Modeling and Optimization of Self-Healing Polymers

Amy Murdaugh

*Clemson University, agmurdaugh@gmail.com*

Follow this and additional works at: [https://tigerprints.clemson.edu/all_theses](https://tigerprints.clemson.edu/all_theses)

**Recommended Citation**


[https://tigerprints.clemson.edu/all_theses/3285](https://tigerprints.clemson.edu/all_theses/3285)

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
MODELING AND OPTIMIZATION OF SELF-HEALING POLYMERS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

by
Amy Murdaugh
May 2020

Accepted by:
Dr. Hyesuk Lee, Committee Chair
Dr. Margaret Wieccek
Dr. Marek Urban
Abstract

Continuous interests in developing self-healable polymers are driven by the desire to extend life spans of existing functional materials. Combining mathematical modeling and optimization approaches with experimental studies, an ultimate aim is to predict the flow and macroscopic wound closures in polymeric materials. Corresponding numerical simulations to experimental results are presented, utilizing two mathematical models that describe bulk or layer flow. An optimization routine for this self-healing process is described and implemented using two different approaches, competitive with each other with respect to accuracy. Finally a multi-layer model is simulated to show the flow profile of multiple layers rather than a single surface one.
Acknowledgments

This research is partially supported by the National Science Foundation EPSCoR through grant MADE in SC.
# Table of Contents

Title Page ................................................................. i
Abstract ................................................................. ii
Acknowledgments ..................................................... iii
List of Tables ........................................................ vi
List of Figures ........................................................ vii

1 Introduction ......................................................... 1

2 Equation and Models ............................................. 3
   2.1 Mathematical Models ......................................... 3
   2.2 Physical Domain ............................................... 4
   2.3 Nondimensionalization ....................................... 5

3 Numerical Methods ............................................... 7

4 Comparison of TFE results with literature ....................... 11

5 Results for the GTFE ............................................... 13
   5.1 Square wound ................................................... 13
   5.2 Fixed percentage wedge wound ............................... 14
   5.3 Fixed angle wedge wound .................................... 16
   5.4 V shaped wound ................................................. 16
   5.5 V shaped wound, sharper initial angles .................... 18
   5.6 Conclusions for GTFE ......................................... 18

6 Optimization of the TFE ........................................... 20
   6.1 Least-squares approach: fmincon ............................ 21
   6.2 Nonlinear system approach: fsolve .......................... 22
   6.3 Method and Errors .............................................. 22
   6.4 Selecting $K_{ref}$ and $K_{init}$ .............................. 23
   6.5 Numerical comparison of fmincon with fsolve ............. 24
   6.6 Conclusions from the tables ................................ 25
   6.7 Plots of $K(X, T_0)$ and Heights ......................... 28
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Multi-layer Numerical Modeling</td>
<td>31</td>
</tr>
<tr>
<td>7.1</td>
<td>Discussion and Results</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Appendices</td>
<td>38</td>
</tr>
<tr>
<td>A</td>
<td>Selected Code</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>56</td>
</tr>
</tbody>
</table>
List of Tables

6.1 Symmetric case (1), $K_{ref} = 1000$ ........................................ 26
6.2 Symmetric case (2) .......................................................... 26
6.3 Symmetric case (3) .......................................................... 27
6.4 Nonsymmetric case (4) ..................................................... 27

7.1 Layer Progression, $H_m = 0.004$, $[a, b] = [-0.25, -0.2], \gamma_{\eta_m} = 4.58796 \times 10^{-9} \frac{m}{s}$ 34
7.2 Layer Progression, $H_m = 0.04$, $[a, b] = [-0.25, -0.2], \gamma_{\eta_m} = 4.5878 \times 10^{-12} \frac{m}{s}$ 34
7.3 Layer Progression, $H_m = 0.004$, $[a, b] = [-0.25, -0.2], \gamma_{\eta_m} = 6 \times 10^{-10} \frac{m}{s}$ 34
7.4 Layer Progression, $H_m = 0.04$, $[a, b] = [-0.25, -0.2], \gamma_{\eta_m} = 6 \times 10^{-10} \frac{m}{s}$ 35
List of Figures

4.1 Numerical solution: One-sided stepped TFE height profile . . . . . . . . . . 12
4.2 Solution from reference [7]: One-sided stepped TFE height profile . . . . . 12
4.3 Two-sided TFE height profile; plotted in computational domain . . . . . . 12

5.1 Square initial wound. $H_m = 0.004$ . . . . . . . . . . . . . . . . . . . . . . . 14
5.2 Initial wedge wound with fixed percentage. $H_m = 0.04$ . . . . . . . . . . 15
5.3 Initial wedge wound with fixed angle. $H_m = 0.04$ . . . . . . . . . . . . 16
5.4 Initial V shaped wound with fixed percentage. $H_m = 0.04$ . . . . . . . 17
5.5 Initial V shaped wound with fixed percentage. $H_m = 0.004$ . . . . . . 17
5.6 Initial V shaped wound with fixed percentage. $H_m = 0.008$ . . . . . . 18
5.7 Initial V shaped wound with 15° angle. $H_m = 0.008$ . . . . . . . . . 19

6.1 Initial Height Profile . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
6.2 Fsolve: Target (500:1000), Initial (1000:100) . . . . . . . . . . . . . . . . . . 28
6.3 Fmincon: Target (10,100), Initial (50,100) . . . . . . . . . . . . . . . . . . . 28
6.4 Fsolve: Target (10,100), Initial (1,10) . . . . . . . . . . . . . . . . . . . . . . 29
6.5 Fmincon: Target (10,100), Initial (1,10) . . . . . . . . . . . . . . . . . . . . 29
6.6 Fsolve: Target (10,100), Initial (50,100) . . . . . . . . . . . . . . . . . . . . 29
6.7 Fsolve: Target (10,100), Initial (50,500) . . . . . . . . . . . . . . . . . . . . 29
6.8 Fmincon: Target (1,100), Initial (5,100) . . . . . . . . . . . . . . . . . . . . 29
6.9 Fsolve Target (1,100), Initial (5,100) . . . . . . . . . . . . . . . . . . . . . . 29
6.10 Fsolve: Target (1,100), Initial (1,10) . . . . . . . . . . . . . . . . . . . . . . 30
6.11 Fsolve: Target(1,100), Initial (10,1) . . . . . . . . . . . . . . . . . . . . . . . 30
6.12 $H_{opt}$ and $\overline{H}$ profiles; Fsolve . . . . . . . . . . . . . . . . . . . . . 30
6.13 $H_{opt}$ and $\overline{H}$ profiles; Fsolve . . . . . . . . . . . . . . . . . . . . . . 30

7.1 Multi-layer model for $H_m = 0.04$ . . . . . . . . . . . . . . . . . . . . . . . 35
7.2 Multi-layer model for $H_m = 0.004$ . . . . . . . . . . . . . . . . . . . . . . . 35
Chapter 1

Introduction

Continuous interests in developing self-healable polymers are driven by the desire to extend life spans of existing functional materials [3]. The objective of these studies is to develop mathematical models that will elucidate the origin of self-healing mechanisms in polymeric materials. Specifics goals are to identify time-dependent relationships governing the kinetics of self-healing and correlate to in-situ shape variations of macroscopic damage during self-healing. Combining mathematical modeling and optimization approaches with experimental studies, an ultimate aim is to predict the flow and macroscopic wound closures in polymeric materials.

Previous work shows the derivation of appropriate mathematical models from physical principles [2],[7]. These have been utilized to simulate the leveling behavior of a stepped polymer film. In particular, the temperature dependence of the polymer’s mobility is shown, with only a thin surface layer retaining mobility when the polymer temperature is below the glass transition temperature [2]. Recently, experimental studies have been performed that demonstrate the healing process in a cut polymer film and identify surface energy/tension as a driving force in the process [3].

The first aim of the work presented here is to provide the corresponding numerical simulations to these experimental results, utilizing both of the mathematical models given in [2]. The second aim of this work is to describe and implement an optimization routine
for this self-healing process. In [5], an optimal control problem that focuses on matching a
target profile has been presented, and the conceptual framework of the control problem is
followed here. Lastly, the third aim of this work is to describe and simulate a multi-layer
self-healing model. To the author’s knowledge, no previous mathematical work has been
done in this direction. We extend the single layer case to a multi-layer model using the
physical properties of the layer rather than the bulk material.

In accordance with these aims, the thesis contains three parts. In the first part,
Chapters 2-5, we study two mathematical models to simulate in 1D the self-healing process
of a polymeric material. We present the numerical methods used to solve both models and
then display results for each, studying the dependence of the self-healing progression on
different initial shapes and depths of wounds. Next, in Chapter 6, we consider optimizing
this modeling process by matching a target profile at a specific point in time. Results are
shown using two pre-programmed algorithms in Matlab, each competitive to the other with
respect to accuracy. Finally, in Chapter 7, we show an extension of the single layer models
to a multi-layer case. The Appendix contains several samples of the Matlab code used in
the implementation and optimization of the models.
Chapter 2

Equation and Models

In this chapter, we present two mathematical models used to simulate the self-healing process of the polymer. Physical parameters and properties are given, as well as an explanation of the procedure of transferring between a physical, real-world problem and a numerical, dimensionless one.

2.1 Mathematical Models

Let \( h(x,t) \) be a function of space and time; \( h : [a, b] \times [0, T_f] \rightarrow \mathbb{R} \), with \([a, b] \in \mathbb{R} \) and \([0, T_f] \in \mathbb{R}_+ \). In order to describe the progression of wound closure in a polymeric material, let the height function \( h \) describe the depth of the wound over the horizontal dimension \( x \) from time 0 to a final time \( T_f \). Throughout this paper we denote:

\[
\begin{align*}
\gamma &= \text{surface tension} \\
\eta_b &= \text{bulk viscosity} \\
\eta_m &= \text{layer viscosity} \\
h_m &= \text{thickness of a mobile thin surface layer}
\end{align*}
\]
We utilize two one-dimensional, fourth order model equations. The thin film equation (TFE):

\[
\frac{\partial h}{\partial t} + \gamma \frac{\partial}{\partial x} \left( h^3 \frac{\partial^3 h}{\partial x} \right) = 0 \tag{2.5}
\]

describes flow of a homogeneous viscous film where the polymer temperature is above the glass transition temperature \( T_g \) [7]. The glass transition temperature is the range of temperatures at which a polymer transitions from a less viscous, glassy state into a more viscous or rubbery state. Secondly, the glassy thin film equation (GTFE):

\[
\frac{\partial h}{\partial t} + \gamma h^3 m^3 \left( \frac{\partial^4 h}{\partial x} \right) = 0 \tag{2.6}
\]

describes flow in situations where the polymer temperature is below the glass transition temperature \( T_g \). Here, the bulk of the film is unable to move, so we consider the film as a thin mobile layer (with thickness described by \( h_m \)) on top of an immobile layer [2].

### 2.2 Physical Domain

The physical domain is the size of the polymeric material in which the wound is made. The domain size given is a 0.1 mm square, with layer thickness \( h_m = 2 \cdot 10^{-5} mm \). Note that \( h_m \) is 0.02% of the domain thickness. The shape of the initial damage will be discussed in Chapter 5, but its size is roughly 10% of the domain size, (for example, a 0.01 mm square). Recalling the parameters defined in (2.1)-(2.3), the physical parameters used in the GTFE simulations are [3]:

\[
\gamma = 30 \text{ mN m}^{-1} \tag{2.7}
\]

\[
\eta_m = 50 \text{ MPa.s} \tag{2.8}
\]

\[
\eta_b = 100 \text{ MPa.s} \tag{2.9}
\]
2.3 Nondimensionalization

In order to numerically solve Eqs. (2.5) and (2.6), we first nondimensionalize them so that there is no reference to units. First, define:

\[ H = \frac{h}{h_0} \]  \hspace{1cm} (2.10)
\[ H_m = \frac{h_m}{h_0} \]  \hspace{1cm} (2.11)
\[ X = \frac{x}{h_0} \]  \hspace{1cm} (2.12)
\[ T = \frac{\gamma t}{3\eta h_0} \]  \hspace{1cm} (2.13)

with \( h_0 \) a reference height scale. Note that

\[ \eta = \begin{cases} 
\eta_m, & \text{for the GTFE} \\
\eta_b, & \text{for the TFE} 
\end{cases} \]

Now, \( H(X,T) \) is the dimensionless height function we will use in computations. It corresponds exactly to \( h(x,t) \) except for scaling. The reference scale \( h_0 \) will allow us to translate from the physical domain involving variables \( h, x, t \) to the computational domain involving variables \( H, X, T \). The reference scale \( h_0 \) must be calculated uniquely for each computational domain; this process is elaborated in Chapter 3. Substituting Eqs. (2.10), (2.12) and (2.13) into Eq. (2.5), the nondimensionalized TFE becomes:

\[ \partial_T H + \partial_X (H^3 \partial_X^2 H) = 0 \]  \hspace{1cm} (2.14)

Now, we can work numerically without units, and then translate the unitless results back to their real-world counterparts using the reference scale \( h_0 \).
Similarly, we nondimensionalize the GTFE in Eq. (2.6) to obtain:

$$\partial_T H + (H^2 m \partial_X^4 H) = 0$$  \hspace{1cm} (2.15)

In the next chapter, we describe the computational procedure used to solve Eqs. (2.14) and (2.15).
Chapter 3

Numerical Methods

Since Eqs. (2.14) and (2.15) are still continuous, we need to discretize them in order to compute the numerical solutions to Eqs. (2.5) and (2.6). To this end, a finite difference approach is applied to discretize Eqs. (2.14) and (2.15) with respect to the spatial variable following the method from [7]. As stated in [7], this method ensures volume conservation and positivity of the height profile $H(X,T)$.

First, we discretize the $X$ axis of the computational domain into $M$ grid points, centered around $X = 0$. The dimensionless step size $\Delta X$ is found by $\Delta X = \frac{X_{\text{left}} - X_{\text{right}}}{M - 1}$, with $X_{\text{left}}$ and $X_{\text{right}}$ being the leftmost and rightmost grid points of the $X$ axis and $M$ the number of grid points. Recall that $h(x,t)$ represents the height with units and $H(X,T)$ is the nondimensionalized height. We will let the nondimensionalized, discretized height values be represented by $\mathcal{H}$. Thus we calculate $H(X,T)$ at each grid point $X_i$ for $i \in [1, M]$ by:

$$\mathcal{H}_i = H \left( (i - \frac{M + 1}{2}) \Delta X, T \right)$$

(3.1)
Now we define the rates:

\[ H_{X,i} = \frac{H_{i+1} - H_i}{\Delta X} \]  
(3.2)

\[ H_{X,i} = H_{X,i-1} \]  
(3.3)

\[ H_{XX,i} = \frac{H_{XX,i+1} - H_{XX,i}}{\Delta X} \]  
(3.4)

\[ H_{XX,i} = \frac{H_{XX,i} - H_{XX,i-1}}{\Delta X} \]  
(3.5)

We introduce the auxiliary function \( A_i \) for \( i \in [1, M] \) as:

\[ A_i = 2 \frac{H_i^2 - H_{i-1}^2}{H_i - H_{i-1}} \]  
(3.6)

Note that \( A_i \approx H_i^3 \) when \( \Delta X \) is small. Thus, we describe an approximation of Eq. (2.14) that is continuous in time and discrete in space:

\[ \frac{dH_i}{dT} = \frac{A_i H_{XX,i} - A_{i+1} H_{XX,i+1}}{\Delta X} \]  
(3.7)

for \( i \in [1, M - 1] \).

At each point \( X_i \), we now have the first order differential equation (3.7), continuous with respect to \( T \). A similar finite difference method is followed to discretize the GTFE (2.15). Since the GTFE is linear, the auxiliary function is no longer needed to approximate the nonlinear term of Eq. (2.14), and we include the constant \( H_{3M}^3 \) in its place.

Different initial conditions used in solving Eq. (3.7) are presented in Chapters 4 and 5. The most physically accurate initial conditions are those designed to ensure mass
conservation, shown in Figure 6.1. The boundary conditions for \( T > 0 \) are also from [7]:

\[
A_1 = H_1^3 
\]

(3.8)

\[
\mathcal{H}_{X,1} = 0 
\]

(3.9)

\[
\mathcal{H}_{XX,M} = 0 
\]

(3.10)

\[
\mathcal{H}_{XXX,1} = 0 
\]

(3.11)

\[
\frac{dH_M}{dT} = \frac{A_M H_{XXX,M}}{\Delta X} 
\]

(3.12)

The computational domain refers to the nondimensional time region and 1D spatial region over which we numerically solve the equation. We employ a variety of different nondimensionalized square domain sizes, including 2 by 2, 10 by 10, 20 by 20, and 40 by 40. The step sizes \( \Delta X \) and \( \Delta T \) of the spatial and time variables, respectively, vary from domain to domain. As discussed in [7], due to the general orders of the TFE, we should use step sizes roughly consistent with \( \Delta T \approx \Delta X^4 \). For larger domains, we generally keep \( \Delta X \) around 0.1 or 0.2 to reduce the computational time, whereas in smaller domains we may decrease \( \Delta X \) to as small as 0.025. As discussed in Chapter 7, we can also use an adaptive approach that varies the value of \( \Delta X \) based on the location in the domain, decreasing the size only in areas where we see significant change happening. \( \Delta T \) tends to be around \( 10^{-5} \) in the larger domains. In some trials, as we utilize a smaller \( \Delta X \), we use a very small time step as low as \( \Delta T = 10^{-9} \).

The solutions to Eq. (2.15) presented in Chapter 5 are computed using two different computational domain sizes: a nondimensionalized 20 by 20 square and a nondimensionalized 40 by 40 square. In each domain, the reference scale \( h_0 \) is computed to convert between the physical and computational domains.

Specifically, \( h_0 \) is found as follows. We set the nondimensional height, \( H \), to be the true height scaled by \( h_0 \) as seen in Eq. (2.10). Recalling Chapter 2.2, since the physical domain is a 0.1 mm square, in the 20 by 20 computational domain, \( h_0 = \frac{0.1 \text{ mm}}{20} = 0.005 \text{ mm} \). We can then find the nondimensionalized layer thickness using Eq. (2.11),
\[ H_m = \frac{2 \times 10^{-5} \text{ mm}}{0.005 \text{ mm}} = 0.004. \] For the 40 by 40 domain, \( h_0 = 0.0025 \) and \( H_m = 0.008. \) These nondimensional \( H_m \) values are used in solving Eq. (2.15) over a specific computational domain.

We also compute the correspondence between the physical time and the number of steps of the solver. The nondimensionalized time \( T \) can be represented as \( T = j \Delta T \), with \( j \) the number of steps of the solver and \( \Delta T \) the nondimensionalized time step. Combining this with Eq. (2.13), we can solve for the physical time \( t \) given the number of time steps \( j \). In the 20 by 20 domain, using the parameter values presented in Chapter 2.2 along with \( \Delta T = 10^{-5} \), we can then see that one day in physical time corresponds to 345,600 steps of the solver. In the 40 by 40 domain, similarly, one day in physical time corresponds to 691,200 steps of the solver.

As seen in Chapter 5, the 40 by 40 square for the computational domain works best in terms of allowing for a larger time step \( \Delta T \) in solving the differential equation, a smaller reference scale \( h_0 \), and thereby a larger constant \( H_m \) used in solving the GTFE. The 40 by 40 domain seems to be the best balance between increasing the domain size enough to have a larger \( \Delta T \) and smaller \( h_0 \), while keeping the domain size small enough to be computationally reasonable at a relatively small \( \Delta X \).

Lastly, we write code to solve Eq. (3.7) implementing the fourth order Runge Kutta method [1]. The reader is referred to the Appendix for the code implementing these methods.

In the next two chapters, we discuss the results obtained from solving the TFE (2.14) and the GTFE (2.15).
Chapter 4

Comparison of TFE results with literature

For the first trials of the code for the TFE, we solve the problem considering a one-sided initial wound to try to produce similar results to the literature [7]. The one-sided wound is represented by a sharp step at the midpoint of the domain, while in later sections we use a two-sided wound that appears rectangular or parabolic at the initial time. Figures 4.1 and 4.2 display the height profile $H - H_1$ at various time steps, where $H(X,T)$ is the computed height profile and $H_1$ is a constant representing the fixed initial height on the left hand side of the wound. The time step $\Delta T$ used here is $10^{-6}$, and $\Delta X = 0.1$. Figure 4.1 shows the numerical results and Figure 4.2 shows the corresponding results from the literature [7]. The obtained plots show perfect correspondence to literature at each of the five nondimensionalized times, $T$.

Considering the two-sided initial wound, the nondimensionalized TFE (2.14) exhibits the self-healing process well by rising quickly and smoothly, as seen in Figure 4.3. This example is computed over a 2 by 2 computational domain, with the time step $\Delta T = 10^{-10}$, and the spatial increment $\Delta X = 0.02$. Using an initial square wound, the height profile rises and begins evening out smoothly as expected. Compared to the solutions of the GTFE transferred back to the physical domain shown in Figure 5.1, the TFE results shown in Fig-
Figure 4.1
Numerical solution: One-sided stepped TFE height profile

Figure 4.2
Solution from reference [7]: One-sided stepped TFE height profile

Figure 4.3
Two-sided TFE height profile; plotted in computational domain

Figure 4.3 only briefly exhibit the sharp peaks we see in Figure 5.1. Even though Figure 4.3 does show the wound penetrating deeper into the domain briefly, the profile quickly begins rising and flattens out evenly, as expected based on the experimental profiles [3]. We return to the TFE for the purpose of optimization in Chapter 6 using a two-sided initial wound.
Chapter 5

Results for the GTFE

In order to compare the obtained numerical results with the experimental results, we turn to solving the GTFE (2.15), implementing the physical domain and parameters given in [3]. We experiment with 4 different shapes of initial wounds. The results are mostly discussed for the 20 by 20 computational domain, as elaborated on in Chapter 3, and displayed in Figures 5.1-5.7. Each figure shows the height profile over time for a different initial wound shape. The layer thickness $H_m$ has the largest impact on the model, and in several figures, we present results where $H_m$ is increased artificially to speed up the progression.

5.1 Square wound

The first wound shape is the most basic, a square wound that is 10% of the domain size. So, for the 20 by 20 computational domain, we have a 2 by 2 square wound. For this initial condition, we run the model for 102 million time steps, which represents 295 days in real time, using the conversion given in Chapter 3. As seen in Figure 5.1, the main characteristic of this initial wound is the development of very thin points, going deeper into the domain, at the corners of the square. This may be an artifact of the numerical method. At the end of the 295 days, we only see the points getting deeper and coming to a sharper
point, but the actual wound has not filled in or risen any above the initial depth.

Compare this to Figure 4.3, in which the TFE begins to exhibit those points also but soon recovers. It is possible that the GTFE would echo the same behavior here if run for a sufficient number of time steps, since including $H_m$ in the equation significantly slows down the progression of the wound healing. However, even if the GTFE were to begin rising here, it would not be in a computationally or physically reasonable amount of time. Thus, we move on to considering other initial wound shapes.

![GTTE Initial Square: Hm = 0.004](image)

Figure 5.1
Square initial wound. $H_m = 0.004$

### 5.2 Fixed percentage wedge wound

The next two wounds are more realistic in shape and align more closely with the physical information given. Both are a wedge shape, angled in on the sides and meeting up with a 850 nm wide flat part at the bottom; i.e., a V shape wound with the bottom point cut off. For this version, we assume the wedge still takes up 10% of the domain, i.e., it has a maximum width and depth of 0.01 mm. The angle of the wound comes out to be about 50 degrees. This wound is tried on the 20 by 20 computational domain, setting the parameter $H_m$ to 0.04, 0.02, and 0.008. Recall that the value of $H_m$ corresponding to this domain size is 0.004.

For $H_m = 0.04$, we see in Figure 5.2 that the wound fills in about a third of the
way by approximately 102 million time steps. If we consider the transition back to physical times (Eq. 2.13) in terms of the $h_0 = 0.005$ mm necessitated by the 20 by 20 computational domain, 102 million time steps corresponds to 295 days in real time. However, since we artificially increased $H_m$ from 0.004 to 0.04, the value of $h_0$ corresponding to $H_m = 0.04$ would be $h_0 = 0.0005$ mm. Using this value to convert to real time, 102 million time steps corresponds to 29.5 days in real time.

Because we no longer have the sharp corners of the square wound conditions, the phenomenon of developing the very thin, sharp peaks that go deeper into the domain is not exhibited here. We compare Figure 5.2 with Figures 5.3 and 5.4 to see the effect that three different initial conditions have on the healing process, all for up to 102 million time steps with $H_m = 0.04$.

As $H_m$ is decreased, the rising happens proportionally slower. By about 100 million time steps with $H_m = 0.02$, the wound has only risen roughly a tenth of the way. And using $H_m = 0.008$, which is still twice as large as the true value $H_m = 0.004$ that corresponds to the domain size, there is barely any visible progress after several hundred million steps.

![Initial wedge wound with fixed percentage. $H_m = 0.04$](image)

Figure 5.2
Initial wedge wound with fixed percentage. $H_m = 0.04$
5.3 Fixed angle wedge wound

This initial wound is the same shape as the previous, with the only change being that instead of fixing the percentage of the domain it takes up, we fix the angle to be $35^\circ$. This initial condition shows almost precisely the same behavior in the rate of healing to the previous for each respective value of $H_m$, with the only difference being the initial depth of the wound (Figure 5.3). Again, decreasing $H_m$ slows the process down significantly.

![Initial wedge wound with fixed angle: $H_m = 0.04$](image)

Figure 5.3
Initial wedge wound with fixed angle. $H_m = 0.04$

5.4 V shaped wound

The last initial condition we present exhibits the quickest healing. Instead of cutting off the bottom of the V shape to create a wedge, we utilize a true V shape whose maximum depth and width are 10% of the domain. Compared to the wedge shape initial wounds for respective values of $H_m = 0.04$, the V shape rises between 1.5 and 4 times faster than either of the wedge shapes. Thus, whereas we must artificially increase $H_m$ for the two wedge shape wounds to see any visible progression, we can run the V shaped initial conditions for the true value of $H_m = 0.004$ that corresponds to the computational domain. Even after 1.78 billion time steps (5165 days in real time), we only see about a 4% rise, which implies that both of the wedge shaped initial wounds would take even longer to show visible change.
using $H_m = 0.004$. Figures 5.4 and 5.5 show the results for the V shaped wound with $H_m = 0.04$ and $H_m = 0.004$.

![Initial V shape with fixed percent: $H_m = 0.04$](image1)

**Figure 5.4**
Initial V shaped wound with fixed percentage. $H_m = 0.04$

![Initial V shape with fixed percent: $H_m = 0.004$](image2)

**Figure 5.5**
Initial V shaped wound with fixed percentage. $H_m = 0.004$

However, using the 40 by 40 computational domain with corresponding $H_m = 0.008$, we see a 5.86% rise by 1.78 billion time steps in Figure 5.6. The 1.78 billion time steps corresponds to about 5165 days for the 20 by 20 domain, and about 2582 days for the 40 by 40 domain. So, increasing the domain size does provide a more significant change in the height profile in even fewer days, but it still does not correspond to the time frame that
experimental simulations portray.

We note that while $H_m$, and thus the domain size, have the largest affect on the actual speed of healing in the nondimensional setting, the factor that affects the transition back to real time the most is the ratio $\frac{\gamma}{\eta}$.

Lastly, we try decreasing the initial angle of the V shape. The actual V shape has an angle of about 53.13°. Decreasing the angle to 15°, we see enormous improvement in the rate of healing. Using the 40 by 40 domain with $H_m = 0.008$, we see in Figures 5.6 and 5.7 that while the original V shape has only filled in about 5.86% by 1.78 billion steps, this new shape has filled in about 20% by only 153 million steps. Figure 5.7 shows these results on the 40 by 40 computational domain.

5.5 V shaped wound, sharper initial angles

Lastly, we try decreasing the initial angle of the V shape. The actual V shape has an angle of about 53.13°. Decreasing the angle to 15°, we see enormous improvement in the rate of healing. Using the 40 by 40 domain with $H_m = 0.008$, we see in Figures 5.6 and 5.7 that while the original V shape has only filled in about 5.86% by 1.78 billion steps, this new shape has filled in about 20% by only 153 million steps. Figure 5.7 shows these results on the 40 by 40 computational domain.

5.6 Conclusions for GTFE

The presented work shows the relative success of each of the types of initial wounds. Eliminating the corners of the square wound shows great improvement. The true V shape outperforms the others in terms of the speed and smoothness of the healing process. As
expected, a narrower angle increases the rate of healing. The main hindrance to the self-healing process is the small size of the \( H_m \) parameter necessitated by the domain size. Possible fixes to this include deriving the domain size based on the desired value of \( H_m \), and then using an adaptive discretization method to choose the grid points \( X_i \) at which we evaluate the height. Combining this with a different choice in the starting angle may speed up the progression, as would implementing the physically realistic initial conditions shown in Figure 6.1.

Using a smaller value for \( \Delta X \), whether across the whole domain or only in localized regions, will also improve performance. One trial has been run using adaptive step sizes for the fixed percentage wedge wound with \( H_m = 0.04 \), and the wound has healed about 17% more by the same \( T = 255 \) with the adaptive \( \Delta X \) than with the nonadaptive. Lastly, we note the dependence of the time conversion on the reference scale \( h_0 \) and on the ratio \( \frac{\gamma}{\eta} \). Finding the most accurate parameters \( \gamma \) and \( \eta \) and ensuring that we use a domain size that allows the true \( H_m \) value to be employed will improve the precision of the time conversion.
Chapter 6

Optimization of the TFE

We now move from pure modeling of the wound closure progression to optimization of the process. We formulate the optimization problem to find a control $K(X,T)$ which produces a height profile that matches a target height $\bar{H}(X,T)$ at a fixed time $T_0$ [5].

For the purpose of optimization, we use the TFE (2.14) instead of the GTFE (2.15), as the nondimensionalized models for this equation have been showing the change over time more quickly, and thus are better candidates for optimization. We change the right hand side (RHS) from $0$ to $K(X,T)$, where $K : [a,b] \times [0,T_f] \rightarrow \mathbb{R}$. Here, $K$ represents an external force. Thus, a given height profile at any fixed time $T_0$ can be obtained by solving the TFE as a function of the chosen $K(X,T_0)$. The goal of the optimization problem is to find a control $K(X,T_0)$ that produces a height profile $H(X,T)$ which matches a target height profile $\bar{H}(X,T)$ at a specific time $T_0$ for all $X$ in the domain.

The updated TFE for optimization reads:

$$\frac{\partial H}{\partial T} + \frac{\partial}{\partial X} \left( H^3 \frac{\partial^3 H}{\partial X^3} \right) = K(X,T) \quad (6.1)$$

The corresponding discretized equation is given by

$$\frac{dH_i}{dT} = \frac{A_i H_{XX,i} - A_{i+1} H_{XX,i+1}}{\Delta X} + K(X_i, T) \quad (6.2)$$
for $i \in [1, M - 1]$, with the same boundary conditions presented in Chapter 3. The computational domain we solve over is approximately a 2 by 2 square. In the X direction, we discretize the interval $[-1, 1]$ into $M = 101$ equally spaced grid points $X_i$. Thus, in practice, $K(X, T_0), H(X, T_0), \overline{P}(X, T_0) \in \mathbb{R}^{101}$, where the height vector $H(X, T_0)$ contains the height at each point $X_i$, which is produced by solving Eq.(6.2) for the corresponding vector $K(X, T_0)$.

The shape of the initial profile is given by Figure 6.1, where the lowest point of the wound has a height of about 1.5 and the highest peaks are a height of roughly 2.3 in the computational domain. This general shape is used to ensure mass conservation.

We pose the optimization problem in terms of constrained least-squares optimization and in terms of solving a system of nonlinear equations, and MATLAB routines are implemented to solve each formulation.

6.1 Least-squares approach: \texttt{fmincon}

In the first method, we consider the optimization problem in terms of least squares minimization. This approach uses MATLAB’s \texttt{fmincon} [6] to solve

$$
\min_K \| C \left( H(X, T_0) - \overline{P}(X, T_0) \right) \|_2^2
$$

subject to Eq. (6.1)

where $\overline{P}(X, T_0)$ is the target height at time $T_0$ produced by a reference $K(X, T_0)$, and $C$ is a scaling constant used as a weight of the objective functional. To solve the discretized problem, we provide \texttt{fmincon} with an initial guess for $K$, as described in Chapter 6.4.
6.2 Nonlinear system approach: \textit{fsolve}

In the second method, the problem is considered in terms of root finding instead of minimization, using \textit{fsolve} \cite{6} to solve (6.3) as a system of equations. Considered as a continuous problem, we want the height profile $H(X, T_0)$ to equal the target height $\overline{H}(X, T_0)$ for all $X \in [a, b]$ at a fixed time $T_0$. However, to express this problem as a system of equations, it must be presented discretely, with one equation for every point $X_i$ in the discretized domain. Thus, for $F : \mathbb{R}^{101} \rightarrow \mathbb{R}^{101}$, the MATLAB function \textit{fsolve} returns a $K$ that satisfies the vector function $F(K) = 0$, with $F = [f_1, f_2, ..., f_{101}]'$. In effect, we solve the system of equations:

$$F(K) = 0,$$

where $f_i = C \left(H(X_i, T_0) - \overline{H}(X_i, T_0)\right)$, \hspace{1cm} (6.4)

subject to Eq. (6.2)

for $i = 1, ..., 101$, with $C$ the same scaling constant used as a weight. We implement (6.4) in \textit{fsolve} in MATLAB, again providing an initial guess for the vector $K$.

6.3 Method and Errors

For both approaches, we follow the process described below. In the following, $K$ and $H$ denote vectors with entries $K_i = K(X_i, T_0)$ and $H_i = H(X_i, T_0)$. Before solving, we pick a final time $T_0$ and a reference RHS function $K_{ref}(X, T_0)$. We then solve Eq. (6.2) using $K_{ref}$ to produce the target height $\overline{H}$. Various selections of $K_{ref}$ and $K_{init}$ are discussed in Chapter 6.4. We then solve Eq. (6.2) using $K_{init}$ to create $H_{init}$. Next, we pass $K_{init}$ into either \textit{fmincon} or \textit{fsolve} as an initial guess, and the solver produces an optimal solution $K_{opt}$. Lastly, we use $K_{opt}$ to solve Eq. (6.2) for the solution $H_{opt}$. The goal of this optimization process is to reduce $||H_{opt} - \overline{H}||_2$, where $|| \cdot ||_2$ is the $l_2$ vector norm.

We measure three errors in this process using the $l_2$ norm. The first is $||K_{opt} -$
which shows the difference between the optimal control $K$ that produces the target height and the optimal control $K$ that produces the optimal height. As seen in Tables 6.1-6.4, this error is often very large, demonstrating that there are multiple solutions $K$ to the problems presented in (6.3) and (6.4).

The second and third errors are between the height vectors: $||H_{\text{init}} - \overline{H}||_2$ and $||H_{\text{opt}} - \overline{H}||_2$. Ideally, the error between the optimal and target heights is noticeably less than the error between the initial and target heights, showing that the optimization process did indeed produce a closer match in height profiles.

Multiplying by a larger constant $C$ in problems (6.3) and (6.4) reduces error. Notice that these problems both include the constant $C$ to speed up results, but the errors we calculate in Tables 6.1-6.4 do not include $C$, since we want to examine the true difference between the target and output height vectors.

### 6.4 Selecting $K_{\text{ref}}$ and $K_{\text{init}}$

We utilize both symmetric and nonsymmetric RHS functions in the optimization schemes. In the symmetric case, we choose the vector $K$ as containing either one, two, or three distinct values. The symmetry of the vector $K$ is reminiscent of the symmetry in the initial wound shown in Figure 6.1. Thus, for the case with two distinct values, we use one value $k_0$ towards the ends of the horizontal domain where the initial conditions were flat, and a second value $k_1$ toward the middle of the domain where the actual wound is located. Recall that $K$ is a vector with 101 components. The following four cases are used in producing the reference vector $K_{\text{ref}}$ and the initial vector $K_{\text{init}}$. We list the four cases below, with $X_i$ the discrete grid points:

1. Symmetric; constant value. The value is displayed as $k_0$ in the tables.

   $$K(X_i) = k_0, \text{ for } X_i \in [-1, 1], \quad i = 1:101$$
(2) Symmetric; 2 distinct values. The values are displayed as \((k_0, k_1)\) in the tables.

\[
K(X_i) = \begin{cases} 
  k_0, & \text{for } X_i \in [-1, -0.66] \cup [0.66, 1], \quad i = 1:18, \, 84:101 \\
  k_1, & \text{for } X_i \in [-0.64, 0.64], \quad i = 19:83 
\end{cases}
\]

(3) Symmetric; 3 distinct values. The values are displayed as \((k_0, k_1, k_2)\) in the tables.

\[
K(X_i) = \begin{cases} 
  k_0, & \text{for } X_i \in [-1, -0.66] \cup [0.66, 1], \quad i = 1:18, \, 84:101 \\
  k_1, & \text{for } X_i \in [-0.64, -0.14] \cup [0.14, 0.64], \quad i = 19:44, \, 58:83 \\
  k_2, & \text{for } X_i \in [-0.12, 0.12], \quad i = 45:57 
\end{cases}
\]

(4) Nonsymmetric; increasing from \(k_0\) to \(k_1\) in 5 intervals. The values are displayed as \((k_0 : k_1)\) in the tables.

\[
K(X_i) = \begin{cases} 
  k_0, & \text{for } X_i \in [-1, -0.66], \quad i = 1:18, \\
  k_0 + \frac{k_1-k_0}{4}, & \text{for } X_i \in [-0.64, -0.14], \quad i = 19:44 \\
  k_0 + \frac{k_1-k_0}{2}, & \text{for } X_i \in [-0.12, 0.12], \quad i = 45:57 \\
  k_0 + \frac{3(k_1-k_0)}{4}, & \text{for } X_i \in [0.14, 0.64], \quad i = 58:83 \\
  k_1, & \text{for } X_i \in [0.66, 1], \quad i = 84:101 
\end{cases}
\]

6.5 Numerical comparison of \textit{fmincon} with \textit{fsolve}

In the following tables, we display results from using Matlab’s \textit{fsolve} to solve (6.4), and compare them with results from using Matlab’s \textit{fmincon} to solve (6.3). For \textit{fmincon}, the maximum number of function evaluations is set to 50,000. For all these results, we fix \(\Delta T = 10^{-9}\) and take \(j = 1000\) time steps. Thus \(T_0 = j\Delta T = 10^{-6}\). Tolerances for \textit{fsolve} and \textit{fmincon} are set to \(10^{-8}\) and \(10^{-9}\) respectively for the results presented here. In general, \textit{fmincon} takes longer to get to the same level of accuracy as \textit{fsolve}, but the two methods produce results that are competitive with respect to the approximation error \(||H_{opt} - \overline{H}|||\).

Tables 6.1-6.4 display the 3 errors computed in \(l_2\) norm: between \(K_{opt}\) and \(K_{ref}\),
between $H_{\text{init}}$ and $\overline{H}$, and between $H_{\text{opt}}$ and $\overline{H}$, as described in Chapter 6.3. The corresponding case described in Chapter 6.4 for choosing $K_{\text{ref}}$ and $K_{\text{init}}$ is stated in the table’s title. We observe that $||H_{\text{opt}} - \overline{H}|| \leq ||H_{\text{init}} - \overline{H}||$, which implies the optimization routine is indeed finding a height profile closer to the target than the height profile with which we started. Additionally, we note that $||K_{\text{opt}} - K_{\text{ref}}||$ is often quite large, but the resulting error in the heights is small, leading us to believe that an optimal solution for the control $K$ is not unique.

6.6 Conclusions from the tables

The main conclusions we draw are as follows: using a larger scalar, such as $C = 1000$, improves convergence in the height profiles. The fmincon approach tends to work better than the fsolve approach as far as approximating the expected target RHS and height, but it takes much longer to compute. Also, the nonsymmetric cases seem to have a tendency to converge better to the target height. There does not seem to be an explicit pattern for which combinations of initial guesses work best, since the variation is large from sample to sample. The range of errors in heights tend to be between $10^{-4}$ and $10^{-9}$. 
**Table 6.1: Symmetric case (1), $K_{ref} = 1000$**

| $C$ | $K_{init}$ | $||H_{init} - \overline{H}||_2$ | $fsolve$ | $fmincon$ | $fsolve$ | $fmincon$ |
|-----|-------------|-------------------------------|----------|-----------|----------|-----------|
|     |             | $||H_{opt} - \overline{H}||_2$ | $||H_{opt} - \overline{H}||_2$ | $||K_{opt} - K_{ref}||$ | $||K_{opt} - K_{ref}||$ |
| 1   | 1           | 1.00e-02                      | 1.00e-02 | 1.00e-02  | 10040    | 10040     |
| 10  | 1           | 1.00e-02                      | 1.00e-02 | 1.19e-04  | 10040    | 162       |
| 100 | 1           | 1.00e-02                      | 1.00e-02 | 1.19e-04  | 10040    | 162       |
| 10  | 100         | 9.05e-03                      | 2.26e-05 | 1.10e-04  | 1590     | 152       |
| 100 | 100         | 9.05e-03                      | 1.90e-06 | 1.09e-04  | 124      | 151       |
| 1000| 100         | 9.05e-03                      | 4.25e-08 | 4.60e-07  | 2.04     | 15        |
| 1000| 900         | 1.01e-03                      | 3.73e-09 | 2.68e-08  | 0.19     | 1.26      |

**Table 6.2: Symmetric case (2)**

| $C$ | $K_{ref}$ | $K_{init}$ | $||H_{init} - \overline{H}||_2$ | $fsolve$ | $fmincon$ | $fsolve$ | $fmincon$ |
|-----|-----------|------------|-------------------------------|----------|-----------|----------|-----------|
|     |           |            | $||H_{opt} - \overline{H}||_2$ | $||H_{opt} - \overline{H}||_2$ | $||K_{opt} - K_{ref}||$ | $||K_{opt} - K_{ref}||$ |
| 1   | (10,100)  | (1,10)     | 5.97e-04                      | 5.97e-04 | 5.98e-04  | 607.48   | 607       |
| 10  | (10,100)  | (1,10)     | 5.97e-04                      | 5.33e-04 | 2.83e-05  | 543.89   | 101       |
| 10  | (1,10)    | (10,100)   | 5.97e-04                      | 5.97e-04 | 5.97e-04  | 607.48   | 608       |
| 100 | (1,10)    | (10,100)   | 5.97e-04                      | 1.05e-05 | 4.55e-06  | 1825     | 79        |
| 1000| (1,10)    | (10,100)   | 5.97e-04                      | 2.35e-06 | 1.31e-06  | 809      | 66        |
| 1000| (10,100)  | (100,1000) | 5.97e-03                      | 1.10e-05 | 2.92e-05  | 2811     | 735       |
| 1000| (10,100)  | (100,1000) | 5.97e-03                      | 2.58e-06 | 1.30e-05  | 887      | 660       |
| 1000| (100,1000)| (10,100)   | 5.97e-03                      | 1.32e-06 | 1.26e-05  | 514      | 657       |
Table 6.3: Symmetric case (3)

<table>
<thead>
<tr>
<th>$C$</th>
<th>$K_{ref}$</th>
<th>$K_{init}$</th>
<th>$|H_{init} - \overline{H}|_2$</th>
<th>$|H_{init} - \overline{H}|_2$</th>
<th>$|H_{opt} - \overline{H}|_2$</th>
<th>$|H_{opt} - \overline{H}|_2$</th>
<th>$|K_{opt} - K_{ref}|$</th>
<th>$|K_{opt} - K_{ref}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,5,10)</td>
<td>(1,10,20)</td>
<td>4.15e-05</td>
<td>4.15e-05</td>
<td>4.15e-05</td>
<td>4.15e-05</td>
<td>42.4264</td>
<td>42.4264</td>
</tr>
<tr>
<td>100</td>
<td>(1,5,10)</td>
<td>(1,10,20)</td>
<td>4.15e-05</td>
<td>2.19e-06</td>
<td>5.32e-07</td>
<td>8.13</td>
<td>7.23</td>
<td>7.23</td>
</tr>
<tr>
<td>100</td>
<td>(1,10,20)</td>
<td>(1,5,10)</td>
<td>4.15e-05</td>
<td>3.75e-06</td>
<td>5.41e-07</td>
<td>9.33</td>
<td>7.23</td>
<td>7.23</td>
</tr>
<tr>
<td>100</td>
<td>(1,10,100)</td>
<td>(5,50,500)</td>
<td>1.13e-03</td>
<td>1.14e-05</td>
<td>9.08e-06</td>
<td>2556</td>
<td>336</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>(1,10,100)</td>
<td>(5,50,500)</td>
<td>1.13e-03</td>
<td>3.19e-06</td>
<td>5.72e-06</td>
<td>830</td>
<td>299</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>(1,50,100)</td>
<td>(100,50,1)</td>
<td>7.74e-04</td>
<td>5.83e-06</td>
<td>7.62e-06</td>
<td>551</td>
<td>139.50</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>(1,50,100)</td>
<td>(100,50,1)</td>
<td>7.74e-04</td>
<td>1.14e-06</td>
<td>1.99e-06</td>
<td>391</td>
<td>112.28</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Nonsymmetric case (4)

<table>
<thead>
<tr>
<th>$C$</th>
<th>$K_{ref}$</th>
<th>$K_{init}$</th>
<th>$|H_{init} - \overline{H}|_2$</th>
<th>$|H_{opt} - \overline{H}|_2$</th>
<th>$|H_{opt} - \overline{H}|_2$</th>
<th>$|H_{opt} - \overline{H}|_2$</th>
<th>$|K_{opt} - K_{ref}|$</th>
<th>$|K_{opt} - K_{ref}|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1:100)</td>
<td>(10:100)</td>
<td>5.78e-05</td>
<td>5.78e-05</td>
<td>5.78e-05</td>
<td>5.78e-05</td>
<td>57.9784</td>
<td>57.9784</td>
</tr>
<tr>
<td>1</td>
<td>(10:100)</td>
<td>(100:1000)</td>
<td>5.94e-03</td>
<td>5.94e-03</td>
<td>5.94e-03</td>
<td>5.94e-03</td>
<td>5951</td>
<td>5951</td>
</tr>
<tr>
<td>100</td>
<td>(10:100)</td>
<td>(100:1000)</td>
<td>5.94e-03</td>
<td>1.11e-05</td>
<td>1.01e-05</td>
<td>1.01e-05</td>
<td>2501</td>
<td>246</td>
</tr>
<tr>
<td>1000</td>
<td>(10:100)</td>
<td>(100:1000)</td>
<td>5.94e-03</td>
<td>3.85e-06</td>
<td>3.42e-06</td>
<td>3.42e-06</td>
<td>1090</td>
<td>224</td>
</tr>
<tr>
<td>100</td>
<td>(100:1000)</td>
<td>(500:5000)</td>
<td>2.64e-02</td>
<td>5.64e-06</td>
<td>8.52e-05</td>
<td>8.52e-05</td>
<td>551</td>
<td>1320</td>
</tr>
<tr>
<td>1000</td>
<td>(100:1000)</td>
<td>(500:5000)</td>
<td>2.64e-02</td>
<td>7.24e-07</td>
<td>1.28e-05</td>
<td>1.28e-05</td>
<td>248</td>
<td>989</td>
</tr>
<tr>
<td>1000</td>
<td>(1000:10000)</td>
<td>(2000:20000)</td>
<td>6.60e-02</td>
<td>7.44e-09</td>
<td>2.48e-05</td>
<td>2.48e-05</td>
<td>391</td>
<td>2418</td>
</tr>
</tbody>
</table>
6.7 Plots of $K(X, T_0)$ and Heights

We show several selected plots for the $K_{ref}$ and $K_{opt}$ solutions. Figure 6.2 shows a nonsymmetric example. This target produced the best approximation to $K_{ref}$. Figures 6.3-6.7 display selected results from symmetric case (2). Next, with $k_0, k_1) = (1, 100)$, we show examples in Figures 6.8-6.11. Lastly, we show two plots that compare the target height $H$ with $H_{opt}$. The first in Figure 6.12 is a closeup of results from the first entry of Table 6.1 with $C = 1$ and $K_{init} = 1$. This is the largest error of any of the trials, and the only one visible in plotting. In Figure 6.13, we show an example of the full height profiles compared. As the difference is never visible, we only show the one case as exemplar of the rest. In 6.12 and 6.13, $T_0$ refers to the number of time steps $j$, not the nondimensional time $T = j\Delta T$. 

Figure 6.2
Fsolve: Target (500:1000), Initial (1000:100)  
Figure 6.3
Fmincon: Target (10,100), Initial (50,100)
Figure 6.4
Fsolve: Target (10,100), Initial (1,10)

Figure 6.5
Fmincon: Target (10,100), Initial (1,10)

Figure 6.6
Fsolve: Target (10,100), Initial (50,100)

Figure 6.7
Fsolve: Target (10,100), Initial (50,500)

Figure 6.8
Fmincon: Target (1,100), Initial (5,100)

Figure 6.9
Fsolve Target (1,100), Initial (5,100)
Figure 6.10
Fsolve: Target (1,100), Initial (1,10)

Figure 6.11
Fsolve: Target (1,100), Initial (10,1)

Figure 6.12: \( H_{opt} \) and \( \overline{H} \) profiles; Fsolve

Figure 6.13: \( H_{opt} \) and \( \overline{H} \) profiles; Fsolve
Lastly, we consider a numerical model to describe wound healing in a multi-layer polymer [4]. The GTFE (2.15) uniquely captures this case as the layer viscosity $\eta_m$ (2.3) and the layer thickness $H_m$ (2.4) can be used to model the progression of a thin surface layer instead of the bulk material. As explained in [2], when the temperature is below the glass transition temperature $T_g$, most of the material is unable to flow. Thus, we can only expect movement in a thin layer near the surface.

The goal of the work presented in this section is to simulate the healing process for multiple layers instead of solely the one surface layer. For the following tests, we use a 20 by 20 square computational domain, with $\Delta X = 0.025$ and $\Delta T = 10^{-5}$. Note that this differs from the previous trials in Chapter 5 where we used a value of $\Delta X = 0.1, 0.2$. Since the region of the wound is such a small percentage of the total domain size, decreasing the value of $\Delta X$ improves convergence and more accurately captures the progression of the healing process. For any $0 < \Delta X < 0.025$, the results remain relatively the same but the computational time increases greatly, so we use $\Delta X = 0.025$ in the following computations.

Since the GTFE (2.15) only contains information about the top surface layer, we cannot model the second layer until it is exposed to the air. Thus, we need to find the time at which the first layer has moved to fill in the wound, leaving the second layer exposed. We choose the time at which to switch to the next layer as follows. For a particular interval $[a, b]$
chosen in the computational domain $[-10,10]$, we calculate the height $H([a,b], T_0)$ over the interval $[a,b]$, with $T_0$ the initial time. We choose $[a,b]$ in the raised region immediately to the left or right of the wound. Since the first layer is considered to be the mobile part of the polymer, we wish to find the time $T_1$ at which the first layer has moved down from the raised region $[a,b]$ to fill in the wound. At that time, we consider the second layer to be exposed to the air in the interval $[a,b]$. Using the same layer thickness $H_m$, but a higher layer viscosity $\eta_m$, we then calculate the next time $T_2$ when the second layer has moved from the interval $[a,b]$ to fill in the wound.

Mathematically, $T_i$ is the minimum time satisfying:

$$H([a,b], T_i) < H([a,b], T_{i-1}) - H_m$$

for $i=1,2,3,...,n$ (7.1)

where $n$ is the number of layers we wish to simulate.

Recall that $T = j \Delta T$ for the nondimensional time $T$ and the number of time steps $j$. Thus, in the tables below we show the number of time steps $j$ corresponding to $T_i$ for each layer.

7.1 Discussion and Results

Consider again the problem of converting from computational time to physical time, with the conversion described by Eq. (2.13). With a fixed reference scale $h_0 = 0.005$ mm for the 20 by 20 computational domain, the ratio $\frac{\alpha}{\eta_m}$ affects this conversion. With a different ratio, the same computational time corresponds to a different real time scale.

We present two different ratios for $\frac{\alpha}{\eta_m}$. The first is to simply use the given values presented in Eq. (2.7) and (2.8), arriving at $\frac{\alpha}{\eta_m} = 6 \times 10^{-10}$ m s. This value is used in the conversion for Tables 7.3 and 7.4. In the second method, used in Tables 7.1 and 7.2, this ratio is calculated by using the correspondence between the numerical and experimental results. In [3], we observe that it takes roughly 15 days for the depth of the wound to have
healed one-third of the way. Additionally, we note that the numerical results show healing one-third of the way by \( j = 39,640,000 \) time steps for \( H_m = 0.004 \). We compute:

\[
\frac{\gamma}{\eta_m} = \frac{3j \Delta T h_0}{t} = \frac{0.005946 \text{ m}}{15 \text{ days}} = 4.58796 \times 10^{-9} \frac{\text{m}}{\text{s}}
\]

Likewise, for \( H_m = 0.04 \), the numerical results display healing by only \( j = 39,639 \) steps. As expected, this process is much quicker as \( H_m \) greatly affects the speed of the healing. We compute:

\[
\frac{\gamma}{\eta_m} = \frac{3j \Delta T h_0}{t} = \frac{5.95 \times 10^{-6} \text{ m}}{15 \text{ days}} = 4.5878 \times 10^{-12} \frac{\text{m}}{\text{s}}
\]

In Tables 7.1-7.4, we display results for the time taken for each subsequent layer of thickness \( H_m \) to slide down to fill in the wound. Tables 7.1 and 7.3 contain results for \( H_m = 0.004 \), which corresponds precisely to the domain size, and Tables 7.2 and 7.3 show results for \( H_m = 0.04 \). In each table, the second column represents the number of time steps that the solver takes for Eq. (7.1) to be satisfied for the corresponding layer. The third column contains the physical time that corresponds to that number of time steps, entirely dependent on the choice of \( \frac{\gamma}{\eta_m} \). Note that the number of time steps is measured from the time the previous layer moved, not from the initial time.

The initial physical time is computed using the values for \( \frac{\gamma}{\eta_m} \) given above. For each layer, we assume \( \eta_m \) doubles, as we are moving further into the bulk of the material. Thus the ratio \( \frac{\gamma}{\eta_m} \) decreases by a half when calculating the physical time for each progressive layer. The plots corresponding to these results are seen in Figures 7.1 and 7.2.
As a separate note, we observe that using $H_m = 0.04$ in computation, the height profile reaches $H([a, b], T_0) - H_m = H([a, b], T_0) - 0.04$ by 410 time steps, whereas using $H_m = 0.004$, the height profile reaches $H([a, b], T_0) - H_m = H([a, b], T_0) - 0.004$ by 410,000 time steps as seen in Table 7.3. This corresponds with exactly what we expect: the value used in the GTFE (2.15) is $H_m^3$, thus when using a value of $H_m = 0.04$ that is 10 times larger, we should reach the same height profile at $\frac{1}{10}$ of the time that it took to reach that height profile with the smaller $H_m$ value.

Table 7.1: Layer Progression, $H_m = 0.004$, $[\gamma, \eta] = [−0.25, −0.2], \frac{\gamma}{\eta_m} = 4.58796 \times 10^{-9} m/s$

<table>
<thead>
<tr>
<th>Layer Number</th>
<th>Time Steps</th>
<th>Physical Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First layer</td>
<td>409,179</td>
<td>0.155 days</td>
</tr>
<tr>
<td>Second layer</td>
<td>416,013</td>
<td>0.315 days</td>
</tr>
<tr>
<td>Third layer</td>
<td>470,724</td>
<td>0.712 days</td>
</tr>
<tr>
<td>Fourth layer</td>
<td>510,381</td>
<td>1.545 days</td>
</tr>
<tr>
<td>Fifth layer</td>
<td>555,720</td>
<td>3.364 days</td>
</tr>
</tbody>
</table>

Table 7.2: Layer Progression, $H_m = 0.04$, $[\gamma, \eta] = [−0.25, −0.2], \frac{\gamma}{\eta_m} = 4.5878 \times 10^{-12} m/s$

<table>
<thead>
<tr>
<th>Layer Number</th>
<th>Time Steps</th>
<th>Physical Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First layer</td>
<td>5,755</td>
<td>2.18 days</td>
</tr>
<tr>
<td>Second layer</td>
<td>9,922</td>
<td>7.51 days</td>
</tr>
<tr>
<td>Third layer</td>
<td>13,914</td>
<td>21.06 days</td>
</tr>
<tr>
<td>Fourth layer</td>
<td>22,252</td>
<td>67.36 days</td>
</tr>
<tr>
<td>Fifth layer</td>
<td>46,349</td>
<td>280.63 days</td>
</tr>
</tbody>
</table>

Table 7.3: Layer Progression, $H_m = 0.004$, $[\gamma, \eta] = [−0.25, −0.2], \frac{\gamma}{\eta_m} = 6 \times 10^{-10} m/s$

<table>
<thead>
<tr>
<th>Layer Number</th>
<th>Time Steps</th>
<th>Physical Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First layer</td>
<td>409,179</td>
<td>1.184 days</td>
</tr>
<tr>
<td>Second layer</td>
<td>416,013</td>
<td>2.407 days</td>
</tr>
<tr>
<td>Third layer</td>
<td>470,724</td>
<td>5.448 days</td>
</tr>
<tr>
<td>Fourth layer</td>
<td>510,381</td>
<td>11.814 days</td>
</tr>
<tr>
<td>Fifth layer</td>
<td>555,720</td>
<td>25.728 days</td>
</tr>
</tbody>
</table>
Table 7.4: Layer Progression, $H_m = 0.04$, $[a,b] = [-0.25,-0.2]$, $\frac{\gamma}{\eta_m} = 6 \times 10^{-10} \text{m} \text{s}^{-1}$

<table>
<thead>
<tr>
<th>Layer Number</th>
<th>Time Steps</th>
<th>Physical Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First layer</td>
<td>5,755</td>
<td>0.017 days</td>
</tr>
<tr>
<td>Second layer</td>
<td>9,922</td>
<td>0.057 days</td>
</tr>
<tr>
<td>Third layer</td>
<td>13,914</td>
<td>0.161 days</td>
</tr>
<tr>
<td>Fourth layer</td>
<td>22,252</td>
<td>0.515 days</td>
</tr>
<tr>
<td>Fifth layer</td>
<td>46,349</td>
<td>2.146 days</td>
</tr>
</tbody>
</table>

Figure 7.1
Multi-layer model for $H_m = 0.04$

Figure 7.2
Multi-layer model for $H_m = 0.004$
Chapter 8

Conclusions

We study and implement two mathematical models that describe the process of self-healing in polymers. The numerical results of the TFE correspond perfectly to the literature [7]. For the GTFE, we find that including the layer thickness $H_m$ in the model greatly decreases the rate of healing. Numerically, the domain size and step size need to be adjusted to accommodate for a larger $H_m$. The initial shape and size of the wound has a large effect on the self-healing process, and the two that have shown the best results are the V shaped conditions and the area-preserving conditions shown in Figure 6.1. The two optimization approaches presented in Chapter 6 each produce accurate results for matching a target height profile at a specific time point. Convergence is sped up when using a larger scaling constant. The transition between the dimensionless number of time steps and the physical time is mostly dependent upon the value of the factor $\frac{2}{\eta}$. Finding more accurate approximations of this value for each layer will improve the estimates given in the multi-layer modeling.

Some future directions for this work include extending the models from one dimensional to multi dimensional. This involves changing from a finite difference approach to a finite element approach for solving numerically, as well as completing the applicable analysis. For optimization, different scaling constants $C$ that are domain dependent could be utilized, as well as changing the problem setup to match target heights over all times $[0, T_0]$.
instead of just at one point $T_0$. In general, implementing adaptive step sizes will improve performance by allowing for a larger domain size, and thereby a larger $H_m$, while providing a small $\Delta X$ only in regions where the change is concentrated, which mitigates the large increase in computational time.
Appendices
Appendix A  Selected Code

In this Appendix, we include three sample codes. The first code presented implements the finite difference method presented in Chapter 3 to solve the GTFE (2.15). The second code implements the same method to solve the TFE (6.1), specifically in the case of an optimization problem such as Chapter 6. This code includes the nonzero right hand side function $K(x)$, but it could be easily changed to have the right hand side set equal to 0 again in order to solve the original TFE (2.14). The third code given is the routine implementing $fsolve$ as the second approach to the optimization problem (Chapter 6.2).

In Listing 1, the function GTFE takes as input the final number of time steps, $t_{end}$, the nondimensional time step $delt$, the nondimensional layer thickness $HM$, the boundaries of the horizontal computational domain $x_{start}, x_{end}$, and the number of grid points $nx$. The output given is a vector $store.H1n$ that contains the height profile at each grid point $X_i$ every 5000 time steps.

In Listing 2, the function TFE.RHSfxn also takes $delt, x_{start}, x_{end}$ and $nx$. Additionally, it requires the right hand side vector $K(x)$ given to be input in the parameter $RHS_{fxn}$. The output is only a single height profile this time; $hout$ gives the height at every grid point $X_i$ at the final time step $t_{end}$.

In Listing 3, we follow the process described in Chapter 6.3. The script sets up $K_{ref}$ and solves the TFE by calling the function TFE.RHSfxn in Listing 2 to produce corresponding target height $\bar{H}$. Next the script sets up $K_{init}$ and uses $fsolve$ to find the optimal solution $K_{opt}$ and then computes corresponding $H_{opt}$. Plots and error computations are included at the end of the script.

Listing 1: RK4 code to solve the GTFE

```matlab
1 % Solve the GTFE
2 % Area preserving ICS implemented as a cubic spline
3
4 function [store.H1n, delx] = GTFE(t_end, delt, HM, xstart, xend, nx)
```
delx = (xend - xstart)/(nx-1);
xspan=inspace(xstart,xend,nx);
H1n = zeros(nx,1);
H2n = zeros(nx,1);
H3n = zeros(nx,1);
H4n = zeros(nx,1);
k1 = zeros(nx,1);
k2 = zeros(nx,1);
k3 = zeros(nx,1);
k4 = zeros(nx,1);

%% Area preserving ICS:
x = [-0.6, -0.55, -0.45, -0.4, -0.34, -0.22, -0.15073, 0, 0.15073, 0.22, ... 
  0.34, 0.4, 0.45, 0.55, 0.6];
  20.04986, 20.01, 20, 20];
sp = spline(x,y,xspan);
for i = 1:nx
    if xspan(i) >= 0.6 || xspan(i) <= -0.6
        H1n(i,1) = 20;
    else
        H1n(i,1) = sp(i);
    end
end
store_H1n = zeros(nx,(t_end/5000)+1); %save every 5000 steps
store_H1n(:,1) = H1n(:,1);
count = 2;

%% Time Loop for RK4
for t = 1:t_end %increments of delt

    %To find k1: calculate H2n/H3n/H4n with no changes to H1
    for i = 1:nx
if i == 1  % BC
    H2n(i,1) = 0;
else
    H2n(i,1) = (H1n(i,1) - H1n(i-1,1))/(delx);
end
end
for i = nx:-1:1
    if i == nx  % BC
        H3n(i,1) = 0;
    else
        H3n(i,1) = (H2n(i+1,1) - H2n(i,1))/(delx);
    end
end
for i = 1:nx
    if i == 1  % BC
        H4n(i,1) = 0;
    else
        H4n(i,1) = (H3n(i,1) - H3n(i-1,1))/(delx);
    end
end
for j = 1:nx
    if j == nx
        k1(j,1) = ((HM^3)*H4n(j,1))/(delx);
    else
        k1(j,1) = (HM^3)*(H4n(j,1) - H4n(j+1,1))/(delx);
    end
end
% Now, calculate k2 using the updated H1n + 0.5*delt*k1
for i = 1:nx
    if i == 1  % BC
H2n(i,1) = 0;
else
    H2n(i,1) = (H1n(i,1) + 0.5*delt*k1(i,1) - (H1n(i,1) + 0.5*
    delt*k1(i-1,1)))/(delx);
end
end
for i = nx:-1:1
    if i == nx  % BC
        H3n(i,1) = 0;
    else
        H3n(i,1) = (H2n(i+1,1) - H2n(i,1))/(delx);
    end
end
for i = 1:nx
    if i == 1  % BC
        H4n(i,1) = 0;
    else
        H4n(i,1) = (H3n(i,1) - H3n(i-1,1))/(delx);
    end
end
for j = 1:nx
    if j == nx
        k2(j,1) = ((HM^3)*H4n(nx,1))/(delx);
    else
        k2(j,1) = (HM^3)*(H4n(j,1) - H4n(j+1,1))/(delx);
    end
end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Now, calculate k3 using the updated H1n + 0.5*delt*k2
for i = 1:nx
    if i == 1  % BC
H2n(i,1) = 0;
else
H2n(i,1) = (H1n(i,1) + 0.5 * delt * k2(i,1) - (H1n(i-1,1) + 0.5 * delt * k2(i-1,1)) / (delx);
end
end
for i = nx:-1:1
if i == nx % BC
H3n(i,1) = 0;
else
H3n(i,1) = (H2n(i+1,1) - H2n(i,1)) / (delx);
end
end
for i = 1:nx
if i == 1 % BC
H4n(i,1) = 0;
else
H4n(i,1) = (H3n(i,1) - H3n(i-1,1)) / (delx);
end
end
for j = 1:nx
if j == nx
k3(j,1) = ((HM^3) * H4n(nx,1)) / (delx);
else
k3(j,1) = (HM^3) * (H4n(j,1) - H4n(j+1,1)) / (delx);
end
end
% Now, calculate k4 using the updated H1n + delt*k3
for i = 1:nx
if i == 1 % BC
H2n(i,1) = 0;
else
H2n(i,1) = (H1n(i,1)+delt*k3(i,1) − (H1n(i−1,1)+delt*k3(i−1,1)))/(delx);
end
end
for i = nx:−1:1
if i == nx % BC
H3n(i,1) = 0;
else
H3n(i,1) = (H2n(i+1,1) − H2n(i,1))/(delx);
end
end
for i = 1:nx
if i == 1 % BC
H4n(i,1) = 0;
else
H4n(i,1) = (H3n(i,1) − H3n(i−1,1))/(delx);
end
end
for j = 1:nx
if j == nx
k4(j,1) = ((HM^3)*H4n(nx,1))/(delx);
else
k4(j,1) = (HM^3)*(H4n(j,1) − H4n(j+1,1))/(delx);
end
end
% Do the RK4 calculation for H1n(:,i+1)
H1n(:,1) = H1n(:,1) + (delt/6) * (k1(:,1)+2*k2(:,1)+2*k3(:,1)+k4(:,1));
%every 5000 time steps, save current height profile
if mod(t,5000) == 0
    store_H1n(:,count) = H1n(:,1);
    count = count + 1;
end
end

Listing 2: RK4 code to solve the TFE for optimization

% Compute the TFE with a nonzero right hand side fxn K(x).
% Output the height profile at the final time for optimization.
function hout = TFE_RHSfxn(RHS_fxn, xstart, xend, nx, delt)
    t_end = 1000;
    delx = (xend - xstart)/(nx-1);
    xspan=linspace(xstart,xend,nx);
    H1n=zeros(nx,1);
    H2n=zeros(nx,1);
    H3n=zeros(nx,1);
    H4n=zeros(nx,1);
    A1 = zeros(nx,1);
    A2 = zeros(nx,1);
    A3 = zeros(nx,1);
    A4 = zeros(nx,1);
    k1 = zeros(nx,1);
    k2 = zeros(nx,1);
    k3 = zeros(nx,1);
    k4 = zeros(nx,1);

    %ICs for H1n used for Tables 6.1–6.4
    x = [-.45, -.4, -.35, -.3, -.2, -.15, -.085, -.06, -.05, 0, 0.05, 0.06, 0.085, .15, .2, .3, .35, .4, .45];
    y = [2.005, 2.015, 2.025, 2.04, 2.1, 2.17, 2.3, 2.0, 1.8, 1.4, 1.8,
2.0, 2.3, 2.17, 2.1, 2.04, 2.025, 2.015, 2.005];
sp = spline(x,y,xspan);
for i=1:nx
    if xspan(i) < 0.45 && xspan(i) > -0.45
        H1n(i,1) = sp(i);
    else
        H1n(i,1) = 2;
    end
end
store_H1n(:,1) = H1n(:,1);
count=2;

%% Time Loop for RK4
for t = 1:t_end
    %%%%%% To find k1: calculate H2n/H3n/H4n with no changes to H1
    for i = 1:nx
        if i == 1 % BC
            H2n(i,1) = 0;
        else
            H2n(i,1) = (H1n(i,1) - H1n(i-1,1))/(delx);
        end
    end
    for i = nx:-1:1
        if i == nx % BC
            H3n(i,1) = 0;
        else
            H3n(i,1) = (H2n(i+1,1) - H2n(i,1))/(delx);
        end
    end
end
for i = 1:nx
    if i == 1 % BC
        H4n(i,1) = 0;
    else
        H4n(i,1) = (H3n(i,1) - H3n(i-1,1))/(delx);
    end
end

for j = 1:nx
    if j == 1
        A1(j,1) = H1n(j,1)^3;
    else
        A1(j,1) = (2*((H1n(j-1,1))^2)*((H1n(j,1))^2))/(H1n(j-1,1) + H1n(j,1));
    end
end

for j = 1:nx
    if j == nx
        k1(j,1) = ((A1(j,1)*H4n(j,1))/(delx)) + RHS_fxn(j);
    else
        k1(j,1) = ((A1(j,1)*H4n(j,1) - A1(j+1,1)*H4n(j+1,1))/(delx)) + RHS_fxn(j);
    end
end

% Now, calculate k2 using the updated H1n + 0.5*delt*k1
for i = 1:nx
    if i == 1 % BC
        H2n(i,1) = 0;
    else
        H2n(i,1) = ( (H1n(i,1) + 0.5*delt*k1(i,1)) - (H1n(i-1,1)+0.5*delt*k1(i-1,1)) )/(delx);
    end
end
for i = nx:-1:1
    if i == nx % BC
        H3n(i,1) = 0;
    else
        H3n(i,1) = (H2n(i+1,1) - H2n(i,1))/(delx);
    end
end
for i = 1:nx
    if i == 1 % BC
        H4n(i,1) = 0;
    else
        H4n(i,1) = (H3n(i,1) - H3n(i-1,1))/(delx);
    end
end
for j = 1:nx
    if j == 1
        A2(j,1) = (H1n(j,1) + 0.5*delt*k1(j,1))^3;
    else
        A2(j,1) = (2*((H1n(j-1,1) + 0.5*delt*k1(j-1,1))^2)*((H1n(j,1) + 0.5*delt*k1(j,1))^2)/(H1n(j-1,1) + 0.5*delt*k1(j-1,1) + H1n(j,1) + 0.5*delt*k1(j,1));
    end
end
for j = 1:nx
    if j == nx
        k2(j,1) = ((A2(j,1)*H4n(nx,1))/(delx)) + RHS_fxn(j);
    else
        k2(j,1) = ((A2(j,1)*H4n(j,1) - A2(j+1,1)*H4n(j+1,1))/(delx)) + RHS_fxn(j);
    end
end
% Now, calculate k3 using the updated H1n + 0.5*del*t*k2

for i = 1:nx
    if i == 1  % BC
        H2n(i,1) = 0;
    else
        H2n(i,1) = ((H1n(i,1) + 0.5*del*t*k2(i,1)) - (H1n(i-1,1) + 0.5*del*t*k2(i-1,1))) / (delx);
    end
end

for i = nx:-1:1
    if i == nx  % BC
        H3n(i,1) = 0;
    else
        H3n(i,1) = (H2n(i+1,1) - H2n(i,1)) / (delx);
    end
end

for i = 1:nx
    if i == 1  % BC
        H4n(i,1) = 0;
    else
        H4n(i,1) = (H3n(i,1) - H3n(i-1,1)) / (delx);
    end
end

for j = 1:nx
    if j == 1
        A3(j,1) = (H1n(j,1) + 0.5*del*t*k2(j,1))^3;
    else
        A3(j,1) = (2*((H1n(j-1,1) + 0.5*del*t*k2(j-1,1))^2)*((H1n(j,1))))
\[ +0.5 \cdot \text{del} \cdot k2(j,1)^2) / (H1n(j-1,1) + 0.5 \cdot \text{del} \cdot k2(j-1,1) + H1n(j,1) + 0.5 \cdot \text{del} \cdot k2(j,1)) \]

```
end
end
for j = 1:nx
  if j == nx
    k3(j,1) = ((A3(j,1) \cdot H4n(nx,1)) / (delx)) + RHS_fxn(j);
  else
    k3(j,1) = ((A3(j,1) \cdot H4n(j,1) - A3(j+1,1) \cdot H4n(j+1,1)) / (delx)) + RHS_fxn(j);
  end
end
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Now, calculate k4 using the updated H1n + del \ast k3
for i = 1:nx
  if i == 1  % BC
    H2n(i,1) = 0;
  else
    H2n(i,1) = ( (H1n(i,1)+\text{del} \cdot k3(i,1)) - (H1n(i-1,1)+\text{del} \cdot k3(i-1,1)) ) / (delx);
  end
end
for i = nx:-1:1
  if i == nx  % BC
    H3n(i,1) = 0;
  else
    H3n(i,1) = (H2n(i+1,1) - H2n(i,1)) / (delx);
  end
end
for i = 1:nx
  if i == 1  % BC
```
H4n(i,1) = 0;
else
H4n(i,1) = (H3n(i,1) - H3n(i-1,1))/(delx);
end
end
for j = 1:nx
if j == 1
    A4(j,1) = (H1n(j,1)+delt*k3(j,1))^3;
else
    A4(j,1) = (2*((H1n(j-1,1)+delt*k3(j-1,1))ˆ2)*(H1n(j,1)+delt*k3(j,1)^2))/(H1n(j-1,1)+delt*k3(j-1,1) + H1n(j,1)+delt*k3(j,1));
end
end
for j = 1:nx
if j == 1
    k4(j,1) = ((A4(j,1)*H4n(nx,1))/(delx)) + RHS_fxn(j);
else
    k4(j,1) = ((A4(j,1)*H4n(j,1) - A4(j+1,1)*H4n(j+1,1))/(delx)) + RHS_fxn(j);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Do the RK4 calculation for H1n(:,i+1)
H1n(:,1) = H1n(:,1) + (delt/6) * (k1(:,1)+2*k2(:,1)+2*k3(:,1)+k4(:,1));
if t == t_end
    hout = H1n(:,1); %output height at time step t_end
end
end

Listing 3: Optimization approach 2: fsolve routine
% Implements the process described in Ch. 6.3 using the nonlinear
    systems approach with fsolve
CC = 100; % scaling constant
Symtype = 1; % symmetric: 1, nonsymmetric: 2
ICStype = 1; % if symmetric, how many constants in RHS vec?
sym_targ1 = 0;
sym_targ2 = 1000;
sym_targ3 = 100;
nonsym_start = 1000;
nonsym_end = 10000;
% Set the initial RHS vector
sym_init1 = 10;
sym_init2 = 900;
sym_init3 = 1;
nonsym_init1 = 2000;
nonsym_init2 = 20000;
% Computational domain
nx = 101;
xstart = −1; xend = 1;
xspan = linspace(xstart,xend,nx);
delt = 10^−9;
% Compute target RHS and height
k_targ = Setup_InitialK(Symtype,ICStype,sym_targ1,sym_targ2,sym_targ3,
    nonsym_start,nonsym_end,nx);
h_targ = TFE_eq_RHSf(xn(k_targ,xstart,xend,nx,delt);
% Define the function to be solved; F(k) = 0
F = @(RHS_ICS) CC*TFE_eq_RHSf(RHS_ICS,xstart,xend,nx,delt)−CC*h_targ;
k_init = Setup_InitialK(Symtype,ICStype,sym_init1,sym_init2,sym_init3,
    nonsym_init1,nonsym_init2,nx);
% Use fsolve to solve for the optimal RHS
options = optimoptions(’fsolve’,’Display’,’iter’,’TolFun’,1e−8,’
MaxFunEval',50000);

[kout,fval] = fsolve(F,k_init,options);

%Plot solutions and display errors
fsolve_sol = TFE_eq_RHSfxn(kout,xstart,xend,nx,delt);
plot(xspan,h_targ,'LineWidth',2)
hold on
plot(xspan,fsolve_sol,'LineWidth',2)
switch Symtype
    case 1
        switch ICStype
            case 1
                str1 = sprintf('Target Height, reference k0 (%d)', sym_targ1);
                str2 = sprintf('Fsolve Height, initial k0 (%d)', sym_init1);
                str = sprintf('T0=1000, C=%d; Symmetric case (1)', CC);
                stri = sprintf('Compare RHS K(x); C=%d; K init = (%d)', CC, sym_init1);
                stri2 = sprintf('K target = (%d)', sym_targ1);
                title(str)
            case 2
                str1 = sprintf('Target Height, reference (k0,k1)=(%d,%d)', sym_targ1, sym_targ2);
                str2 = sprintf('Fsolve Height, initial (k0,k1)=(%d,%d)', sym_init1, sym_init2);
                str = sprintf('T0=1000, C=%d, Symmetric RHS, 2 constants', CC);
                stri = sprintf('Compare RHS K(x); C=%d; K init = (%d,%d)', CC, sym_init1, sym_init2);
                stri2 = sprintf('K target = (%d,%d)', sym_targ1, sym_targ2);
        end
    end
end
case 3

str1 = sprintf('Target Height, reference (k0,k1,k2)=(%d,%d,%d)', sym_targ1, sym_targ2, sym_targ3);
str2 = sprintf('Fsolve Height, initial (k0,k1,k2)=(%d,%d,%d)', sym_init1, sym_init2, sym_init3);
str = sprintf('T0=1000, C=%d, Symmetric RHS, 3 constants', CC);
stri = sprintf('Compare RHS K(x); C=%d; K init = (%d,%d,%d)', CC, sym_init1, sym_init2, sym_init3);
stri2 = sprintf('K target = (%d,%d,%d)', sym_targ1, sym_targ2, sym_targ3);
title(str)
end

end

title(str)

case 2

str1 = sprintf('Target Height, reference k0:k1=%d:%d', nonsym_start, nonsym_end);
str2 = sprintf('Fsolve Height, initial k0:k1=%d:%d', nonsym_init1, nonsym_init2);
stri = sprintf('Compare RHS K(x); C=%d; K init = %d:%d', CC, nonsym_init1, nonsym_init2);
stri2 = sprintf('K target = %d:%d', nonsym_start, nonsym_end);
str = sprintf('T0=1000, C=%d, Nonsymmetric RHS', CC);
title(str)
end

legend(str1, str2)

figure()
hold on
plot(xspan, k_targ, '-', 'LineWidth', 2)
plot(xspan, kout, ':', 'LineWidth', 2)
legend(stri2, 'K computed by Fsolve')
title(stri)

% Display errors
disp('L2 error in RHS')
disp(norm((k_targ - kout), 2))

% disp('l2 error in height profile (optimized)')
disp(norm((h_targ - fsolve_sol), 2))

% disp('H_initial - H_targ')
disp(norm((h_targ - TFE_eq_RHSfxn(k_init)), 2))
Bibliography


