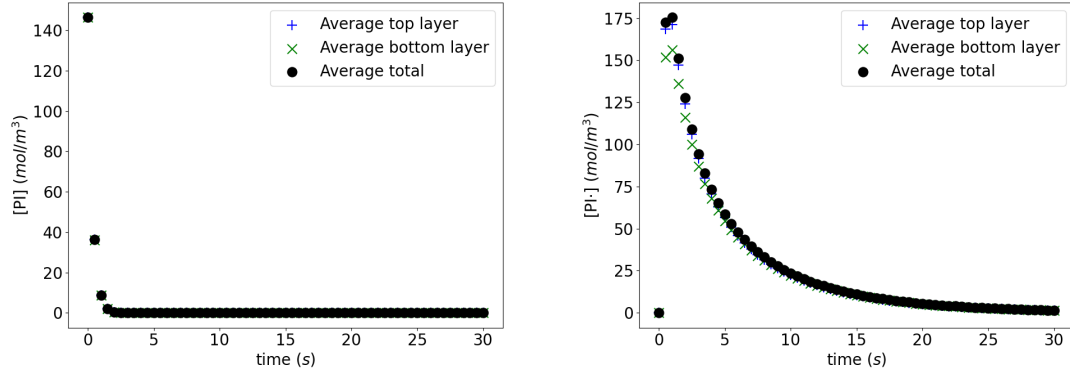
The polymer system begins to form gradients during the propagation phase of the process. The concentration profile for the primary radical $PI\cdot$ initially starts at zero, undergoes a near-step increase, and then slowly polymerizes over the 30-second simulation period (Fig. 4.5b). Spatial gradients begin to form between the top and bottom layers, with concentrations along the bottom layer initially lagging behind those near the top. Similar dynamics are observed for the formation and depletion of $M\cdot$ (Fig. 4.5c), indicating that the gradients across the z-axis impact the reaction profiles downstream from the primary radical $PI\cdot$. The concentration of monomer M (Fig. 4.5d) decreases rapidly and stabilizes after about 5 seconds, suggesting that monomers quickly convert to active chains but freeze in place with a low likelihood of further polymerization after the initial few seconds. Note that a more common representation in polymer literature is the percent conversion, as shown in Fig. 4.5e, which estimates 84% monomer conversion (as defined by Eqn. 4.12) after 30 seconds.

of exposure to UV light. This prediction is inline with the experimental results reported in Chapter 3 Fig. 3.11 in which FTIR performed on samples exposed to UV light for 30 seconds (without subsequent heating) achieve 81% conversion.

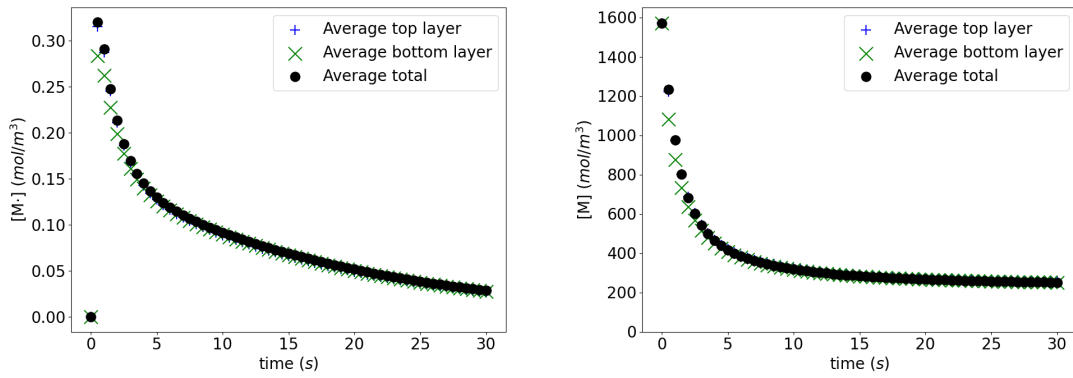
Close examination of the temperature field reveals that most of the dynamics occur within the first 5 seconds of UV light exposure. The largest temperature increase peak matches the dynamics of Eqn. 4.9, Eqn. 4.10, and Eqn. 4.11 at similar times. These simulations indicate that temperature fluctuations are primarily dominated by reactions rather than UV exposure suggesting that at these intensity levels, the UV light source can be safely used with controlled risk for temperature-sensitive materials.

Comments:

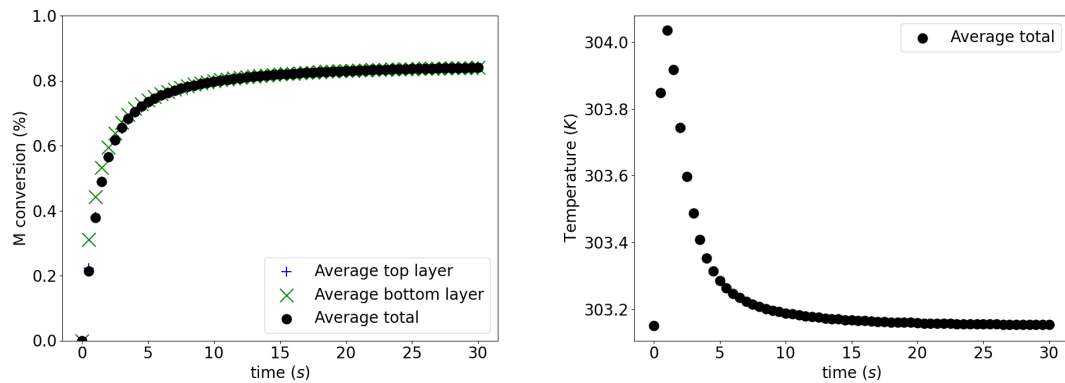
The spatial gradients in Fig. 4.5b Fig. 4.5c indicate that reactants are forming at different rates at different layers within the material. As time goes on, the concentrations of these intermediary species eventually converge to roughly the same average quantity. It is important to note that these spatial gradients during reaction profile indicate that differ polymer structures are forming across the different layers. The purpose of this work is to identify overarching trends resulting from input parameters that may be critical to advanced manufacturing applications. However if a deeper understanding is required for polymer structure or average branch length, performing molecular dynamics simulations is critical to quantifying the molecular differences at different layers.



(a) $[PI]$ over time, averaged over the entire domain, top and bottom layers. (b) $[PI\cdot]$ over time, averaged over the entire domain, top and bottom layers.



(c) $[M\cdot]$ over time, averaged over the entire domain, top and bottom layers. (d) $[M]$ over time, averaged over the entire domain, top and bottom layers.



(e) Conversion of M over time, averaged over the entire domain, top and bottom layers. (f) Temperature evolution averaged across the entire simulation domain.

Figure 4.5: Temperature and concentration evolution over 30s simulations.

Ink vs Resin

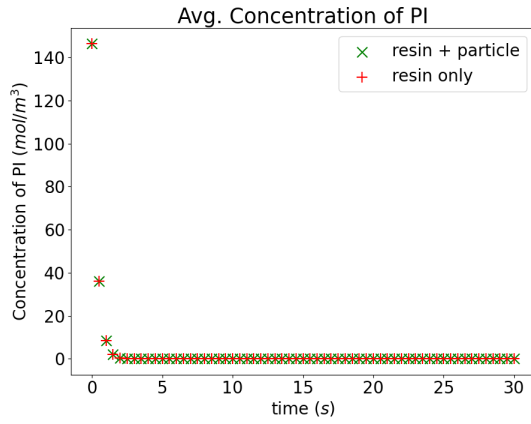
Particles embedded within a resin to formulate an ink or composite appropriate for advanced manufacturing applications are typically chosen to be inert relative to the polymerization mechanism. However, since the direct factors that drive the reactions forward are the reaction rate and diffusion coefficients, which in turn are functions of temperature, it becomes necessary to consider the thermal properties of the particle and its larger effect on the reactivity of the polymer system. The results shown in Fig. 4.6 indicate that the NaCl particles act as thermal sinks and have a significant effect on the temperature of the overall system. This impact on temperature does affect the concentration profile, but these simulations indicate that overall conversion is unaffected by the NaCl particles.

Temperature sweep

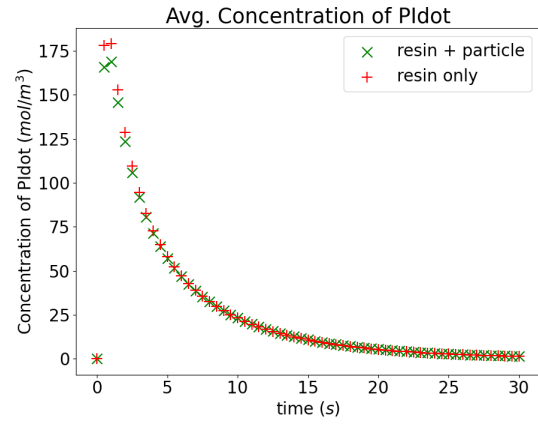
Ambient temperature also plays a critical role in the evolution of the polymer system since all fields are fundamentally functions of temperature (Fig. 4.7). The results shown in Fig. 4.7f confirm the impact of ambient temperature on the reaction profiles for chemical species, having a variable impact on the final state for each field. As the ambient temperature sweeps from $298.15K$ to $335.15K$, the conversion of monomer approaches 100%. However, this comes at a cost since active radicalized species $PI\cdot$ and $M\cdot$ appear not to terminate at higher temperatures, leaving higher concentrations of active chains in the final polymer state. Having lower concentrations of monomer $[M]$ in the final material may prevent monomers leaching from the polymer in the long run, but higher concentrations of radicalized polymer chains will serve as degradation sites and promote aging, particularly when applications of the polymer require cyclical stress and strain forcing mobility of these active chains with damaged sites from creep or fatigue.

UV intensity sweep

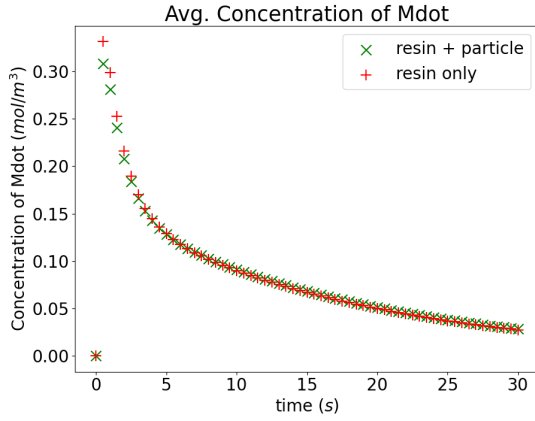
Another parameter worth observing is the intensity of the UV light source. A UV intensity sweep was performed for values ranging from $2\text{--}100 \frac{mW}{cm^2}$ and the results are reported in Fig. 4.8. It is clear that the UV intensity has minimal impact on the conversion of monomer M Fig. 4.8d and the final state of $PI\cdot$ Fig. 4.8b. However, there appears to be a direct effect on the rate of dissociation Fig. 4.8a and on the termination of active chains $M\cdot$ Fig. 4.8c. The results Fig. 4.8a are expected and match the UV-sweep experimental results in Chapter 3, however the results in Fig. 4.8c indicate that the reported UV intensities in the experimental work of Chapter 3 are too low to manage unreacted chains. Increasing the UV intensity to about $10 \frac{mW}{cm^2}$ appears to bring the highest benefit for polymerization while still constraining UV intensity for safety and energy purposes.



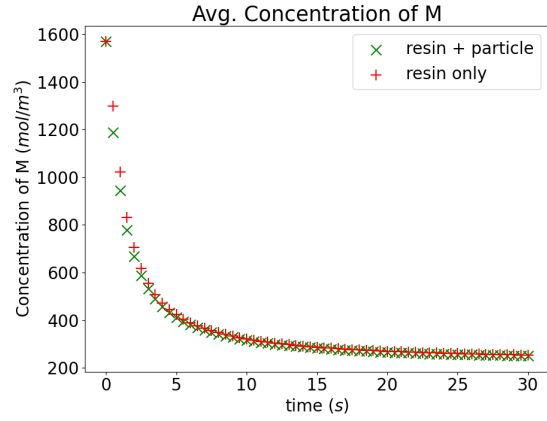
(a) Average concentration of PI over time.



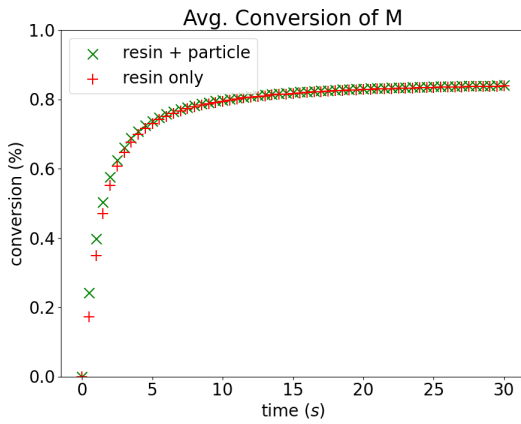
(b) Average concentration of $PI\cdot$ over time.



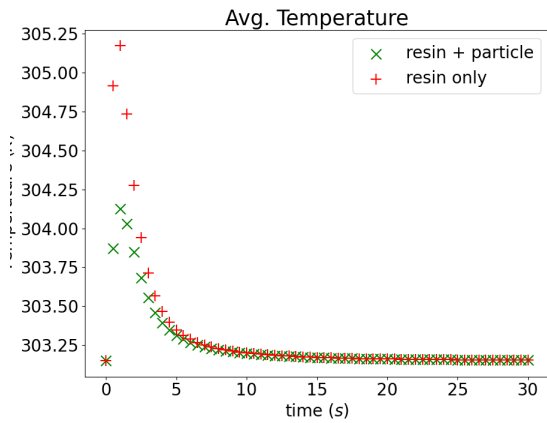
(c) Average concentration of $M\cdot$ over time.



(d) Average concentration of M over time.

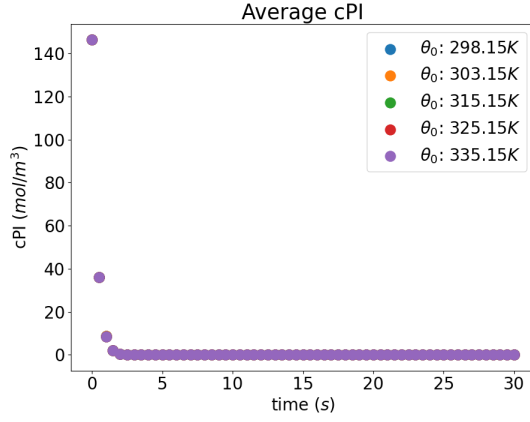
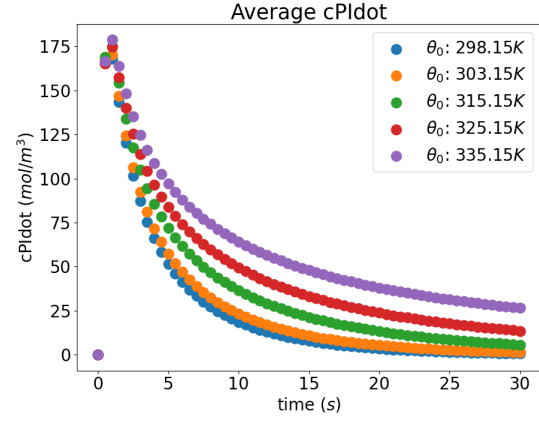
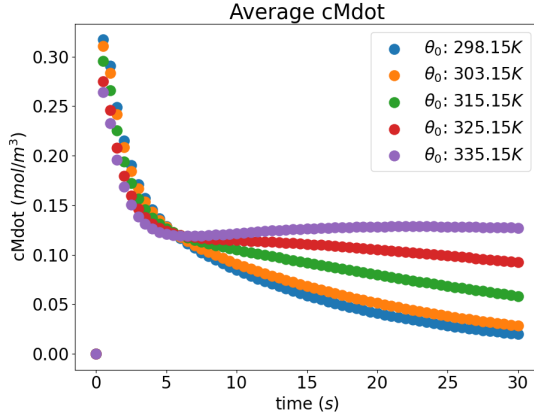
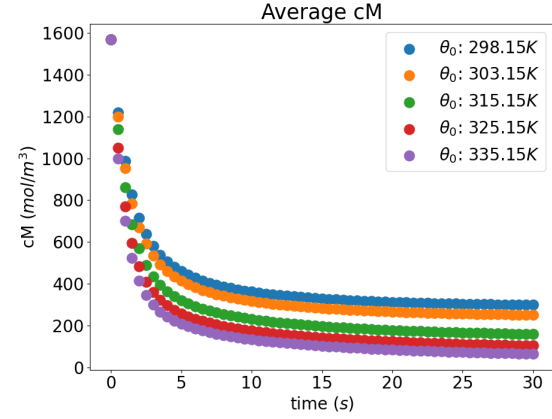
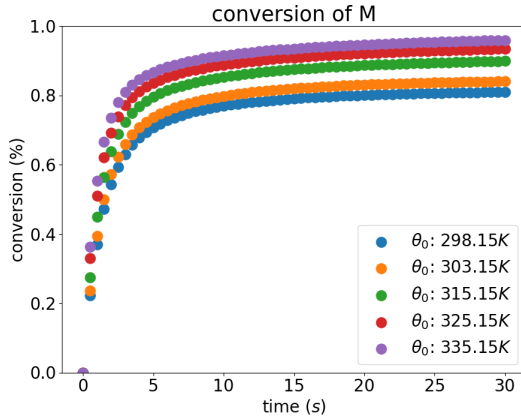
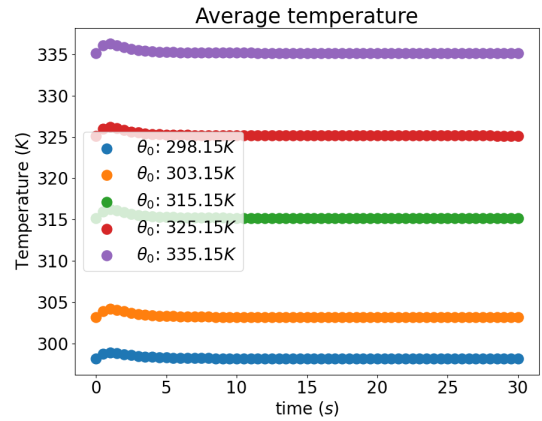


(e) Conversion of average M over time.



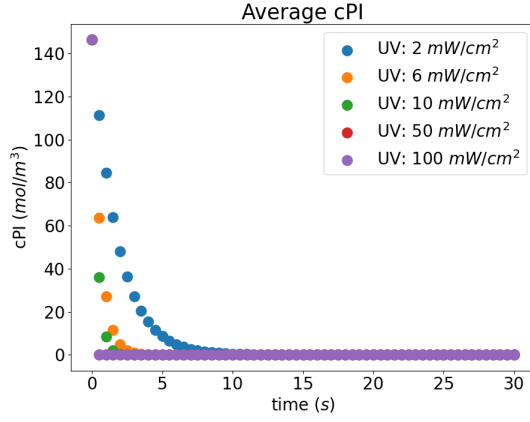
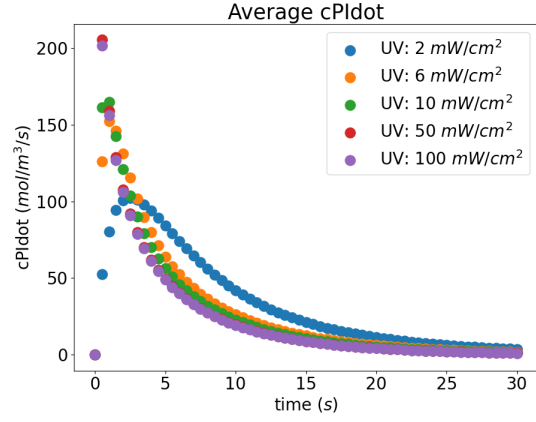
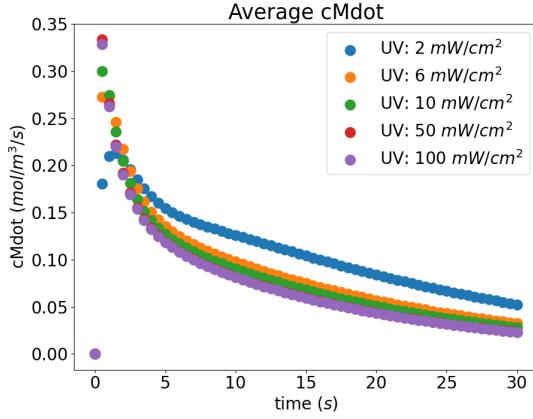
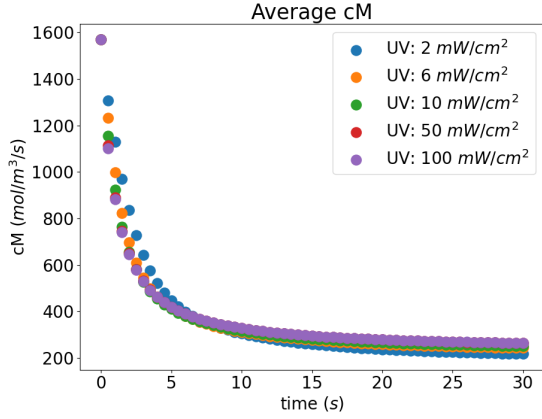
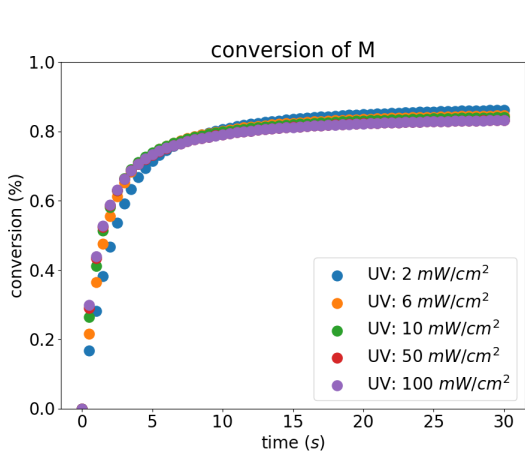
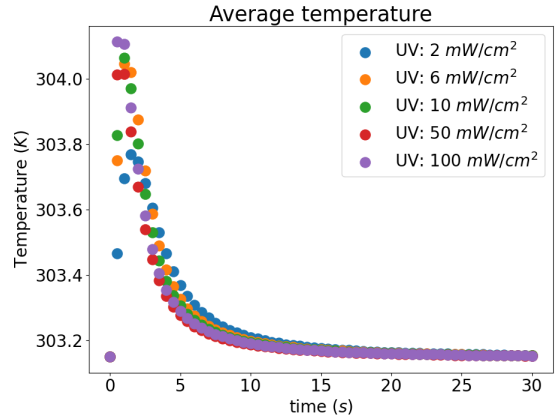
(f) Temperature evolution averaged across the entire simulation domain.

Figure 4.6: Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations.


 (a) Average concentration of PI over time.

 (b) Average concentration of $PI\cdot$ over time.

 (c) Average concentration of $M\cdot$ over time.

 (d) Average concentration of M over time.

 (e) Conversion of average M over time.


(f) Temperature evolution averaged across the entire simulation domain.

Figure 4.7: Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations for a sweep of ambient temperatures.

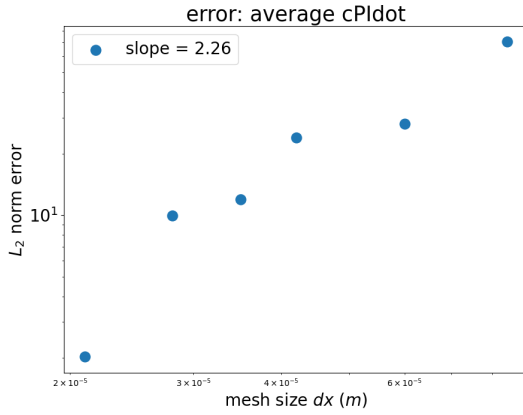

 (a) Average concentration of PI over time.

 (b) Average concentration of $PI\cdot$ over time.

 (c) Average concentration of $M\cdot$ over time.

 (d) Average concentration of M over time.

 (e) Conversion of average M over time.


(f) Temperature evolution averaged across the entire simulation domain.

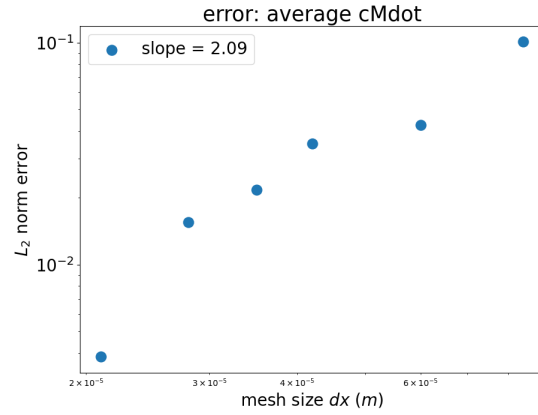
Figure 4.8: Temperature and concentration evolution of ink (particle-laden) vs resin (only polymer) over 30s simulations for a sweep of UV intensities.

Numerical Stability

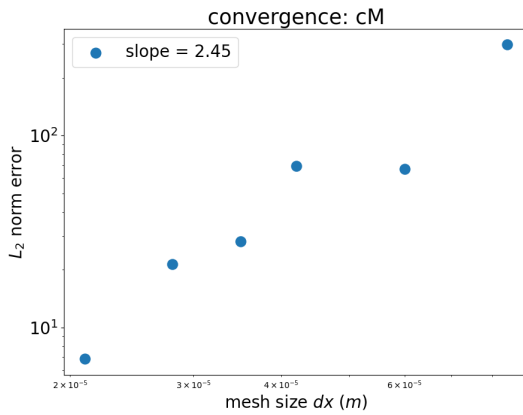
Convergence is a fundamental aspect of assessing the stability and accuracy of computational models. The figures presented in this subsection illustrate the convergence behavior of various key equations. These include Eqn. 4.9, Eqn. 4.10, Eqn. 4.11, and Eqn. 4.20, analyzed through the numerical stability of temperature and concentration profiles under temporal and spatial discretization refinement. The errors are quantified using $k = 5.5 \times 10^{-5}$ and $h = 1.65 \times 10^{-5}$ as the benchmarks for the estimated “true” solution. The results, depicted in Figures 4.9a through 4.9d, demonstrate that the logarithmic slopes of the error curves confirm the attainment of second-order accuracy.



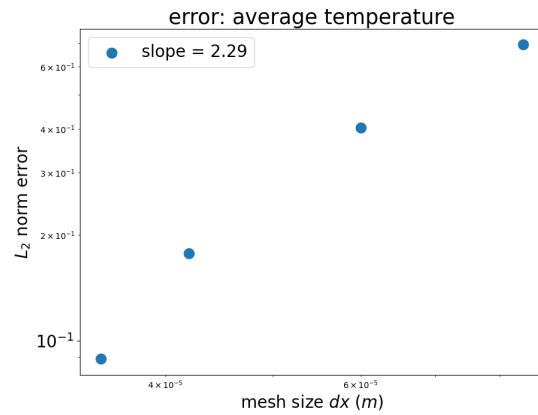
(a) Convergence of Eqn. 4.9.



(b) Convergence of Eqn. 4.10.



(c) Convergence of Eqn. 4.11



(d) Convergence of Eqn. 4.20.

Figure 4.9: Numerical stability for temperature and concentration profiles over a temporal/spatial discretization refinement.

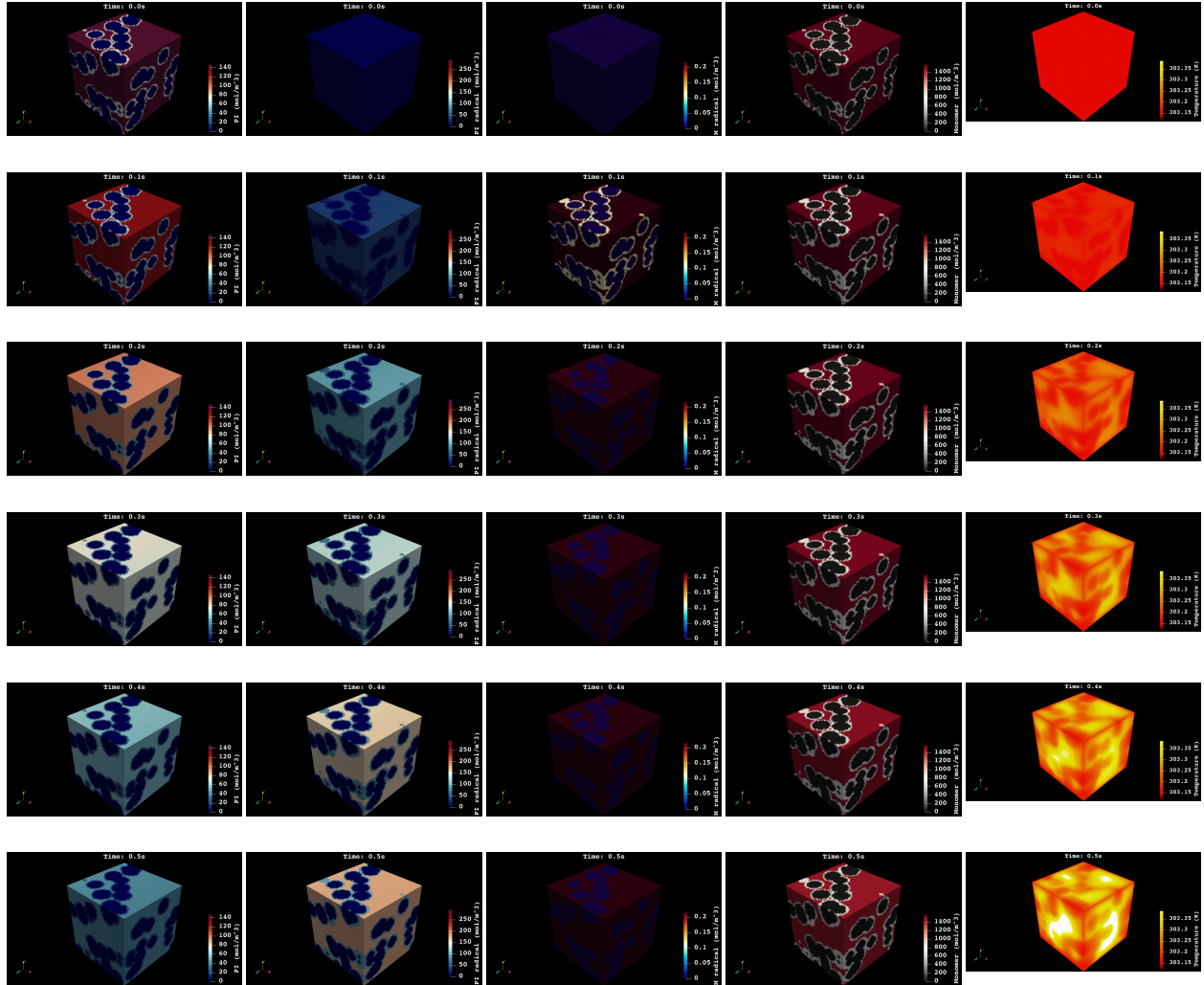


Figure 4.10: **Dynamics of the first half second:** Columns 1-5 show the time evolution of the concentration of PI, the primary radical (PI $\cdot$), the propagating radical (M $\cdot$), and the monomer (M), each with units mol/m^3 . The dynamics of the first 0.5 seconds are dominated by the rapid conversion of PI (column 1) to the primary radical (column 2). During this rapid reaction, heat is generated from within and is convected from the surface to the environment, as indicated by the temperature distribution in column 5.



Figure 4.11: **Dynamics of next 29s**: Columns 1-5 show the time evolution of the concentration of PI, the primary radical ($\text{PI}\cdot$), the propagating radical ($\text{M}\cdot$), and the monomer (M), each with units mol/m^3 . The dynamics from 1-30s are dominated by the slower conversion of primary radical (column 2) to the propagating radical (column 3) to consumption of monomer (column 4). During this slower reaction phase, the convection to the surrounding environment provides a more stable temperature distribution.

4.4 Summary and Extensions

This chapter has explored the complex and coupled dynamics of a UV-curing system within an advanced manufacturing context. The simulations conducted revealed critical insights into the evolution of reaction rate coefficients, fractional free volume, concentrations, and temperature fields over time, highlighting the influence of various parameters such as solid particles, ambient temperature, and UV intensity on the reaction profiles.

Key findings include:

- **Reaction Rate and Diffusion Coefficients:** The rapid decrease in reaction potential within the first 5 seconds of UV exposure was observed, with the termination rate coefficient decreasing more rapidly than the propagation rate coefficient. This has significant implications for the quality and aging of the final polymer.
- **Temperature and Concentration Fields:** Temperature fluctuations were found to be primarily dominated by reactions rather than UV exposure. The formation of spatial gradients in concentration fields, particularly in the propagation phase, indicates different polymer structures forming across the layers.
- **Impact of Solid Particles:** NaCl particles, acting as thermal sinks, significantly impacted the system's temperature, though they did not affect the overall conversion.
- **Ambient Temperature Sweep:** Increasing ambient temperature led to nearly 100% monomer conversion but resulted in higher concentrations of active radicalized species, potentially promoting material degradation over time.
- **UV Intensity Sweep:** UV intensity was found to have a minimal impact on monomer conversion but significantly influenced the dissociation rate and termination of active chains, with an optimal intensity of around $10 \frac{mW}{cm^2}$ suggested for balancing polymerization efficiency and safety.

These findings suggest that careful control of reaction conditions, particularly within the first few seconds of UV exposure, is crucial for optimizing the polymerization process and the quality of the final product. The insights gained from the temperature and concentration profiles provide a deeper understanding of the underlying dynamics and highlight the importance of considering spatial gradients in advanced manufacturing applications.

Extensions

Future work can build upon these findings by exploring several key areas:

- **Optimization of Particle Properties:** Investigating the thermal and chemical properties of different particles can further optimize the ink or composite formulations for advanced manufacturing applications.

- **Molecular Dynamics Simulations:** To gain a more detailed understanding of polymer structure and average branch length, molecular dynamics simulations can be performed. This will help quantify molecular differences across layers and provide deeper insights into the material properties.
- **Long-term Aging Studies:** Conducting long-term studies on the aging and degradation of the polymer under various environmental conditions and stressors will help assess the materials' durability and performance over time.
- **Experimental Validation:** Validating the simulation results with experimental data, particularly focusing on the spatial gradients and reaction profiles, will enhance the reliability and applicability of the findings.
- **Advanced Control Strategies:** Developing advanced control strategies for UV intensity, ambient temperature, and exposure duration can further optimize the manufacturing process and improve product quality.

By addressing these areas, future research can contribute to the development of more efficient and reliable UV-curing systems, advancing the fields of advanced manufacturing and polymer science.

Chapter 5

Computational Optimization for Material Design

“Bottomless wonders come from simple rules repeated without end.”

- Benoit Mandelbrot

The source code for the optimization algorithms, simulations, and experiments in this chapter can be found in the following GitHub repositories:

https://github.com/bmhowell/ugap_bopt
https://github.com/bmhowell/ugap_cmaes
https://github.com/bmhowell/ugap_ga

5.1 Introduction

In modern manufacturing processes, the optimization of chemical properties, material composition, and processing parameters is crucial for achieving desired performance within design and manufacturing constraints. Traditionally, formulation engineers and materials scientists have relied on iterative strategies involving trial-and-error methods or brute-force design of experiments (DOE). This conventional approach begins with the selection of materials and operating parameters based on intuition and experience. Subsequently, a series of batch formulations and tests are designed to identify trends that inform additional rounds of testing. This cycle typically continues until either the material meets critical benchmarks or available resources—time, materials, or funding—are depleted.

Recent advancements in computing have revolutionized this process by enabling rapid simulation of multi-physics processes at unprecedented scales [69, 70, 71]. These physics-based simulations offer significant advantages over traditional DOE approaches. They are faster to execute, more cost-effective, and have a reduced environmental impact [3, 6]. Moreover, they minimize engineering time and manufacturing resources while decreasing overall risk in the development process.

The logical progression from these computational advancements is to leverage these frameworks to simulate various configurations, thereby identifying optimal input parameters for desired performance metrics *within a virtual environment*. To further accelerate the search for optimal material design and operating parameters, numerical optimization techniques can be employed to efficiently traverse complex search spaces using stochastic [72, 4, 7] or gradient-based [73, 74] methods. Additionally, machine learning techniques offer efficient surrogate models that may be used in the parameter sampling process allowing for faster convergence to optimal parameters [32, 75, 73, 76]. In the limit, these optimization approaches can be integrated with hardware and robotics to achieve a fully automated material-discovery process [77, 78], accelerating industry’s capability of improving product quality while lowering operational costs, environmental impact, and safety risks associated with chemical exposure and waste.

This chapter presents the final component with the simulation-informed optimization and machine learning framework proposed in Chapter 1. In this chapter, I apply a variety of stochastic optimization methods to the simulation framework detailed in Chapter 4. The results from this process can be directly reapplied to the formulation work presented in Chapter 3, creating a holistic iteration process. This approach is particularly well-suited to product cycles in industrial environments, allowing for continuous performance improvements.

In essence, this work demonstrates a comprehensive approach that combines advanced simulation techniques with stochastic optimization methods and machine learning algorithms. This integrated framework offers a powerful tool for efficient and effective material design and process optimization in modern manufacturing. By closing the loop between simulation, optimization, and practical application, a more robust and adaptive approach is enabled to formulation and manufacturing challenges.

5.2 Materials Optimization Strategy

Optimization, whether in the lab or simulation, requires identifying input parameters that may affect the final material quality [5, 79]. These parameters span multiple domains: material properties, chemical formulation, processing conditions, and environmental variables. By systematically varying these inputs, researchers and engineers can optimize materials for specific applications and performance requirements.

For every application, there are metrics of “goodness” or performance. These metrics may be challenging to condense into a quantitative formula due to unintended consequences in areas not initially considered (e.g., cost, processing time). In numerical optimization, it is common to compile many of these metrics into a composite objective that weights each individual metric based on the application’s priority.

A critical step in materials optimization is identifying which input parameters can be varied. By systematically adjusting these parameters and measuring their impact on performance metrics, researchers can develop optimized materials for specific applications. This

Domain	Examples of Parameters
Material Properties	• Composition (elemental/compound ratios) • Microstructure (particle size/distribution, phase distribution, crystallinity) • Physical properties (density, thermal conductivity, electrical resistivity)
Chemical Formulation	• Reactant ratios • Catalyst types/concentrations • Additives, stabilizers • pH levels • Solvent selection and concentration
Processing Parameters	• Temperature profiles (heating/cooling rates, ambient temperature) • Pressure conditions • Applied energy sources (e.g., laser intensity) • Mixing speeds • Processing time • Flow rates • Application methods (spraying, dipping, casting)
Environmental Variables	• Humidity levels • Atmospheric composition • Light exposure • Electromagnetic fields

Table 5.1: Optimization parameters in materials development

Category	Performance Metrics
Mechanical Properties	Tensile strength, elasticity (Young's modulus), hardness, toughness, fatigue and wear resistance
Chemical Properties	Reaction conversion/completion, yield of desired products, selectivity in multi-product reactions, stability against degradation, corrosion resistance
Physical Properties	Thermal/electrical conductivity, optical transmittance/reflectance/color, magnetic properties
Processing Metrics	Cure time, viscosity, aging, shelf life, ease of application
Economic and Environmental Factors	Cost of raw materials, energy consumption, environmental impact, scalability of production

Table 5.2: Performance metrics in materials optimization

process often involves trade-offs between different properties, necessitating a holistic approach that considers the interplay between various factors.

5.3 No Free Lunch

Efficient exploration of the parameter space typically involves carefully designed experiments or optimization algorithms, often leveraging computational simulations (or their surrogates) to accelerate the process and reduce resource consumption. Similar to experimental approaches such as trial-and-error and DOE, a simulation engineer must identify a search approach that is appropriate given the problem at hand. The best search approach depends on three main factors:

- Assumptions of underlying dynamics: access to gradients, correlated outputs, continuously differentiable
- Computational cost: required resources, length of simulation
- Number of optimization variables: e.g., $\mathcal{O}(10)$, $\mathcal{O}(10^3)$, $> \mathcal{O}(10^6)$

The No Free Lunch (NFL) Theorem, introduced by Wolpert and Macready [80], provides a fundamental insight into algorithm selection. It states that no single optimization algorithm outperforms all others across all possible problems. Mathematically, the theorem states that for any pair of algorithms a_1 and a_2 :

$$\sum_f P(d_m^y \mid f, m, a_1) = \sum_f P(d_m^y \mid f, m, a_2) \quad (5.1)$$

where d_m^y denotes the ordered set of size m of the cost values y associated with input values $x \in X$, $f : X \rightarrow Y$ is the function being optimized and $P(d_m^y \mid f, m, a)$ is the conditional probability of obtaining a given sequence of cost values from algorithm a run m times on function f (see [80]). This theorem implies that when an algorithm performs well on a certain class of problems, it necessarily pays for that with degraded performance on the set of all remaining problems. Therefore, the choice of optimization algorithm should be tailored to the specific characteristics of the problem at hand. The factors mentioned above significantly influence algorithm selection. For instance:

1. If gradients are available and the function is smooth, gradient-based methods like gradient descent or BFGS are often efficient.
2. For problems with correlated outputs, methods that can exploit this structure, such as GP-based optimization, can be particularly effective.
3. If compute is cheap and the problem is low-dimensional, grid search or random search can be the best approach due to their simplicity and ability to explore the entire space.

4. For expensive computations without access to gradients, BO is a good choice as it attempts to minimize the number of function evaluations.
5. In high-dimensional spaces with access to gradients, first-order methods such as stochastic gradient descent or its variants (e.g., Adam) are often preferred due to their ability to handle large-scale problems efficiently.

These factors and the NFL theorem allow practitioners to make informed decisions when selecting optimization algorithms for their specific problems, balancing the trade-offs between exploration, exploitation, and computational resources (see Fig. 5.1).

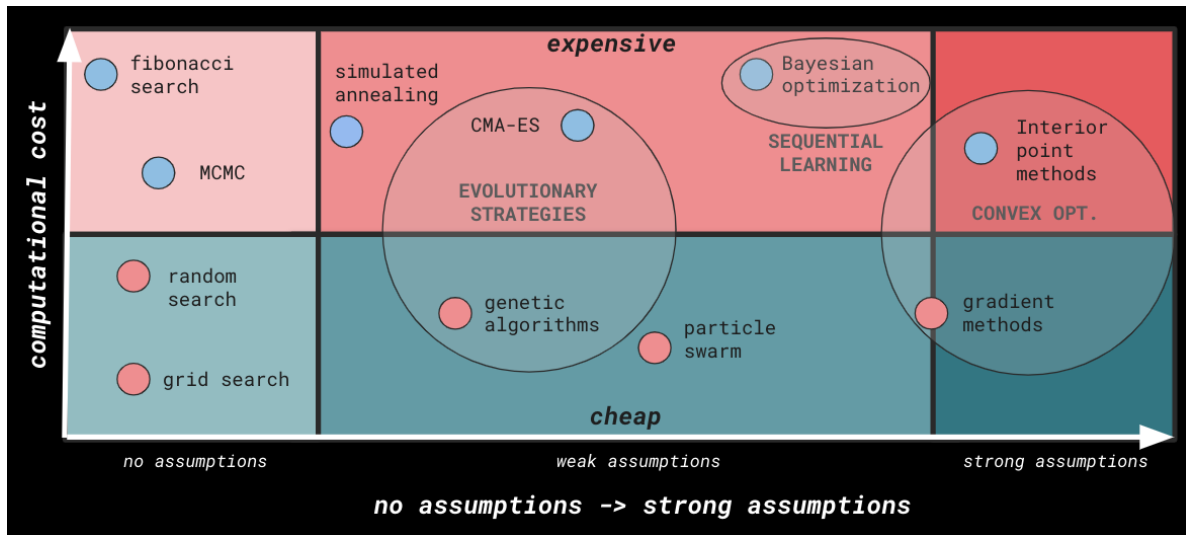


Figure 5.1: Example space of potential algorithms parameterized by the assumptions that can be made about the underlying objective and the computational cost.

5.4 Applications to Advanced Manufacturing

The work in Chapter 3 provides the framework in which simulation-informed optimization can be applied to additive manufacturing environments. The results from the experiments and simulations in Chapter 2 & 3 provide insights on the key parameters that may have an impact on the quality of materials used in UV-dependent inks in direct-ink-write (DIW) printing. In this chapter, I focus on optimizing five critical variables:

- particle volume fraction,
- particle diameter,
- UV exposure time,

- UV exposure intensity,
- ambient temperature.

These parameters were chosen based on their estimated impact on the curing process and final material properties, as demonstrated by experimentation and simulation. By systematically varying these parameters within the voxel-based model, numerical optimization offers a mechanism by which exploration of a vast design space can be performed to identify optimal combinations that yield desired material properties. This approach allows for:

- minimization of uncured monomers to enhance long-term stability,
- optimization of curing speed for improved manufacturing efficiency,
- control of spatial gradients in polymer properties for functionally-graded materials,
- balancing of thermal effects to prevent overheating and maintain dimensional accuracy.

The optimization process can be guided by objectives such as maximizing degree of cure, minimizing curing time, or achieving specific mechanical properties. Constraints related to manufacturing limitations, such as maximum allowable temperatures and/or UV intensities, can be readily incorporated into the optimization framework.

5.5 Designing the Multi-Objective Function

The specific objective function should directly relate to the use case of the ink. In the context of Chapter 3, UV-curable inks used as support material for other materials may require a different objective formulation than those applied to the manufacturing of objects intended for actual use. As a support material, rapid stiffening is the ultimate objective, thus it is intuitive to construct an objective function that measures the rate of depletion of the starting monomer. For the purposes of this work, the objective will be to choose the best input parameters to achieve a high quality material. One way to quantify the quality of the material is to minimize concentration of starting and intermediary chemical species obtained from the last state of the solution. For example, for an arbitrary species A , a potential objective might be defined as:

$$\min_{\boldsymbol{\lambda}} \left(\frac{[\bar{A}]_{tf}}{\max([\bar{A}]_t)} \right) \quad (5.2)$$

where $\boldsymbol{\lambda}$ is a vector of optimization variable, $[\bar{A}]$ is the average concentration of species A (across all voxels representing the polymer), t represents the time step and tf represents the final time step. Explicitly, the objective function computes the ratio of the average concentration across all polymer voxels relative the max (averaged) concentration across

all time steps. This set up enables a basic metric for reaction completion of starting or intermediary species.

For all species of interest in Chapter 3 & 4, a weighted sum of each of the individual objectives can be composed:

$$\begin{aligned}
 \min_{\boldsymbol{\lambda}} \quad & w_0 \left(\frac{[\bar{P}I]_{tf}}{\max([\bar{P}I]_t)} \right) \\
 & + w_1 \left(\frac{[\bar{P}I\cdot]_{tf}}{\max([\bar{P}I\cdot]_t)} \right) \\
 & + w_2 \left(\frac{[\bar{M}\cdot]_{tf}}{\max([\bar{M}\cdot]_t)} \right) \\
 & + w_3 \left(\frac{[\bar{M}]_{tf}}{\max([\bar{M}]_t)} \right) \\
 \text{subject to } & \boldsymbol{l} \leq \boldsymbol{\lambda} \leq \boldsymbol{u}
 \end{aligned} \tag{5.3}$$

where $\mathbf{1}^\top \boldsymbol{w} = 1$, $\boldsymbol{w}$ are weights of relative importance in the multi-objective optimization formulation and $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower and upper bounds.

For physical systems, the operating constraints may come in a variety of forms. For process parameters such as operating temperature or UV intensity, the constraints are typically defined by the equipment, safety protocols, or by some natural lower and upper bounds. For material parameters such as volume fraction of solids or particle size, parameter sweeps in experimental environments can identify the bounds of feasibility. The constraints used in this work were identified as:

$$\begin{aligned}
 \theta_{amb} &\in [273.15K, 350.0K] \\
 r_p &\in [4.2\mu m, 84\mu m] \\
 v_p &\in [50 \text{ vol.}\%, 80 \text{ vol.}\%] \\
 I_{uv} &\in [2.0W/m^2, 100.0W/m^2] \\
 t_{uv} &\in [1.0s, 30.0s]
 \end{aligned} \tag{5.4}$$

where θ_{amb} , r_p , v_p , I_{uv} , t_{uv} are the ambient temperature, particle radius, volume fraction, UV intensity and UV exposure time respectively.

Pareto Optimality

Multi-objective optimization is a notoriously challenging topic in the context of industry, primarily due to the bias introduced when tuning the weights $\boldsymbol{w}$ by hand. One approach is to simulate the pairwise combinations to identify the Pareto optimal set of weights. A solution is considered Pareto optimal if it cannot be improved in any of the objectives without

degrading at least one of the other objectives. The set of all Pareto optimal solutions forms what is known as the Pareto front.

In the context of the UV-curable ink optimization problem, the Pareto frontier represents the set of solutions where the concentration of one chemical species cannot be further minimized without increasing the concentration of another. This approach allows us to visualize and understand the trade-offs between the different objectives, without the need to explicitly define weights.

Fig. 5.2 illustrates pairwise Pareto fronts for the objectives obtained using evolutionary algorithms (NSGA-II). Each point on these fronts represents a Pareto optimal solution and the curves show the trade-offs between pairs of objectives.

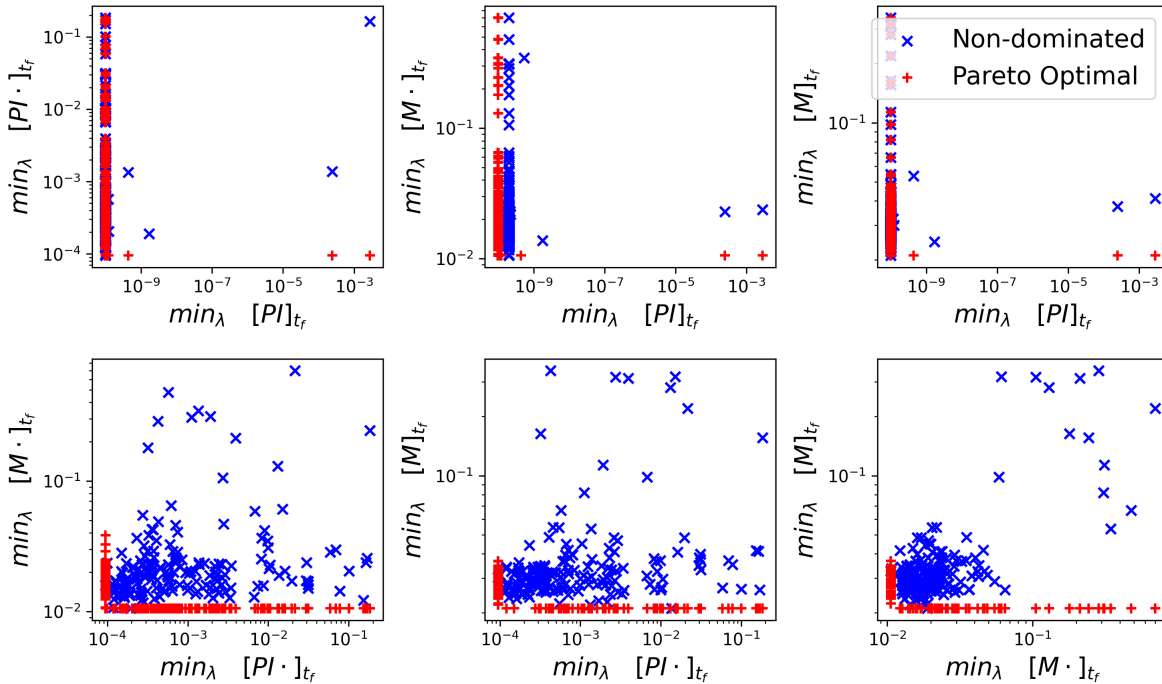


Figure 5.2: Pairwise Pareto fronts for objective pair in which the relative weights for each pairwise objectives are varied.

Each of these plots produce a point along the Pareto front that minimizes both objectives. These relative ratios of these values were computed and compiled into the following system of equations:

$$w_0 \cdot 6.5e^5 = w_1 \quad (5.5)$$

$$w_0 \cdot 8.0e^7 = w_2 \quad (5.6)$$

$$w_0 \cdot 2.0e^8 = w_3 \quad (5.7)$$

$$w_1 \cdot 1.0e^2 = w_2 \quad (5.8)$$

$$w_1 \cdot 3.0e^2 = w_3 \quad (5.9)$$

$$w_2 \cdot 2.5 = w_3 \quad (5.10)$$

$$\mathbf{1}^\top \mathbf{w} = 1 \quad (5.11)$$

which in turn can be represented in the following way:

$$\begin{bmatrix} 6.5e5 & -1 & 0 & 0 \\ 8.0e7 & 0 & -1 & 0 \\ 2.0e8 & 0 & 0 & -1 \\ 0 & 1e2 & -1 & 0 \\ 0 & 3e2 & 0 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \quad (5.12)$$

The Pareto optimal weights for the remaining work was obtained by using the pseudo-inverse:

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} = \begin{bmatrix} 3.56574286e^{-9} \\ 2.42560512e^{-3} \\ 0.280839829 \\ 0.714916061 \end{bmatrix}. \quad (5.13)$$

Comments:

Computing Pareto optimal weights for multi-objective optimization is the standard approach when attempting to balance objectives equally. However, there may be scenarios in which hand tuning these weights may prove beneficial. One example related to the formulation of UV-curable inks may be that the appropriate use case for an ink may purely be as a support structure for an alternative ink, in which case the final quality of the polymer may not be of concern, and we would like to optimize for rapid solidification. Under such requirements, and example weighting may look more like:

$$\begin{aligned}
\min_{\lambda} \quad & .02 \left(\frac{[\bar{P}I]_{tf}}{\max([\bar{P}I]_t)} \right) \\
& + 0.78 \left(\frac{[PI\cdot]_{tf}}{\max([PI\cdot]_t)} \right) \\
& + .1 \left(\frac{[\bar{M}\cdot]_{tf}}{\max([\bar{M}\cdot]_t)} \right) \\
& + .1 \left(\frac{[\bar{M}]_{tf}}{\max([\bar{M}]_t)} \right).
\end{aligned} \tag{5.14}$$

5.6 Bayesian Optimization

Bayesian Optimization (BO) is a technique particularly well-suited for optimizing expensive objective functions within low-dimensional search spaces (a *common rule of thumb* ≤ 20). It is especially effective in scenarios where the computational cost of evaluating the objective is costly (e.g., high-fidelity physics simulations, hyperparameter tuning in training deep learning models, etc.) in terms of computational resources or energy. As a stochastic method, BO treats the objective as a blackbox and is beneficial to when optimization variables implicitly affect the objective function (e.g., the particle radius within voxelized representation of a material) or when the computation cost of computing the gradients is high.

At its core, BO is an algorithmic approach to decision making under uncertainty that leverages machine learning techniques to model the provided data points and make intelligent decisions on where to sample next. This approach allows BO to rapidly find high-potential solutions with relatively few objective function evaluations making it ideal for expensive optimization problems.

Bayesian Optimization makes several key assumptions about the underlying data and the function being optimized:

1. **Smoothness:** The objective function is assumed to be smooth, meaning that similar inputs will produce similar outputs. This assumption is crucial for the surrogate model to make meaningful predictions.
2. **Stationarity:** The statistical properties of the function are assumed to be constant across the input space. This means that the relationship between inputs and outputs doesn't fundamentally change in different regions of the search space.
3. **Correlation Structure:** BO assumes that the outputs of two input points are correlated, and this correlation is typically modeled using a kernel function.

4. **Gaussian Process Prior:** BO typically assumes that the objective function can be modeled as a sample from a Gaussian Process (GP). This means that any finite set of function values has a joint Gaussian distribution:

$$f(\mathbf{X}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \quad (5.15)$$

where $\boldsymbol{\mu}$ is the mean vector and $\mathbf{K}$ is the covariance matrix defined by the kernel function.

These assumptions allow BO to make probabilistic predictions about the objective function's behavior in unobserved regions of the search space, guiding the optimization process efficiently.

Machine Learning and Surrogate Modeling

Gaussian Processes (GP) are typically used as the default surrogate model in BO, primarily for its capabilities in providing a probabilistic model of the objective function landscape. This is particularly useful for uncertainty quantification, a critical component in balancing the exploration-exploitation trade-off. Additionally, a tremendous amount of flexibility is available to scientist/engineers in designing custom kernel functions that more appropriately capture the complexity of the data. These characteristics make GPs a powerful and popular choice for surrogate modeling in Bayesian Optimization, particularly for expensive, black-box optimization problems in low to moderate dimensions. However, alternative machine learning techniques can also serve as surrogate models within the BO framework:

- **Random Forests:** Offer good performance and can handle mixed continuous and categorical inputs.
- **Neural Networks:** Can capture highly complex relationships but may require more data to train effectively.
- **Support Vector Machines:** Effective for moderate-dimensional problems and can be extended to provide probabilistic outputs.

Gaussian Processes

Brief summary of section 2.3 provided here.

Recall that the univariate Gaussian distribution of a random variable x_1 is defined by its probability density function (PDF):

$$\text{prob}(x_1|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_1 - \mu)^2\right) \quad (5.16)$$

Here, $x_1 \in \mathbb{R}$ is a random sample from a set of m measurements $\mathbf{x}_1 \in \mathbb{R}^m$, $\mu \in \mathbb{R}$ is the mean, and $\sigma^2 \in \mathbb{R}$ is the variance. The mean and variance are computed as:

$$\mu = \frac{\mathbf{1}^\top \mathbf{x}_1}{m} \quad (5.17)$$

$$\sigma^2 = \frac{(\mathbf{x}_1 - \mu \mathbf{1})^\top (\mathbf{x}_1 - \mu \mathbf{1})}{m} \quad (5.18)$$

Similarly, for another variable x_2 , the mean and variance are computed analogously.

The joint probability distribution of x_1 and x_2 , given means μ_1 , μ_2 , and covariance matrix Σ , is:

$$\text{prob} \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \middle| \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma \right) = \frac{1}{\sqrt{(2\pi)^2 |\Sigma|}} \exp \left(-\frac{1}{2} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^\top \Sigma^{-1} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix} \right) \quad (5.19)$$

In the context of Gaussian processes, we typically extend this concept to an infinite collection of random variables, indexed by x , where any finite subset has a joint Gaussian distribution characterized by a mean function $\mu(x) = \mathbb{E}[f(x)]$ and a covariance function $k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))]$.

The Gaussian process is fully specified by its mean function and covariance function. For a set of observations $\mathcal{D} = \{(x_i, f(x_i))\}_{i=1}^N$, where $f(x)$ is modeled as a Gaussian process, the joint distribution of function values $\mathbf{f} = [f(x_1), f(x_2), \dots, f(x_N)]^\top$ at inputs $\mathbf{x} = [x_1, x_2, \dots, x_N]^\top$ is:

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, K) \quad (5.20)$$

where $\boldsymbol{\mu} = [\mu(x_1), \mu(x_2), \dots, \mu(x_N)]^\top$ and $K_{ij} = k(x_i, x_j)$.

Given new inputs $\mathbf{x}_*$, the joint distribution of predictions $\mathbf{f}_* = [f(\mathbf{x}_*)]$ and observed data $\mathbf{y}$ is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \boldsymbol{\mu}(\mathbf{x}) \\ \boldsymbol{\mu}_*(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} K(\mathbf{x}, \mathbf{x}) + \sigma_n^2 I & K(\mathbf{x}, \mathbf{x}_*) \\ K(\mathbf{x}_*, \mathbf{x}) & K(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right) \quad (5.21)$$

where $\boldsymbol{\mu}_*(\mathbf{x}) = K_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y}$ and $K_* = K(\mathbf{x}, \mathbf{x}_*)$.

The posterior mean and covariance functions for predictions $\mathbf{f}_*$ given $\mathbf{y}$ are:

$$\mathbb{E}[\mathbf{f}_* | \mathbf{y}] = \boldsymbol{\mu}_*(\mathbf{x}) = K_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \quad (5.22)$$

$$\text{cov}[\mathbf{f}_*(\mathbf{x})] = K_{**} - K_*^\top (K + \sigma_n^2 I)^{-1} K_* \quad (5.23)$$

These equations summarize the predictive distribution of Gaussian processes, making them powerful tools for regression and probabilistic modeling tasks.

Kernel and Model Selection

As stated before, the choice of the kernel function is a critical step in GP modeling as it defines the covariance structure of the process. More specifically, the kernel defines the degree of correlation between two outputs given two inputs. In this work, I explore three kernel functions: the Squared Exponential Kernel (SEK Eqn. 5.24), known for its smoothness and infinite differentiability; the Rational Quadratic Kernel (RQK Eqn. 5.25), which can be interpreted as a scale mixture of Squared Exponential kernels with different length scales; and the Local-Periodic Kernel (LPK Eqn. 5.26), which captures locally periodic functions. These kernels offer different characteristics in terms of smoothness, stationarity, and the ability to model various types of relationships between input variables.

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp \left(- \frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2} \right) \quad (5.24)$$

$$k_{RQ}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \left(1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\alpha l^2} \right)^{-\alpha} \quad (5.25)$$

$$k_{LOCPER}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp \left(- \frac{(\mathbf{x} - \mathbf{x}')^2}{2l^2} \right) \exp \left(- \frac{2 \sin^2(\pi|\mathbf{x} - \mathbf{x}'|/p)}{l^2} \right) \quad (5.26)$$

In addition to kernel selection, model selection (hyperparameter selection) is required each time dataset updates prior to calling the GP for inference. This can be accomplished by maximizing the negative log likelihood of the data in Eqn. 5.27 using evolutionary methods, gradient descent, or Monte-Carlo Markov-Chain sampling [24]. This approach allows BO to automatically tune the kernel parameters to best fit the observed data, providing a principled way to compare different models and select the one that best captures the underlying structure of the optimization problem. An example convergence plot for a single BO iteration for the hyperparameters required for the squared exponential kernel is shown in Fig. 5.3, as well as the variation of the hyperparameter during the meta-optimization shown in Fig. 5.4.

$$\max_{\boldsymbol{\theta}} \log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \left(-\frac{1}{2} \mathbf{y}^\top \mathbf{K}_{\boldsymbol{\theta}}^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}_{\boldsymbol{\theta}}| - \frac{n}{2} \log 2\pi \right) \quad (5.27)$$

Results

The optimization of Eqn. 5.3 was performed over 10 iterations for SEK, RPK and LPK functions. At each step, the BO evaluates 16 samples in parallel on a 16-thread Intel Core Processor i9 - 9900 (8 Cores/16 MB/16T/up to 5.0GHz/65W) for a total budget of 160 function evaluations. At each optimization step, maximization of the negative log-likelihood was performed using a genetic algorithm with the following parameters:

- Population: $pop = 24$

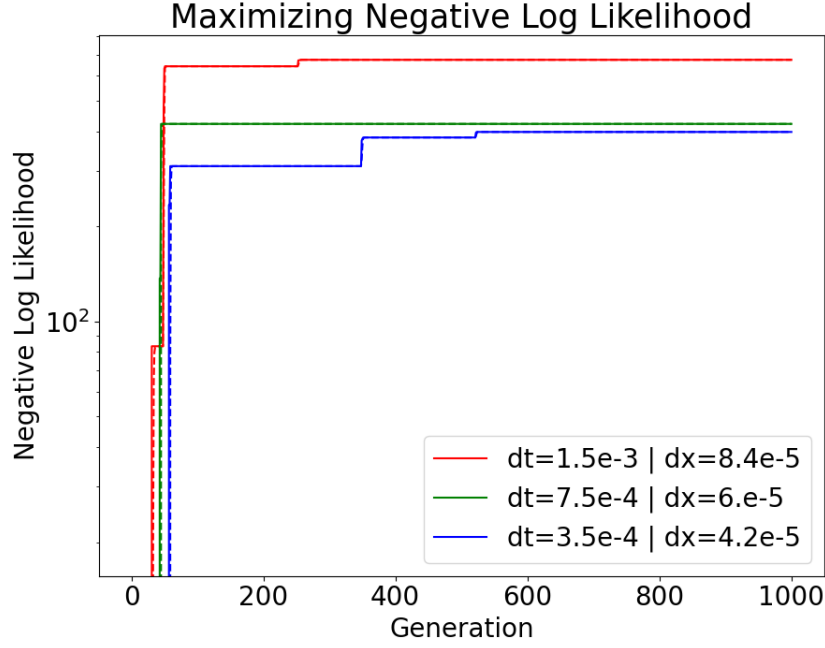


Figure 5.3: Maximization of the negative log likelihood over 1000 generations.

- Parents: $P = 4$
- Children: $C = 4$
- Generations: $G = 100$,

with the following constraints for all kernel functions used:

$$\begin{aligned}
 l &\in [1 \times 10^{-2}, 1 \times 10^1] \\
 \sigma &\in [1 \times 10^{-2}, 1.0] \\
 \sigma_n &\in [1 \times 10^{-7}, 1 \times 10^{-3}] \\
 p &\in [1 \times 10^{-2}, 10.0] \\
 a &\in [1 \times 10^{-2}, 10.0].
 \end{aligned}$$

The optimization for each kernel was also run across the following mesh sizes:

$$\begin{aligned}
 nodes &= [11, 15, 21, 25] \\
 h &= [8.4 \times 10^{-5}, 6 \times 10^{-5}, 4.2 \times 10^{-5}, 3.5 \times 10^{-5}] \\
 dt &= [1.45 \times 10^{-3}, 7.5 \times 10^{-4}, 3.5 \times 10^{-4}, 2.5 \times 10^{-4}]
 \end{aligned}$$

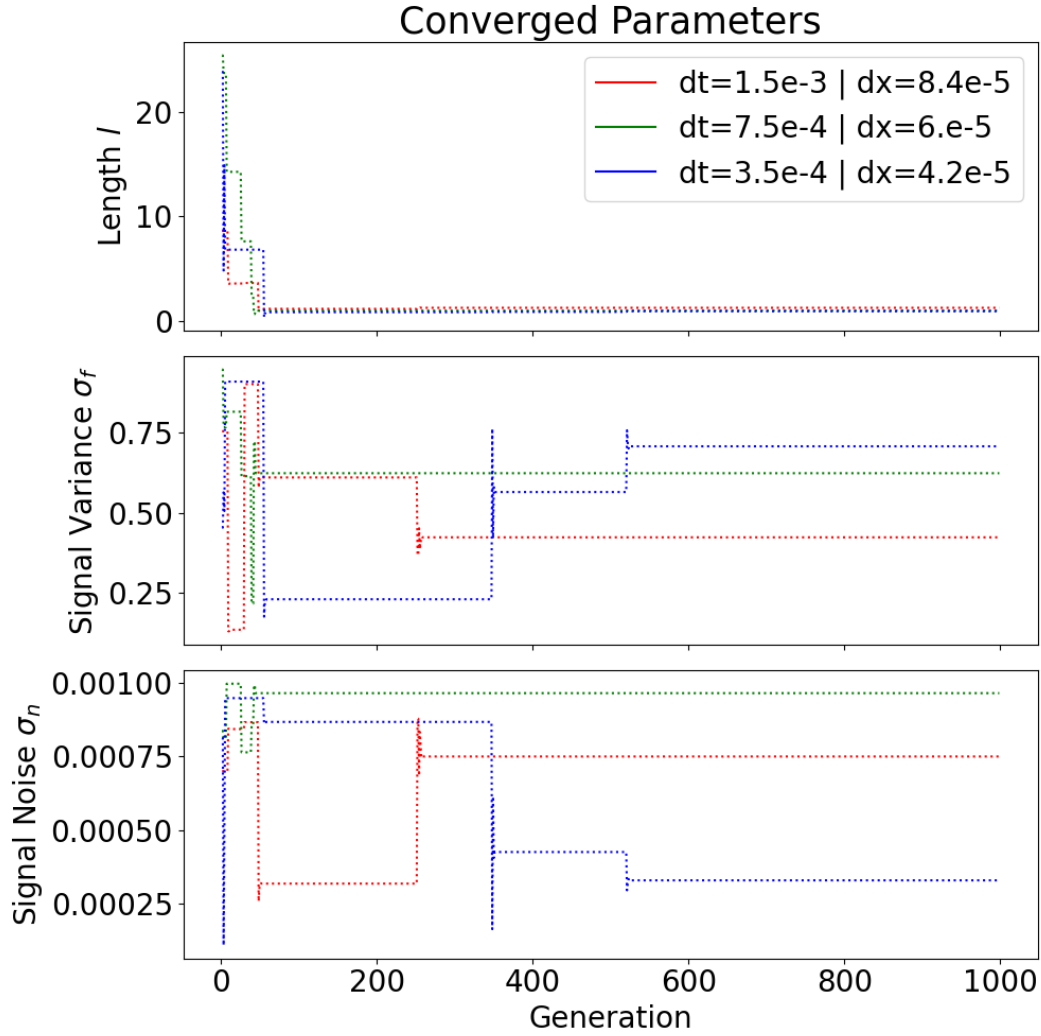


Figure 5.4: Convergence of squared exponential kernel hyperparameters over 1000 generations.

for the purpose of ensure identifying how the performance changes over a refining mesh.

Hyperparameter Evolution

From the initial results in Fig. 5.3, Fig. 5.4 indicate the total number of generations. In the BO implementation used for these results, the maximization of the negative log-likelihood was run for a static 100 generations using the same GA parameters in the initial results. The parameter evolution over all 10 BO steps was tracked for variable mesh sizes, and evolution of the hyperparameters are show in Fig. 5.5, Fig. 5.6, & Fig. 5.7 for SEK, RQK and LPK respectively.

After each iteration, BO chooses the best sample points for successive objective function calls. As more data are collected after each optimization iteration, the hyperparameter selection begins to stabilize for each mesh as the Gaussian Process *learns* the parameter space and typically converges around a set value. Occasionally, new data points cause a jump to a new hyperparameter that better fits the data, however some general trends can still be identified. In particular, the mesh size appears affect the the values for each parameter. For example, the parameter σ_n for SEK and for LPK is lower for each refined mesh, indicating that the higher accuracy the solver, the lower the noise parameter, and consequently the tighter the certainty bands around sampled points within the GP. For other hyperparameters, mesh refinement appears to affect in structured way. This may indicate that the stochastic nature of the particle size and location has a greater effect on the GP model than the noise introduced from numerical accuracy.

Material and Process Parameter Convergence

The convergence patterns of individual optimization parameters provide insight into their relative impact on the overall objective. Parameters that consistently converge to a central value across all models (or mesh sizes) are typically of higher importance compared to those that exhibit stochastic behavior. In optimization problems where the explicit form of the dynamics or model is available along with its gradients, this approach would be analogous to computing the gradient and identifying the direction of steepest change.

An examination of the material parameter evolution for SEK, RQK, and LPK models over BO iterations (Fig. 5.8) reveals a convergence of particle radius to a narrow range between approximately $6 - 8 \times 10^{-5}$ m. Similarly, the optimal particle volume fractions consistently fall within the 50-65 vol.% range. These trends persist across various mesh sizes for each model, as evidenced in Fig. 7.7, Fig. 7.12, and Fig. 7.17 in the Appendix.

For process parameters (Fig. 5.9), ambient temperature also demonstrates convergence to a relatively narrow range ($\sim 314 - 320$ K). UV intensity shows a broader, yet still defined, range of optimal values ($\sim 20 - 80 \frac{W}{m^2}$). Interestingly, UV exposure time exhibits random sampling across its entire domain, bounded only by the imposed lower and upper constraints. This behavior is consistent across different mesh sizes (see Fig. 7.6, Fig. 7.11, and Fig. 7.16 in the Appendix).

These findings challenge an assumption made in Chapter 2, where UV exposure time was considered a critical parameter due to its impact on the total processing time of an extruded filament in additive manufacturing. The optimization results highlight an aspect of the physics that was not fully accounted for in the original development phase: the reaction $PI \rightarrow PI\cdot$ is primed to initiate immediately upon UV exposure. Once this reaction begins, subsequent reactions become largely independent of the UV light source, save for a slight additional increase in temperature. This insight underscores the importance of considering all aspects of the underlying physics when developing and optimizing manufacturing processes.

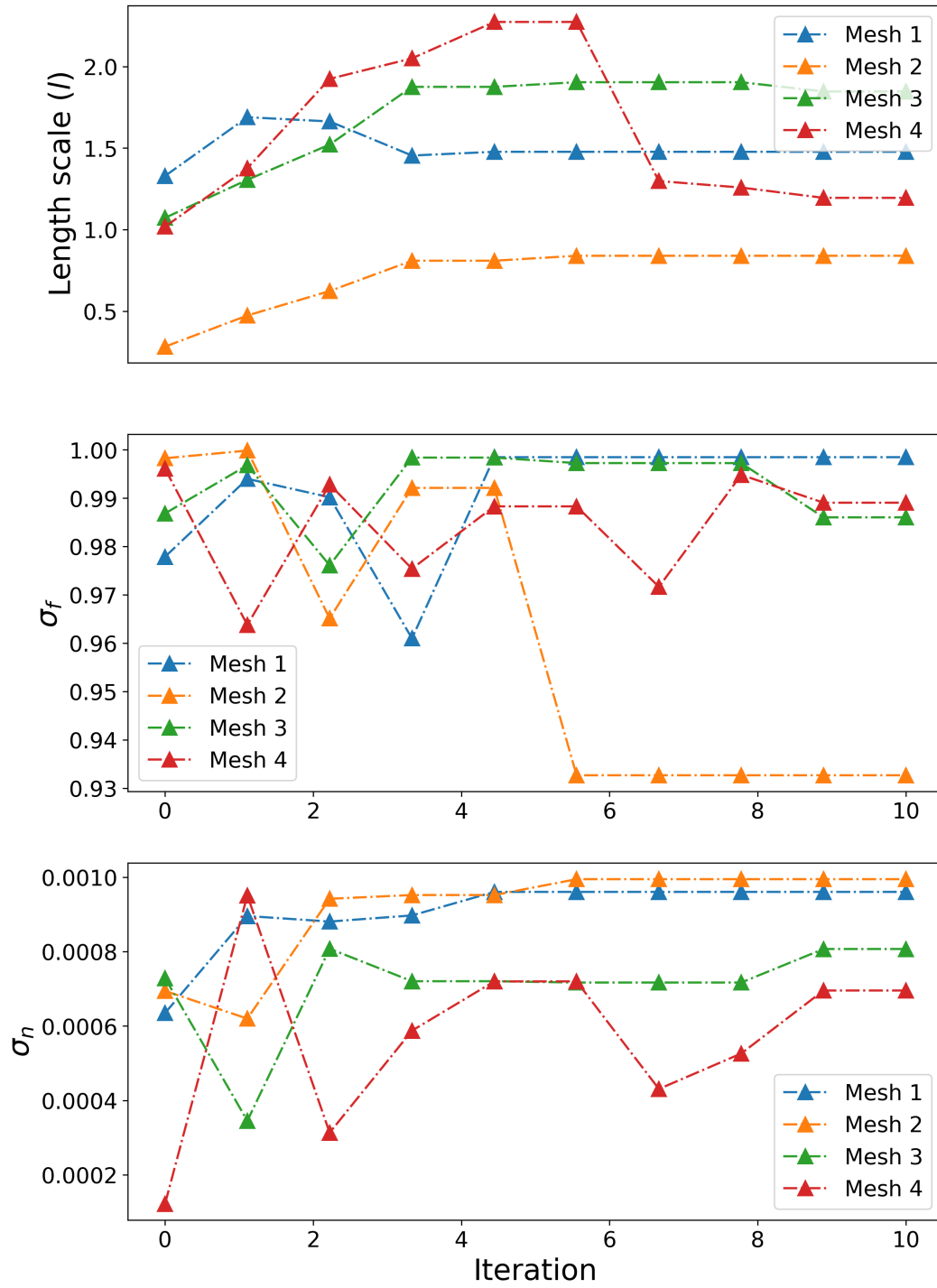


Figure 5.5: Hyper-parameter Convergence of the SEK.

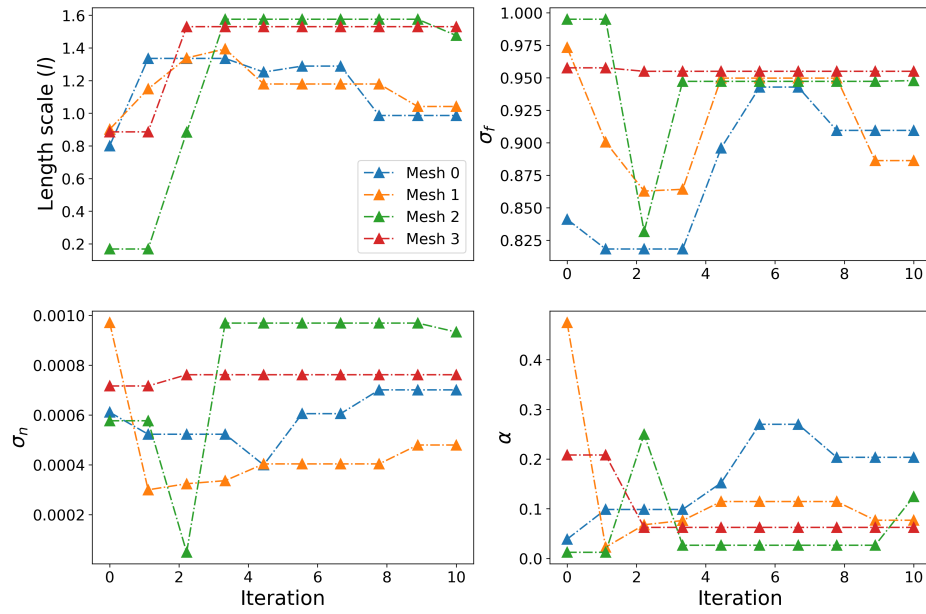


Figure 5.6: Hyper-parameter Convergence of the RQK.

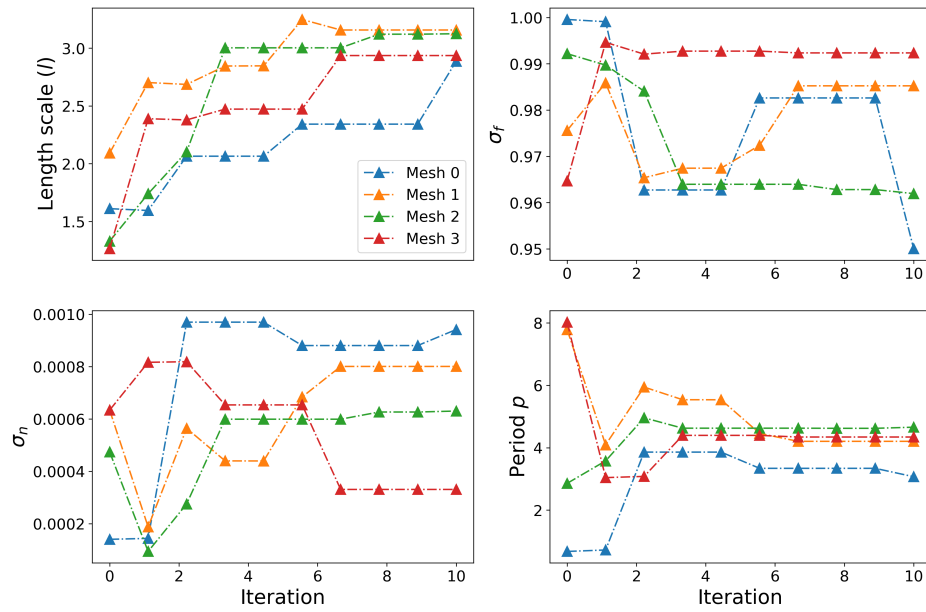


Figure 5.7: Hyper-parameter Convergence of the LPK.

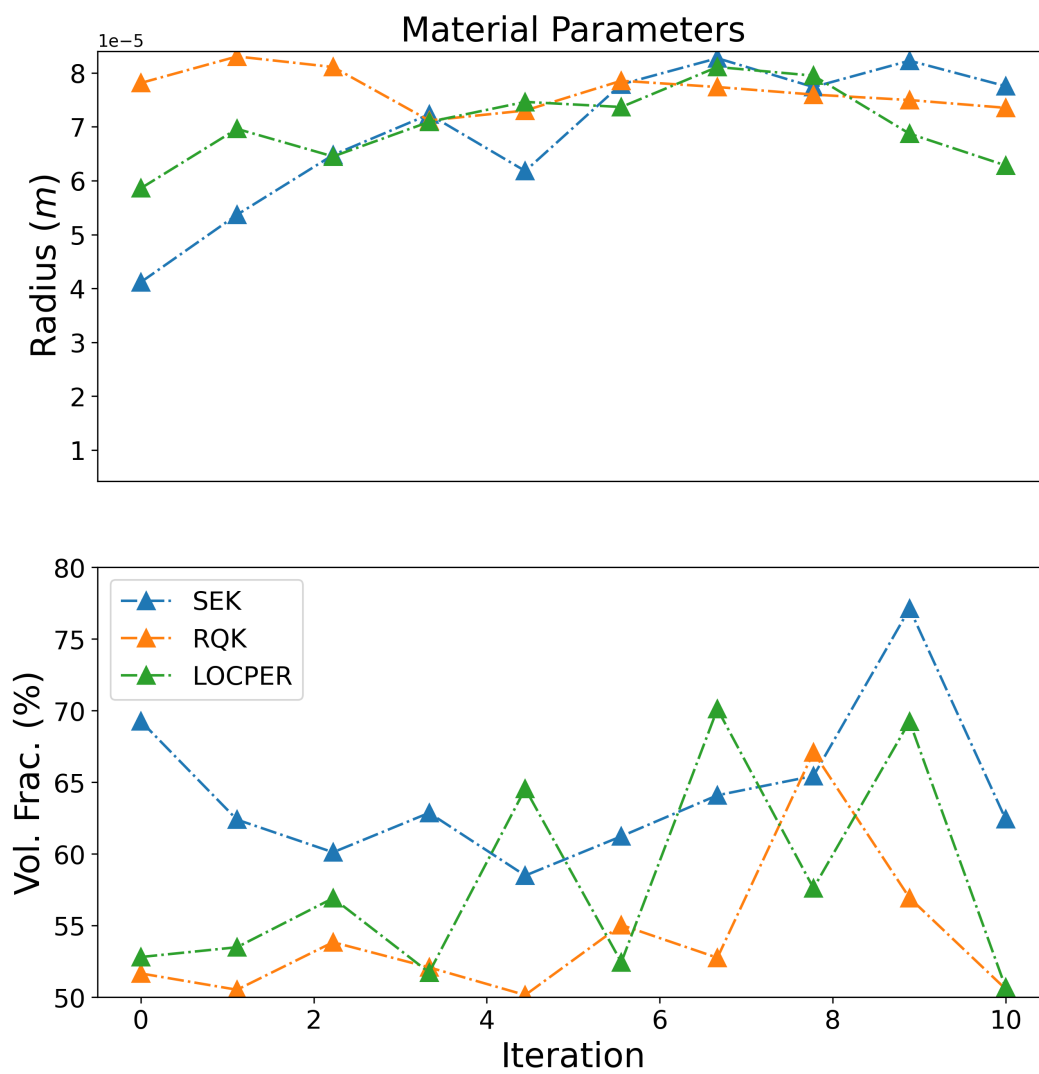


Figure 5.8: Material parameter convergence of for SEK, RQK, LPK functions.

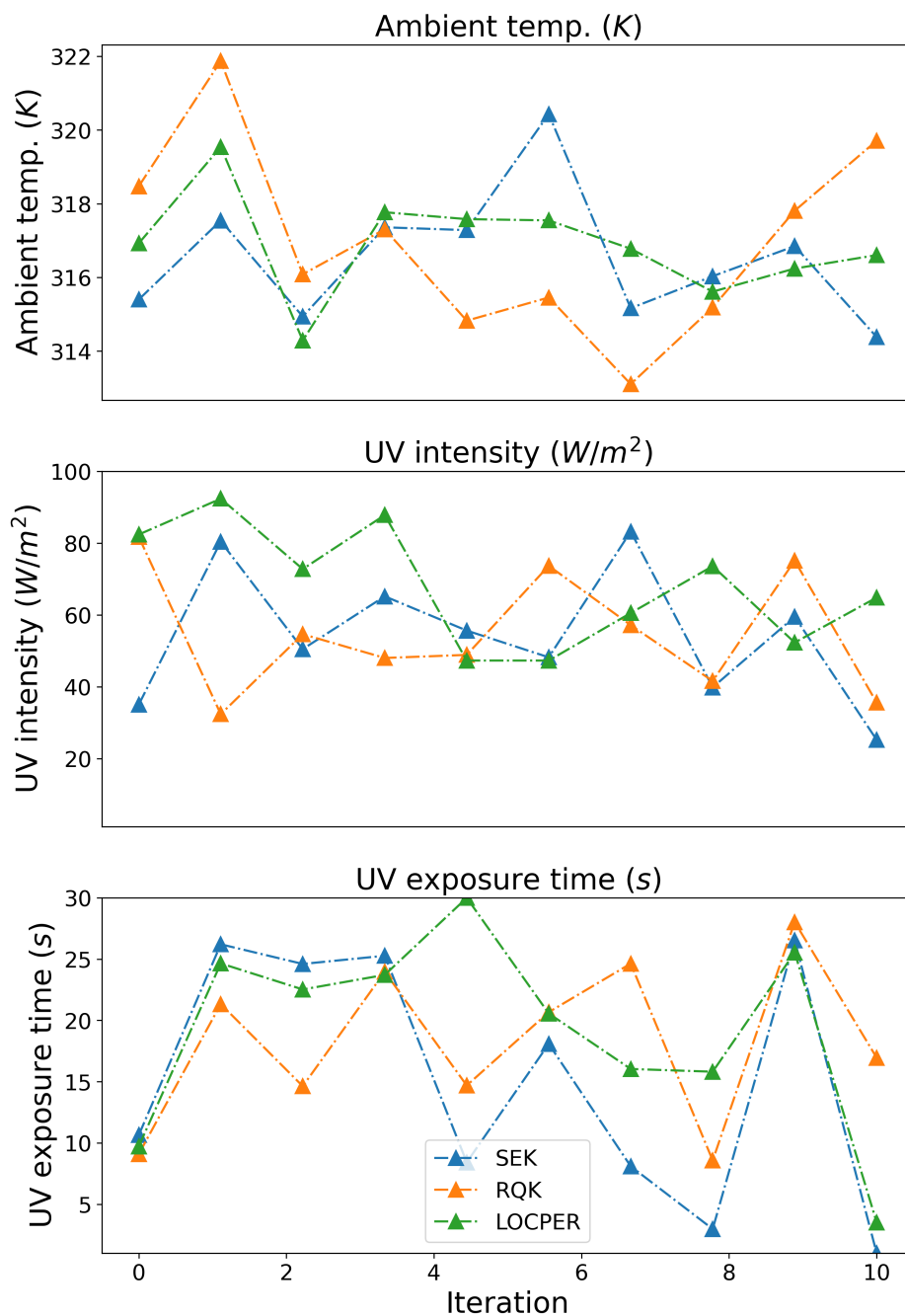


Figure 5.9: Process parameter convergence of for SEK, RQK, LPK functions.

Results for modified objective

Fig. 5.10 presents the overall performance of the Squared Exponential Kernel (SEK), Rational Quadratic Kernel (RQK), and Local Periodic Kernel (LPK) functions, showcasing the top performer, the average of the top 4 performers, and the average performer for each. The convergence of top performers indicates competitive final objective values across all models. While the rate of convergence appears to differ between these functions, it is crucial to note that iteration 0 employs random search, making the average top 4 objectives a more valuable indicator of BO’s convergence rate to an optimal value.

Analysis of the average top 4 objectives reveals that RQK achieves the fastest convergence, obtaining an objective value within approximately 1.6% of the final optimal objective in a single step. This rapid convergence highlights the efficiency of the RQK model in this optimization context. Furthermore, the overall average performer within the BO sampling approach demonstrates a smooth convergence to a range similar to the optimal value, indicating the overall robustness of the BO method.

For deeper insights, we examine the decomposition of the multi-objective value from Eqn. 5.3 into its individual objectives. Fig. 5.11 illustrates the convergence plots for each term in Eqn. 5.3, revealing a less smooth progression compared to their linear combination. The first term in Eqn. 5.3, representing the minimization of the starting concentrations of PI , shows expected behavior. Given that PI molecules are specifically tuned to dissociate when exposed to a specific wavelength of light, and based on the results from the Pareto Frontier analysis, it is understandably challenging to obtain input parameters that don’t drive $[PI]$ to zero.

For the remaining three convergence plots, it is evident that the BO algorithm faces a more complex challenge in balancing these objectives. This complexity arises from the interplay between different reaction components and process parameters, highlighting the intricate nature of the optimization problem at hand. The varying convergence patterns across these objectives underscore the importance of a multi-objective approach in capturing the full complexity of the UV-curable ink system.

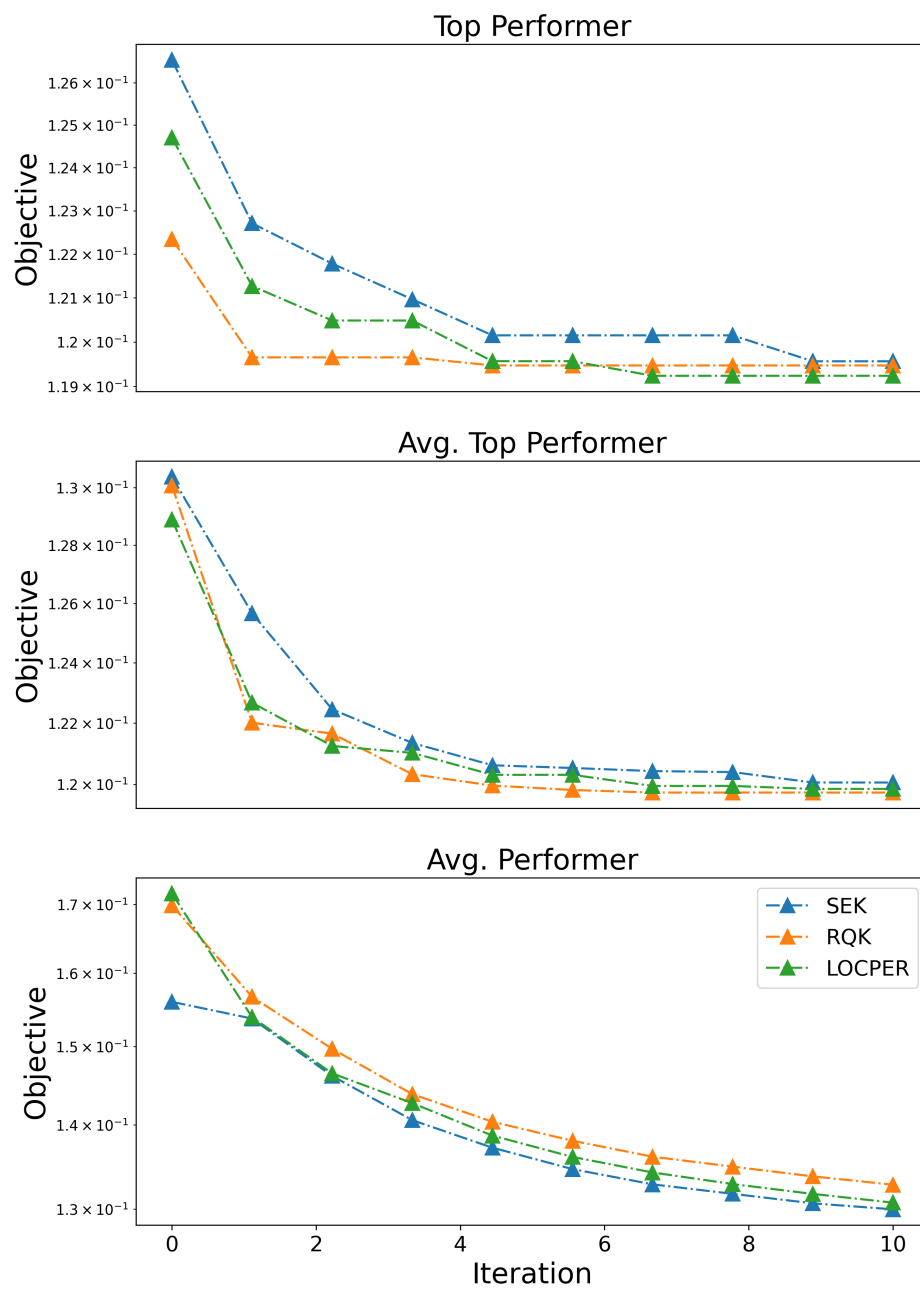


Figure 5.10: Overall convergence of BO for SEK, RQK and LPK.

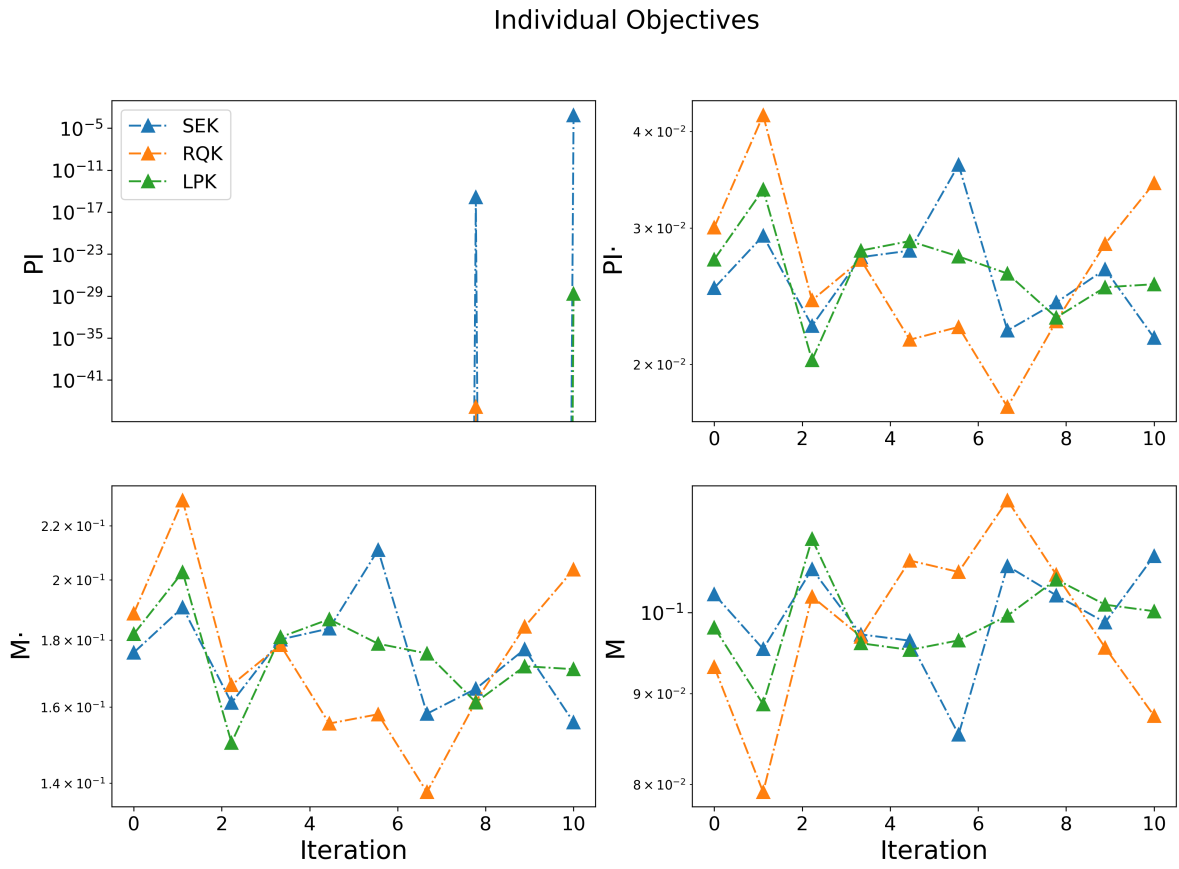


Figure 5.11: Convergence of individual objectives.

5.7 Covariance Matrix Adaptation - Evolutionary Strategy

Like BO, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a stochastic, derivative-free method for numerical optimization of nonlinear or non-convex continuous optimization problems. Developed by Hansen and Ostermeier [81], CMA-ES is particularly effective for solving difficult optimization problems in continuous domains, but does not require the same computational effort as BO.

The optimization was run across the following mesh sizes:

$$\begin{aligned} \text{nodes} &= [11, 15, 21, 25] \\ h &= [8.4 \times 10^{-5}, 6 \times 10^{-5}, 4.2 \times 10^{-5}, 3.5 \times 10^{-5}] \\ dt &= [1.45 \times 10^{-3}, 7.5 \times 10^{-4}, 3.5 \times 10^{-4}, 2.5 \times 10^{-4}]. \end{aligned}$$

Algorithm Overview

CMA-ES works by iteratively updating a multivariate normal distribution that models the distribution of good solutions in the search space. The key idea is to adapt the covariance matrix of this distribution to improve the sampling of promising solutions.

At each generation g , the algorithm:

1. Samples a population of λ new candidate solutions.
2. Evaluates these solutions.
3. Updates the distribution parameters based on the best solutions.

Mathematical Formulation

Let $\mathbf{x} \in \mathbb{R}^n$ be an n -dimensional vector of optimization parameters. The core of CMA-ES is the sampling of new candidate solutions according to a multivariate normal distribution:

$$\mathbf{x}_k^{(g+1)} \sim \mathcal{N}(\mathbf{m}^{(g)}, (\sigma^{(g)})^2 \mathbf{C}^{(g)}) \quad (5.28)$$

where:

- $\mathbf{x}_k^{(g+1)}$ is the k -th candidate solution in generation $g + 1$
- $\mathbf{m}^{(g)}$ is the mean of the distribution at generation g
- $\sigma^{(g)}$ is the overall step size at generation g
- $\mathbf{C}^{(g)}$ is the covariance matrix at generation g

The key components of the algorithm are updated as follows:

1. Mean Update:

$$\mathbf{m}^{(g+1)} = \mathbf{m}^{(g)} + c_m \sum_{i=1}^{\mu} w_i (\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}) \quad (5.29)$$

where $c_m \approx 1$ is the learning rate, w_i are the recombination weights, and $\mathbf{x}_{i:\lambda}^{(g+1)}$ are the μ best solutions.

2. Step Size Update:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{\mathbb{E}\|\mathcal{N}(0, \mathbf{I})\|} - 1 \right) \right) \quad (5.30)$$

where $c_\sigma < 1$ is the learning rate, $d_\sigma \approx 1$ is a damping parameter, and $\mathbf{p}_\sigma^{(g+1)}$ is the evolution path.

3. Covariance Matrix Update:

$$\begin{aligned} \mathbf{C}^{(g+1)} = & (1 - c_1 - c_\mu) \mathbf{C}^{(g)} + c_1 \mathbf{p}_c^{(g+1)} (\mathbf{p}_c^{(g+1)})^T \\ & + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_i^{(g+1)} (\mathbf{y}_i^{(g+1)})^T \end{aligned} \quad (5.31)$$

where c_1 and c_μ are learning rates, $\mathbf{p}_c^{(g+1)}$ is the evolution path for the covariance matrix, and $\mathbf{y}_i^{(g+1)}$ are the selected steps.

The evolution paths $\mathbf{p}_\sigma^{(g+1)}$ and $\mathbf{p}_c^{(g+1)}$ accumulate information about the correlation between consecutive steps.

CMA-ES is self-adaptive, meaning it can adjust its own parameters during the optimization process. This makes it particularly robust for a wide range of optimization problems, including those with complex fitness landscapes or noise.

Results

The optimization of [Eqn. 5.3](#) was performed using CMA-ES over 10 iterations, mirroring the approach taken with BO. Each iteration involved the evaluation of 16 samples in parallel, leveraging a 16-thread Intel Core Processor i9 - 9900 (8 Cores/16 MB/16T/up to 5.0GHz/65W). This setup resulted in a total budget of 160 function evaluations, providing a basis for comparison with the BO implementation.

Material and Process Parameter Convergence

While the ranges for material and process parameters obtained through CMA-ES align closely with those from BO, the convergence paths differ substantially. The material parameters, when examined across variable mesh sizes, exhibit a tendency to become trapped in local minima, as evidenced in Fig. 5.12. This behavior contrasts with the more exploratory nature observed in the BO results.

Fig. 5.13 offers deeper insights into the search strategy employed by CMA-ES for process parameters. Unlike the broad initial exploration seen in BO, CMA-ES begins by sampling within a narrow range for each parameter. As iterations progress, the algorithm gradually expands its search, branching out to explore a wider parameter space. This approach appears to invert the exploration-exploitation balance observed in BO, with CMA-ES favoring exploitation in early iterations and conservatively increasing exploration over time. At the final iteration, the exploration converges on similar ranges provided by BO.

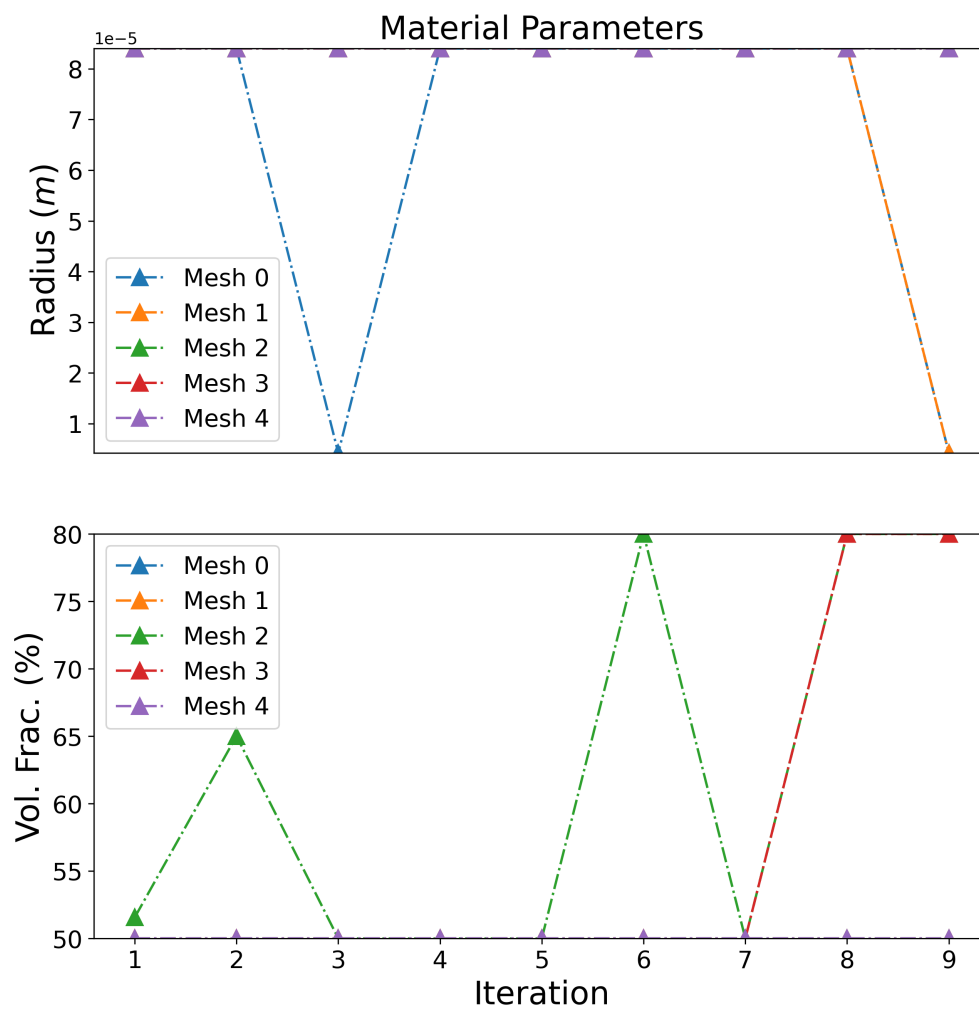


Figure 5.12: Material parameter convergence for CMA-ES.

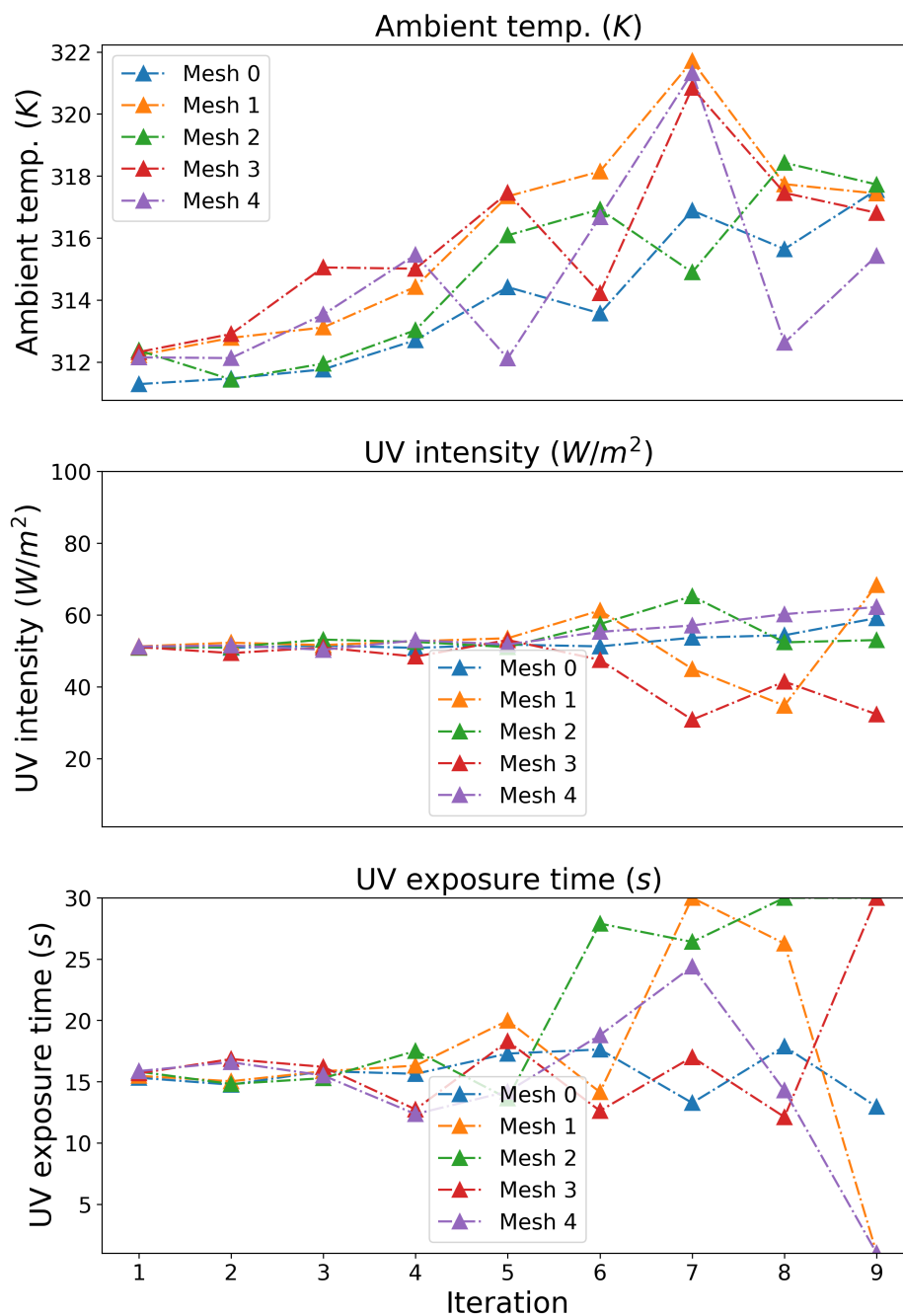


Figure 5.13: Process parameter convergence for CMA-ES.

Results for modified objective

Fig. 5.14 illustrates the overall performance of the CMA-ES algorithm across variable mesh sizes. The figure presents three key metrics: the top performer, the average of the top 4 performers, and the average performer across all models. These convergence patterns reveal nuanced insights into CMA-ES's parameter selection strategy. For the top performer, CMA-ES achieves objective values comparable to those obtained through BO. However, the algorithm appears to encounter difficulties with coarser meshes, suggesting a sensitivity to discretization that warrants further investigation.

More intriguing are the trends observed in the average top 4 performers and overall average performers. Rather than strict convergence, these plots reveal a divergent behavior as iterations progress. This pattern indicates that while the algorithm technically converges, it increasingly weighs exploration over exploitation in later iterations. Consequently, the overall variance of the population begins to increase, potentially leading to computational inefficiencies.

Further insights can be gleaned by decomposing the multi-objective function (Eqn. 5.3) into its constituent objectives. Fig. 5.15 depicts this breakdown, revealing a more controlled search across the different objectives. The behavior of CMA-ES in this context bears similarities to gradient descent and greedy algorithms, suggesting a more directed exploration of the objective space compared to the broader sampling approach of BO.

This analysis underscores the distinct characteristics of CMA-ES in navigating complex, multi-objective optimization landscapes. While effective in identifying optimal solutions, the algorithm's tendency towards increased exploration in later stages presents both opportunities and challenges for efficient computational resource utilization in materials optimization problems.

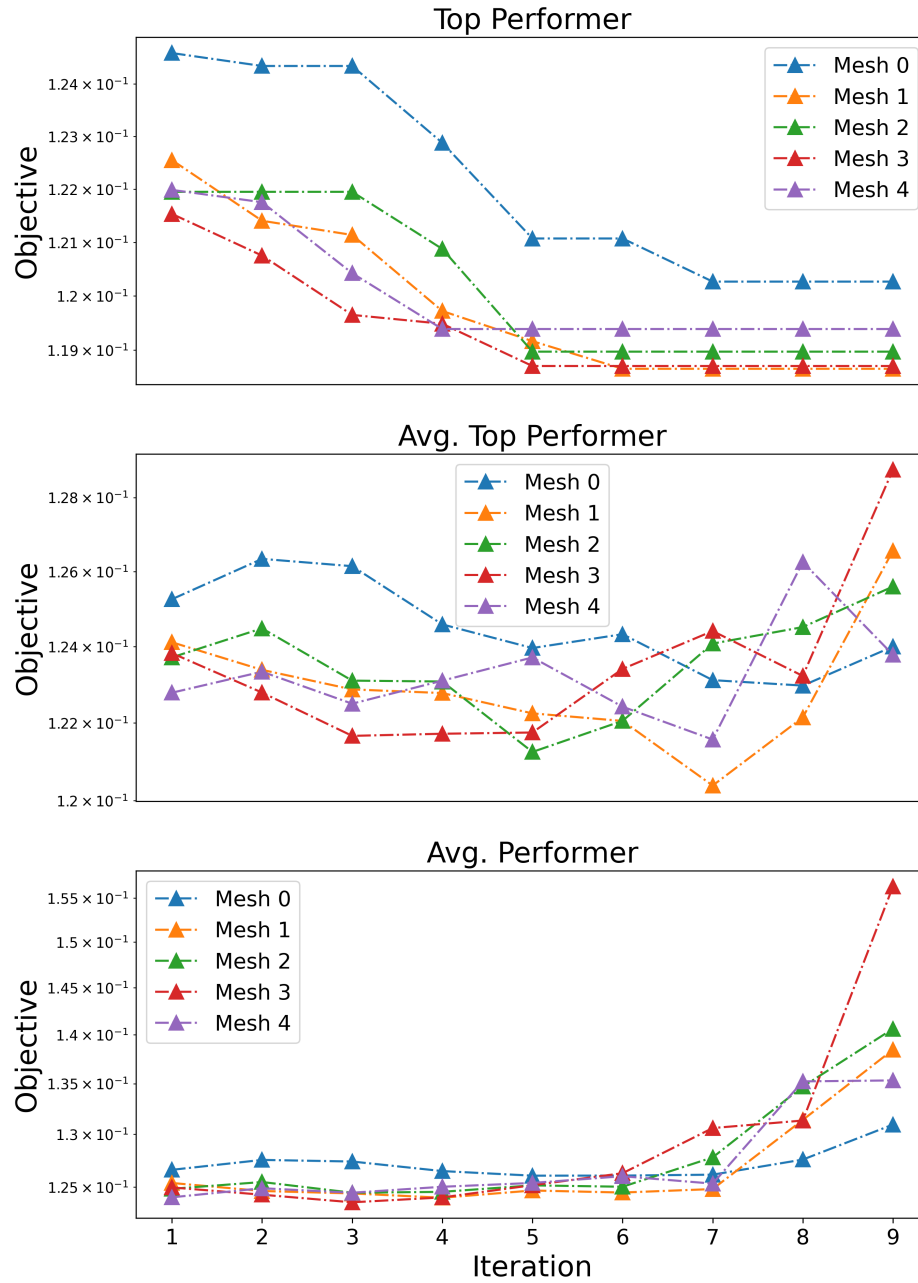


Figure 5.14: Optimization convergence plots for CMA-ES: The candidate solution performance over each optimization iteration for the best parent, average parents, and overall average entire population.

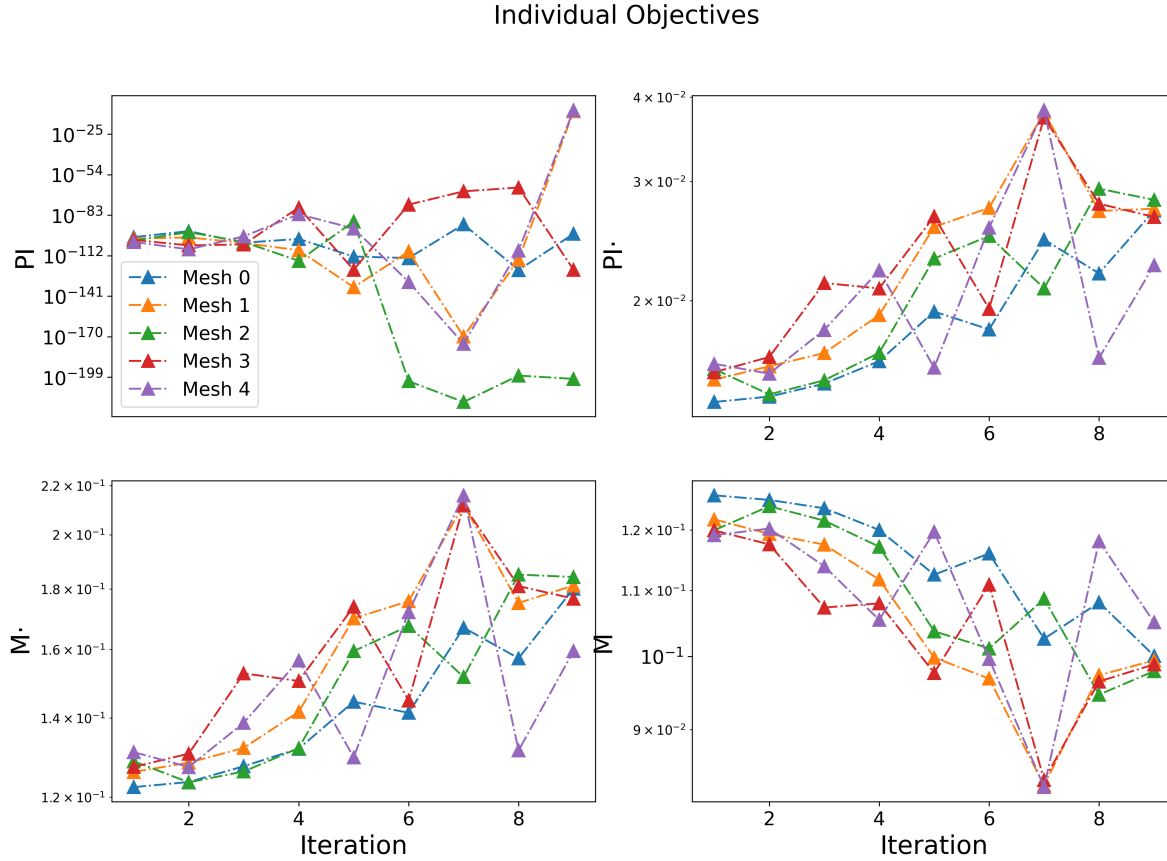


Figure 5.15: Optimization convergence plots for CMA-ES: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI·), propagating radical (M·) and starting monomer (M) concentrations.

5.8 Genetic Algorithms

Brief summary of section 2.4 provided here.

Genetic Algorithms (GAs) are a subset of Machine Learning Algorithms (MLAs) that are particularly well-suited for solving nonconvex, nonsmooth, multicomponent, and multistage optimization problems. These algorithms draw inspiration from the principles of natural selection and evolution, leveraging random search for exploration and linearly combining top-performing parameters for exploitation. The simplicity of this approach makes GAs a robust and generalizable tool for complex system optimization where the stronger assumptions required for BO or CMA-ES may be too stringent for complex coupled-physics application. Like BO and CMA-ES, the GA is a stochastic, derivative-free method for numerical optimization of nonlinear or non-convex continuous optimization problems. Developed

by John Holland [82], the GA is particularly effective for solving diverse optimization problems with a relatively straight-forward and simple approach to implement in code. At its core, the algorithm is an exploration-prioritized approach that leverages random search in an intelligent way.

As with BO and CMA-ES, the optimization was run across the following mesh sizes:

$$\begin{aligned} nodes &= [11, 15, 21, 25] \\ h &= [8.4 \times 10^{-5}, 6 \times 10^{-5}, 4.2 \times 10^{-5}, 3.5 \times 10^{-5}] \\ dt &= [1.45 \times 10^{-3}, 7.5 \times 10^{-4}, 3.5 \times 10^{-4}, 2.5 \times 10^{-4}] . \end{aligned}$$

Algorithm Structure

Following the work of Zohdi [25, 26, 27, 28, 29, 30], the basic structure of a genetic algorithm can be described as follows:

Algorithm 5 Genetic Algorithm

- 1: Generate an initial population of S genetic strings
 - 2: **while** termination condition not met **do**
 - 3: Compute fitness of each string
 - 4: Rank genetic strings
 - 5: Mate nearest pairs to produce offspring
 - 6: Eliminate bottom M strings
 - 7: Keep top K parents and their K offspring
 - 8: Introduce M new random strings
 - 9: **end while**
 - 10: **return** best solution found
-

Genetic String Representation

Each candidate solution in the population is represented as a genetic string λ_i , defined as:

$$\lambda_i \stackrel{\text{def}}{=} [\lambda_1^i, \lambda_2^i, \lambda_3^i, \dots, \lambda_N^i] \quad (5.32)$$

where each λ_j^i represents a parameter in the solution, constrained within specific ranges: $\lambda_j^{(-)} \leq \lambda_j^i \leq \lambda_j^{(+)}$.

Mating and Offspring Generation

The algorithm generates offspring through a mating process between pairs of parent strings. For a pair of parent strings λ_i and λ_{i+1} , two offspring are produced:

$$\boldsymbol{\lambda}_i^{\text{new}} = \boldsymbol{\phi} \odot \boldsymbol{\lambda}_i + (1 - \boldsymbol{\phi}) \odot \boldsymbol{\lambda}_{i+1} \quad (5.33)$$

$$\boldsymbol{\lambda}_{i+1}^{\text{new}} = \boldsymbol{\psi} \odot \boldsymbol{\lambda}_i + (1 - \boldsymbol{\psi}) \odot \boldsymbol{\lambda}_{i+1} \quad (5.34)$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are vectors of random numbers between 0 and 1, and $\odot$ denotes the Hadamard (element-wise) product.

Results

The optimization of Eqn. 5.3 was performed using the GA over 10 iterations, mirroring the approach taken with BO and CMA-ES. Each iteration involved the evaluation of 16 samples in parallel, leveraging a 16-thread Intel Core Processor i9 - 9900 (8 Cores/16 MB/16T/up to 5.0GHz/65W). This setup resulted in a total budget of 160 function evaluations, providing a basis for comparison with the BO and CMA-ES implementations.

Material and Process Parameter Convergence

The ranges for the material and process parameters obtained through GA are nearly identical to those obtained through BO, confirming the conclusions made in the prior sections. The convergence paths also appear to follow a similar pattern as BO, differing from CMA-ES as the exploration component of the GA continuously enables the optimizer to avoid local minima traps.

As with BO, the mesh refinement appears to have little effect on the convergence pattern for r_p and v_p (Fig. 5.16) and θ_{amb} , I_{uv} , and t_{uv} (Fig. 5.17). This indicates that a range of values is appropriate given the stochastic nature of particle size and location for each simulation.

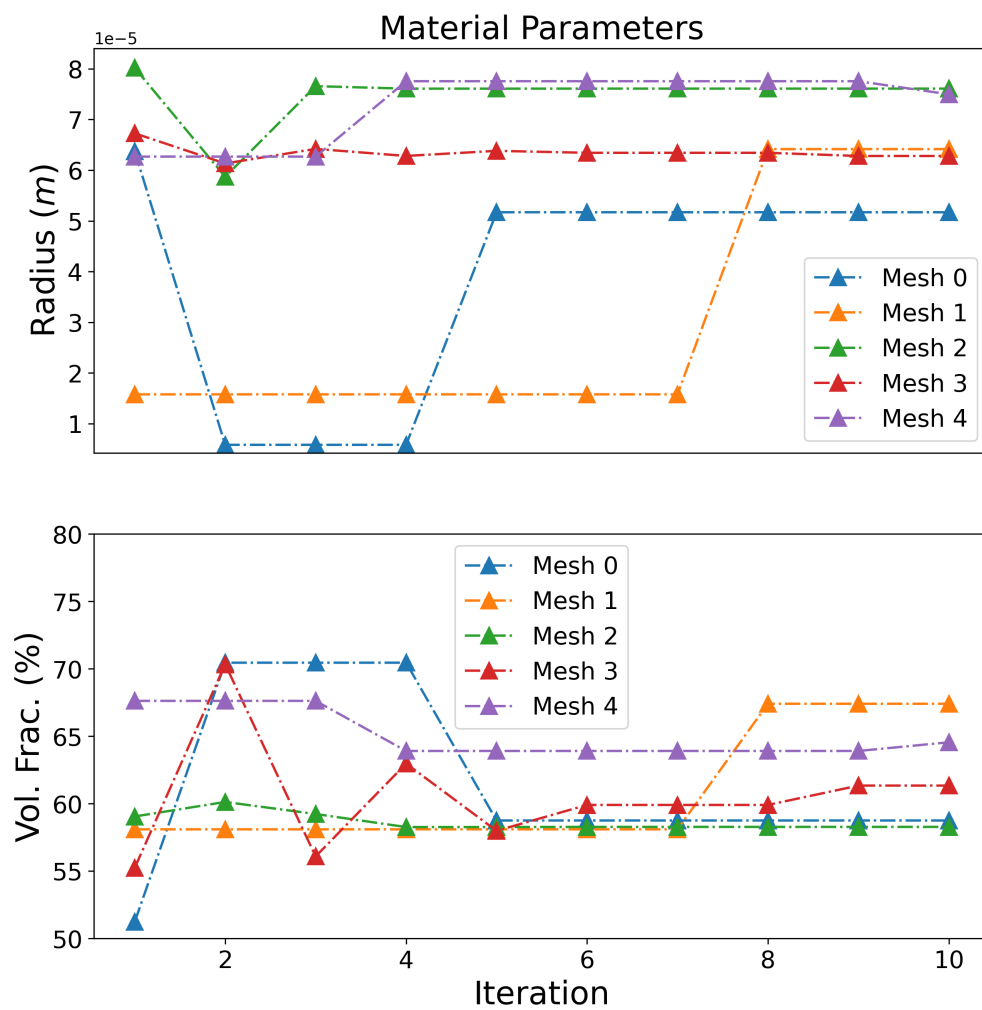


Figure 5.16: Material parameter convergence for GA.

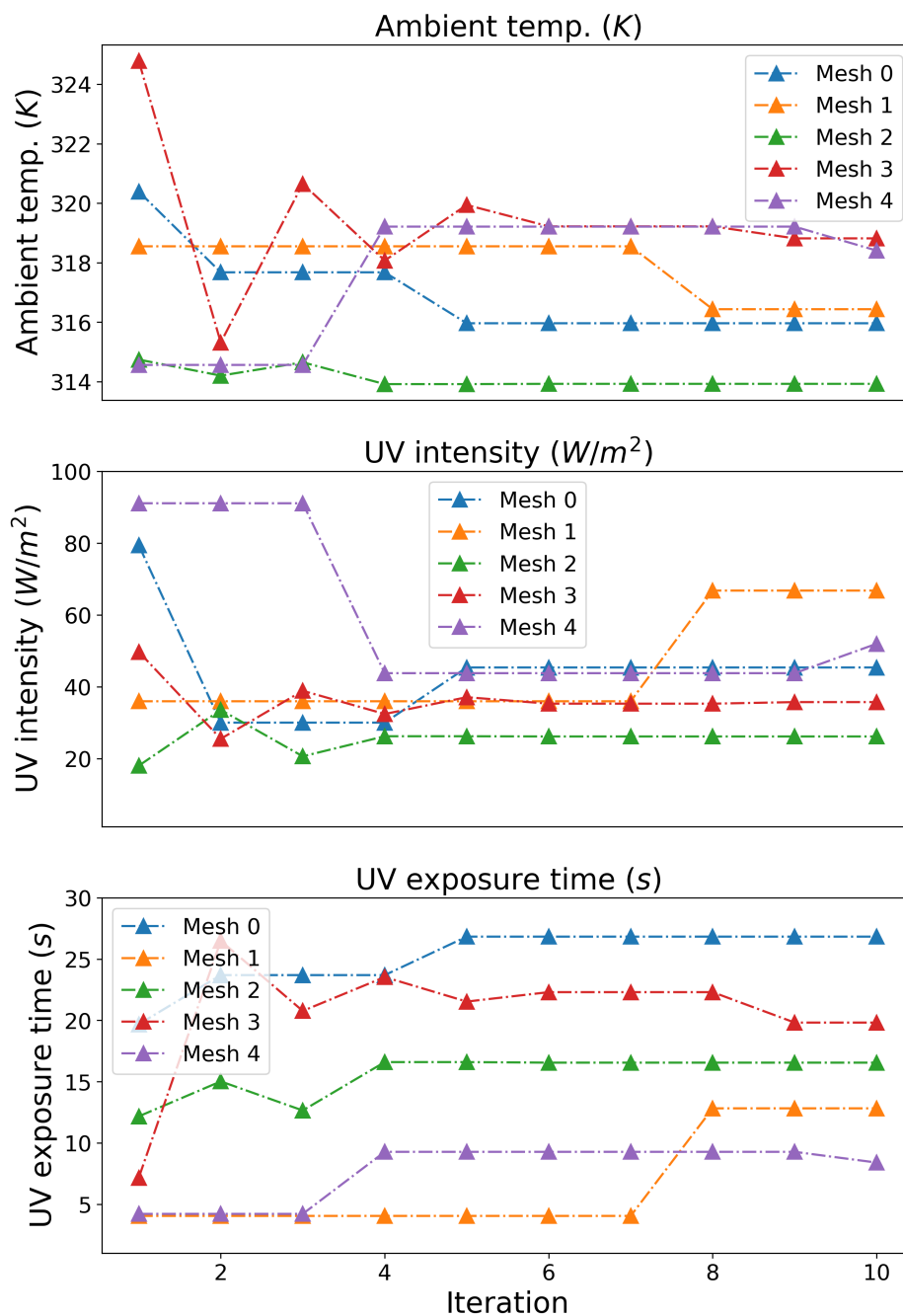


Figure 5.17: Process parameter convergence:

Results for Modified Objective

Fig. 5.18 presents the overall performance of the GA across variable mesh sizes, showing the top performer, the average of the top 4 performers, and the average performer for each iteration. The convergence patterns reveal interesting insights into GA's search strategy. The top performer in GA shows a gradual increase in performance over each iteration (strongly influenced by the performance of the random search in the initial iteration), achieving final objective values comparable to those of BO and CMA-ES.

The average of the top 4 performers and the overall average demonstrate a steady improvement throughout the optimization process, indicating GA's ability to maintain a diverse population while consistently improving solution quality. This behavior contrasts with the divergent trends observed in CMA-ES, suggesting that GA may be more robust in maintaining a balance between exploration and exploitation throughout the optimization process.

Fig. 5.19 shows the decomposition of the multi-objective function (Eqn. 5.3) into its individual objectives. The GA exhibits a more balanced approach in optimizing each objective compared to CMA-ES, with smoother convergence curves for each component. This suggests that GA's population-based approach and genetic operators allow for a more comprehensive exploration of the objective space, potentially leading to more robust solutions in complex, multi-objective optimization problems.

The GA's performance in this context highlights its versatility in handling complex optimization landscapes without requiring strong assumptions about the underlying function. While it may not converge as rapidly as BO in the initial stages, its ability to maintain diversity and consistently improve solutions makes it a robust choice for materials optimization problems, especially when the relationship between input parameters and objectives is not well understood or highly nonlinear.

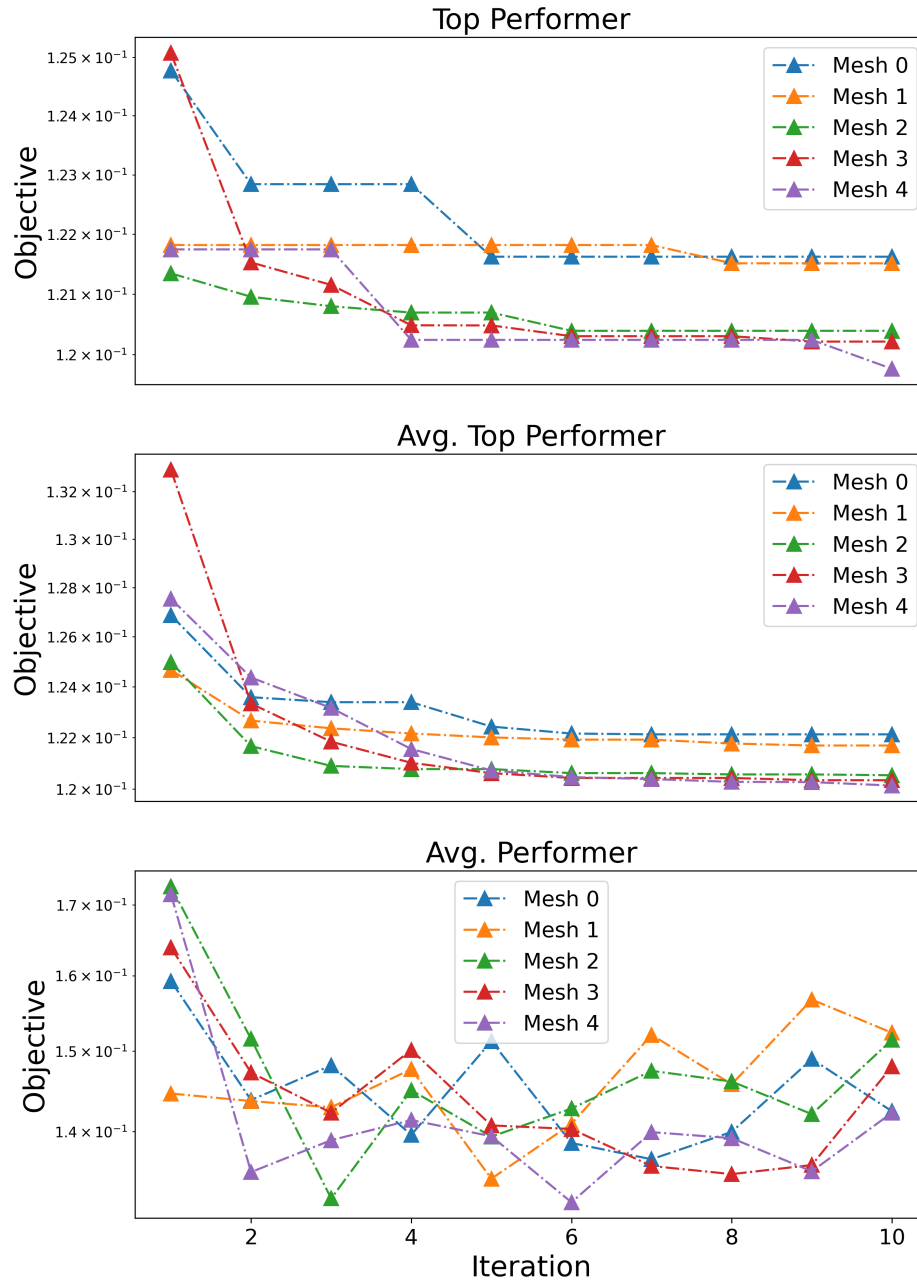


Figure 5.18: Optimization convergence plots for GA: The candidate solution performance over each optimization iteration for the best parent, average parents, and overall average entire population.

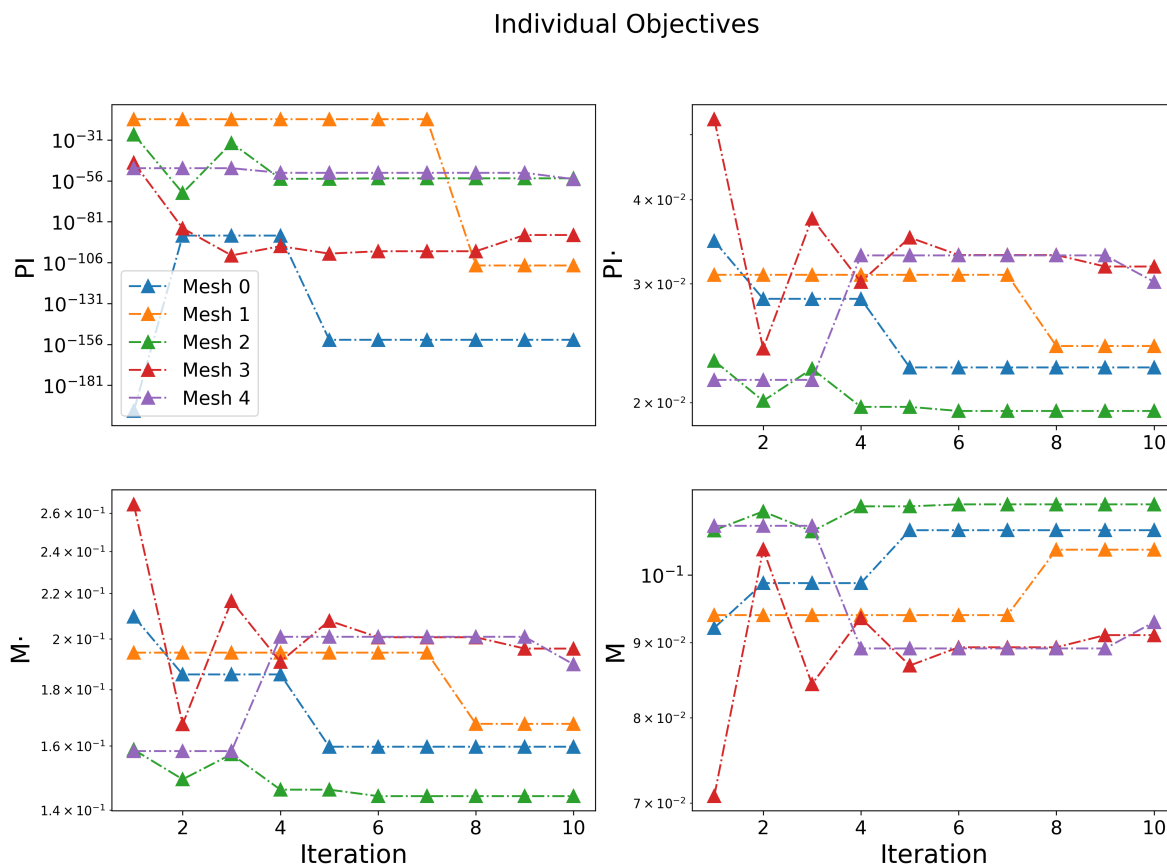


Figure 5.19: Optimization convergence plots: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI·), propagating radical (M·) and starting monomer (M) concentrations.

5.9 Discussion and Comparison

The optimization approaches explored in this chapter demonstrate significant improvements over the design of experiments (DOE) approach presented in Chapter 2. The parameters determined through DOE yielded an average objective value of approximately 0.1395, while the best objective value obtained through computational optimization was approximately 0.1119, representing a 14.6% improvement in performance relative to the DOE results. This substantial enhancement underscores the potential role of optimization in formulation and development environments, enabling engineers and scientists to identify performance-boosting trends at minimal cost.

Interestingly, the results indicate that given sufficient time and computational resources, BO, CMA-ES, and GA perform comparably. All three methods converge to parameters

within similar regions of the solution space. Therefore, a more nuanced analysis is necessary to differentiate these algorithms and determine their relative value in optimizing physical systems.

Performance Gains

Fig. 5.20 illustrates the convergence plots for each algorithm’s top performer, average of the top 4 performers, and overall average performer (using a time step of $dt = 2.5 \times 10^{-4}$ and mesh size of $h = 3.5 \times 10^{-5}$). The top performer plot suggests that BO, CMA-ES, and GA all identify parameters achieving similar objectives within the allotted 160 simulation budget.

However, comparing convergence rates directly may not provide an accurate assessment, as random high performers could unfairly skew the perceived performance of an algorithm in a single case. The average top 4 performers plot offers a more balanced perspective, mitigating the effects of potential outliers in the initial iterations. This plot reveals that the SEK model for BO rapidly converges to the optimal value within a few steps, while the GA requires several more iterations to achieve comparable parameters. The comparison with CMA-ES is more challenging to interpret, as all but the top performer begin to diverge as the algorithm prioritizes exploration in later iterations.

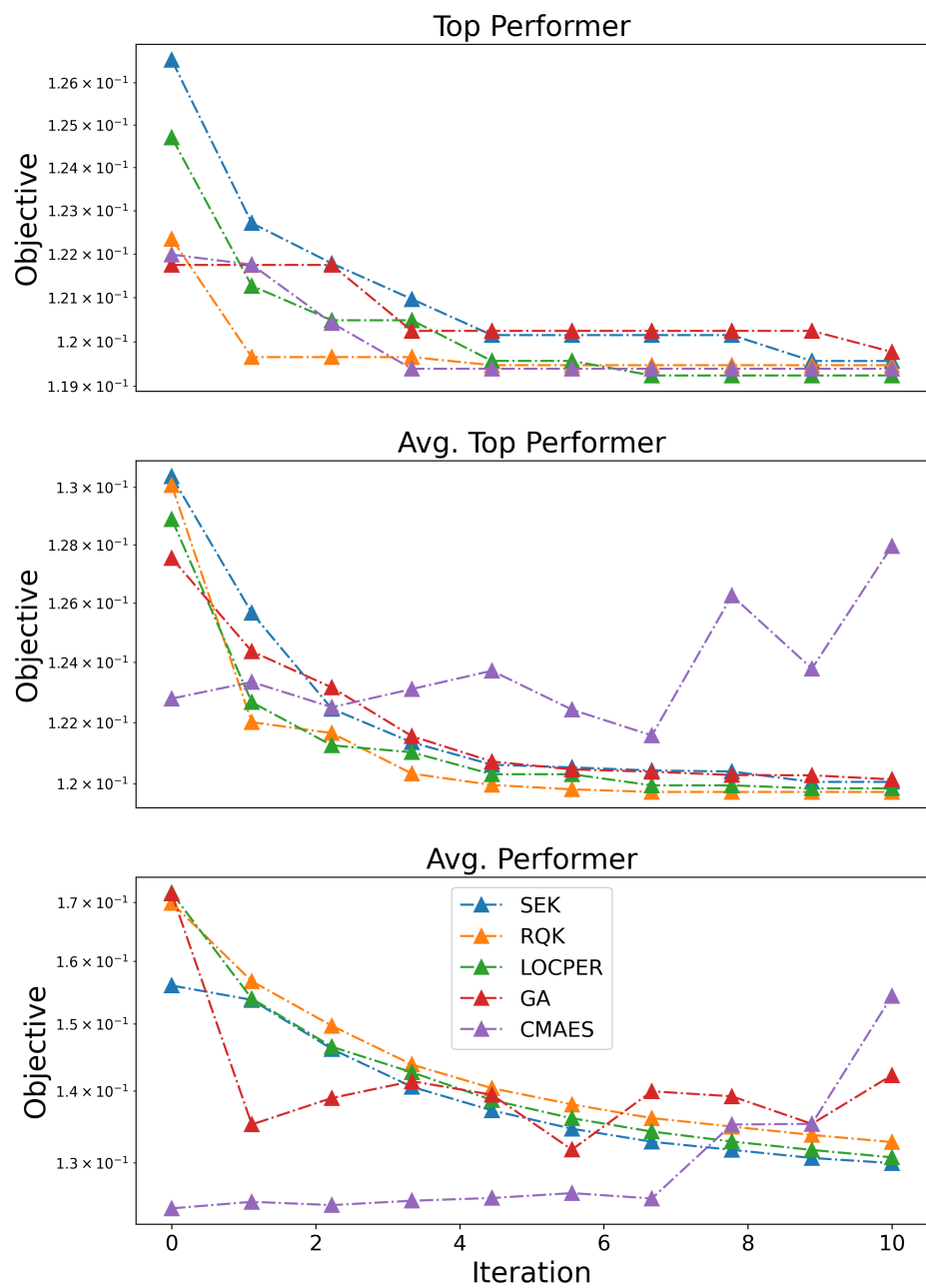


Figure 5.20: Performance comparison between all optimization approaches.

Computational Gains

Examining the average performer convergence plots for each algorithm provides further insights. BO exhibits smooth improvement across all samples, indicating high efficiency in information gain for each sample. In contrast, the GA employs random search for 8 out of 16 samples, representing an unstructured exploration approach. This can be inefficient, as sampling around previously explored areas may lead to low-information gain evaluations of the objective function, with no penalization within that area.

These findings highlight the importance of understanding the underlying assumptions about the objective function, as illustrated in Fig. 5.1. When no assumptions can be made about the correlation between inputs and outputs, random search remains the most effective approach. In such cases, GA may converge faster than BO, as randomly selected values can potentially discover unexplored regions of the objective space. However, for objectives where basic assumptions can be made — such as correlated outputs for nearby input parameters — methods like BO can actively avoid low-information areas and prioritize sampling in regions likely to yield high information gain.

From a computational cost perspective, BO with SEK converges within a single step, requiring only 32 function evaluations. To achieve a comparable final objective value, the GA requires all 10 iterations (160 evaluations). With each iteration taking approximately 18 hours for the fine mesh ($h = 1.68 \times 10^{-5}$, $dt = 5.5 \times 10^{-5}$), BO/SEK potentially saves six days of computation time compared to GA. For problems with high computational cost, low-dimensional parameter spaces, and basic assumptions about the underlying objective function structure, the computational savings of BO outweigh its increased overhead costs, as demonstrated in Fig. 5.21.

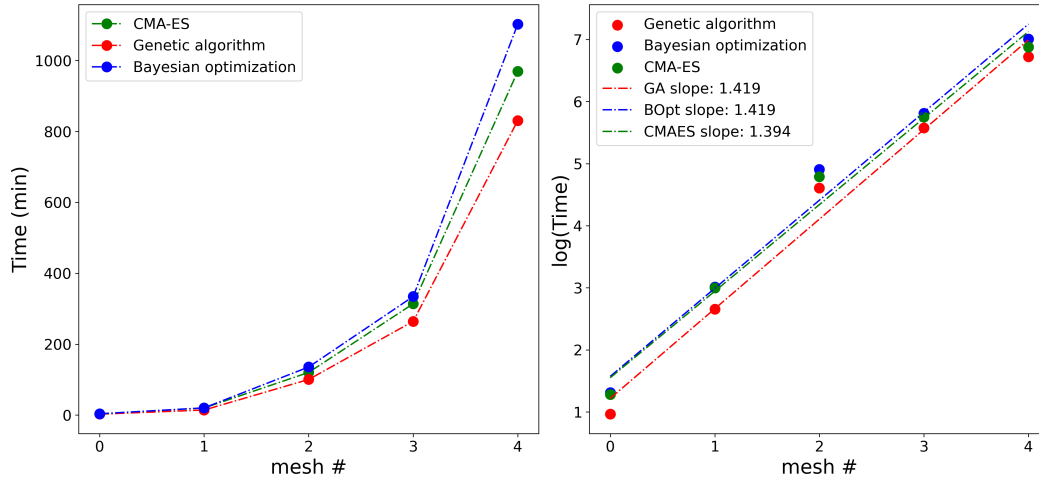


Figure 5.21: Time complexity comparison of optimization approaches over increasingly fine meshes: BOpt, CMA-ES, GA.

Assessing Value & Managing Expectations

The value derived from computational optimization can be measured in two primary areas: increased performance and decreased computational cost. It is crucial to identify *a priori* where the expected value will be obtained when designing optimization methods and to manage expectations, particularly when there is a risk of overestimating potential returns from an algorithm.

For managerial positions, optimization provides numerous opportunities to potentially inflate performance metrics due to nebulous benchmarks that depend on stakeholder or customer perspectives. However, the true beneficiaries of simulation and optimization tools are often engineers and scientists whose benchmarks are their alternative workflows. Creating compelling reasons for adopting simulation and optimization requires demonstrating tangible benefits relative to these existing workflows. This underscores the importance of reporting performance gains from optimization methods relative to the optimized formulation identified by DOE, as done in Chapter 2.

In the field of non-convex optimization, one should manage expectations regarding performance gains (in terms of the final objective value). Substantial performance boosts are unlikely for physics-based optimizations, particularly when the number of parameters is low, as scientists and engineers generally possess solid physics-based intuition and can select reasonable initial guesses. If an objective function yields a 200% increase in performance, it likely indicates either a highly complex system or a fundamental misunderstanding of the underlying physics.

While optimization will likely provide some performance boost as per the objective, an often-overlooked benchmark is the overall reduction in cost throughout the development workflow. For example, if the traditional workflow requires manual parameter tweaking in simulations or systematic DOE, the benefit of an optimization approach can be measured in reduced costs associated with each function evaluation (as indicated by the computational savings of BO/SEK relative to CMA-ES or GA). These savings can be even greater when benchmarked against workflows involving rudimentary brute-force experimentation, which are still common in many fields.

5.10 Conclusions

When benchmarking optimization methods, it's crucial to consider not just performance metrics and computational cost, but also the effort required for implementation. More complex optimization techniques demand a deeper understanding of the underlying mathematics and computational implementations to make intelligent algorithm choices. For instance, BO requires users to define kernel and acquisition functions, and make decisions about model fitting and implementation techniques - all of which take considerable time and energy to understand. The GA is simpler, but still requires systematic choices of optimization hyperparameters such as number of parents, children, generations, and mutation settings.

In contrast, random search requires no assumptions and is a truly blackbox approach. Interestingly, for each optimization approach shown in Fig. 5.20, the first iteration relies purely on random search. This highlights an often-overlooked strength of this simple method. As the cost of compute trends toward zero, random search (and in the limit, grid search) becomes an increasingly attractive approach to system optimization. Its lack of assumptions and ease of implementation make it particularly appealing as computational resources become more abundant. The rise of cloud computing enables programmers to harness an arbitrary number of CPUs/GPUs, limited only by budget and energy constraints. This trend naturally lends itself to random/grid search methods, especially for smaller-scale problems.

However, it's important to note that the effectiveness of random search isn't universal. For high-dimensional problems or those with complex, highly nonlinear objective functions, more sophisticated optimization methods may still offer significant advantages. The choice between simple and complex optimization strategies should be guided by a thorough understanding of the problem at hand, available computational resources, and the specific goals of the optimization process.

This work has demonstrated the power of advanced optimization techniques in materials design, but also underscores the enduring value of simpler approaches like random search. As these approaches are integrated into industrial workflows for materials development in advanced manufacturing, balancing the sophistication of the methods with their practical implementability and computational efficiency will be key. Future work in this field could explore hybrid approaches that leverage the strengths of both simple and complex optimization strategies, potentially leading to more robust and accessible tools for materials scientists and engineers.

Chapter 6

Summary and Future Work

“I don’t believe in the empirical science. I only believe in a priori truth.”

- Kurt Gödel

6.1 Summary

This dissertation has presented a comprehensive framework for the simulation-informed optimization of UV-curable materials in advanced manufacturing processes. Through the integration of experimental data, multi-physics modeling, and advanced optimization techniques, I have demonstrated a more integrative approach to materials design that balances theoretical rigor with practical applicability.

Key contributions of this work include:

- Development of a detailed multi-physics model for UV-curing processes, incorporating reaction kinetics, heat transfer, and light propagation.
- Implementation and comparison of various optimization techniques, including Bayesian Optimization, Covariance Matrix Adaptation Evolution Strategy, and Genetic Algorithms, for navigating complex material design spaces.
- Demonstration of significant improvements over traditional design of experiments (DOE) approaches, with optimization methods achieving a 14.6% improvement in performance relative to DOE results.
- Identification of critical process parameters and their impacts on final material properties, providing valuable insights for manufacturing process control.

This work has highlighted the potential of computational methods in accelerating materials discovery and process optimization, while also underscoring the importance of balancing sophisticated techniques with practical implementability.

6.2 Future Work

While this dissertation has covered major steps in improving formulation processes in the field of UV-curable materials in additive manufacturing applications, several avenues for future research remain:

1. **Advanced Material Models:** Develop more sophisticated material models that capture additional complexities of UV-curable systems, such as the formation of polymer networks and the impact of crosslinking on material properties with the use of molecular dynamics and uncertainty quantification.
2. **Multi-Objective Optimization:** Extend the optimization framework to handle alternative objectives that encapsulate more manufacturing applications.
3. **Deep Learning Integration:** Explore the integration of deep learning approaches (i.e., neural networks) as an alternative surrogate model that can further accelerate the optimization process through model compression.
4. **Experimental Validation:** Conduct comprehensive experimental studies to validate and refine the computational models, particularly focusing on the predicted spatial gradients in material properties.
5. **Process-Structure-Property Relationships:** Investigate the relationships between processing parameters, resulting material structure, and final product properties to enable more targeted optimization strategies.
6. **Real-Time Process Control:** Develop algorithms for real-time process control based on the insights gained from the simulation and optimization studies, enabling adaptive manufacturing processes.
7. **Extension to Other Manufacturing Processes:** Apply and adapt the developed framework to other advanced manufacturing processes beyond UV-curing, such as selective laser sintering or fused deposition modeling.

Some initial work has been made in some of these applications and remain as unfinished work which can be found in the Chapter 7: Appendix.

In conclusion, this work represents a step forward in the application of simulation-informed optimization to advanced manufacturing. By providing a robust framework for material design and process optimization, I have laid the groundwork for future advancements in the field, ultimately contributing to the development of more sophisticated, efficient, and tailored manufacturing processes.

Chapter 7

Appendix

“There was a footpath leading across fields to New Southgate, and I used to go there alone to watch the sunset and contemplate suicide. I did not, however, commit suicide, because I wished to know more mathematics.”

- Bertrand Russell

7.1 Step-Wise Reactions

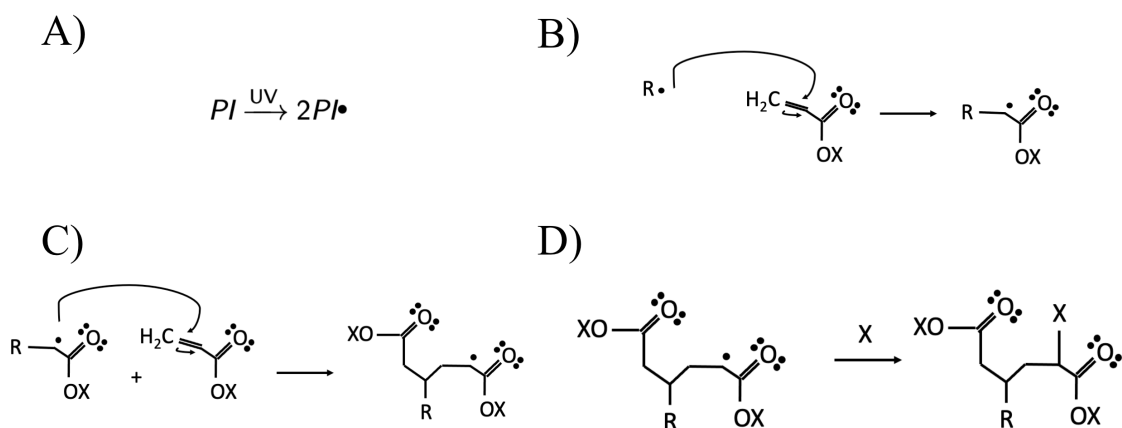


Figure 7.1: The step-wise reaction mechanism for radical initiated acrylate reactions: A) Dissociation, B) Initiation, C) Propagation, D) Termination.

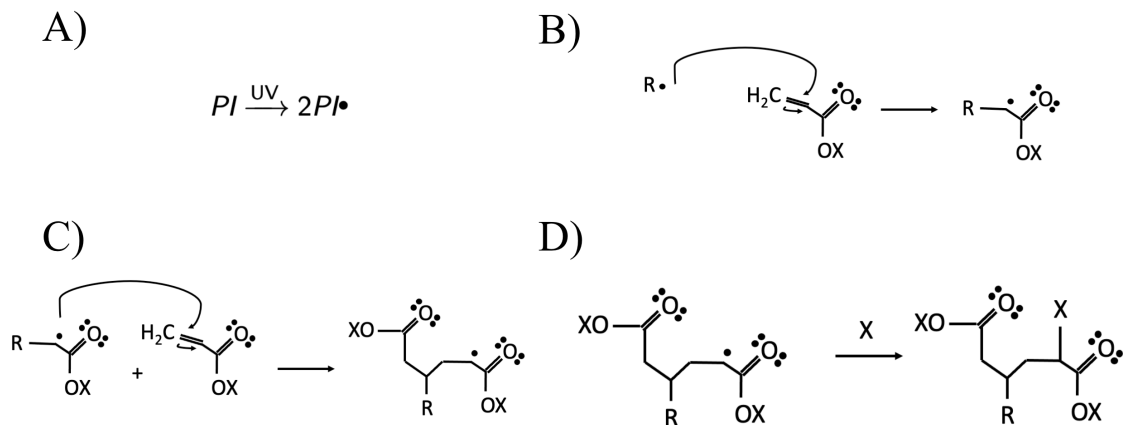


Figure 7.2: The step-wise reaction mechanism for heat initiated urethane reactions: A) hydroxyl group attacks carbon. electrons in double bond of carbon moves to nitrogen, B) hydrogen on oxygen moves to free electrons on nitrogen group, C) urethane segment complete.

7.2 Numerical Stability of Fixed-Point Iteration

Special thanks to Aidan Gould for aid in identifying these constraints.

Consider the simplified heat equation:

$$\mathbf{u}_t = \alpha \mathbf{u}_{xx} \quad (7.1)$$

the Crank-Nicholson solution is

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{k} = \frac{\alpha}{2h^2} (\mathbf{A}\mathbf{u}^{n+1} + \mathbf{A}\mathbf{u}^n) \quad (7.2)$$

where $\mathbf{A}$ is matrix defining the diffusion constants. Combining variables yields:

$$\left(\mathbf{I} - \frac{\alpha}{2h^2} \mathbf{A} \right) \mathbf{u}^{n+1} = \left(\mathbf{I} + \frac{\alpha k}{2h^2} \mathbf{A} \right) \mathbf{u}^n \quad (7.3)$$

where we seek to find an iterative solution for $\mathbf{u}^{n+1}$. Let the first guess be the solution at the previous time step

$$\mathbf{u}^{(0)} = \mathbf{u}^n \quad (7.4)$$

and the iterative estimate of the next value:

$$\mathbf{u}^{(1)} = \frac{\alpha k}{2h^2} \mathbf{A} \mathbf{u}^{(0)} + \mathbf{u}^n + \frac{\alpha k}{2h^2} \mathbf{A} \mathbf{u}^n \quad (7.5)$$

The solution for the next time step is the iterative application of the $\frac{\alpha k}{2h^2} \mathbf{A}$ operator. Assume for simplicity that $\frac{\alpha k}{2h^2} = 1$, then:

$$\mathbf{u}^{n+1} = \sum_{j=0}^{n-1} \left(\frac{\alpha k}{2h^2}\right)^j \mathbf{A}^j \mathbf{u}^{n+1} + \left(\frac{\alpha k}{2h^2}\right)^n \mathbf{A}^n \mathbf{u}^n. \quad (7.6)$$

If the true solution is

$$\mathbf{u}^{n+1} = \frac{\alpha k}{2h^2} \mathbf{A} \mathbf{u}^n \quad (7.7)$$

and we let the fixed-point iterator go to ∞ :

$$\mathbf{u}^\infty = \frac{\alpha k}{2h^2} \mathbf{A} \mathbf{u}^\infty, \quad (7.8)$$

we define the error as:

$$\mathbf{e}^{n+1} = \mathbf{u}^{n+1} - \mathbf{u}^\infty \quad (7.9)$$

and

$$\mathbf{e}^n = \mathbf{u}^n - \mathbf{u}^\infty \quad (7.10)$$

then

$$\mathbf{e}^{n+1} = \left(\frac{\alpha k}{2h^2} \mathbf{A}\right)^{n+1} \mathbf{e}^0 \quad (7.11)$$

indicating that

$$\left\| \frac{\alpha k}{2h^2} \mathbf{A} \right\|_\infty < 1 \quad (7.12)$$

for the error to be capped. More specifically,

$$\frac{\alpha k}{h^2} < \frac{1}{2}. \quad (7.13)$$

7.3 Numerical Examples

The section of the appendix provides detailed implementations of numerical methods and optimization techniques. In the “Numerical Methods” subsection, a Poisson equation is solved using a finite difference approach with Gauss-Seidel iteration. The C++ code demonstrates how to apply boundary conditions, solve the system iteratively, and output the results. In the “Numerical Optimization” subsection, Python code is provided for optimizing the Beale function using different gradient-based methods, including basic gradient descent, gradient descent with momentum, and ADaptive Momentum (ADAM). Each method’s implementation is clearly presented, showcasing their use in navigating the non-convex landscape of the Beale function. A comparative analysis of these optimization techniques is illustrated through a benchmark comparison.

Numerical Solution for Poisson Equation in C++

Consider the Poisson equation:

$$\nabla_x \cdot (\kappa \nabla_x u(x, y)) + f = 0 \quad (7.14)$$

With the following boundary conditions:

$$u(x = 0|y) = 0 \quad (7.15)$$

$$u(x = 1|y) = 0 \quad (7.16)$$

$$u(x|y = 0) = 0 \quad (7.17)$$

$$u(x|y = 1) = 1 \quad (7.18)$$

$$(7.19)$$

Assume a source:

$$f(x/2, y/2) = 1000; \quad (7.20)$$

with the following assumptions:

- let $\kappa = 1$
- expand Laplacian operator $[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}] \cdot \left[\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right] = -f \rightarrow \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f$
- discretize $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \rightarrow (u[i+1] - 2u[i] + u[i-1])/dx^2 + (u[i+N] - 2u[i] + u[i-N])/dy^2 = -f[i]$
- assume $dx = dy$ then $1/dx^2(u[i+1] + u[i-1] + u[i+N] + u[i-N] - 4u[i]) = u[i] = (u[i+1] + u[i-1] + u[i+N] + u[i-N])/4$

- use of Gauss-Seidel iteration is sufficient for solving $Ax = b$

Example: Implementation of 2D Poisson solver in C++

```
void output_results(std::vector<double> &u){
    for (int i = 0; i < N2; ++i){
        std::cout << u[i] << " ";
        if (i % N == N-1){
            std::cout << std::endl;
        }
    }
}
```

Example: Implementation of 2D Poisson solver in C++ continued

```
const int    N    = 10;           // start with 3 nodes along a side
const int    N2   = N*N;         // total number of nodes
const double L    = 1.0;         // length of a side
const double dx   = L / (N - 1); // spatial discretization
const double tol  = 1e-6;        // tolerance for convergence

// function to apply boundary conditions
void apply_bcs(std::vector<double> &u){
    // set u=1 at y=1
    for (int i = N2 - N; i < N2; i++){
        u[i] = 1.0;               // along top of square (y=1)
    }

    // u=0 at y=0, x=0, x=1 already set in pre-initialization
}
```

Example: Implementation of 2D Poisson solver in C++ continued

```
void solver(std::vector<double> &u, std::vector<double> &f){
    bool converge = false;
    double rhs, err, max_err;
    for (int iter = 0; iter < 10000; ++iter){
        max_err = 0.;
        for (int i = N; i < N2-N; ++i){
            if (i % N == 0){
                // left boundary
                continue;
            }
            else if (i % N == N-1){
                // right boundary
                continue;
            }
            else{
                // step 1: compute rhs
                rhs = 0.25*(f[i]*dx*dx+u[i+1]+u[i-1]+u[i+N]+u[i-N]);

                // step 2: check absolute difference between /LHS - RHS/
                err = std::abs(rhs - u[i]);

                // step 3: update u_new with RHS
                u[i] = rhs;

                // iterator until convergence
                if (err > max_err){
                    max_err = err;
                }
            }
        }
        if (max_err < tol){
            std::cout << "converged after "
                      << iter << " steps with max_err: "
                      << max_err << std::endl;

            break;
        }
    }
}
```

Example: Implementation of 2D Poisson solver in C++ continued

```
int main(int argc, char** argv) {  
    // create u for fixed point solver (initialized at zero)  
    std::vector<double> u = std::vector<double>(N*N, 0.0);  
  
    // create f vector  
    std::vector<double> f = std::vector<double>(N*N, 0.0);  
    int f_node = (N/2) + (N/2)*N;  
    f[f_node] = 1000.0;  
  
    // print statements  
    std::cout << "boundary conditions: " << std::endl;  
    apply_bcs(u);  
    output_results(u);  
    std::cout << "\nsource term: " << std::endl;  
    output_results(f);  
    std::cout << std::endl;  
  
    // solve  $Ax = b$   
    solver(u, f);  
    std::cout << "\nsolution: " << std::endl;  
    output_results(u);  
    return 0;  
}
```

Numerical Optimization

An example implementation in Python is shown below while attempting to optimize the non-convex [Beale Function](#)

```
# manual gradients and hessians for each function
def grad_beale(curr_loc, ord=1, method=1):
    x = curr_loc[0]
    y = curr_loc[1]

    # compute gradient
    dfdx = 2 * (1.5 - x + x * y) * (-1 + y) + \
           2 * (2.25 - x + x * y ** 2) * (-1 + y ** 2) + \
           2 * (2.6250 - x + x * y ** 3) * (-1 + y ** 3)
    dfdy = 2 * (1.5 - x + x * y) * x + \
           2 * (2.25 - x + x * y ** 2) * (2 * x * y) + \
           2 * (2.6250 - x + x * y ** 3) * (3 * x * y ** 2)

    gradient = np.array([dfdx, dfdy])

    if ord == 1:
        return gradient

    elif ord == 2:
        # compute hessian
        ddfddx = 2 * (y - 1) * (y - 1) + \
                 2 * (y * y - 1) * (y * y - 1) + \
                 2 * (y * y * y - 1) * (y * y * y - 1)
        ddfdydx = 4 * x * (y ** 2 - 1) + \
                  4 * y * (x * y * y - x + 2.25) + \
                  6 * x * (y * y * y - 1) * y * y + \
                  y * y ** 2 * (x * y * y * y - x + 2.625) + \
                  2 * x * (y - 1) + \
                  2 * (x * y - x + 1.5)
        ddfddy = 18 * x * x * y * y * y * y + \
                 8 * x * x * y * y + \
                 4 * x * (x * y * y - x + 2.25)

        hessian = np.array([[ddfddx, ddfdydx], [ddfdydx, ddfddy]])
        return gradient, hessian
```



```
def grad_descent_momentum(grad_fn, init_loc=[0, 0],
                           eta=1e-6, n_steps=10000, gamma=0.9, f=None):
    """
        simple gradient descent with momentum
    """

    # initialize trajectories
    traj = np.zeros((n_steps, len(init_loc)))
    traj[0, :] = np.array(init_loc)

    # initialize momentum
    vt = np.zeros(2)

    for i in range(1, n_steps):
        if f is not None:
            vt = gamma * vt \
                + eta * grad_fn(f=f, curr_loc=traj[i - 1, :], ord=1)
            traj[i, :] = traj[i - 1, :] - vt
        else:
            vt = gamma * vt \
                + eta * grad_fn(curr_loc=traj[i - 1, :], ord=1)
            traj[i, :] = traj[i - 1, :] - vt

    return traj
```

```

def grad_descent_adam(grad_fn, init_loc=[0, 0], n_steps=10000, f=None):
    eta_adam = 1e-3

    beta1 = 0.9
    beta2 = 0.99
    eps = 1e-8

    # initialize trajectories
    traj = np.zeros((n_steps, len(init_loc)))
    traj[0, :] = np.array(init_loc)

    # initialize momentum
    mt = np.zeros(2)
    st = np.zeros(2)

    for i in range(1, n_steps):
        # method: ADaptive Momemntum
        if f is not None:
            gt = grad_fn(f=f, curr_loc=traj[i - 1, :], ord=1)
            mt = beta1 * mt + (1 - beta1) * gt
            st = beta2 * st + (1 - beta2) * gt ** 2
            mt_hat = mt / (1 - beta1)
            st_hat = st / (1 - beta2)

            traj[i, :] = traj[i - 1, :] - eta_adam * mt_hat
            traj[i, :] /= np.sqrt(st_hat + eps)

        else:
            gt = grad_fn(curr_loc=traj[i - 1, :], ord=1)
            mt = beta1 * mt + (1 - beta1) * gt
            st = beta2 * st + (1 - beta2) * gt ** 2
            mt_hat = mt / (1 - beta1)
            st_hat = st / (1 - beta2)

            traj[i, :] = traj[i - 1, :] - eta_adam * mt_hat
            traj[i, :] /= np.sqrt(st_hat + eps)

    return traj

```

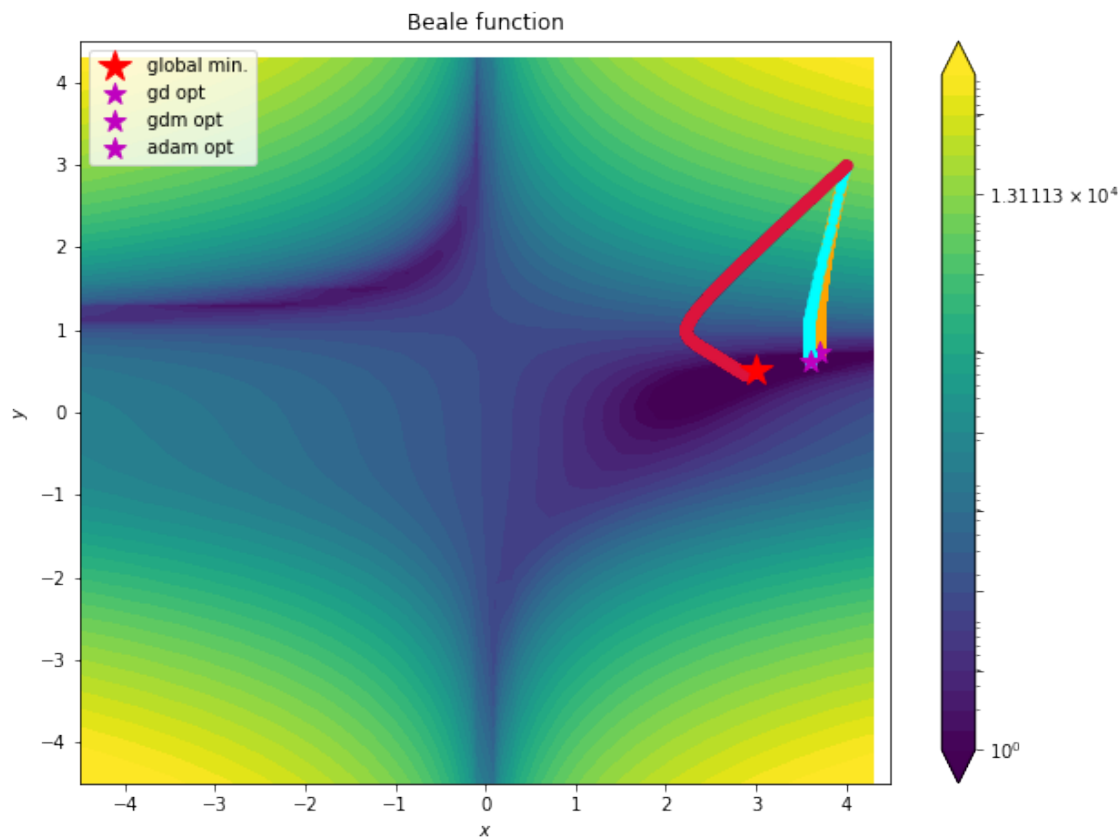


Figure 7.3: A benchmark comparison between gradient descent, gradient descent + momentum, and ADAPtive Momentum (ADAM) gradient descent on the non-convex Beale Function

7.4 Effects of Variable Mesh Size on Optimization for Bayesian Optimization

Squared Exponential Kernel

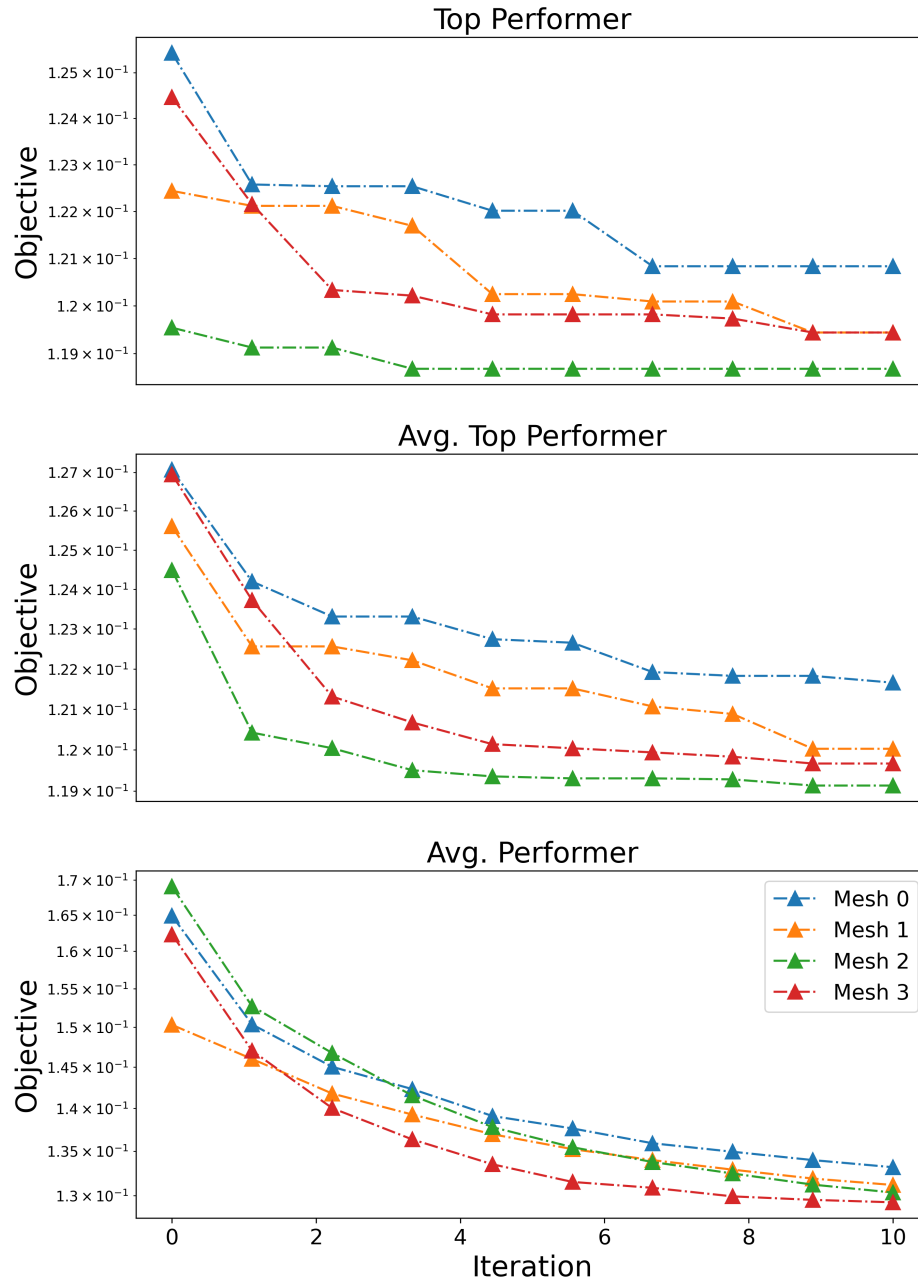


Figure 7.4: Optimization convergence plots of the SEK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.

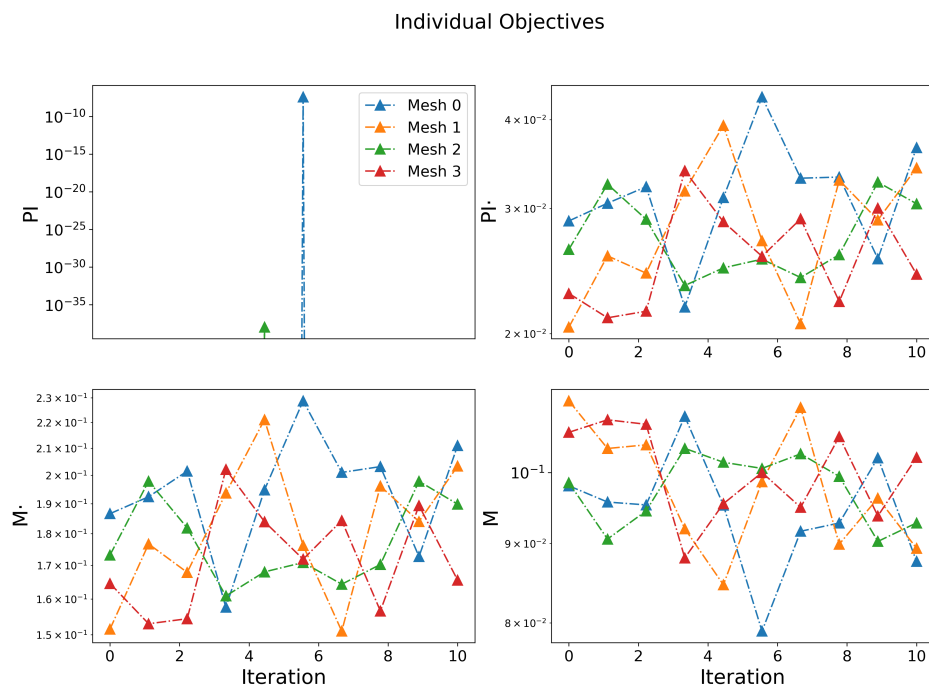


Figure 7.5: Optimization convergence plots of the SEK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI·), propagating radical (M·) and starting monomer (M) concentrations.

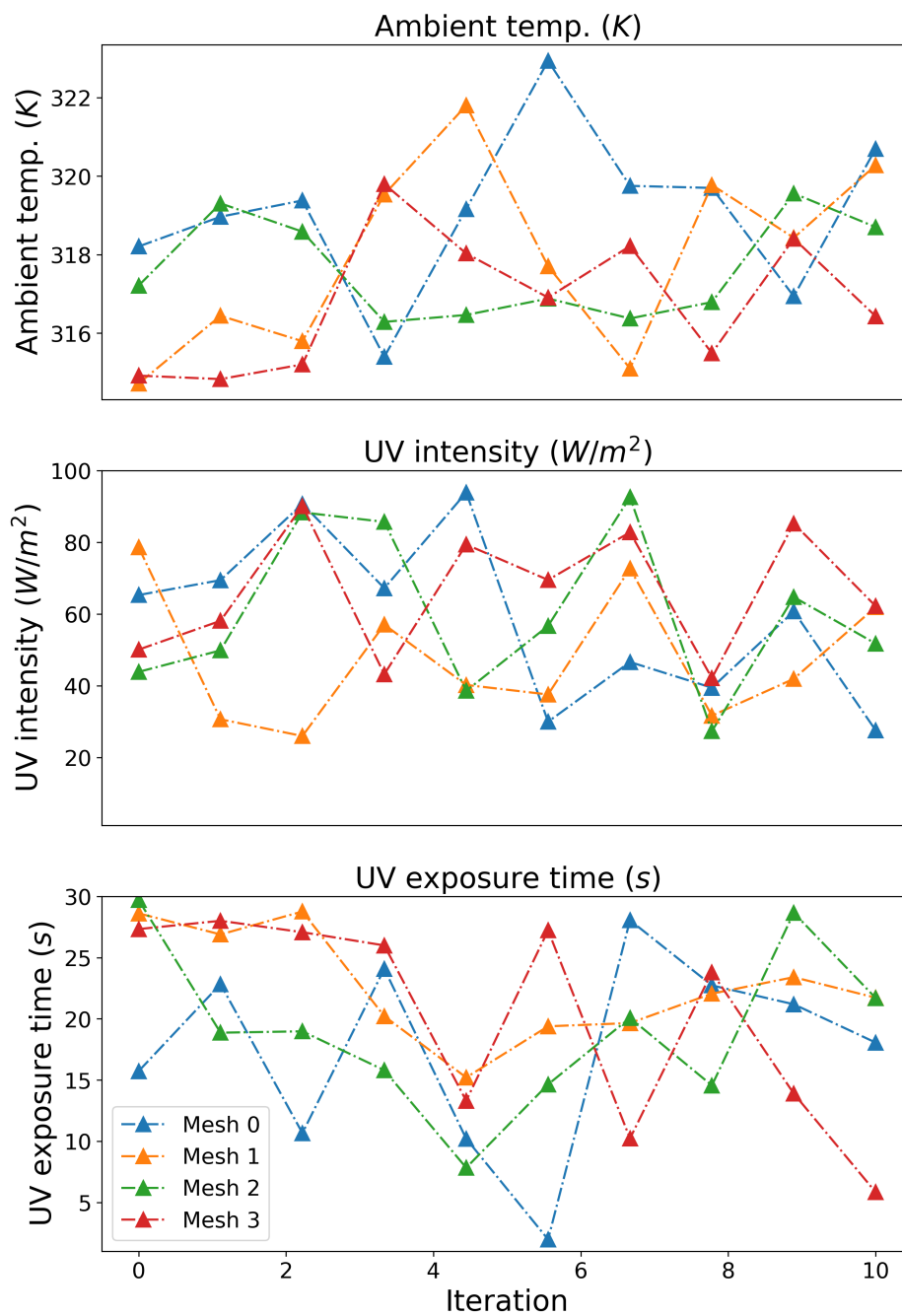


Figure 7.6: Process parameter convergence of the SEK:

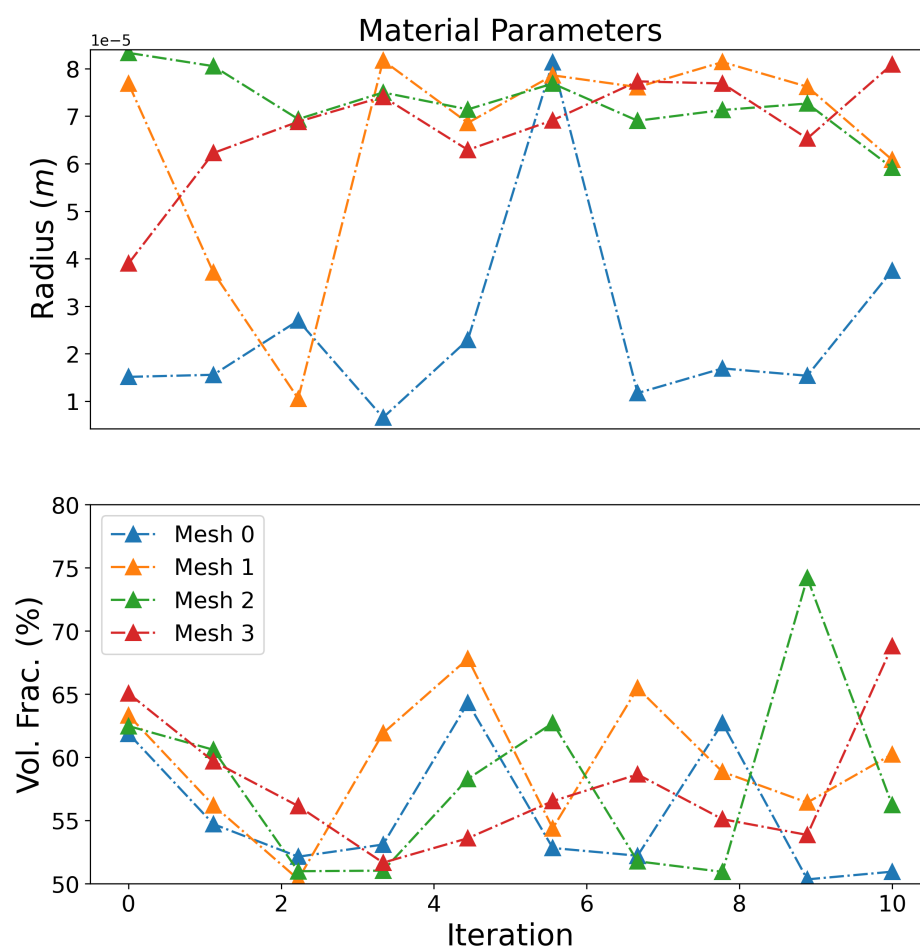


Figure 7.7: Material parameter convergence of the SEK:

Rational Quadratic Kernel

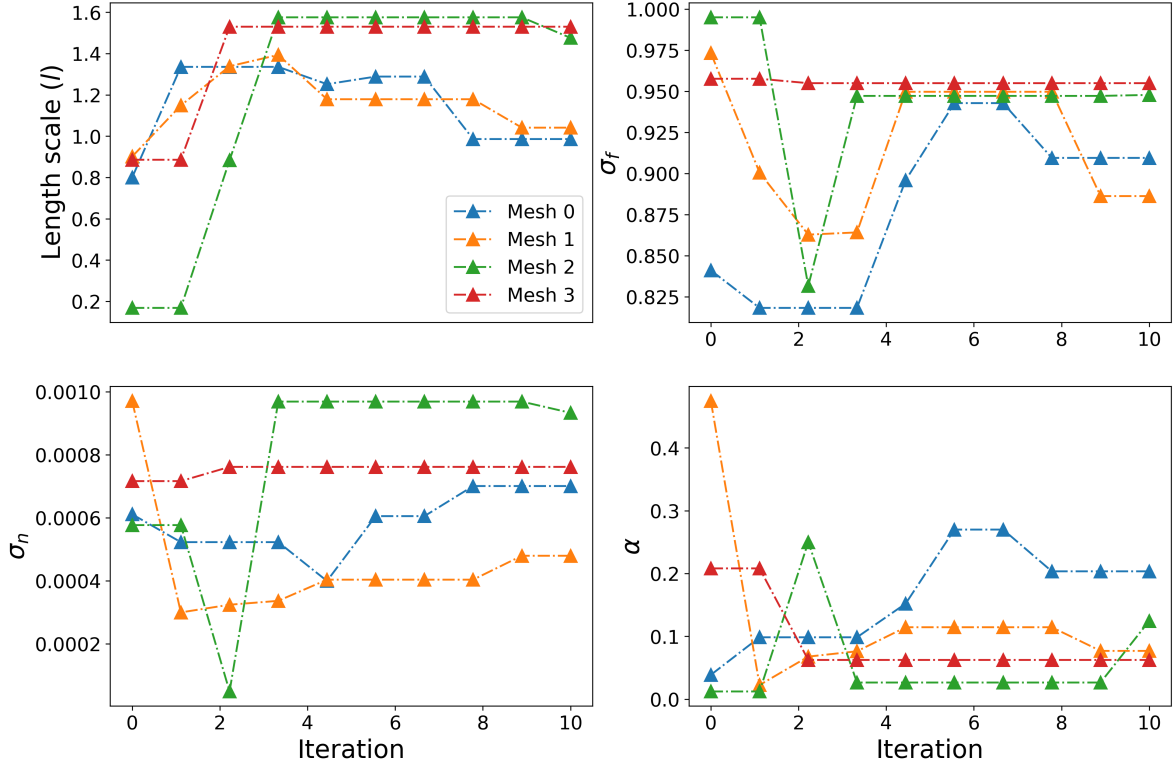


Figure 7.8: Hyper-parameter Convergence of the RQK: As more data is collected after each optimization iteration, the hyperparameters selection begins to stabilize for each mesh as the Gaussian Process *learns* the parameter space.

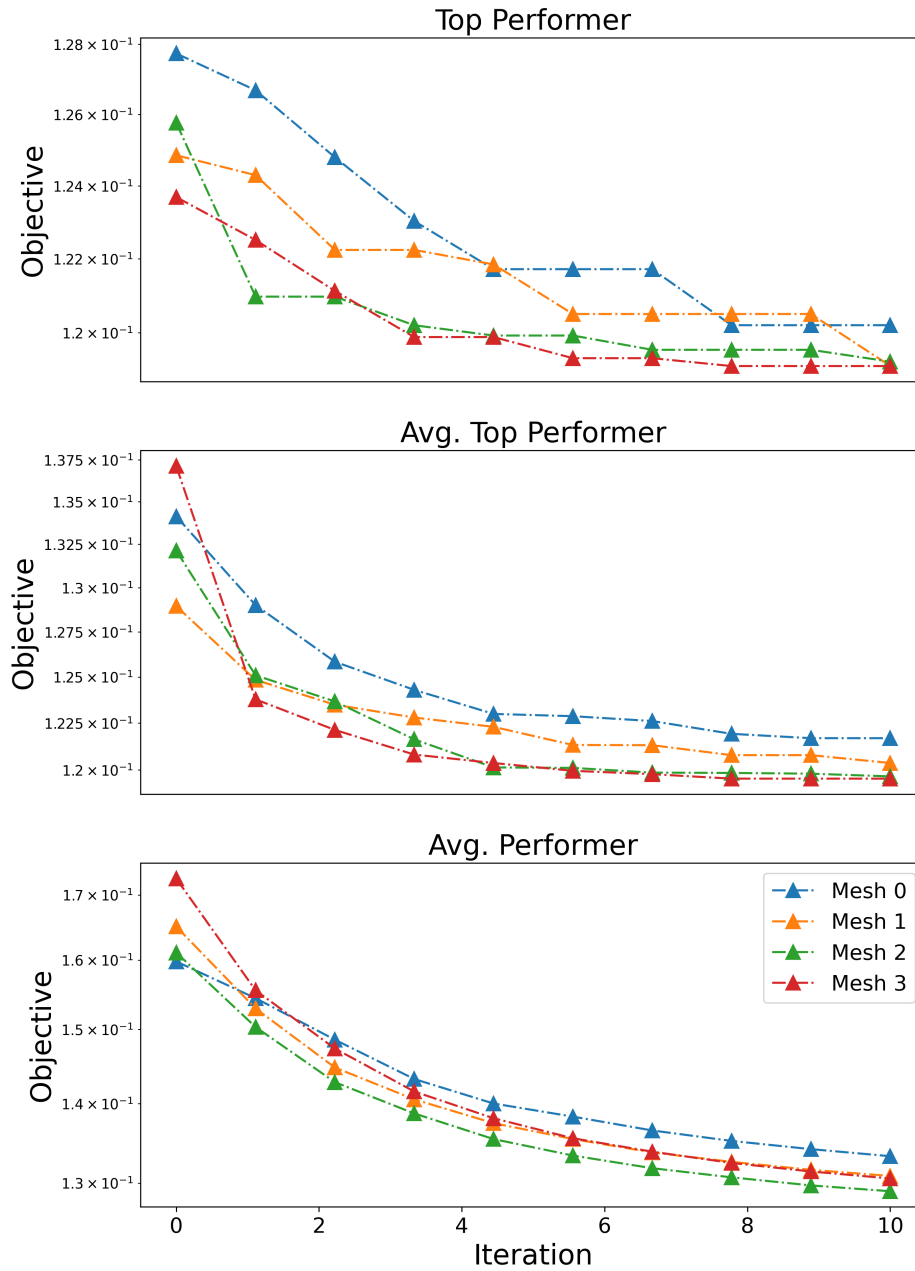


Figure 7.9: Optimization convergence plots for RQK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.

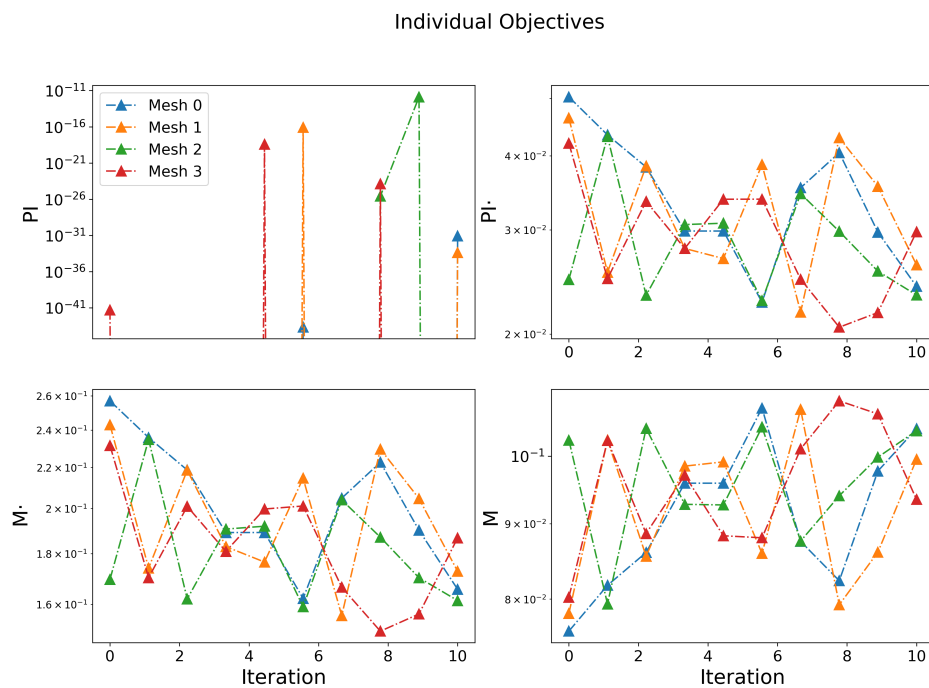


Figure 7.10: Optimization convergence plots for RQK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical (PI·), propagating radical (M·) and starting monomer (M) concentrations.

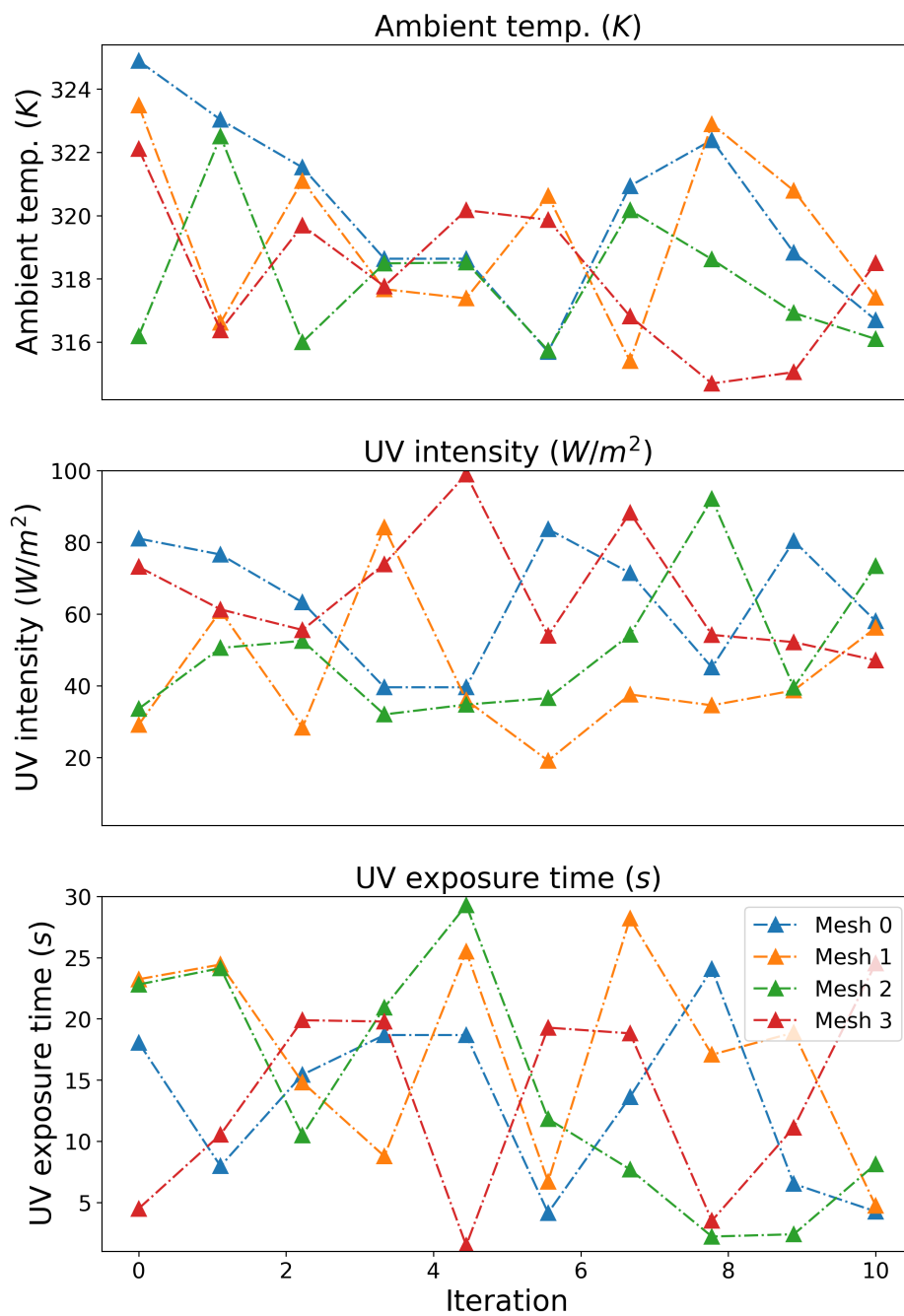


Figure 7.11: Process parameter convergence for RQK.

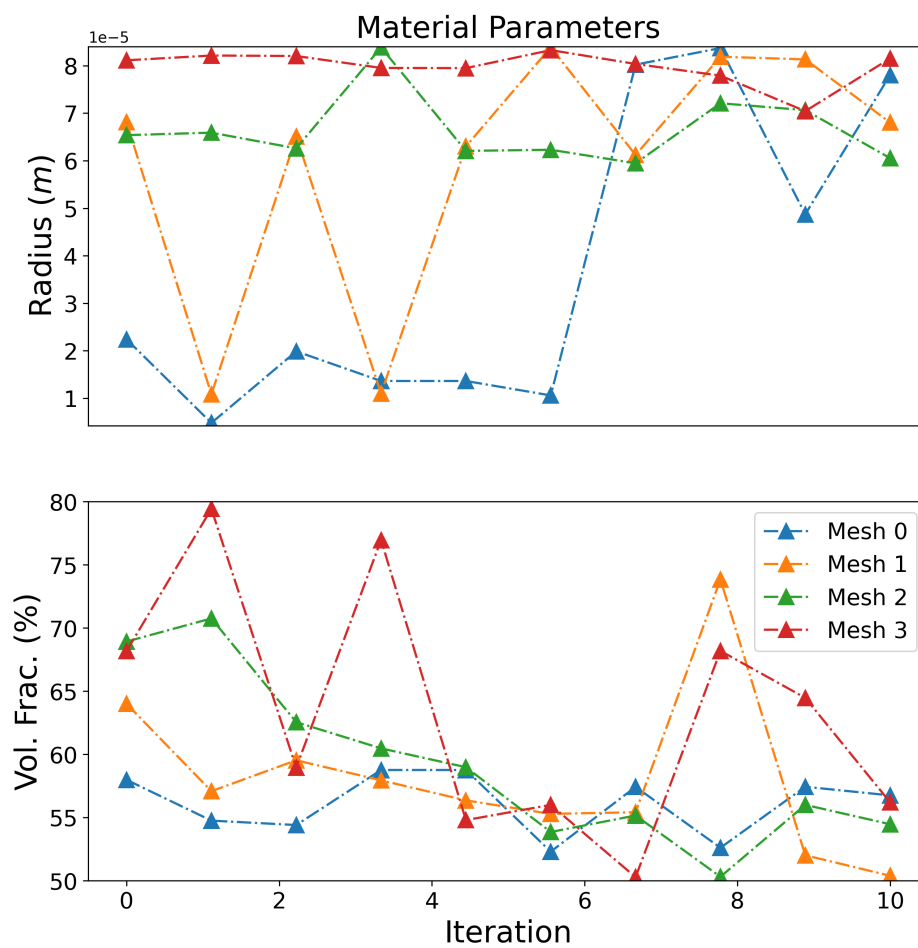


Figure 7.12: Material parameter convergence for RQK.

Local-Periodic Kernel

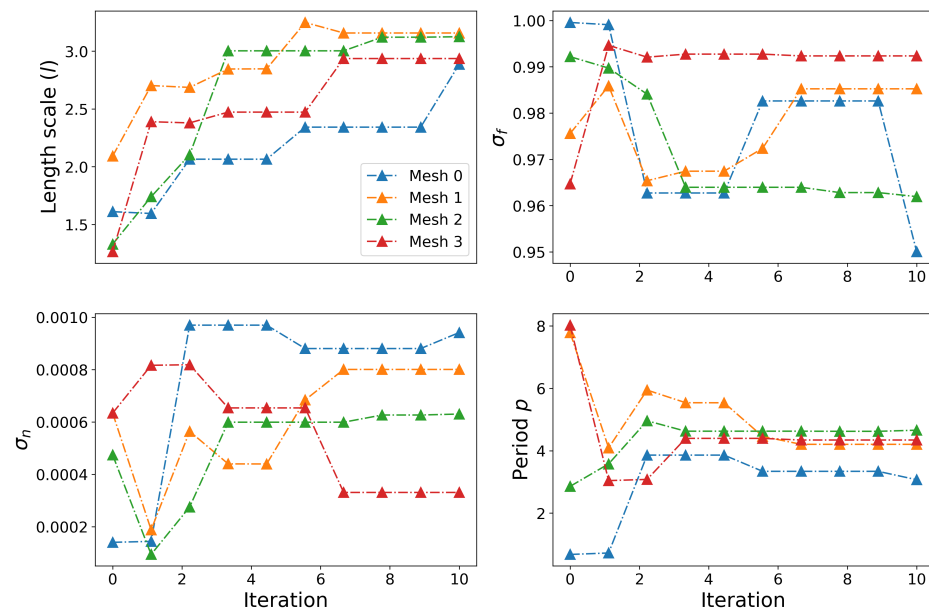


Figure 7.13: Hyper-parameter Convergence of the LPK: As more data is collected after each optimization iteration, the hyperparameters selection begins to stabilize for each mesh as the Gaussian Process *learns* the parameter space.

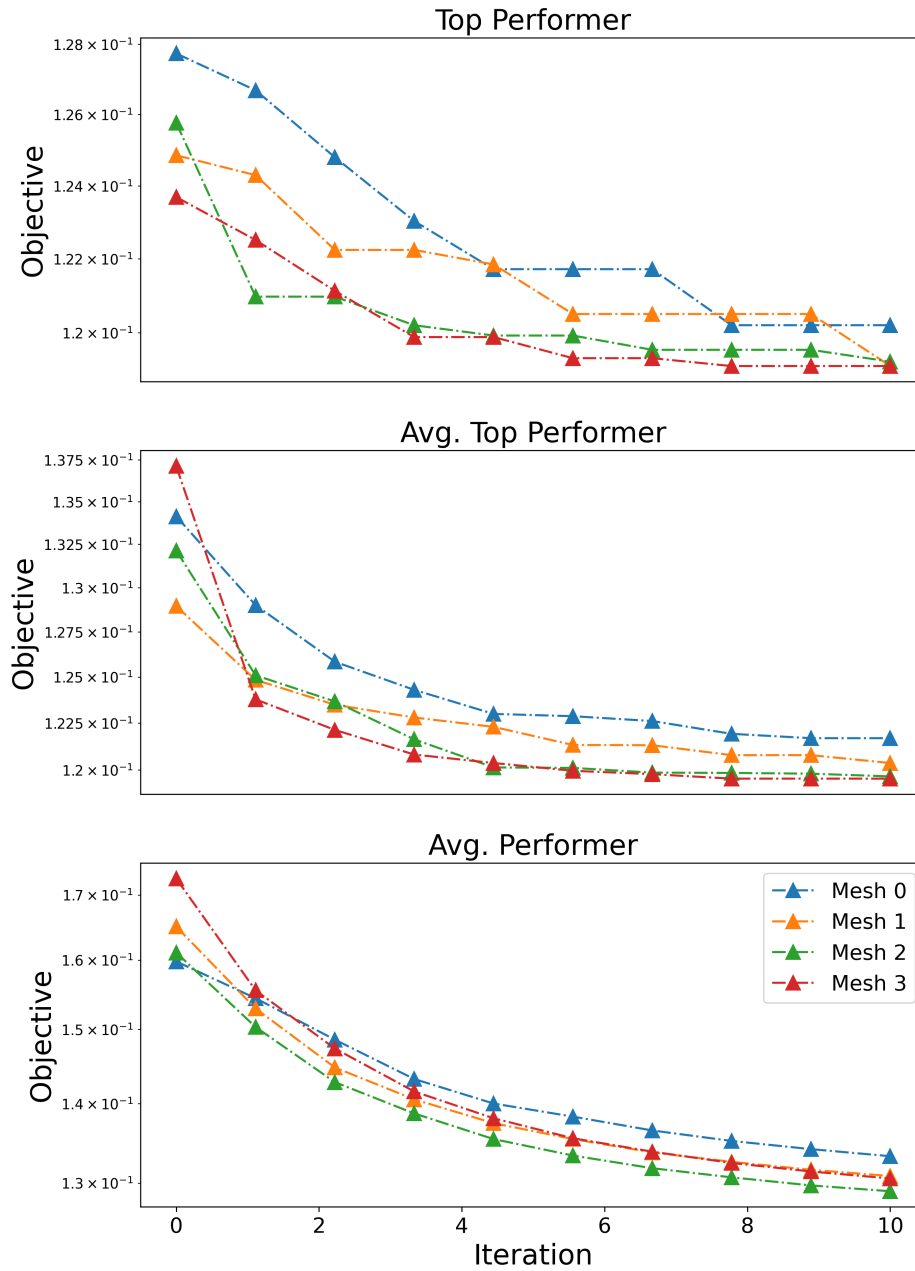


Figure 7.14: Optimization convergence plots for LPK: The candidate solution performance over each optimization iteration for the best candidate, average of the 6 best candidates, and overall average of all sampled candidates.

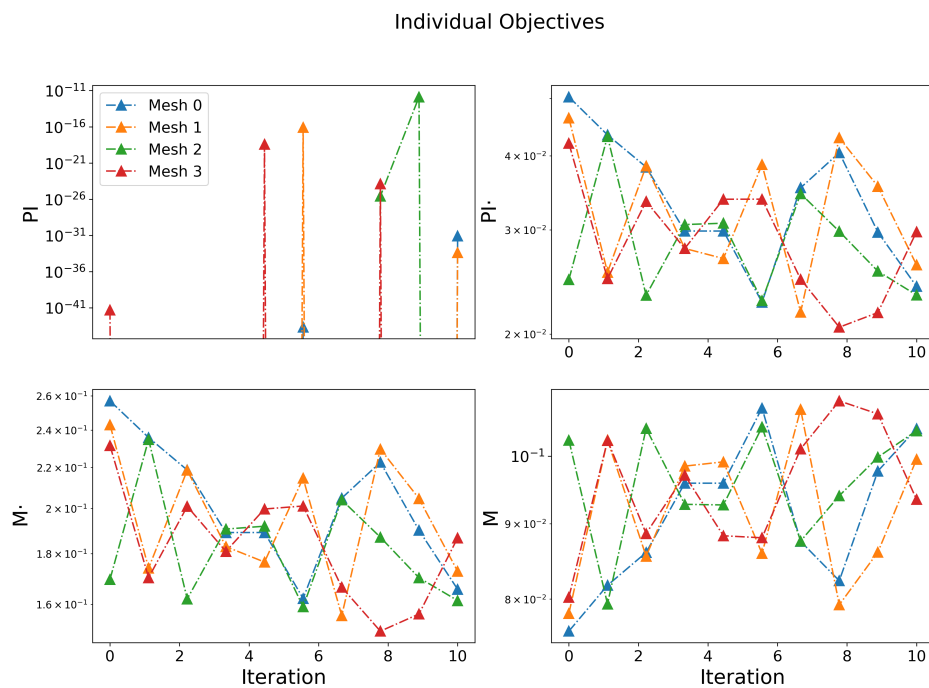


Figure 7.15: Optimization convergence plots for LPK: The best candidate solution performance relative to each individual objective – photoinitiator (PI), primary radical ($PI\cdot$), propagating radical ($M\cdot$) and starting monomer (M) concentrations.

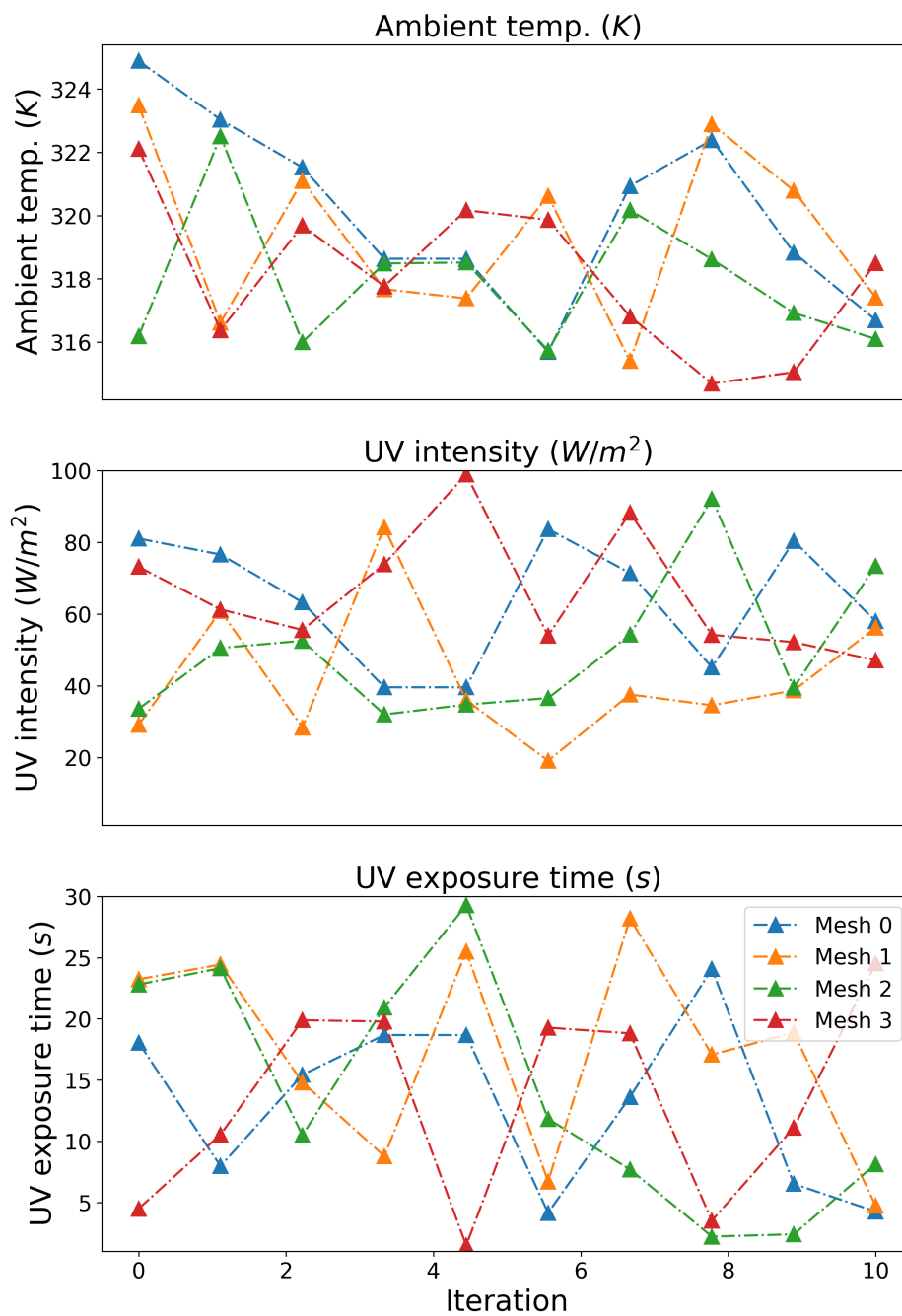


Figure 7.16: Process parameter convergence for LPK.

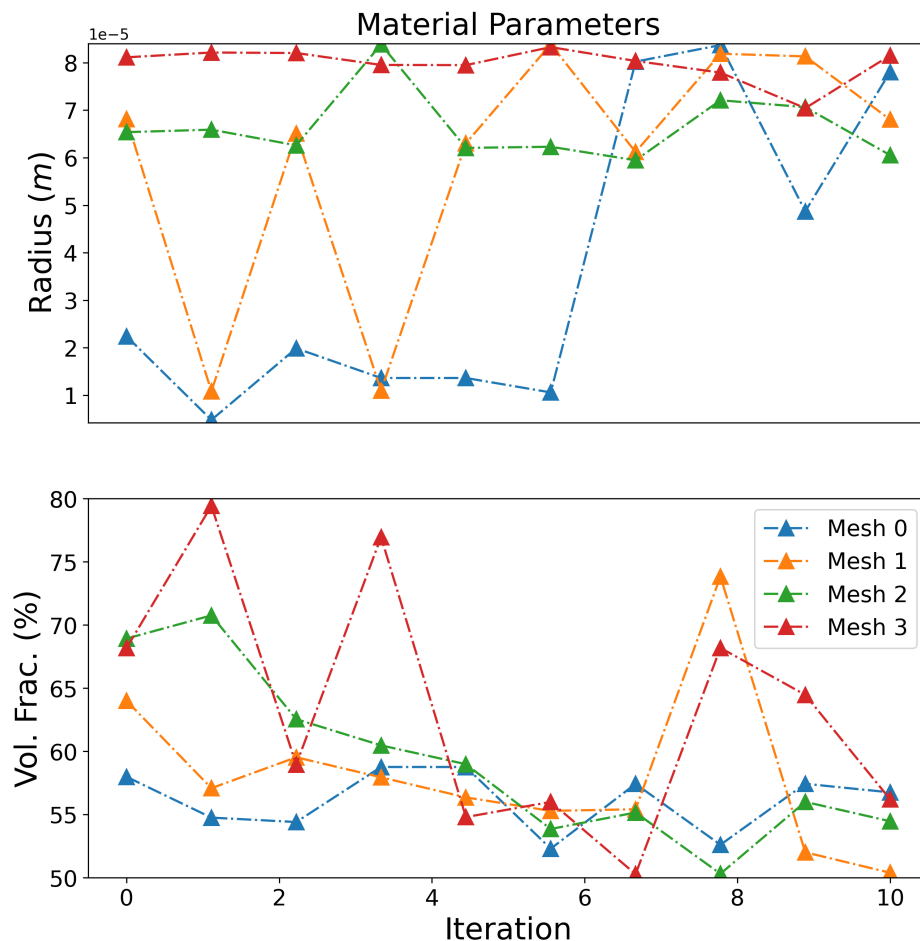


Figure 7.17: Material parameter convergence.

7.5 Graveyard of Unfinished Modeling and Simulation

This section provides general outline of failed projects that I embarked on throughout my years at UC Berkeley. It contains some of my initial derivations and simulations. Nothing in this section is complete and ideas are not fully formulated, however I have documented my train of thought for potential future completion.

A Simple Model for Fluid Dispensing Analysis and Control

If the energy variation is omitted, the incompressible flow in the syringe can be approximated as pipe flow and described by the continuity and momentum equations as seen in 7.21:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \\ \frac{\partial \rho \vec{V}}{\partial t} + \nabla \cdot (\rho \vec{V} \vec{V}) = -\nabla p + \nabla \cdot \vec{\tau} + \rho g \end{cases} \quad (7.21)$$

where $\vec{V}\vec{V}$ is a dyadic product and τ and ρg are stress tensor and gravitation body force, respectively.

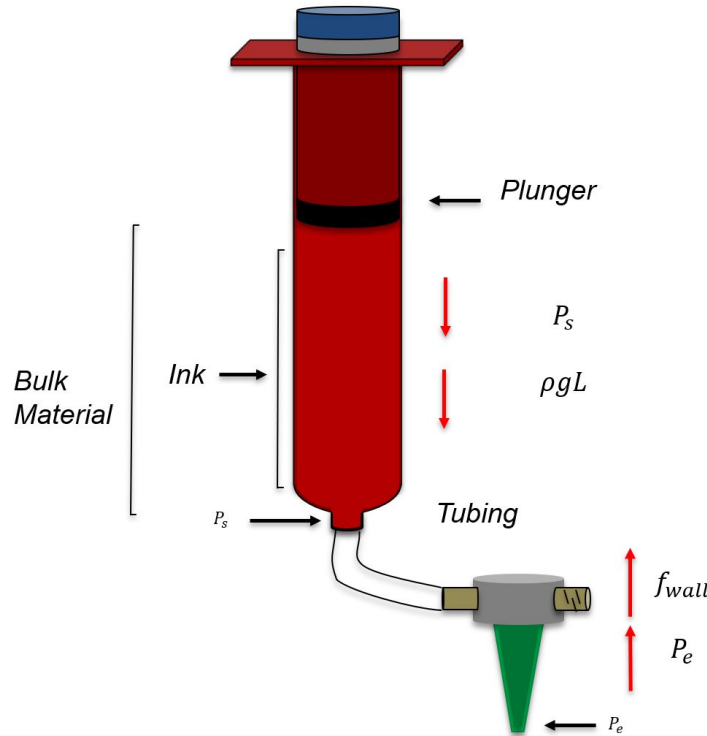


Figure 7.18: Typical DIW system

For a typical syringe and nozzle used for DIW printing, a syringe is filled with an ink and degassed as shown in Figure 7.18. Lines are connected to the syringe that are long enough for sensor attachment, and then subsequently connected to nozzle. The diameters of the line and nozzle are significantly smaller than that of the syringe barrel and it can be assumed that the friction loss of the fluid flow in the syringe can be ignored [6]. The entire system can be considered to be in the vertical orientation, therefore the forces acting in positive z -direction can be the gravitational force, and the pressure from the syringe. The forces in the opposing direction are the pressure at the exit, P_e and the force of friction along the wall, f_{wall} . Note that f_{wall} is a function of the wall shear stress τ_{wall} . The balanced equation in terms of pressure drop and shear stress at the wall are shown below:

$$\pi R^2 (P_s + \rho g L) = \pi R^2 P_e + f_{wall} \quad (7.22)$$

where R is the inner radius of the line and the nozzle, and

$$f_{wall} = \left(2\pi RL + 2\pi RL \right) \tau_{wall}. \quad (7.23)$$

Letting $\Delta P = P_s - P_e + \rho g L$ and plugging 7.23 into 7.22 and solving for τ_{wall} yields:

$$\tau_{wall} = \frac{R \left(\Delta P + \rho g L \right)}{2L}. \quad (7.24)$$

Similarly, the shear stress as a function of the radius r is given by:

$$\tau(r) = \frac{r \left(\Delta P - \rho g L \right)}{2L}. \quad (7.25)$$

Then, taking the ratio of τ and τ_{wall} :

$$\frac{\tau}{\tau_{wall}} = \frac{r}{R} \quad (7.26)$$

and the shear rate is given as the negative of the velocity profile:

$$\dot{\gamma} = -\frac{du_z}{dr} \quad (7.27)$$

where u_z is the flow velocity through the cylinder with the radius r . The boundary condition of $u_z = 0|_{r=R}$ and $u_z < \infty|_{r=0}$. Also recall that the definition of the shear stress is given by:

$$\tau = \eta \dot{\gamma}. \quad (7.28)$$

Plugging Equations 7.27 and 7.28 into Equation 7.26 is as follows:

$$\tau = -\eta \frac{du_z}{dr} \Rightarrow -\eta \frac{du_z}{dr} = \tau_{wall} \frac{r}{R} \Rightarrow du_z = -\tau_{wall} \frac{r}{\eta R} dr \quad (7.29)$$

Integrating Equation 7.29:

$$\int du_z = -\frac{\tau_{wall}}{\eta R} \int r dr \Rightarrow u_z(r) = -\frac{\tau_{wall} r^2}{2\eta R} + C_2 \quad (7.30)$$

where C_2 can be found through the boundary condition $u_z = 0|_{r=R}$.

$$u_z(r = R) = 0 = -\frac{\tau_{wall} R^2}{2\eta R} + C_2 \Rightarrow C_2 = \frac{\tau_{wall} R}{2\eta} \quad (7.31)$$

The resulting velocity profile is:

$$u_z(r) = -\frac{\tau_{wall}r^2}{2\eta R} + \frac{\tau_{wall}R}{2\eta} \quad (7.32)$$

Simplifying Equation 7.32 by substituting Equation 7.24 yields:

$$u_z(r) = \frac{R^2\Delta P}{4\eta L} \left[1 - \left(\frac{r}{R} \right)^2 \right] \quad (7.33)$$

The flow rate Q can be obtain from Equation 7.33 by integrating $u_z(r)$:

$$2\pi \int_0^R r u_z(r) dr = 2\pi \frac{R^2\Delta P}{4\eta L} \int r dr - 2\pi \frac{R^2\Delta P}{4\eta L R^2} \int r^3 dr \quad (7.34)$$

Let $K_1 = 2\pi \frac{R^2\Delta P}{4\eta L}$:

$$K_1 \int r dr - \frac{K_1}{R^2} \int r^3 dr = \frac{K_1}{2} r^2 - \frac{K_1}{4R^2} r^4 \Big|_{r=0}^{r=R} \quad (7.35)$$

$$\frac{K_1}{2} r^2 - \frac{K_1}{4R^2} r^4 \Big|_{r=0}^{r=R} = \frac{\pi R^4 \Delta P}{4\eta L} - \frac{\pi R^6 \Delta P}{8\eta L R^2} \Rightarrow \boxed{Q = \frac{\pi R^4 \Delta P}{8\eta L}} \quad (7.36)$$

Power law fluid

Recall that for Newtonian fluids that the relationship between shear stress τ and shear rate $\dot{\gamma}$ is linear and given by Equation 7.37.

$$\tau = \eta \dot{\gamma} \quad (7.37)$$

However, the relationship between shear stress τ and shear rate $\dot{\gamma}$ is not linear for non-Newtonian fluids, due to the viscosity η being a function of shear rate $\dot{\gamma}$.

$$\eta(\dot{\gamma}) = \begin{cases} \eta_0 & \tau : \tau < 2\tau_y^2 \\ k\dot{\gamma}^{n-1} + \frac{\tau_y}{\dot{\gamma}} & \tau : \tau \geq 2\tau_y^2 \end{cases} \quad (7.38)$$

Where τ_y is the yield stress, η is viscosity at very low shear rates, k is the consistency index, and n is the power-law index. Equation 7.38 was obtained empirically from [11], where $\tau_y > 0$ and $n \neq 1$. For such a fluid, the rheological behavior function for power-law fluids is obtained after integrating 7.26 and 7.27.

For conditions where $\tau < 2\tau_y^2$

Recall that from equations 7.37 and 7.38 that:

$$\begin{aligned}
\tau = -\eta_0 \frac{du_z}{dr} &\Rightarrow -\eta_0 \frac{du_z}{dr} = \frac{r\tau_w}{R} \\
\Rightarrow \int du_z &= \frac{-\tau_w}{R\eta_0} \int r dr \\
u_z(r) &= -\frac{\tau_w r^2}{2R\eta_0}
\end{aligned}$$

$$\therefore u_z(r) = -\frac{\tau_w r^2}{2\eta_0 R} + C_1 \quad (7.39)$$

The constant can be found from the boundary conditions $r = R$.

$$\begin{aligned}
u_z(r = R) = 0 &= -\frac{\tau_w R^2}{2\eta_0 R} + C_1 \Rightarrow C_1 = \frac{\tau_w R}{2\eta_0} \\
u_z(r) &= \frac{\tau_w R}{2\eta_0} - \frac{\tau_w r^2}{2\eta_0 R} \\
u_z(r) &= \frac{\tau_w}{2\eta_0} R \left(1 - \left(\frac{r}{R} \right)^2 \right) \quad (7.40)
\end{aligned}$$

The flow rate can be found by integrating $u_z(r)$ multiplied by the perimeter of the nozzle between $r = 0$ and $r = R$:

$$\begin{aligned}
\int 2\pi r u_z(r) dr &= \frac{\pi \tau_w R}{\eta_0} \int_{r=0}^{r=R} \left(r - \frac{r^3}{R^2} \right) dr \\
\Rightarrow \frac{\pi \tau_w R}{\eta_0} \left[\int_{r=0}^{r=R} r dr - \int_{r=0}^{r=R} \left(\frac{r^3}{R^2} \right) dr \right] &\Rightarrow \frac{\pi \tau_w R}{2\eta_0} r^2 - \frac{\pi \tau_w R}{4\eta_0} \frac{r^4}{R^2} \Big|_{r=0}^{r=R} \\
\Rightarrow Q &= \frac{\pi \tau_w R^3}{2\eta_0} - \frac{\pi \tau_w R^3}{4\eta_0} = \frac{\pi \tau_w R^3}{4\eta_0} \\
\text{substitute } \tau_w &= \frac{R \Delta P^*}{2L}
\end{aligned}$$

$$\boxed{Q = \frac{\Delta P^* R^4}{8\eta_0}} \quad \text{where} \quad \Delta P^* = \left(P_s - P_e + \rho g L \right) \quad (7.41)$$

For conditions where $\tau \geq 2\tau_y^2$

Recall from Equations 7.37 and 7.38 that:

$$\tau = \eta(\dot{\gamma})\dot{\gamma} \quad \text{where} \quad \eta(\dot{\gamma}) = K\dot{\gamma}^{n-1} + \frac{\tau_y}{\dot{\gamma}} \quad \Rightarrow \quad \tau = K\dot{\gamma}^n + \tau_y$$

$$\text{now plug into Equation 7.26} \quad \Rightarrow \quad K\left(-\frac{du_z}{dr}\right)^n + \tau_y = \frac{r\tau_w}{R}$$

$$\int du_z = -\int \left(\frac{r\tau_w - R\tau_y}{KR}\right)^{1/n}$$

$$\text{Let } U = \frac{r\tau_w - R\tau_y}{KR} \quad \text{and} \quad \frac{du}{dr} = \frac{\tau_w}{KR} \quad \Rightarrow \quad dr = \frac{KR}{\tau_w} du$$

plugging back into the integrand:

$$u_z(r) = -\frac{KR}{\tau_w} \int U^{1/n} du = -\frac{KR}{\tau_w} \left(\frac{n}{n+1}\right) U^{\frac{n+1}{n}} + C_1$$

$$\Rightarrow \quad u_z(r) = -\frac{KR}{\tau_w} \left(\frac{n}{n+1}\right) \left(\frac{r\tau_w - R\tau_y}{KR}\right)^{\frac{n+1}{n}} + C_1$$

The constant can be found from the boundary conditions $r = R$.

$$u_z(r = R) = 0 \quad \Rightarrow \quad C_1 = \frac{KR}{\tau_w} \left(\frac{n}{n+1}\right) \left(\frac{\tau_w - \tau_y}{K}\right)^{\frac{n+1}{n}}$$

$$\boxed{u_z(r) = \left(\frac{KR}{\tau_w}\right) \left(\frac{n}{n+1}\right) \left(\frac{\tau_w - \tau_y}{K}\right)^{\frac{n+1}{n}} - \left(\frac{KR}{\tau_w}\right) \left(\frac{n}{n+1}\right) \left(\frac{r\tau_w - R\tau_y}{KR}\right)^{\frac{n+1}{n}}} \quad (7.42)$$

The flow rate for these conditions can be found by $\int_{r=0}^{r=R} 2\pi r u_z(r) dr$.

$$\begin{aligned} Q &= 2\pi \int_{r=0}^{r=R} u_z(r) r dr \\ &= 2\pi \left(\frac{KR}{\tau_w}\right) \left(\frac{n}{n+1}\right) \int_{r=0}^{r=R} r \left[\left(\frac{\tau_w - \tau_y}{K}\right)^{\frac{n+1}{n}} - \left(\frac{r\tau_w - R\tau_y}{RK}\right)^{\frac{n+1}{n}} \right] dr \\ &= 2\pi \left(\frac{KR}{\tau_w}\right) \left(\frac{n}{n+1}\right) \left[\int r \left(\frac{\tau_w - \tau_y}{K}\right)^{\frac{n+1}{n}} dr - \int r \left(\frac{r\tau_w - R\tau_y}{RK}\right)^{\frac{n+1}{n}} dr \right] \end{aligned} \quad (7.43)$$

Term 2

$$\begin{aligned}
\text{Let } u &= \left(\frac{r\tau_w - R\tau_y}{RK} \right) \quad \text{and} \quad \frac{du}{dr} = \frac{\tau_w}{KR} \quad \Rightarrow \quad dr = \frac{KR}{\tau_w} du \\
\text{where } r &= \left(\frac{KR}{\tau_w} \right) \left(u + \frac{\tau_y}{K} \right) \\
\Rightarrow \text{Term 2} &= - \left(\frac{KR}{\tau_w} \right)^2 \int u^{\frac{n+1}{n}} \left(u + \frac{\tau_y}{K} \right) du \\
&\Rightarrow - \left(\frac{KR}{\tau_w} \right)^2 \int \left(u^{\frac{2n+1}{n}} + \frac{\tau_y u^{\frac{n+1}{n}}}{K} \right) du
\end{aligned}$$

Distribute integral over both terms:

$$\begin{aligned}
&- \left(\frac{KR}{\tau_w} \right)^2 \left[\int u^{\frac{2n+1}{n}} du + \frac{\tau_y}{K} \int u^{\frac{n+1}{n}} du \right] \\
\Rightarrow &- \left(\frac{KR}{\tau_w} \right)^2 \int u^{\frac{2n+1}{n}} du - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{\tau_y}{K} \right) \int u^{\frac{n+1}{n}} du
\end{aligned}$$

Integrating each term yields:

$$- \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) u^{\frac{3n+1}{n}} - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) u^{\frac{2n+1}{n}} \quad (7.44)$$

Plugging these two terms from 7.44 back into Equation 7.43 yields:

$$\begin{aligned}
Q &= 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) \left[\int r \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} dr \right. \\
&\quad - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(\frac{r\tau_w - R\tau_y}{RK} \right)^{\frac{3n+1}{n}} \\
&\quad \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) \left(\frac{r\tau_w - R\tau_y}{RK} \right)^{\frac{2n+1}{n}} \right] \quad (7.45)
\end{aligned}$$

Now simplifying the first integral:

$$\int r \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} dr = \frac{r^2}{2} \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} \quad (7.46)$$

Plugging Term 7.46 into Equation 7.45 results in:

$$\begin{aligned}
Q = & 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) \left[\frac{r^2}{2} \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} \right. \\
& - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(\frac{r\tau_w - R\tau_y}{RK} \right)^{\frac{3n+1}{n}} \\
& \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) \left(\frac{r\tau_w - R\tau_y}{RK} \right)^{\frac{2n+1}{n}} \right] \Bigg|_{r=0}^{r=R}
\end{aligned} \tag{7.47}$$

Evaluate Equation 7.47 at $r = R$ and $r = 0$.

$$\begin{aligned}
Q(r = R) = & 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) \left[\frac{R^2}{2} \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} \right. \\
& - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{3n+1}{n}} \\
& \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{2n+1}{n}} \right]
\end{aligned} \tag{7.48}$$

$$\begin{aligned}
Q(r = 0) = & 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) \left[\frac{(0)^2}{2} \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} \right. \\
& - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(\frac{(0)\tau_w - R\tau_y}{RK} \right)^{\frac{3n+1}{n}} \\
& \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) \left(\frac{(0)\tau_w - R\tau_y}{RK} \right)^{\frac{2n+1}{n}} \right]
\end{aligned} \tag{7.49}$$

$\Rightarrow$

$$\begin{aligned}
Q(r = 0) = & 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) \left[- \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(\frac{-\tau_y}{K} \right)^{\frac{3n+1}{n}} \right. \\
& \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{2n+1} \right) \left(\frac{\tau_y}{K} \right) \left(\frac{-\tau_y}{K} \right)^{\frac{2n+1}{n}} \right]
\end{aligned} \tag{7.50}$$

Term 7.48 - term 7.50:

$$\begin{aligned}
Q(r) = 2\pi \left(\frac{KR}{\tau_w} \right) \left(\frac{n}{n+1} \right) & \left[\frac{R^2}{2} \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{n+1}{n}} \right. \\
& - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{\tau_y}{K} \right) \left(\frac{n}{n+1} \right) \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{2n+1}{n}} \\
& - \left(\frac{n}{3n+1} \right) \left(\frac{\tau_w - \tau_y}{K} \right)^{\frac{3n+1}{n}} \\
& - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{\tau_y}{K} \right) \left(\frac{n}{n+1} \right) \left(-\frac{\tau_y}{K} \right)^{\frac{2n+1}{n}} \\
& \left. - \left(\frac{KR}{\tau_w} \right)^2 \left(\frac{n}{3n+1} \right) \left(-\frac{\tau_y}{K} \right)^{\frac{3n+1}{n}} \right]
\end{aligned} \tag{7.51}$$

Recall Equation 7.24 that $\tau_w = \frac{R(\Delta P + \rho g L)}{2L}$ and plug into Equation 7.50, resulting in a final equation for flow rate give that $\tau \geq 2\tau_y^2$.

$$\begin{aligned}
Q(\Delta P) = 2\pi \left(\frac{2LKR}{\Delta P + \rho g L} \right) \left(\frac{n}{n+1} \right) & \left[\frac{R^2}{2} \left(\frac{\Delta P + \rho g L}{2LK} - \frac{\tau_y}{K} \right)^{\frac{n+1}{n}} \right. \\
& - \left(\frac{2LKR}{\Delta P + \rho g L} \right)^2 \left(\frac{\tau_y}{K} \right) \left(\frac{n}{n+1} \right) \left(\frac{\Delta P + \rho g L}{2LK} - \frac{\tau_y}{K} \right)^{\frac{2n+1}{n}} \\
& - \left(\frac{n}{3n+1} \right) \left(\frac{\Delta P + \rho g L}{2LK} - \frac{\tau_y}{K} \right)^{\frac{3n+1}{n}} \\
& - \left(\frac{2LKR}{\Delta P + \rho g L} \right)^2 \left(\frac{\tau_y}{K} \right) \left(\frac{n}{n+1} \right) \left(-\frac{\tau_y}{K} \right)^{\frac{2n+1}{n}} \\
& \left. - \left(\frac{2LKR}{\Delta P + \rho g L} \right)^2 \left(\frac{n}{3n+1} \right) \left(-\frac{\tau_y}{K} \right)^{\frac{3n+1}{n}} \right]
\end{aligned} \tag{7.52}$$

Problems with each of these assumptions

The velocity profile was derived by plugging in the Herschel Bulkley model – for $\tau > \tau_0$ the shear stress is given by $\tau = \tau_0 + K\dot{\gamma}^n$ – into the force or momentum balance of the system. In each of these cases, it steady state was assumed which yielded a constant velocity profile. The DIW system does not operate at steady-state conditions and is dependent on the input pressure, the quality of the mixing, and potential clogging that might be occurring on the inside. Variable input pressure results in a variable shear stress, one that needs to be a function of flow rate and/or velocity.

Discrete Element Modeling for Complex Fluid Flow

Newton's Equations of Motion

$$m\ddot{\mathbf{x}} = \sum \Psi_i = \Psi^{\text{tot}} \quad (7.53)$$

$$\sum \Psi_i = \Psi^{\text{grav}} + \Psi^{\text{contact}} + \Psi^{\text{fric}} + \Psi^{\text{sub}} + \Psi^{\text{bond}} + \Psi^{\text{diss}} \quad (7.54)$$

Particle-to-particle Contact Forces

$$\Psi^{\text{contact}} = \sum \Phi_{ij}^{(\text{con},n)} \quad (7.55)$$

Activation of contact force:

$$\|\mathbf{r}_i - \mathbf{r}_j\|_2 \leq R_i + R_j$$

$$\Phi_{ij}^{(\text{con},n)} \propto K_p |\mathcal{E}_{ij}|^{P_p} \mathbf{n}_{ij} A_{ij}^c \quad (7.56)$$

Overlap defined as:

$$\delta_{ij} := (R_i + R_j) - \|\mathbf{r}_i - \mathbf{r}_j\|_2 \quad (7.57)$$

“Strain-like” deformation:

$$\mathcal{E}_{ij} = \frac{\delta_{ij}}{R_i + R_j} \quad (7.58)$$

Vector normal to the force:

$$\mathbf{n}_{ij} = \frac{\mathbf{r}_j - \mathbf{r}_i}{\|\mathbf{r}_i - \mathbf{r}_j\|_2} \quad (7.59)$$

Contact area:

$$A_{ij}^c = \pi(R_i^2 - L_i^2) \quad (7.60)$$

Regularized Contact Friction Model

$$\Phi^{\text{fric}} = K^f \|\mathbf{v}_{j,\tau} - \mathbf{v}_{i,\tau}\|_2 A_{ij}^c \Delta t \tau_{ij}^c \quad (7.61)$$

Static friction threshold comparison:

If

$$(K^f \|\mathbf{v}_{j,\tau} - \mathbf{v}_{i,\tau}\|_2 A_{ij}^c \Delta t \tau_{ij}^c) < \mu_s \|\Phi_{ij}^{(\text{con},n)}\|_2 \quad (7.62)$$

Then

$$\Phi^{\text{fric}} = K^f \|\mathbf{v}_{j,\tau} - \mathbf{v}_{i,\tau}\|_2 A_{ij}^c \Delta t \tau_{ij}^c \quad (7.63)$$

where

$$\tau_{ij}^c = \frac{\mathbf{v}_{j,\tau}^c - \mathbf{v}_{i,\tau}^c}{\|\mathbf{v}_{j,\tau} - \mathbf{v}_{i,\tau}\|_2} \quad (7.64)$$

Slip model if threshold is met or exceeded:

$$\Phi^{\text{fric}} = \mu_d \|\Phi_{ij}^{(\text{con},n)}\| \tau_{ij}^c \quad (7.65)$$

Contact Substrate

$$\Phi^{\text{sub}} = \Phi_{ij}^{(\text{con},n)} \propto K_{pij} |\mathcal{E}_{ij}|^{P_p} \mathbf{n}_{ij} A_{ij}^c \quad (7.66)$$

Contact Dissipation

$$\Phi^{\text{diss}} = c^{cd} (\mathbf{v}_{nj} - \mathbf{v}_{ni}) A_{ij}^c \quad (7.67)$$

c^{cd} is the contact dissipation constant.

Particle-to-Particle Bonding Relationship

$$\sum_i \Theta_i = \Theta^{\text{grav}} + \Theta^{\text{contact}} + \Theta^{\text{fric}} + \Theta^{\text{sub}} + \Theta^{\text{bond}} + \Theta^{\text{diss}} \quad (7.68)$$

Adhesive/Attractive Force (Normal Bond):

$$\delta_{ij}^{(\text{bond},n)} = K_{ij}^{nb} |\mathcal{E}_{ij}|^{P_p} \mathbf{n}_{ij} \cdot \mathbf{A}_{ij}^c$$

Rotational Bond:

$$\delta_{ij}^{(\text{bond},r)} = K_{ij}^{nb} \|\mathbf{v}_{j,\tau} - \mathbf{v}_{i,\tau}\|_2^{A_{ij}^c} \Delta t \cdot \Theta_{ij}^c$$

Torsional Bond:

$$M_{ij}^{(\text{bond},t)} = K_{ij}^{rt} ((\Theta_i - \Theta_j) \cdot a^2 \cdot \mathbf{n}) \mathbf{A}_{ij}^c \Delta t \cdot n$$

Newton's Second Law

$$m\ddot{\mathbf{x}} = \Theta^{\text{grav}} + \Theta^{\text{contact}} + \Theta^{\text{fric}} + \Theta^{\text{sub}} + \Theta^{\text{bond}} + \Theta^{\text{diss}}$$

Discretize Differential Equation for Each Particle:

$$m\ddot{\mathbf{x}}_i = \Theta_i^{\text{grav}} + \Theta_i^{\text{contact}} + \Theta_i^{\text{fric}} + \Theta_i^{\text{sub}} + \Theta_i^{\text{bond}} + \Theta_i^{\text{diss}}$$

where

$$\Theta_i^{\text{grav}} = -m_i \mathbf{g}_3$$

$$\Theta_i^{\text{contact}} = \sum_j \delta_{ij}^{(\text{contact},n)} + \sum_j \delta_{ij}^{(\text{sub},n)}$$

$$\Theta_i^{\text{fric}} = \sum_j \delta_{ij}^{(\text{fric})} + \sum_j \delta_{iw}^{(\text{fric})}$$

Numerical Integration Methods**Forward Euler:**

For time t_j :

$$\mathbf{X}(t_{j+\Delta t}) \equiv \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \Delta t \mathbf{f}(\mathbf{X}^{(k)}, t_j)$$

Backward Euler:

For time t_j :

$$\mathbf{X}(t_{j+\Delta t}) \equiv \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \Delta t \mathbf{f}(\mathbf{X}^{(k+1)}, t_{j+\Delta t})$$

Runge-Kutta 4th Order:

For time t_j :

$$\mathbf{K}_1 = \Delta t \mathbf{f}(\mathbf{X}^{(k)})$$

$$\mathbf{K}_2 = \Delta t \mathbf{f}(\mathbf{X}^{(k)} + \mathbf{K}_1/2)$$

$$\mathbf{K}_3 = \Delta t \mathbf{f}(\mathbf{X}^{(k)} + \mathbf{K}_2/2)$$

$$\mathbf{K}_4 = \Delta t \mathbf{f}(\mathbf{X}^{(k)} + \mathbf{K}_3)$$

$$\mathbf{X}(t_{j+\Delta t}) \equiv \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \frac{\mathbf{K}_1}{6} + \frac{\mathbf{K}_2}{3} + \frac{\mathbf{K}_3}{3} + \frac{\mathbf{K}_4}{6} + \mathcal{O}(\Delta t^5)$$

Initial Conditions and Integration

Define $\mathbf{X}_i = (\mathbf{v}_i \mid \mathbf{x}_i)$:

$$\dot{\mathbf{X}}_i = \left(\frac{\mathbf{F}_i^{\text{tot}}}{m_i} \mid \mathbf{v}_i \right) \equiv \mathbf{f}(\mathbf{X}_i, t)$$

Initialize Particle Parameters at initial time t_0 :

$$\boldsymbol{\nu}_i \equiv \boldsymbol{\nu}_i, \quad \mathbf{x}_i \equiv \mathbf{x}_i, \quad \mathbf{v}_i \equiv \mathbf{v}_i, \quad \mathbf{a}_i \equiv \mathbf{a}_i, \quad \mathbf{R}_i \equiv \mathbf{R}_i, \quad m_i \equiv m_i \quad \text{for } i = 0, 1, \dots, N_p$$

Initial results

Force	K	Pp	cd
Gravity			
Contact	1	3	
Substrate/ Nozzle	1	3	
Dissipate			1e-2
Friction			1e-14
Bond			

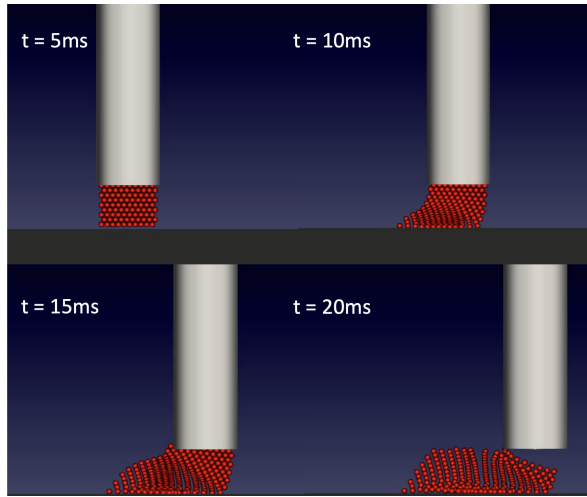


Figure 7.19: Initial results for discrete element modeling of a complex fluid being dispensed from a nozzle on to a build plate.

References

- [1] Ke Wang and Alexander W Dowling. “Bayesian optimization for chemical products and functional materials”. In: *Current Opinion in Chemical Engineering* 36 (2022), p. 100728.
- [2] Turab Lookman, Francis J Alexander, and Krishna Rajan. *Information science for materials discovery and design*. Vol. 1. Springer, 2016.
- [3] Tarek I Zohdi. “Modeling and simulation of laser processing of particulate-functionalized materials”. In: *Archives of Computational Methods in Engineering* 24 (2017), pp. 89–113.
- [4] Tarek I Zohdi. “Genetic design of solids possessing a random–particulate microstructure”. In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 361.1806 (2003), pp. 1021–1043.
- [5] TI Zohdi. “An upper bound on the particle-laden dependency of shear stresses at solid–fluid interfaces”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2211 (2018), p. 20170332.
- [6] TI Zohdi. “Rapid voxel-based digital-computation for complex microstructured media”. In: *Archives of Computational Methods in Engineering* 26.5 (2019), pp. 1379–1394.
- [7] Tarek I Zohdi and Mona Zohdi-Mofid. “Rapid machine-learning enabled design and control of precise”. In: (2023).
- [8] Brian M Howell et al. “Spatially Controlled 3D Printing of Dual-Curing Urethane Elastomers”. In: *Advanced Materials Technologies* 7.3 (2022), p. 2100700.
- [9] Ezra Feilden. “Additive manufacturing of ceramics and ceramic composites via robocasting”. In: (2017).
- [10] Brett E Kelly et al. “Volumetric additive manufacturing via tomographic reconstruction”. In: *Science* 363.6431 (2019), pp. 1075–1079.
- [11] Xiaoyu Zheng et al. “Ultralight, ultrastiff mechanical metamaterials”. In: *Science* 344.6190 (2014), pp. 1373–1377.

- [12] Xirui Peng et al. “Integrating digital light processing with direct ink writing for hybrid 3D printing of functional structures and devices”. In: *Additive Manufacturing* 40 (2021), p. 101911.
- [13] Michael Wehner et al. “An integrated design and fabrication strategy for entirely soft, autonomous robots”. In: *Nature* 536.7617 (2016), pp. 451–455.
- [14] Daniel J Shiwarski et al. “Emergence of FRESH 3D printing as a platform for advanced tissue biofabrication”. In: *APL bioengineering* 5.1 (2021), p. 010904.
- [15] MG Li, XY Tian, and XB Chen. “A brief review of dispensing-based rapid prototyping techniques in tissue scaffold fabrication: role of modeling on scaffold properties prediction”. In: *Biofabrication* 1.3 (2009), p. 032001.
- [16] Saif Khalil and Wei Sun. “Biopolymer deposition for freeform fabrication of hydrogel tissue constructs”. In: *Materials Science and Engineering: C* 27.3 (2007), pp. 469–478.
- [17] Tieshu Huang et al. “Freeze-form extrusion fabrication of ceramic parts”. In: *Virtual and Physical Prototyping* 1.2 (2006), pp. 93–100.
- [18] R Vaidyanathan et al. “The extrusion freeforming of functional ceramic prototypes”. In: *Jom* 52 (2000), pp. 34–37.
- [19] MG Li, XY Tian, and XB Chen. “Modeling of flow rate, pore size, and porosity for the dispensing-based tissue scaffolds fabrication”. In: (2009).
- [20] Jennifer A Lewis. “Direct ink writing of 3D functional materials”. In: *Advanced Functional Materials* 16.17 (2006), pp. 2193–2204.
- [21] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.
- [22] Masayuki Yano et al. *Math, Numerics, & Programming (for Mechanical Engineers)*. 2012.
- [23] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.
- [24] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.
- [25] TI Zohdi. “Electrodynamic machine-learning-enhanced fault-tolerance of robotic free-form printing of complex mixtures”. In: *Computational Mechanics* 63 (2019), pp. 913–929.

- [26] TI Zohdi. “A machine-learning framework for rapid adaptive digital-twin based fire propagation simulation in complex environments”. In: *Computer Methods in Applied Mechanics and Engineering* 363 (2020), p. 112907.
- [27] TI Zohdi. “A digital-twin and machine-learning framework for ventilation system optimization for capturing infectious disease respiratory emissions”. In: *Archives of Computational Methods in Engineering* 28.6 (2021), pp. 4317–4329.
- [28] TI Zohdi. “A digital twin framework for machine learning optimization of aerial fire fighting and pilot safety”. In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113446.
- [29] Tarek I Zohdi. “A digital-twin and machine-learning framework for the design of multiobjective agrophotovoltaic solar farms”. In: *Computational Mechanics* 68.2 (2021), pp. 357–370.
- [30] TI Zohdi. “A digital-twin and machine-learning framework for precise heat and energy management of data-centers”. In: *Computational Mechanics* 69.6 (2022), pp. 1501–1516.
- [31] Bobak Shahriari et al. “Taking the human out of the loop: A review of Bayesian optimization”. In: *Proceedings of the IEEE* 104.1 (2015), pp. 148–175.
- [32] Peter I Frazier and Jialei Wang. “Bayesian optimization for materials design”. In: *Information science for materials discovery and design* (2016), pp. 45–75.
- [33] Peter I Frazier. “Bayesian optimization”. In: *Recent advances in optimization and modeling of contemporary problems*. Informs, 2018, pp. 255–278.
- [34] Peter I Frazier. “A tutorial on Bayesian optimization”. In: *arXiv:1807.02811* (2018).
- [35] Neng Li et al. “Progress in additive manufacturing on new materials: A review”. In: *Journal of Materials Science & Technology* 35.2 (2019). Recent Advances in Additive Manufacturing of Metals and Alloys, pp. 242–269. ISSN: 1005-0302.
- [36] Pedram Parandoush and Dong Lin. “A review on additive manufacturing of polymer-fiber composites”. In: *Composite Structures* 182 (2017), pp. 36–53.
- [37] Dong-Gap Shin, Tae-Hyeong Kim, and Dae-Eun Kim. “Review of 4D printing materials and their properties”. In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 4.3 (2017), pp. 349–357.
- [38] Xin Wang et al. “3D printing of polymer matrix composites: A review and prospective”. In: *Composites Part B: Engineering* 110 (2017), pp. 442–458.

- [39] Federico Caviggioli and Elisa Ughetto. “A bibliometric analysis of the research dealing with the impact of additive manufacturing on industry, business and society”. In: *International journal of production economics* 208 (2019), pp. 254–268.
- [40] Qiyi Chen, Peng-Fei Cao, and Rigoberto C Advincula. “Mechanically robust, ultraelastic hierarchical foam with tunable properties via 3D printing”. In: *Advanced Functional Materials* 28.21 (2018), p. 1800631.
- [41] Qiangqiang Zhang et al. “Flyweight, Superelastic, Electrically Conductive, and Flame Retardant 3D Multi-Nanolayer Graphene/Ceramic Metamaterial”. In: *Advanced materials* 29.28 (2017), p. 1605506.
- [42] Mark A Skylar-Scott et al. “Voxelated soft matter via multimaterial multinozzle 3D printing”. In: *Nature* 575.7782 (2019), pp. 330–335.
- [43] Dimitri Kokkinis, Manuel Schaffner, and André R Studart. “Multimaterial magnetically assisted 3D printing of composite materials”. In: *Nature communications* 6.1 (2015), pp. 1–10.
- [44] Kaijuan Chen et al. “Fabrication of tough epoxy with shape memory effects by UV-assisted direct-ink write printing”. In: *Soft Matter* 14.10 (2018), pp. 1879–1886.
- [45] Louis Laberge Lebel et al. “UV-assisted direct-write fabrication of carbon nanotube/polymer nanocomposite microcoils”. In: *Advanced Materials* 22.5 (2010), pp. 592–596.
- [46] Gianmarco Griffini et al. “3D-printable CFR polymer composites with dual-cure sequential IPNs”. In: *Polymer* 91 (2016), pp. 174–179.
- [47] Giovanni Postiglione et al. “UV-assisted three-dimensional printing of polymer nanocomposites based on inorganic fillers”. In: *Polymer Composites* 38.8 (2017), pp. 1662–1670.
- [48] Tao Wu et al. “Additively manufacturing high-performance bismaleimide architectures with ultraviolet-assisted direct ink writing”. In: *Materials & Design* 180 (2019), p. 107947.
- [49] Qiyi Chen et al. “A Dual Approach in Direct Ink Writing of Thermally Cured Shape Memory Rubber Toughened Epoxy”. In: *ACS Applied Polymer Materials* 2.12 (2020), pp. 5492–5500.
- [50] Zhen Jiang et al. “Extrusion 3D printing of polymeric materials with advanced properties”. In: *Advanced Science* 7.17 (2020), p. 2001379.
- [51] X Mu et al. “Porous polymeric materials by 3D printing of photocurable resin”. In: *Materials Horizons* 4.3 (2017), pp. 442–449.

- [52] Nicole Kleger et al. “3D Printing: 3D Printing of Salt as a Template for Magnesium with Structured Porosity (Adv. Mater. 37/2019)”. In: *Advanced Materials* 31.37 (2019), p. 1970260.
- [53] AE Jakus et al. “3D-printing porosity: A new approach to creating elevated porosity materials and structures”. In: *Acta biomaterialia* 72 (2018), pp. 94–109.
- [54] Nazanin Farokhnia et al. “A Catheter-Deployable Soft Robotic Inflatable Basket for Enhanced Conformability to the Left Atrium of the Heart”. In: *Advanced healthcare materials* 9.4 (2020), p. 1900951.
- [55] Tuhin Ghosh, Brigitte Voit, and Niranjan Karak. “Polystyrene/thermoplastic polyurethane interpenetrating network-based nanocomposite with high-speed, thermo-responsive shape memory behavior”. In: *Polymer* 200 (2020), p. 122575.
- [56] Melissa Cadena et al. “3D bioprinting of neural tissues”. In: *Advanced healthcare materials* 10.15 (2021), p. 2001600.
- [57] Yang Yang et al. “Elastic 3D-printed hybrid polymeric scaffold improves cardiac remodeling after myocardial infarction”. In: *Advanced healthcare materials* 8.10 (2019), p. 1900065.
- [58] James E Smay, Joseph Cesarano, and Jennifer A Lewis. “Colloidal inks for directed assembly of 3-D periodic structures”. In: *Langmuir* 18.14 (2002), pp. 5429–5437.
- [59] Hong-Ru Lin et al. “Preparation of macroporous biodegradable PLGA scaffolds for cell attachment with the use of mixed salts as porogen additives”. In: *Journal of biomedical materials research* 63.3 (2002), pp. 271–279.
- [60] Mohammad Rafiee, Rouhollah D Farahani, and Daniel Therriault. “Multi-material 3D and 4D printing: a survey”. In: *Advanced Science* 7.12 (2020), p. 1902307.
- [61] Xue Wan et al. “Direct ink writing based 4D printing of materials and their applications”. In: *Advanced Science* 7.16 (2020), p. 2001000.
- [62] Alexander D Trifunac and Vadim V Krongauz. *Processes in Photoreactive Polymers*. Chapman & Hall, 1995.
- [63] Shanna Angelique Hays. *Particle finite element modeling of nanoparticle-photopolymer composites for additive manufacturing*. University of California, Berkeley, 2019.
- [64] Kentaro Taki et al. “Effect of oxygen inhibition on the kinetic constants of the UV-radical photopolymerization of diurethane dimethacrylate/photoinitiator systems”. In: *Macromolecules* 47.6 (2014), pp. 1906–1913.

- [65] Kiyoshi ARAKAWA. “Free Volume Theory of Viscosity”. In: *Journal of the Society of Materials Science, Japan* 14.139 (1965), pp. 245–248.
- [66] Michael D Goodner and Christopher N Bowman. “Development of a comprehensive free radical photopolymerization model incorporating heat and mass transfer effects in thick films”. In: *Chemical Engineering Science* 57.5 (2002), pp. 887–900.
- [67] Michael D Goodner, Hyun R Lee, and Christopher N Bowman. “Method for determining the kinetic parameters in diffusion-controlled free-radical homopolymerizations”. In: *Industrial & engineering chemistry research* 36.4 (1997), pp. 1247–1252.
- [68] Michael D Goodner and Christopher N Bowman. “Modeling primary radical termination and its effects on autoacceleration in photopolymerization kinetics”. In: *Macromolecules* 32.20 (1999), pp. 6552–6559.
- [69] Michal Januszewski and Marcin Kostur. “Accelerating numerical solution of stochastic differential equations with CUDA”. In: *Computer Physics Communications* 181.1 (2010), pp. 183–188.
- [70] Scott Rostrup and Hans De Sterck. “Parallel hyperbolic PDE simulation on clusters: Cell versus GPU”. In: *Computer Physics Communications* 181.12 (2010), pp. 2164–2179.
- [71] Oliver Hennigh et al. “NVIDIA SimNet™: An AI-accelerated multi-physics simulation framework”. In: (2021), pp. 447–461.
- [72] Tu C Le and David A Winkler. “Discovery and optimization of materials using evolutionary approaches”. In: *Chemical reviews* 116.10 (2016), pp. 6107–6132.
- [73] Youngsoo Choi et al. “Gradient-based constrained optimization using a database of linear reduced-order models”. In: *Journal of Computational Physics* 423 (2020), p. 109787. ISSN: 0021-9991.
- [74] Jichao Li, Xiaosong Du, and Joaquim RRA Martins. “Machine learning in aerodynamic shape optimization”. In: *Progress in Aerospace Sciences* 134 (2022), p. 100849.
- [75] Youngkyu Kim et al. “A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder”. In: *Journal of Computational Physics* 451 (2022), p. 110841. ISSN: 0021-9991.
- [76] David Amsallem et al. “Design optimization using hyper-reduced-order models”. In: *Structural and Multidisciplinary Optimization* 51.4 (2015), pp. 919–940.

- [77] Connor W. Coley et al. “A robotic platform for flow synthesis of organic compounds informed by AI planning”. In: *Science* 365.6453 (2019), eaax1566.
- [78] Connor W. Coley, William H. Green, and Klavs F. Jensen. “Machine Learning in Computer-Aided Synthesis Planning”. In: *Accounts of Chemical Research* 51.5 (2018), pp. 1281–1289.
- [79] TI Zohdi. “Incorporation of microfield distortion into rapid effective property design”. In: *Mathematics and Mechanics of Solids* 7.3 (2002), pp. 237–254.
- [80] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.
- [81] Nikolaus Hansen. “The CMA evolution strategy: A tutorial”. In: *arXiv:1604.00772* (2016).
- [82] John H Holland. “Genetic algorithms”. In: *Scientific american* 267.1 (1992), pp. 66–73.