An Integral Equation Method for Solving Second-Order Viscoelastic Cell Motility Models

A Master’s Thesis

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

by

Kyle Dunn

April 30, 2014

Approved

Professor Roger Y. Lui
Thesis Advisor

Professor Luca Capogna
Department Head
Abstract

For years, researchers have studied the movement of cells and mathematicians have attempted to model the movement of the cell using various methods. This work is an extension of the work done by Zheltukhin and Lui (2011), Mathematical Biosciences 229:30-40, who simulated the stress and displacement of a one-dimensional cell using a model based on viscoelastic theory.

The report is divided into three main parts. The first part considers viscoelastic models with a first-order constitutive equation and uses the standard linear model as an example. The second part extends the results of the first to models with second-order constitutive equations. In this part, the two examples studied are Burger model and a Kelvin-Voigt element connected with a dashpot in series. In the third part, the effects of substrate with variable stiffness are explored. Here, the effective adhesion coefficient is changed from a constant to a spatially-dependent function. Numerical results are generated using two different functions for the adhesion coefficient.

Results of this thesis show that stress on the cell varies greatly across each part of the cell depending on the constitute equation we use, while the position and velocity of the cell remain essentially unchanged from a large-scale point of view.
Acknowledgments

First, I would like to thank my extremely patient advisor and mentor, Professor Roger Lui, for his guidance and support since the beginning of this work. I would also like to thank the Mathematical Sciences Department of Worcester Polytechnic Institute for funding my studies and research. For my background in the areas of mathematics necessary to complete this work, I thank the professors of WPI, in particular Roger Lui, Marcus Sarkis, Sarah Olson, Homer Walker, and Burt Tilley.

Second, I would like to thank my friends and colleagues for their support and understanding, without whom I would have much less motivation to succeed. In particular, I express my sincere gratitude to all those who have listened and contributed ideas, including Joshua Plasse, Nathan Marculis, Daniel Brady, Brendan Hamm, and Nguyenho Ho. An additional thanks goes out to my fiancé Danielle Cloutier for helping to edit this report and her comments on the biological content.

Finally, I would like to thank Danielle, my parents, and my brothers for their love, support, and effort to understand my work. Without them I would not be where I am today.
# Contents

1 Introduction ................................................. 1  
  1.1 How Cells Move ........................................... 1  
  1.2 Viscoelastic Materials ................................... 1  
  1.3 Viscoelastic Cell Motility Model ...................... 3  

2 First-Order PDE Model ........................................ 5  
  2.1 Derivation of Model ...................................... 5  
  2.2 Formulation of Numerical Methods ..................... 7  
  2.3 Analysis of the Steady State Equation ................. 14  
  2.4 Numerical Solutions of the Standard Linear Model .... 15  

3 Second-Order PDE Model ...................................... 19  
  3.1 Examples of Second-Order Models ...................... 19  
  3.2 Derivation of Model ...................................... 21  
  3.3 Numerical Methods for Second-Order Models .......... 24  
  3.4 Numerical Results for Second-Order Models .......... 29  

4 Models with Spatially Dependent Adhesion Coefficients .... 35  
  4.1 Derivation of Model Equations ......................... 35  
  4.2 Derivation of Numerical Methods for Models with Variable Adhesion Coefficient 39  
  4.3 Numerical Results for Models with Variable Adhesion Coefficient ............ 42  
    4.3.1 Numerical Results for the Standard Linear Model .................. 42  
    4.3.2 Numerical Results for the Burger Model ......................... 47  

5 Conclusion .................................................. 51  

A Matlab Code ................................................... 53
# List of Figures

1.1 Diagrams of fundamental viscoelastic models. ........................................ 3  
1.2 A schematic diagram of a motile cell under the assumptions of the model presented in [8]. ................................................................. 4  

2.1 Surface plot of \( \sigma(y, t) \) for the standard linear model. .................. 16  
2.2 Side view of the surface plot of \( \sigma(y, t) \) for the standard linear model. .... 16  
2.3 Plot of the steady state solution \( \sigma_{ss}(y) \) and \( \sigma(y, T_{\text{final}}) \). .................. 17  
2.4 Surface plot of the displacement \( u(y, t) \) for the standard linear model. .... 17  
2.5 Plot of \( r(t) \) and \( f(t) \) for the standard linear model. .......................... 18  

3.1 Illustration of (a) Burger model and (b) Kelvin-Voigt and dashpot in series. 20  
3.2 Surface plot of \( \sigma(y, t) \) for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series. ................................................................. 31  
3.3 Side view of \( \sigma(y, t) \) for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series. ................................................................. 32  
3.4 Surface plots of the displacement \( u(y, t) \) for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series ........................................... 33  
3.5 Plot of the front and rear boundaries. ..................................................... 34  

4.1 Plot of the two functions used for \( \beta(x) \). ........................................ 43  
4.2 Two views of the surface plot of \( \sigma(y, t) \) for standard linear model, where \( \beta(x) \) is the “hump” function. ......................................................... 44  
4.3 Two views of the surface plot of \( \sigma(y, t) \) for the standard linear model, where \( \beta(x) \) is oscillatory. ......................................................... 45  
4.4 Plot of \( f_{\beta} - f_{\beta(x)} \) (blue) and \( r_{\beta} - r_{\beta(x)} \) (green) for the standard linear model. ......................................................... 46  
4.5 Two views of the surface plot of \( \sigma(y, t) \) for the Burger model, where \( \beta(x) \) is the hump function. ......................................................... 48  
4.6 Two views of the surface plot of \( \sigma(y, t) \) for the Burger model, where \( \beta(x) \) is oscillatory. ......................................................... 49  
4.7 Plot of \( f_{\beta} - f_{\beta(x)} \) (blue) and \( r_{\beta} - r_{\beta(x)} \) (green) for the Burger model. ......................................................... 50
Chapter 1

Introduction

1.1 How Cells Move

This paper considers the movement of a crawling cell as modeled by Larripa and Mogilner [7]. Two examples of crawling cells are fibroblasts and \textit{Dictyostelium discoideum} (Dicty) [5]. Fibroblasts, a type of cell involved in wound healing, move relatively slowly, while the adhesion between \textit{Dictyostelium discoideum} and the surface is relatively weak leading to a faster-moving cell. Fibroblasts also form focal adhesions to the surface, while Dicty change shape more rapidly.

Various sources describe the movement of a crawling cell and each source agrees that the cell simultaneously executes at least three steps [5, 7]. As described by Mogilner in [7], the cell cycles through protrusion, adhesion and contraction. In protrusion, the intracellular gel consisting of actin polymers pushes the front of the cell outward to advance the anterior. The cell adheres to the surface at the front edge of the cell while weakening its grip on the surface at the rear boundary in the adhesion step. In the contraction step, myosin motors pull the rear edge toward the front. These three steps repeat continuously as the cell moves forward. The inside of the cell is mostly fluid but contains an actin network which gives the cell its shape and structure. Thus, the material inside the cell is neither strictly viscous nor strictly elastic in the traditional sense; it is best to consider it as viscoelastic. Viscoelastic materials have memory, so integral equations (in time) are used to model viscoelastic materials [4].

1.2 Viscoelastic Materials

A viscoelastic material is a material exhibiting both elastic and viscous properties, each characterized by its stress-strain relationship. A linear elastic material obeys Hooke’s Law, $F = -ku$, where $F$ is force and $u$ is displacement. Let $\varepsilon$ denote strain, a unitless quantity, and let $\gamma$ denote stress. Then since $\varepsilon = u_x$ and $\gamma = F_x$, the constitutive law for an elastic material is

$$\gamma_s = E\varepsilon_s,$$  \hspace{1cm} (1.1)
where $E$ is the Young’s modulus$^1$[6]. For a material modeled by a viscous dashpot, the stress is proportional to the velocity, so $F = -cu_t = -cv$, where $v$ is velocity. Using the above identities for stress and strain, we find that the constitutive law is

$$\gamma_d = \mu \dot{\varepsilon}_d,$$  \hspace{1cm} (1.2)

where $\mu$ is a constant specific to the dashpot, often called viscosity. Using (1.1) and (1.2) we can connect various combinations of springs and dashpots in series and parallel for an infinite number of viscoelastic models. The Maxwell model, Kelvin-Voigt model, and standard linear model are three common combinations of viscous and elastic elements. The configuration for each of these models is shown in Figure 1.1.

For any combination of viscous and elastic elements we are able to derive the constitutive equation. For example, to derive the equations for the Maxwell model we use the facts that the total inactive stress $\gamma = \gamma_d = \gamma_s$ and the total strain $\varepsilon = \varepsilon_d + \varepsilon_s$ for two elements in series. Then from (1.1) and (1.2),

$$\dot{\varepsilon}_d = \frac{1}{\mu_0} \gamma \text{ and } \dot{\varepsilon}_s = \frac{1}{E_0} \dot{\gamma}.$$  \hspace{1cm} (1.6)

Adding the two equations, we obtain $\dot{\varepsilon} = \gamma / \mu_0 + \dot{\gamma}_t / E_0$. Similarly, combining a spring and a dashpot in series yields the Maxwell model,

$$\mu_0 \dot{\varepsilon} = \gamma + \frac{\mu_0}{E_0} \dot{\gamma}.$$  \hspace{1cm} (1.5)

The constitutive equations for the three fundamental models shown in Figure 1.1 are

\begin{align*}
\text{Maxwell Model: } & \frac{\mu_0}{E_0} \dot{\gamma} + \gamma = \mu_0 \dot{\varepsilon} \hspace{1cm} (1.3) \\
\text{Kelvin-Voigt Model: } & \gamma = \mu_0 \dot{\varepsilon} + E_0 \varepsilon \hspace{1cm} (1.4) \\
\text{Standard Linear Model: } & \frac{\mu_0}{E_0} \dot{\gamma} + \gamma = \mu_0 \left(1 + \frac{E_1}{E_0}\right) \dot{\varepsilon} + E_1 \varepsilon \hspace{1cm} (1.5)
\end{align*}

Studies have shown that the properties of the actin gel is best approximated by a combination of nonlinear Maxwell and Kelvin-Voigt models [7]. The basis of our research lies in this common belief. In this work we will consider various combinations of these fundamental viscoelastic elements in series and in parallel. Previous work in [8] considers the Maxwell and Kelvin-Voigt models and can be generalized to any viscoelastic model with a constitutive equation of the form

$$p_1 \dot{\gamma} + q_1 \gamma = p_2 \dot{\varepsilon} + q_2 \varepsilon,$$  \hspace{1cm} (1.6)

where $p_1$, $q_1$, $p_2$, and $q_2$ are unique to each combination of viscoelastic elements. The work of Zheltukhin and Lui [8] uses the stress and displacement of the viscoelastic models to calculate the velocity of the front and rear boundaries of a one-dimensional cell. We look to

$^1$In this paper, all $\mu$ and $E$ are assumed to be constant.
Figure 1.1: Diagrams of fundamental viscoelastic models.

extend the work of Zheltukhin and Lui in [8] by considering a more general constitutive law of the form

\[ r_1 \dot{\gamma} + p_1 \dot{\gamma} + q_1 \gamma = r_2 \ddot{\varepsilon} + p_2 \dot{\varepsilon} + q_2 \varepsilon. \]  

Equation (1.7) will be our constitutive stress-strain relation. This constitutive law must be supplemented with balance laws and we assume they hold in the interior of the cell. The front and rear boundaries of the cell are governed by functions \( f(t) \) and \( r(t) \), respectively. The result is a moving boundary problem (MBP). We now turn to describe the fundamental balance laws of our model.

### 1.3 Viscoelastic Cell Motility Model

We begin with the force balance equation

\[ \sigma_x + b = 0, \]

where \( b \) is a body force and \( \sigma \) is Cauchy stress [6]. Since the stiffness of the substrate affects the amount of force exerted by the cell and the velocity of the cell is directly related to the force, then the velocity of the cell will be dependent on the properties of the substrate [7]. The adhesion to the surface is modeled by a viscous dashpot, as seen in Figure 1.2 and equation (1.2). Thus, our force balance equation is

\[ \sigma_x = \beta v, \]

where \( \beta \) is the effective adhesion viscous drag per unit area, assumed constant in this work unless otherwise stated, and \( v = u_t \) is the velocity of the cell [7]. The coefficient \( \beta \) can be viewed as a coefficient related to the stiffness of the substrate. The change in position of the front and rear boundaries of the cell is modeled by the equations

\[ r'(t) = r_0 + v(r, t), \quad f'(t) = \frac{L_0}{f(t) - r(t)} + v(f, t), \]  

where \( r_0 \) represents a constant retraction rate at the rear and \( L_0/(f(t) - r(t)) \) represents the rate of actin gel protrusion at the front [8]. Using the identities \( \varepsilon_t = v_x = \sigma_{xx}/\beta \) and \( \varepsilon = u_x \), (1.6) becomes

\[ p_1 \dot{\gamma} + q_1 \gamma = p_2 \sigma_{xx}/\beta + q_2 u_x. \]
Assuming the Cauchy stress $\sigma = \gamma + \tau_0$, where $\tau_0$ is the constant contractile stress, the partial differential equation becomes

$$p_1 \sigma_t + q_1 \sigma = p_2 \frac{\sigma_{xx}}{\beta} + q_2 u_x + q_1 \tau_0.$$ (1.9)

Denoting the interior of the cell by $Q_T = \{(x, t) \mid r(t) \leq x \leq f(t), \; t \geq 0\}$, we combine (1.9) and (1.8) with homogeneous boundary conditions for $\sigma(x, t)$ to get our model equations

$$
\begin{align*}
&\begin{cases}
p_1 \gamma_t + q_1 \gamma = p_2 \frac{\sigma_{xx}}{\beta} + q_2 u_x + q_1 \tau_0 & \text{for } (x, t) \in Q_T, \\
\sigma(r, t) = \sigma(f, t) = 0, \\
r'(t) = r_0 + \frac{1}{\beta} \sigma_x(r, t), \\
f'(t) = \frac{L_0}{f(t) - r(t)} + \frac{1}{\beta} \sigma_x(f, t).
\end{cases}
\end{align*}$$ (1.10)
Chapter 2

First-Order PDE Model

The results of Zheltukhin and Lui in [8] modeled the actin gel by a Maxwell element and a Kelvin-Voigt element. In this chapter we explore a numerical approach to solving (1.10).

2.1 Derivation of Model

To numerically solve the MBP, we map the domain $Q_T \rightarrow \{(y,t) | 0 \leq y \leq 1, t \geq 0\}$. This requires a change of variables from $x$ to $y(t)$, where

$$y(t) = \frac{x - r(t)}{l(t)}.$$ 

We make the change of variables and solve the equivalent system in the unit strip.

**Theorem 2.1.** Let $p_1 > 0$. After the change of variables from $x$ to $y(t) = (x - r(t))/l(t)$, the MBP (1.10) is equivalent to the system of equations

$$\begin{align*}
\sigma(y,t) &= \int_0^t F(t - \tau) \left[ \frac{p_2}{p_1\beta l^2(\tau)} \sigma_{yy}(y,\tau) - \sigma_y(y,\tau)g_r(y,\tau) + \frac{p_2}{p_1 l(\tau)} u_y(y,\tau) \right] d\tau + H(t) \\
\sigma(0,t) &= \sigma(1,t) = 0, \\
r'(t) &= r_0 + \frac{1}{\beta l(t)} \sigma_y(0,t), \\
f'(t) &= \frac{L_0}{f(t) - r(t)} + \frac{1}{\beta l(t)} \sigma_y(1,t), \\
\sigma(y,0) &= 0,
\end{align*}$$

(2.1)

**to be solved in the domain** $\{(y,t) | 0 \leq y \leq 1, t \geq 0\}$, **where** $F(t) = e^{-q_1 t/p_1}$ and $H(t) = \tau_0(1 - e^{-q_1 t/p_1})$. 

Proof. We begin with (1.9),

\[ p_1 \sigma_t + q_1 \sigma = \frac{p_2}{\beta} \sigma_{xx} + q_2 u_x + q_1 \tau_0. \]

Making the change of variables to \( y(t) = (x - r(t))/l(t) \), dividing both sides by \( p_1 \) and applying the chain rule yields

\[ \sigma_t + \sigma_y y_t(y, t) + \frac{q_1}{p_1} \sigma = \frac{p_2}{p_1 \beta l^2(t)} \sigma_{yy} + \frac{q_2}{p_1 l(t)} u_y + \frac{q_1}{p_1} \tau_0, \]

where

\[ y_t(y, t) = -(yl'(t) + r'(t))/l(t). \]

Now, using Laplace transform, we simplify both sides and eliminate \( \sigma_t \). Taking the Laplace transform of both sides and using the linearity property of the transform, we have

\[ s L\{\sigma\} - \sigma(y, 0) + L\{\sigma_y y_t(y, t)\} + \frac{q_1}{p_1} L\{\sigma\} = L\left\{ \frac{p_2}{p_1 \beta l^2(t)} \sigma_{yy} \right\} + L\left\{ \frac{q_2}{p_1 l(t)} u_y \right\} + \frac{q_1}{sp_1} \tau_0. \]

Using the fact that \( \sigma(y, 0) = 0 \), this simplifies to

\[ \left(s + \frac{q_1}{p_1}\right) L\{\sigma\} + L\{\sigma_y y_t(y, t)\} = L\left\{ \frac{p_2}{p_1 \beta l^2(t)} \sigma_{yy} \right\} + L\left\{ \frac{q_2}{p_1 l(t)} u_y \right\} + \frac{q_1}{sp_1} \tau_0. \]

Now we divide both sides by \( \left(s + \frac{q_1}{p_1}\right) \) and simplify to get

\[ L\{\sigma\} = \frac{1}{s + \frac{q_1}{p_1}} \left[ L\left\{ \frac{p_2}{p_1 \beta l^2(t)} \sigma_{yy} \right\} - L\{\sigma_y y_t(y, t)\} + L\left\{ \frac{q_2}{p_1 l(t)} u_y \right\} \right] + \frac{q_1 \tau_0}{p_1 s \left(s + \frac{q_1}{p_1}\right)}. \]

Letting

\[ F(t) = L^{-1}\left\{ s + \frac{q_1}{p_1}\right\} = e^{-q_1 t/p_1}, \]

and

\[ H(t) = \frac{q_1 \tau_0}{p_1} L^{-1}\left\{ \frac{1}{s \left(s + \frac{q_1}{p_1}\right)} \right\} = \frac{q_1 \tau_0}{p_1} \cdot \frac{p_1}{q_1} \left(1 - e^{-q_1 t/p_1}\right) = \tau_0 \left(1 - e^{-q_1 t/p_1}\right), \]

we take the inverse Laplace transform of both sides of the above equation to get

\[ \sigma(y, t) = \int_0^t F(t - \tau) \left[ \frac{p_2}{p_1 \beta l^2(\tau)} \sigma_{yy}(y, \tau) - \sigma_y(y, \tau) y_t(y, \tau) + \frac{q_2}{p_1 l(\tau)} u_y(y, \tau) \right] d\tau + H(t). \]

The proof of the first equation is complete.
In the \( y \) variable, \( r = 0 \) and \( f = 1 \), so the boundary conditions become

\[ \sigma(0, t) = \sigma(1, t) = 0. \]

Since

\[ \frac{\partial \sigma}{\partial x} = \frac{\partial \sigma}{\partial y} \cdot \frac{\partial y}{\partial x} = \sigma_y \cdot \frac{1}{l(t)}, \]

the equations for \( r'(t) \) and \( f'(t) \) become

\[ r'(t) = r_0 + \frac{1}{\beta l(t)} \sigma_y(0, t), \quad f'(t) = \frac{L_0}{f(t) - r(t)} + \frac{1}{\beta l(t)} \sigma_y(1, t). \]

The proof of the theorem is complete.

\[ \square \]

### 2.2 Formulation of Numerical Methods

One advantage of mapping the MBP to the unit strip is the simplicity of the necessary numerical methods. We are able to simplify our approach by using a spatially uniform grid. Our discretization in space will be \( 0 = y_0, y_1, \ldots, y_i, \ldots, y_m = 1 \), and the discretization in time is \( 0 = t_0, t_1, \ldots, t_j, \ldots, t_n = T_f \), where \( t_j = j \Delta t \). Throughout this paper we will frequently use the notation \( \sigma_i(t_j) = \sigma(y_i, t_j) \), where \( i = 0, \ldots, m \) and \( j = 0, \ldots, n \).

We rearrange the terms in the result of Theorem 2.1 to a more convenient form,

\[
\sigma(y, t) - \frac{p_2}{p_1 \beta} \int_0^t \frac{F(t - \tau)}{l^2(\tau)} \sigma_{yy}(y, \tau) \, d\tau + \int_0^t F(t - \tau) \sigma_y(y, \tau) y_r(y, \tau) \, d\tau = \frac{p_2}{p_1} \int_0^t \frac{F(t - \tau)}{l(\tau)} u_y(y, \tau) \, d\tau + H(t). \tag{2.4}
\]

We begin by discretizing equation (2.4) at each \((y_i, t_j) \) \( \in \{ (y, t) \mid 0 \leq y \leq 1, t \geq 0 \} \). Since the grid is uniform in space, we use centered difference schemes to approximate the derivatives. The formulas used are

\[
\sigma_y(y_i, t) = \frac{\sigma_{i+1}(t) - \sigma_{i-1}(t)}{2 \Delta y} \quad \text{and} \quad \sigma_{yy}(y_i, t) = \frac{\sigma_{i-1}(t) - 2 \sigma_i(t) + \sigma_{i+1}(t)}{\Delta y^2}.
\]

Note that the above finite difference schemes involve only one neighboring spatial coordinate on each side. This will come into play later when we form a matrix and take advantage of our homogeneous boundary conditions, \( \sigma(0, t) = 0 \) and \( \sigma(1, t) = 0 \).

Substituting the above finite difference formulas in (2.4) yields

\[
\sigma(y_i, t) - \frac{p_2}{p_1 \beta} \int_0^t \frac{F(t - \tau)}{l^2(\tau)} \left( \frac{\sigma_{i-1}(t) - 2 \sigma_i(t) + \sigma_{i+1}(t)}{\Delta y^2} \right) \, d\tau + \int_0^t F(t - \tau) \left( \frac{\sigma_{i+1}(t) - \sigma_{i-1}(t)}{2 \Delta y} \right) y_r(y_i, \tau) \, d\tau = \frac{p_2}{p_1} \int_0^t \frac{F(t - \tau)}{l(\tau)} u_y(y_i, \tau) \, d\tau + H(t). \tag{2.5}
\]
To approximate the two integrals on the left-hand side of (2.5) we apply the composite trapezoid rule. For arbitrary function \( f(t) \), the composite trapezoid rule states that

\[
\int_0^{t_j} f(\tau) \, d\tau \approx \frac{\Delta t}{2} f(0) + \Delta t \sum_{k=1}^{j-1} f(k\Delta t) + \frac{\Delta t}{2} f(t_j).
\]

Applying this formula to each integral and applying the homogeneous boundary conditions \( \sigma(0, t) = \sigma(1, t) = 0 \) yields

\[
\sigma_i(t_j) = \frac{p_2\Delta t}{p_1\beta} \sum_{k=1}^{j-1} \frac{F(t_j - t_k)}{l^2(t_k)} \left( \frac{\sigma_{i-1}(t_k) - 2\sigma_i(t_k) + \sigma_{i+1}(t_k)}{\Delta y^2} \right) - \frac{p_2 F(0)\Delta t}{2p_1l^2(t_j)} \left( \frac{\sigma_{i-1}(t_j) - 2\sigma_i(t_j) + \sigma_{i+1}(t_j)}{\Delta y^2} \right) + \Delta t \sum_{k=1}^{j-1} F(t_j - t_k) \left( \frac{\sigma_{i+1}(t_k) - \sigma_{i-1}(t_k)}{2\Delta y} \right) y_t(y_i, t_k) + \Delta t \frac{F(0)}{2} \left( \frac{\sigma_{i+1}(t_j) - \sigma_{i-1}(t_j)}{2\Delta y} \right) y_t(y_i, t_j) - \int_0^{t_j} \frac{p_2}{p_1(\tau)} F(t_j - \tau) u_y(y_i, \tau) \, d\tau + H(t_j),
\]

where the remaining integral of \( u_y \) can be evaluated numerically. For simplicity we do not expand this integral.

Now the partial differential equation has been reduced to a system of coupled linear equations of the form \( A\vec{\sigma} = \vec{b} \) of the form

\[
\begin{bmatrix}
M_{1,1} & 0 & 0 & \ldots & 0 \\
M_{2,1} & M_{2,2} & 0 & \ldots & 0 \\
M_{3,1} & M_{3,2} & M_{3,3} & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
M_{n,1} & M_{n,2} & \ldots & M_{n,n-1} & M_{n,n}
\end{bmatrix}
\begin{bmatrix}
\overrightarrow{\sigma}_1 \\
\overrightarrow{\sigma}_2 \\
\vdots \\
\overrightarrow{\sigma}_{n-1} \\
\overrightarrow{\sigma}_n
\end{bmatrix}
= \begin{bmatrix}
\overrightarrow{b}_1 \\
\overrightarrow{b}_2 \\
\vdots \\
\overrightarrow{b}_{n-1} \\
\overrightarrow{b}_n
\end{bmatrix},
\]

(2.7)

where each \( M_{j,k} \), for \( j = 1, \ldots, n \) and \( k = 1, \ldots, j \), is a tridiagonal matrix and

\[
\overrightarrow{\sigma}_j = (\sigma_1(t_j), \sigma_2(t_j), \ldots, \sigma_i(t_j), \ldots, \sigma_{m-1}(t_j))^T.
\]

The vector \( \overrightarrow{b} \) is composed of \((m-1) \times 1\) vectors defined by

\[
\overrightarrow{b}_j = \begin{bmatrix}
\int_0^{t_j} \frac{p_2}{p_1(\tau)} F(t_j - \tau) u_y(y_1, \tau) \, d\tau + H(t_j) \\
\int_0^{t_j} \frac{p_2}{p_1(\tau)} F(t_j - \tau) u_y(y_2, \tau) \, d\tau + H(t_j) \\
\vdots \\
\int_0^{t_j} \frac{p_2}{p_1(\tau)} F(t_j - \tau) u_y(y_{m-2}, \tau) \, d\tau + H(t_j) \\
\int_0^{t_j} \frac{p_2}{p_1(\tau)} F(t_j - \tau) u_y(y_{m-1}, \tau) \, d\tau + H(t_j)
\end{bmatrix}.
\]

8
The matrix $A$ is made up of $n \times n$ blocks, where each block $M_{j,k}$ is a $(m-1) \times (m-1)$ matrix. Each block above the diagonal ($M_{j,k}$, where $j < k$) is a zeros matrix.

To construct the blocks of $A$, we must consider the blocks on the diagonal separate from the blocks below the diagonal. The blocks on the diagonal contain coefficients on the left-hand side of (2.6) terms that are not in a summation. The coefficients $D_{1,i,j}^k$ and $D_{2,i,j}^k$ are the coefficients on the first and second derivatives respectively and are defined as

$$D_{1,i,j}^k = \frac{\Delta t}{4\Delta y} F(0)y_t(y_i, t_j),$$

$$D_{2,i,j}^k = -\frac{p_2\Delta t}{2p_1\beta l^2(t_j)\Delta y^2} F(0).$$

(2.8)

Note that these coefficients contain the factors of $\Delta y$ in the denominator of the finite difference schemes. Then an additional 1 appears along the diagonal from the first $\sigma_i(t_j)$ term on the left-hand side and the blocks along the diagonal are defined by

$$M_{j,j} = \begin{bmatrix} 1 - 2D_{1,j}^k + D_{1,j}^k & D_{1,j}^k + D_{2,j}^k & \cdots & 0 \\ -D_{2,j}^k + D_{2,j}^k & 1 - 2D_{2,j}^k & D_{1,j}^k + D_{2,j}^k & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -D_{m-2,j}^k + D_{m-2,j}^k & 1 - 2D_{m-2,j}^k & D_{m-2,j}^k + D_{m-2,j}^k \\ 0 & \cdots & \vdots & \vdots & \vdots \end{bmatrix},$$

where $j = 1, 2, \ldots, n$.

All blocks below the diagonal are defined similarly, but only contain the coefficients on the terms on the left-hand side of (2.6) that are inside a summation. These coefficients are defined by

$$C_{1,i,j}^k = \frac{\Delta t}{2\Delta y} F(t_j - t_k)y_t(y_i, t_k),$$

$$C_{2,i,j}^k = -\frac{p_2\Delta t}{p_1\beta l^2(t_k)\Delta y^2} F(t_j - t_k).$$

Then each tridiagonal block below the diagonal is defined by

$$M_{j,k} = \begin{bmatrix} -2C_{1,j}^k + C_{1,j}^k & C_{1,j}^k + C_{2,j}^k & \cdots & 0 \\ -C_{2,j}^k + C_{2,j}^k & -2C_{2,j}^k & C_{2,j}^k + C_{2,j}^k & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -C_{m-2,j}^k + C_{m-2,j}^k & -2C_{m-2,j}^k & C_{m-2,j}^k + C_{m-2,j}^k \\ 0 & \cdots & \vdots & \vdots & \vdots \end{bmatrix},$$

for $k = 1, 2, \ldots, j-1$. 

9
Now that we have constructed each component of (2.7), we can solve the system $A\tilde{\sigma} = \tilde{b}$ for all $\sigma(y_i, t_j)$, where $(y_i, t_j)$ is in the interior of the cell.

Recall that $u_t = \sigma_x/\beta$ and $y_t(y, t) = -(yl(t) + rl(t))/l(t)$, where $l(t) = f'(t) - r'(t)$. To solve for $u(y, t)$, we consider the identity $u_t(x, t) = \sigma_x(x, t)/\beta$. Changing variables yields

$$u_t + uy_y = \frac{1}{\beta l(t)} \sigma_y. \tag{2.9}$$

We apply a first-order upwind scheme to solve this first-order PDE [2]. In order to find the value of $u(1, t + \Delta t)$, the upwind method requires us to know $u(1 + \Delta y, t)$, which is approximated by $u_r = f'(t) \cdot \Delta t$. Similarly, to find $u(0, t + \Delta t)$, we must know $u(0 - \Delta y, t)$, which is approximated by $u_l = -r'(t) \cdot \Delta t$. The newly computed $\sigma(y, t)$ is used on the right-hand side of (2.9) and a centered difference is used to approximate its derivative. Since the points used in the difference scheme depend on the sign of $y_t(y, t)$, we define

$$a^+(y_i) = \max(y_t(y_i, t_j), 0) \quad \text{and} \quad a^-(y_i) = \min(y_t(y_i, t_j), 0).$$

Using these coefficients, we apply first-order finite difference formulas to the derivatives on $u(y, t)$, giving us the equation

$$\frac{u(y_i, t_j) - u(y_i, t_{j-1})}{\Delta t} + a^+_i(y_i) \frac{u(y_i, t_j) - u(y_{i-1}, t_j)}{\Delta y} - a^-_i(y_i) \frac{u(y_{i+1}, t_j) - u(y_i, t_j)}{\Delta y} = \frac{\Delta t}{\beta l(t_j)} \sigma_y(y_i, t_j)$$

For $1 \leq i \leq m - 1$, this equation can be rearranged to

$$-\frac{\Delta t}{\Delta y} a^+_i(y_i) u(y_{i-1}, t_j) + \left[ 1 + \frac{\Delta t}{\Delta y} (a^+_i(y_i) + a^-_i(y_i)) \right] u(y_i, t_j) - \frac{\Delta t}{\Delta y} a^-_i(y_i) u(y_{i+1}, t_j) = \frac{\Delta t}{\beta l(t_j)} \sigma_y(y_i, t_j) + u(y_i, t_{j-1}) \tag{2.10}$$

If we define the coefficients

$$\alpha^D_{i,j} = 1 + \frac{\Delta t}{\Delta y} \left( a^+_i(y_i) + a^-_i(y_i) \right), \quad \alpha^1_{i,j} = -\frac{\Delta t}{\Delta y} a^-_i(y_i), \quad \text{and} \quad \alpha^{-1}_{i,j} = -\frac{\Delta t}{\Delta y} a^+_i(y_i),$$

then converting (2.10) to a system of equations $U_j u_j = \tilde{b}_j$, the matrix $U_j$ on the left-hand side becomes

$$U_j = \begin{bmatrix}
\alpha^D_{0,j} & \alpha^1_{0,j} & \cdots & 0 \\
\alpha^{-1}_{1,j} & \alpha^1_{1,j} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \alpha^{-1}_{m-1,j} & \alpha^D_{m-1,j} \\
0 & \cdots & \alpha^{-1}_{m,j} & \alpha^D_{m,j}
\end{bmatrix}.$$
Additionally, we have the vectors

\[
\vec{u}_j = \begin{pmatrix}
  u(y_0, t_j) \\
  u(y_1, t_j) \\
  \vdots \\
  u(y_{m-1}, t_j) \\
  u(y_m, t_j)
\end{pmatrix}
\]

and

\[
\vec{b}_j = \begin{pmatrix}
  \frac{\Delta t}{\beta_l(t_j)} \sigma_y(y_0, t_j) + u(y_0, t_{j-1}) + \frac{\Delta t}{\Delta y} \max(y_t(y_0, t_j), 0) u_l(t_j) \\
  \frac{\Delta t}{\beta_l(t_j)} \sigma_y(y_1, t_j) + u(y_1, t_{j-1}) \\
  \vdots \\
  \frac{\Delta t}{\beta_l(t_j)} \sigma_y(y_{m-1}, t_j) + u(y_{m-1}, t_{j-1}) \\
  \frac{\Delta t}{\beta_l(t_j)} \sigma_y(y_m, t_j) + u(y_m, t_{j-1}) - \frac{\Delta t}{\Delta y} \min(y_t(y_m, t_j), 0) u_r(t_j)
\end{pmatrix},
\]

where \(u_l(t_j)\) and \(u_r(t_j)\) are the extrapolated values of \(u(-\Delta y, t_j)\) and \(u(1 + \Delta y, t_j)\), respectively. Finally, solving the system \(U_j \vec{u}_j = \vec{b}_j\) yields \(u(y_0 : y_m, t_j)\). Repeating this for all \(j = 1, 2, \ldots, n\) gives us \(u(y, t)\) for each \((y, t)\) in the unit strip.

Since \(u(y, t)\) and \(\sigma(y, t)\) are coupled, we need to use a fixed-point method. We venture initial guesses for \(u(y, t)\) and \(y_t(y, t)\) for all \((y, t)\) in the unit strip in order to solve for \(\sigma(y, t)\). Then we use the newly computed \(\sigma(y, t)\) to update \(u(y, t)\) using the upwind scheme described above. The position of the boundary points \(f(t)\), and \(r(t)\) are then calculated using the new \(\sigma(y, t)\), allowing us to update \(y_t(y, t)\). The cycle becomes a fixed point problem, as this cycle is repeated until the differences between the most recent and previous versions of \(\sigma(y, t)\) and \(u(y, t)\) are sufficiently small.

Pseudocode for each algorithm is provided below. The formation of the matrix \(A\) and vector \(\vec{b}\) is included in the SolveForSigma algorithm, where we compute \(\sigma(y, t)\) at all interior gridpoints. Methods for the upwind scheme are written in the SolveForU algorithm. The loop controlling the fixed point iteration is outlined in the Driver algorithm.
Algorithm 1: SolveForSigma

\textbf{input}: \( p_1, p_2, q_1, q_2, y_t(y, t), u_y(y, t), l(t), \beta, \tau_0, \Delta y, \Delta t \)

\textbf{output}: \( \sigma(y, t) \) for all \((y, t)\) in the unit strip.

\begin{algorithmic}
\For {k = 1 : N}
\State Define coefficient on first-order derivatives
\State \( D_{1j}^1 = \frac{\Delta t}{4\Delta y} F(0) y_t(y_1 : y_{m-1}, t_j) \);
\State Create submatrix of first-order derivative coefficients
\State \( M_1 = \text{diag} \left( D_{1j}^1(y_1 : y_{m-2}, 1) - \text{diag} \left( D_{1j}^1(y_2 : y_{m-1}, -1) \right) \right) \);
\State Define coefficient on second-order derivatives
\State \( D_{i,j}^2 = -\frac{p_2 \Delta t}{2p_1 \beta l(t_j) \Delta y^2} F(0) \cdot \text{ones}(m-1, 1) \);
\State Create submatrix of second-order derivative coefficients
\State \( M_2 = \text{diag} \left( -2D_{j}^2(y_1 : y_{m-1}) \right) + \text{diag} \left( D_{j}^2(y_1 : y_{m-2}, 1) \right) + \text{diag} \left( D_{j}^2(y_2 : y_{m-1}, -1) \right) \);
\State Store submatrix in correct block in lower-triangular A
\State \( M_{j,j} = I_{M-1} + M_1 + M_2 \);
\EndFor
\For {j = 1 : k}
\State Define coefficient on first-order derivatives
\State \( C_{i,j}^{1,k} = \frac{\Delta t}{2\Delta y} F(t_j - t_k) y_t(y_1 : y_{m-1}, t_k) \);
\State Create submatrix of first-order derivative coefficients
\State \( M_1 = \text{diag} \left( D_{1j}^1(y_1 : y_{m-2}, 1) - \text{diag} \left( D_{1j}^1(y_2 : y_{m-1}, -1) \right) \right) \);
\State Define coefficient on second-order derivatives
\State \( C_{i,j}^{2,k} = -\frac{p_2 \Delta t}{2p_1 \beta l(t_j) \Delta y^2} F(t_j - t_k) \cdot \text{ones}(m-1, 1) \);
\State Create submatrix of second-order derivative coefficients
\State \( M_2 = \text{diag} \left( -2D_{j}^2(y_1 : y_{m-1}) \right) + \text{diag} \left( D_{j}^2(y_1 : y_{m-2}, 1) \right) + \text{diag} \left( D_{j}^2(y_2 : y_{m-1}, -1) \right) \);
\State Store submatrix in correct block in lower-triangular A
\State \( M_{j,k} = M_1 + M_2 \);
\EndFor
\State \( \vec{b}_j = \int_{0}^{t_j} \frac{p_2}{\rho l(\tau)} F(t_j - \tau) u_y(y_1 : y_{m-1}, \tau) d\tau + H(t_j) \);
\EndFor
\State Solve \( A\vec{\sigma} = \vec{b} \) for \( \vec{\sigma}(y, t) \) and reshape to \( m - 1 \) by \( n \) matrix;
\State Concatenate with zeros for initial/boundary conditions at \( y = 0, y = 1 \) and \( t = 0 \);
\end{algorithmic}
**Algorithm 2: Driver**

**input**: $E_0$, $E_1$, $\mu_0$, $\mu_1$, $r_0$, $L_0$, $\beta$, $\tau_0$, $T_{final}$, $\Delta y$, $\Delta t$

**output**: $\sigma(y, t)$, $u(y, t)$, $l(t)$ for all space and time.

Define $p_1$, $p_2$, $q_1$, and $q_2$ in terms of $E_0$, $E_1$, $\mu_0$, $\mu_1$;

Initialize $\sigma(y, t)$, $u(y, t)$, $y_l(y, t)$, $y_r(y, t)$;

while $||\sigma(y, t) - \sigma_{old}(y, t)|| > Tol$ or $||u(y, t) - u_{old}(y, t)|| > Tol$ do

  Compute $u_y$;
  Assign $\sigma_{old}(y, t) = \sigma(y, t)$;
  Solve $\sigma(y, t) = SolveForSigma$;
  $f'(t) = \frac{L_0}{l(t)} + \frac{\sigma_{y}(1,t)}{\beta(t)}$;
  $r'(t) = r_0 + \frac{\sigma_{y}(1,t)}{\beta(t)}$;

  for $j = 1 : N - 1$ do
    if $j == 1$ then
      $r(t_{j+1}) = r(t_j) + \Delta t \cdot r'(t_j)$;
      $f(t_{j+1}) = f(t_j) + \Delta t \cdot f'(t_j)$;
    else
      $r(t_{j+1}) = \frac{(2\Delta t \cdot r'(t_j) + 4r(t_j) - r(t_{j-1}))}{3}$;
      $f(t_{j+1}) = \frac{(2\Delta t \cdot f'(t_j) + 4f(t_j) - f(t_{j-1}))}{3}$;
    end

  Update $y_l(y, t) = -(y(f'(t) - r'(t)) + r'(t))/l(t)$;
  Assign $u_{old}(y, t) = u(y, t)$;
  Solve $u(y, t) = SolveForU$;

end

**Algorithm 3: SolveForU**

**input**: $y_l(y, t)$, $\sigma_y(y, t)$, $\beta$, $l(t)$, $y_l$, $y_r$, $\Delta y$, $\Delta t$

**output**: $u(y, t)$ for all $(y, t)$ in the unit strip

for $j=1:n$ do

  Separate coefficients on spatial derivative
  $a^+ = \max(y_l(y_0 : y_m, t_j), 0)$;
  $a^- = \min(y_l(y_0 : y_m, t_j), 0)$;

  Form the matrix $A$
  $A = \text{diag}(1 + \frac{\Delta t}{\Delta y}(a^+(y_0 : y_m) - a^-(y_0 : y_m))) + \text{diag}(\frac{\Delta t}{\Delta y}a^-(y_0 : y_{m-1}), 1)$
  $+ \text{diag}(-\frac{\Delta t}{\Delta y}a^+(y_1 : y_m), -1)$

  $b = u(y_0 : y_m, t_{j-1}) + \Delta t \frac{\sigma_{y}(y_l(y_m, t_j))}{\beta(t_j)}$

  $b(y_0) = b(y_0) + \frac{\Delta t}{\Delta y} \max(y_l(y_0, t_j), 0) \cdot u_l(t_j)$;

  $b(y_m) = b(y_m) - \frac{\Delta t}{\Delta y} \max(y_l(y_m, t_j), 0) \cdot u_r(t_j)$;

  Solve $Au = b$ for $u(y_0 : y_m, t_j)$;
end
2.3 Analysis of the Steady State Equation

Prior to observing the numerical approximation to the solution, we will explore a possible steady-state solution. In [8], the existence of a traveling cell solution \( \tilde{\sigma}(x-kt) \) is proved, where \( k \) is the speed of the cell at steady state. We also note that, at steady state, \( u_x(x,t) = 0 \), since the cell is moving at a constant rate \( k \) and the change in cell length \( l'(t) = 0 \).

For a steady state length we have \( l'(t) = 0 \), since \( \lim_{t \to \infty} v(f,t) = \lim_{t \to \infty} v(r,t) \). Therefore,

\[
l'(t) = f'(t) - r'(t) = \frac{L_0}{l_{ss}} - r_0 = 0,
\]

and we find that the steady state length of the cell is \( l_{ss} = L_0/r_0 \).

Since the length of the cell does not change and the cell is traveling at a constant rate, we find this rate by considering \( r'(t) = r_0 + \sigma_x(x,t)/\beta \). As seen in Table 2.1, typical values of \( \beta \) are significantly larger than values of \( \sigma_x(x,t) \). Although \( \sigma_x(x,t) \) is not zero, \( k \) can be very well approximated by \( k \approx r_0 \).

So we substitute \( \tilde{\sigma}(x-kt) = \tilde{\sigma}(z) \) and \( u_x = 0 \) into (1.9) yielding

\[
-kp_1 \tilde{\sigma}'(z) + q_1 \tilde{\sigma}(z) = \frac{p_2}{\beta} \tilde{\sigma}''(z) + q_1 \tau_0,
\]

for \( 0 \leq z \leq l_{ss} \). Now we make the change of variables from \( z \in [0,l_{ss}] \) to \( y \in [0,1] \). So \( y = \frac{1}{l_{ss}} z \), implying that \( \tilde{\sigma}(z) = \sigma_{ss}(y) \) and \( \tilde{\sigma}'(z) = \frac{1}{l_{ss}} \sigma'_{ss}(y) \). Substituting each into (2.11) and simplifying,

\[
\sigma''_{ss}(y) + \frac{p_1 \beta l_{ss} k}{p_2} \sigma'_{ss}(y) - \frac{q_1 \beta l_{ss}^2}{p_2} \sigma_{ss}(y) = -\frac{q_1 \beta l_{ss}^2}{p^2} \tau_0,
\]

for \( 0 \leq y \leq 1 \). We rewrite this equation as

\[
\sigma''_{ss}(y) + b \sigma'_{ss}(y) - c \sigma_{ss}(y) = -c \tau_0,
\]

where \( b = \frac{p_1 \beta l_{ss} k}{p_2} \) and \( c = \frac{q_1 \beta l_{ss}^2}{p_2} \).

Solving this ODE yields

\[
\sigma_{ss}(y) = c_1 e^{r_y y} + c_2 e^{-r_y y} + \tau_0,
\]

where \( r_\pm = \frac{1}{2} (-b \pm \sqrt{b^2 + 4c}) \). Applying boundary conditions \( \sigma_{ss}(0) = \sigma_{ss}(1) = 0 \), we solve for \( c_1 \) and \( c_2 \) to get

\[
c_1 = -\tau_0 \left( \frac{e^{r_-} - 1}{e^{r_-} - e^{r_+}} \right) \quad \text{and} \quad c_2 = -\tau_0 \left( \frac{1 - e^{r_+}}{e^{r_-} - e^{r_+}} \right).
\]

Therefore, as time increases, we should see the solution tends toward this steady state for all first order models.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\beta$</th>
<th>$\tau_0$</th>
<th>$r_0$</th>
<th>$L_0$</th>
<th>$E_0$</th>
<th>$E_1$</th>
<th>$\mu_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>5350.4</td>
<td>179.17</td>
<td>0.8</td>
<td>1</td>
<td>10288</td>
<td>10288</td>
<td>53015</td>
</tr>
</tbody>
</table>

Table 2.1: Parameter values for numerical simulation of standard linear model.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>$p_1$</th>
<th>$q_1$</th>
<th>$p_2$</th>
<th>$q_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>$\mu_0/E_0$</td>
<td>1</td>
<td>$\mu_0(1 + E_1/E_0)$</td>
<td>$E_1$</td>
</tr>
<tr>
<td>Value</td>
<td>5.1531</td>
<td>1</td>
<td>106030</td>
<td>10288</td>
</tr>
</tbody>
</table>

Table 2.2: Values for coefficients in (2.1).

### 2.4 Numerical Solutions of the Standard Linear Model

Using the algorithm described above, we solve the system of equations (2.1) where $p_1$, $q_1$, $p_2$ and $q_2$ are chosen consistent with the standard linear model governed by (1.5). The parameter values used are given in Table 2.1. These parameter values translate to the values of $p_1$, $p_2$, $q_1$ and $q_2$ found in Table 2.2.

Figure 2.1 shows the stress for all $(y,t)$ in the unit strip. Initially the stress increases rapidly, but then reaches a maximum and declines below the steady state. This behavior is clear in Figure 2.2. The stress then increases monotonically and appears to approach the steady state described by (2.12). Figure 2.3 compares the steady state solution to the stress at the final time of the simulation, $t = 10$.

We now turn our attention to the displacement $u(y,t)$ pictured in Figure 2.4. As expected, $u(y,t)$ approaches a constant steady state, hence $u_{ss}'(y) = 0$. Because of this, the front and rear also move at a constant rate.

In Figure 2.5, we see the length of the cell initially at 1, then widens to the steady state length, $l_{ss} = L_0/r_0$. In this case the $L_0 = 1$ and $r_0 = 0.8$, resulting in $l_{ss} = 1.25$. Also, the slope of the two parallel lines is the speed the cell is traveling at steady state.
Figure 2.1: Surface plot of $\sigma(y,t)$ for the standard linear model.

Figure 2.2: Side view of the surface plot of $\sigma(y,t)$ for the standard linear model.
Figure 2.3: Plot of the steady state solution $\sigma_{ss}(y)$ and $\sigma(y, T_{\text{final}})$.

Figure 2.4: Surface plot of the displacement $u(y, t)$ for the standard linear model.
Figure 2.5: Plot of $r(t)$ and $f(t)$ for the standard linear model.
Chapter 3

Second-Order PDE Model

The general constitutive law considered in Chapter 2 confines us to a relatively small set of viscoelastic models. In this chapter we consider a much wider range of viscoelastic models with constitutive laws of the form

\[ r_1 \gamma_{tt} + p_1 \gamma_t + q_1 \gamma = r_2 \varepsilon_{tt} + p_2 \varepsilon_t + q_2 \varepsilon. \]  (3.1)

The numerical methods used to solve equations of this form are very similar to those in Chapter 2.

3.1 Examples of Second-Order Models

A classic example of a model with a constitutive law of the form (3.1) is the Burger model [6], depicted in Figure 3.1. This model can be viewed as a combination of a Kelvin-Voigt and a Maxwell element in series, and the constitutive law is derived as follows. Recall that for a combination of elements in series \( \varepsilon = \varepsilon_0 + \varepsilon_1 \) and \( \gamma = \gamma_0 = \gamma_1 \) and for the Kelvin-Voigt model (equation (1.4))

\[ \dot{\varepsilon}_0 = \frac{1}{E_0} \dot{\gamma}_0 - \frac{\mu_0}{E_0} \ddot{\varepsilon}_0, \]

and for the Maxwell model (equation (1.3))

\[ \dot{\varepsilon}_1 = \frac{1}{E_1} \dot{\gamma}_1 + \frac{1}{\mu_1} \gamma_1. \]

Therefore,

\[ \dot{\varepsilon} = \dot{\varepsilon}_0 + \dot{\varepsilon}_1 = \frac{1}{E_0} \dot{\gamma}_0 - \frac{\mu_0}{E_0} \ddot{\varepsilon}_0 + \frac{1}{E_1} \dot{\gamma}_1 + \frac{1}{\mu_1} \gamma_1 \]

\[ = \frac{1}{E_0} \dot{\gamma} - \frac{\mu_0}{E_0} (\ddot{\varepsilon} - \ddot{\varepsilon}_0) + \frac{1}{E_1} \dot{\gamma} + \frac{1}{\mu_1} \gamma \]

\[ = \frac{1}{E_0} \dot{\gamma} - \frac{\mu_0}{E_0} \left( \ddot{\varepsilon} - \frac{1}{E_1} \ddot{\gamma} - \frac{1}{\mu_1} \dot{\gamma} \right) + \frac{1}{E_1} \dot{\gamma} + \frac{1}{\mu_1} \gamma. \]  (3.2)
Figure 3.1: Illustration of (a) Burger model and (b) Kelvin-Voigt and dashpot in series.

Then (3.2) simplifies to our final constitutive law,

$$\frac{\mu_0 \mu_1}{E_0 E_1} \ddot{\gamma} + \left( \frac{\mu_1}{E_0} + \frac{\mu_0}{E_1} + \frac{\mu_0}{E_0} \right) \dot{\gamma} + \gamma = \frac{\mu_0 \mu_1}{E_0} \dot{\varepsilon} + \mu_1 \varepsilon. \quad (3.3)$$

Another example is found by connecting a Kelvin-Voigt element to a dashpot in series. To derive the constitutive law for this system, we combine

$$\gamma_0 = \mu_0 \dot{\varepsilon}_0 + E_0 \varepsilon_0$$

and

$$\gamma_1 = \mu_1 \dot{\varepsilon}_1,$$

the stress-strain relationships for a Kelvin-Voigt element and a dashpot, respectively. Since they are connected in series, $\gamma = \gamma_0 = \gamma_1$ and $\varepsilon = \varepsilon_0 + \varepsilon_1$. Therefore,

$$\dot{\varepsilon} = \frac{1}{E_0} \dot{\gamma} - \frac{\mu_0}{E_0} \dot{\varepsilon}_0 + \frac{1}{\mu_1} \dot{\gamma}$$

$$= \frac{1}{E_0} \dot{\gamma} - \frac{\mu_0}{E_0} \dot{\varepsilon}_0 + \frac{\mu_0}{E_0} \dot{\varepsilon}_1 + \frac{1}{\mu_1} \dot{\gamma}$$

$$= \frac{1}{E_0} \dot{\gamma} - \frac{\mu_0}{E_0} \dot{\varepsilon} + \frac{\mu_0}{\mu_1 E_0} \dot{\gamma} + \frac{1}{\mu_1} \dot{\gamma}$$

Simplifying this result yields our final constitutive law,

$$\frac{\mu_1}{E_0} \left( 1 + \frac{\mu_0}{\mu_1} \right) \dot{\gamma} + \gamma = \mu_1 \dot{\varepsilon} + \frac{\mu_0 \mu_1}{E_0} \dot{\varepsilon}. \quad (3.4)$$

The diagram for this model is also shown in Figure 3.1.

We will now look to transform a general constitutive law of the form (3.1) to an integral equation similar to the equation derived in Chapter 2.
3.2 Derivation of Model

**Theorem 3.1.** Using the change of variables \( y = (x - r(t))/l(t) \), the partial differential equation (3.1) can be transformed into the integral equation

\[
\begin{align*}
\sigma(y, t) + 2r_1K(0)y_t(y, t)\sigma_y(y, t) &- \frac{r_2K(0)}{\beta l^2(t)}\sigma_{yy}(y, t) \\
+ \int_0^t K(t - \tau) &\left[ p_1y_{\tau}(y, \tau) - r_1 \left( y_{\tau\tau}(y, \tau) - \frac{l'(\tau)}{l(\tau)} y_{\tau}(y, \tau) \right) \right] \sigma_y(y, \tau) d\tau \\
+ \int_0^t K'(t - \tau) &\left[ 2r_1y_{\tau}(y, \tau)\sigma_y(y, \tau) - \frac{r_2}{\beta l^2(\tau)} \sigma_{yy}(y, \tau) \right] d\tau \\
+ \int_0^t K(t - \tau) &\left[ -r_1y^2_{\tau}(y, \tau) - \frac{p_2}{\beta l^2(\tau)} \right] \sigma_{yy}(y, \tau) d\tau \\
= q_2 \int_0^t \frac{K(t - \tau)}{l(\tau)} u_y(y, \tau) d\tau + \tau_0 J(t) + r_1K(t)\sigma_t(y, 0)
\end{align*}
\]

where

\[
K(t) = \frac{1}{\sqrt{p_1^2 - 4q_1r_1}}(e^{s_1t} - e^{s_2t}), \quad J(t) = 1 - \frac{p_1}{2\sqrt{p_1^2 - 4q_1r_1}}(e^{s_1t} - e^{s_2t}) - \frac{1}{2}(e^{s_1t} + e^{s_2t})
\]

\[
s_1 = \frac{-2q_1}{p_1 + \sqrt{p_1^2 - 4q_1r_1}} \quad \text{and} \quad s_2 = \frac{-2q_1}{p_1 - \sqrt{p_1^2 - 4q_1r_1}}
\]

**Proof.** Beginning with the partial differential equation and, since \( \sigma = \gamma + \tau_0 \),

\[
r_1\sigma_{tt} + p_1\sigma_t + q_1(\sigma - \tau_0) = r_2\varepsilon_{tt} + p_2\varepsilon_t + q_2\varepsilon,
\]

we substitute \( \varepsilon_t = \sigma_{xx}/\beta \) and \( \varepsilon = u_x \) to get

\[
r_1\sigma_{tt} + p_1\sigma_t + q_1(\sigma - \tau_0) = \frac{r_2}{\beta}(\sigma_{xx})_t + \frac{p_2}{\beta}\sigma_{xx} + q_2u_x.
\]

Then we make the change of variables to \( y = (x - r(t))/l(t) \). This results in

\[
r_1[\sigma_{tt} + \sigma_{ty}y_t + (\sigma_yy_t)_t] + p_1(\sigma_t + \sigma_{yy}y_t) + q_1(\sigma - \tau_0) = \frac{r_2}{\beta}\left( \frac{1}{l^2(t)}\sigma_{yy} \right)_t + \frac{p_2}{\beta}\frac{\sigma_{yy}}{l^2(t)} + q_2\frac{u_y}{l(t)},
\]

where \( \sigma = \sigma(y, t) \), \( u = u(y, t) \), and \( y_t = y_t(y, t) = -(yl'(t) + r'(t))/l(t) \). Next apply the Laplace transform to both sides. Assuming \( \sigma_{ty} = \sigma_{yt} \), this yields

\[
r_1\left( s^2\mathcal{L}\{\sigma\} - s\sigma(y, 0) - \sigma_t(y, 0) \right) + r_1\mathcal{L}\{\sigma_{yt}\} + r_1\mathcal{L}\{(\sigma_{yy}y_t)_t\} + p_1(s\mathcal{L}\{\sigma\} - \sigma(y, 0))
\]

\[
+ p_1\mathcal{L}\{\sigma_{yy}\} + q_1\mathcal{L}\{\sigma\} = \frac{r_2}{\beta}\mathcal{L}\left\{ \frac{1}{l^2(t)}\sigma_{yy} \right\} + \frac{p_2}{\beta}\mathcal{L}\left\{ \frac{\sigma_{yy}}{l^2(t)} \right\} + q_2\mathcal{L}\left\{ \frac{u_y}{l(t)} \right\} + q_1\frac{\tau_0}{s}.
\]
The initial condition \( \sigma(y,0) \) is assumed to be zero, although \( \sigma_t(y,0) \) is not. Additionally, \( (\sigma_y)_t = \sigma_{yt} + \sigma_{yy} y_t \), so \( \sigma_{yt} = (\sigma_y)_t - \sigma_{yy} y_t \). Now we may rearrange terms and reduce the above equation to get

\[
(r_1 s^2 + p_1 s + q_1) \mathcal{L} \{ \sigma \} + r_1 \mathcal{L} \{ (\sigma_y)_t - \sigma_{yy} y_t \} + r_1 \mathcal{L} \{ (\sigma_y y_t)_t \} + p_1 \mathcal{L} \{ \sigma_y y_t \} - r_1 \sigma_t(y,0) - \frac{r_2}{\beta} \mathcal{L} \left\{ \frac{1}{l^2(t)} \sigma_{yy} \right\}_t - \frac{p_2}{\beta} \mathcal{L} \left\{ \frac{\sigma_{yy}}{l^2(t)} \right\} = q_2 \mathcal{L} \left\{ \frac{u_y}{l(t)} \right\} + q_1 \tau_0 \frac{s}{s}.
\]

Dividing both sides by \((r_1 s^2 + p_1 s + q_1)\) and simplifying, we have

\[
\mathcal{L} \{ \sigma \} + \frac{1}{r_1 s^2 + p_1 s + q_1} \left[ r_1 \mathcal{L} \{ (\sigma_y)_t - \sigma_{yy} y_t \} + r_1 \mathcal{L} \{ (\sigma_y y_t)_t \} + p_1 \mathcal{L} \{ \sigma_y y_t \} - r_1 \sigma_t(y,0) - \frac{r_2}{\beta} \mathcal{L} \left\{ \frac{1}{l^2(t)} \sigma_{yy} \right\}_t - \frac{p_2}{\beta} \mathcal{L} \left\{ \frac{\sigma_{yy}}{l^2(t)} \right\} \right] = \frac{1}{r_1 s^2 + p_1 s + q_1} \left[ q_2 \mathcal{L} \left\{ \frac{u_y}{l(t)} \right\} + q_1 \tau_0 \frac{s}{s} \right].
\]

When we take the inverse Laplace transform of both sides, the result is the sum of convolutions, each with the same kernel \( K(t) \) defined as the inverse Laplace transform of \( 1/(r_1 s^2 + p_1 s + q_1) \). Define \( s_1 \) and \( s_2 \) to be the positive and negative roots of the denominator, respectively. Then the roots can be written equivalently as

\[
s_1 = \frac{-2q_1}{p_1 + \sqrt{p_1^2 - 4q_1 r_1}}, \quad s_2 = \frac{-2q_1}{p_1 - \sqrt{p_1^2 - 4q_1 r_1}},
\]

and the kernel is defined as

\[
K(t) = \mathcal{L}^{-1} \left\{ \frac{1}{r_1 s^2 + p_1 s + q_1} \right\} = \frac{1}{\sqrt{p_1^2 - 4q_1 r_1}} \left( e^{s_1 t} - e^{s_2 t} \right).
\]

The function \( J(t) \) on the right-hand side is defined as

\[
J(t) = q_1 \mathcal{L}^{-1} \left\{ \frac{1}{s(r_1 s^2 + p_1 s + q_1)} \right\} = 1 - \frac{p_1}{2 \sqrt{p_1^2 - 4q_1 r_1}} \left( e^{s_1 t} - e^{s_2 t} \right) - \frac{1}{2} \left( e^{s_1 t} + e^{s_2 t} \right)
\]

Then taking the inverse Laplace transform of (3.6), we have

\[
\sigma(y,t) + \int_0^t K(t - \tau) \left[ r_1 (\sigma_y(y,\tau))_\tau y_\tau(y,\tau) - r_1 y_\tau^2(y,\tau) \sigma_{yy}(y,\tau) + r_1 (\sigma_y(y,\tau) y_\tau(y,\tau))_\tau \right. \\
\left. + p_1 y_\tau(y,\tau) \sigma_y(y,\tau) - \frac{r_2}{\beta} \left( \frac{1}{l^2(\tau)} \sigma_{yy} \right)_\tau - \frac{p_2}{\beta l^2(\tau)} \sigma_{yy}(y,\tau) \right] d\tau - r_1 K(t) \sigma_t(y,0) \quad (3.7)
\]

\[
= q_2 \int_0^t K(t - \tau) \frac{u_y(y,\tau)}{l(\tau)} d\tau + \tau_0 J(t).
\]
To further simplify (3.7) and avoid time derivatives on $\sigma(y, \tau)$, we integrate by parts
\[
\int_0^t K(t - \tau)(\sigma_y(y, \tau))_\tau y_\tau(y, \tau) \, d\tau
= K(t - \tau)y_\tau(y, \tau)\sigma_y(y, \tau) \bigg|_0^t - \int_0^t (K(t - \tau)y_\tau(y, \tau))_\tau \sigma_y(y, \tau) \, d\tau
= K(0)y_t(y, t)\sigma_y(y, t) - \int_0^t \left[ K(t - \tau)(y_{\tau\tau}(y, \tau) + y_{\tau y}(y, \tau)y_\tau(y, \tau)) - K'(t - \tau)y_\tau(y, \tau) \right] \sigma_y(y, \tau) \, d\tau
\]
Similarly,
\[
\int_0^t K(t - \tau)\left( \frac{1}{l^2(\tau)} \sigma_{yy} \right)_\tau \, d\tau = K(t - \tau)\sigma_{yy}(y, \tau) \bigg|_0^t + \int_0^t \frac{K'(t - \tau)}{l^2(\tau)} \sigma_{yy}(y, \tau) \, d\tau
= \frac{K(0)}{l^2(t)} \sigma_{yy}(y, t) + \int_0^t \frac{K'(t - \tau)}{l^2(\tau)} \sigma_{yy}(y, \tau) \, d\tau,
\]
and
\[
\int_0^t K(t - \tau)(\sigma_y(y, \tau)y_\tau(y, \tau))_\tau \, d\tau
= K(t - \tau)y_\tau(y, \tau)\sigma_y(y, \tau) \bigg|_0^t + \int_0^t K'(t - \tau)y_\tau(y, \tau)\sigma_y(y, \tau) \, d\tau
= K(0)y_t(y, t)\sigma_y(y, t) + \int_0^t K'(t - \tau)y_\tau(y, \tau)\sigma_y(y, \tau) \, d\tau.
\]
Then (3.7) simplifies to the final form of the integral equation,
\[
\sigma(y, t) + 2r_1 K(0)y_t(y, t)\sigma_y(y, t) - \frac{r_2 K(0)}{\beta l^2(t)} \sigma_{yy}(y, t)
+ \int_0^t K(t - \tau)\left[ p_1 y_\tau(y, \tau) - r_1 \left( y_{\tau\tau}(y, \tau) - \frac{l'(\tau)}{l(\tau)} y_\tau(y, \tau) \right) \right] \sigma_y(y, \tau) \, d\tau
+ \int_0^t K'(t - \tau)\left[ 2r_1 y_\tau(y, \tau)\sigma_y(y, \tau) - \frac{r_2}{\beta l^2(\tau)} \sigma_{yy}(y, \tau) \right] \, d\tau
+ \int_0^t K(t - \tau)\left[ - r_1 y_\tau^2(y, \tau) - \frac{p_2}{\beta l^2(\tau)} \right] \sigma_{yy}(y, \tau) \, d\tau
= q_2 \int_0^t \frac{K(t - \tau)}{l(\tau)} u_y(y, \tau) \, d\tau + \tau_0 J(t) + r_1 K(t)\sigma_t(y, 0).
\]
The proof of the theorem is complete.
3.3 Numerical Methods for Second-Order Models

We turn our attention to the methods used to approximate a solution to this constitutive law in integral form. The numerical methods for the second-order model are very similar to the methods used to solve the first order problem. The use of integration by parts in the derivation of the model ensured only terms involving $\sigma(y,t)$, $\sigma_y(y,t)$ and $\sigma_{yy}(y,t)$, and no higher-order derivatives. This greatly simplifies the numerical methods involved. In Chapter 2 we defined the trapezoid rule and the necessary finite difference formulas. In discretizing the integral equation using these formulas, we implement the spatial discretization $0 = y_0, y_1, \ldots, y_i, \ldots, y_m = 1$, where $y_i = i\Delta y$, and temporal discretization $t_0, t_1, \ldots, t_j, \ldots, t_n$, where $t_0 = 0$, $t_j = j\Delta t$, and $t_n = t_{final}$.

Moving to discrete space, we substitute the finite difference formulas into (3.5) for the respective derivatives. Keeping notation consistent with the previous chapter, the integral equation becomes

$$
\sigma(y_i, t) + 2r_1K(0)y_t(y_i, t)\left(\frac{\sigma_{i+1}(t) - \sigma_{i-1}(t)}{2\Delta y}\right) - \frac{r_2K(0)}{\beta l^2(t)}\left(\frac{\sigma_{i+1}(t) - 2\sigma_i(t) + \sigma_{i-1}(t)}{\Delta y^2}\right)
+ \int_0^t K(t - \tau)p_1y_t(y_i, \tau) - r_1\left(y_{tt}(y_i, \tau) - \frac{l'(\tau)}{l(\tau)}y_t(y_i, \tau)\right)\frac{\sigma_{i+1}(\tau) - \sigma_{i-1}(\tau)}{2\Delta y} d\tau
+ \int_0^t K'(t - \tau)\left[2r_1y_{tt}(y_i, \tau)\frac{\sigma_{i+1}(\tau) - \sigma_{i-1}(\tau)}{2\Delta y} - \frac{r_2}{\beta l^2(\tau)}\sigma_{i+1}(\tau) - 2\sigma_i(\tau) + \sigma_{i-1}(\tau)\right] d\tau
+ \int_0^t K(t - \tau)\left[-r_1y^2_t(y_i, \tau) - \frac{p_2}{\beta l^2(\tau)}\right]\frac{\sigma_{i+1}(\tau) - 2\sigma_i(\tau) + \sigma_{i-1}(\tau)}{\Delta y^2} d\tau
= q_2\int_0^t \frac{K(t - \tau)}{l(\tau)}u_y(y_i, \tau) d\tau + \tau_0J(t) + r_1K(t)\sigma_t(y_i, 0).
$$

(3.8)

Now the only remaining spatial derivative is $u_y(y, \tau)$, which can be evaluated since $u(y, t)$ is assumed to be known when calculating $\sigma(y, t)$. Converting to discrete time, we apply the composite trapezoid rule to each of the three integrals on the left hand side. After applying
this formula and the the initial condition \( \sigma(y, 0) = 0 \), (3.8) becomes

\[
\begin{align*}
\sigma_i(t_j) &+ 2r_1 K(0)y_i(y_i, t_j) \left( \frac{\sigma_{i+1}(t_j) - \sigma_{i-1}(t_j)}{2\Delta y} \right) - r_2 K(0) \left( \frac{\sigma_{i+1}(t_j) - 2\sigma_i(t_j) + \sigma_{i-1}(t_j)}{\Delta y^2} \right) \\
&+ \Delta t \sum_{k=1}^{j-1} K(t_j - t_k) \left[ p_1 y_i(y_i, t_k) - r_1 \left( y_{tt}(y_i, t_k) - \frac{l'(t_k)}{l(t_k)} y_{t}(y_i, t_k) \right) \right] \sigma_{i+1}(t_k) - \sigma_{i-1}(t_k) \\
&+ \frac{\Delta t}{2} K(0) \left[ p_1 y_i(y_i, t_j) - r_1 \left( y_{tt}(y_i, t_j) - \frac{l'(t_j)}{l(t_j)} y_{t}(y_i, t_j) \right) \right] \sigma_{i+1}(t_j) - \sigma_{i-1}(t_j) \\
&+ \Delta t \sum_{k=1}^{j-1} K'(t_j - t_k) \left[ 2r_1 y_i(y_i, t_k) \frac{\sigma_{i+1}(t_k) - \sigma_{i-1}(t_k)}{2\Delta y} \right] \\
&+ \frac{\Delta t}{2} K'(0) \left[ 2r_1 y_i(y_i, t_j) \frac{\sigma_{i+1}(t_j) - \sigma_{i-1}(t_j)}{2\Delta y} - \frac{r_2}{\beta l^2(t_j)} \sigma_{i+1}(t_j) - 2\sigma_i(t_j) + \sigma_{i-1}(t_j) \right] \\
&+ \Delta t \sum_{k=1}^{j-1} K(t_j - t_k) \left[ \sigma_{i+1}(t_k) - \sigma_{i-1}(t_k) \right] \\
&+ \frac{\Delta t}{2} K(0) \left[ \sigma_{i+1}(t_j) - 2\sigma_i(t_j) + \sigma_{i-1}(t_j) \right] \\
&= q \int_0^{t_j} \frac{K(t_j - \tau)}{l(\tau)} u(y_i, \tau) d\tau + \tau_0 J(t_j) + r_1 K(t_j) \sigma_i(y_i, 0). 
\end{align*}
\]

As with the first-order model, we implicitly solve for \( \sigma(y, t) \) as a linear system \( A\ddot{\sigma} = \ddot{b} \). Since each sum depends on \( \sigma(y, t) \) at all previous time steps, the matrix \( A \) will be a block lower-triangular matrix. The system takes the form

\[
\begin{pmatrix}
M_{1,1} & 0 & 0 & \ldots & 0 \\
M_{2,1} & M_{2,2} & 0 & \ldots & 0 \\
M_{3,1} & M_{3,2} & M_{3,3} & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
M_{n,1} & M_{n,2} & \ldots & M_{n,n-1} & M_{n,n}
\end{pmatrix}
\begin{pmatrix}
\bar{\sigma}_1 \\
\bar{\sigma}_2 \\
\vdots \\
\bar{\sigma}_{n-1} \\
\bar{\sigma}_n
\end{pmatrix}
= \begin{pmatrix}
\bar{b}_1 \\
\bar{b}_2 \\
\vdots \\
\bar{b}_{n-1} \\
\bar{b}_n
\end{pmatrix},
\]

where each \( M_{j,k} \) is a tridiagonal matrix and

\[
\bar{\sigma}_j = (\sigma_1(t_j), \sigma_2(t_j), \ldots, \sigma_i(t_j), \ldots, \sigma_{m-1}(t_j))^T.
\]

The subscripts on \( M_{j,k} \) correspond to the indices \( j \) and \( k \) in (3.9), where both index time. We note that on the diagonal of the matrix \( A \), the index \( k = j \), but \( k < j \) in the summation of (3.9), so this index \( k \) does not apply to the diagonal. However, it is convenient to use the
same \( j \) and \( k \) to index the kernel of each integral. We note that the subscripts \( j \) and \( k \) in the following matrices both index time. The spatial indices are hidden inside each matrix.

The content of each block is best described by creating a matrix for each derivative, then summing these to form the final block. From (3.9) we can pull the coefficients, including the kernel \( K(t_j - t_k) \) or \( K'(t_j - t_k) \), off of each derivative. The coefficients inside the blocks on the main diagonal contain the terms that are not involved in the integrals and do not contain any terms from the summations. Letting \( D_{i,j}^1 \) and \( D_{i,j}^2 \) denote the coefficient on the first derivative term and the second derivative term in (3.9) respectively,

\[
D_{i,j}^1 = \frac{r_1}{\Delta y} K(0)y_i(y_i, t_j) + \frac{\Delta t}{4\Delta y} \left[ K(0) \left( p_1 y_t(y_i, t_j) - r_1 \left( y_{tt}(y_i, t_j) - \frac{l'(t_j)}{l(t_j)} y_t(y_i, t_j) \right) \right) \right] + 2K'(0)r_1 y_t(y_i, t_j),
\]

\[
D_{i,j}^2 = -\frac{r_2 K(0)}{\beta l^2(t_j) \Delta y^2} - \Delta t \left[ \frac{r_2 K'(0)}{\beta l^2(t_j)} + K(0) \left( r_1 y_t^2(y_i, t_j) + \frac{p_2}{\beta l^2(t_j)} \right) \right].
\]

(3.11)

Now we return and explore the formation of the blocks on the diagonal of \( A \) in (3.10). In addition to the coefficients in (3.11), the \( \sigma(y_i, t_j) \) contributes ones along the diagonal to this matrix. Combining these yields the tridiagonal matrix

\[
M_{j,j} =
\begin{bmatrix}
1 - 2D_{1,1}^2 & D_{1,1}^1 + D_{1,1}^2 & \cdots & 0 & 0 \\
-D_{2,1}^1 + D_{2,1}^2 & 1 - 2D_{2,2}^2 & D_{2,1}^1 + D_{2,1}^2 & \cdots & 0 \\
& \ddots & \ddots & \ddots & 0 \\
0 & \cdots & -D_{m-2,1}^1 + D_{m-2,1}^2 & 1 - 2D_{m-2,m-2}^2 & D_{m-2,1}^1 + D_{m-2,1}^2 \\
0 & 0 & \cdots & -D_{m-1,1}^1 + D_{m-1,1}^2 & 1 - 2D_{m-1,m-1}^2
\end{bmatrix}.
\]

Now we must define the blocks below the main diagonal in \( A \). Each of the coefficients in this matrix comes from one of the three sums in (3.9). We define the coefficients \( C_{i,j}^{1,k} \) and \( C_{i,j}^{2,k} \) as the coefficients on first and second derivative terms, respectively. These coefficients are defined as

\[
C_{i,j}^{1,k} = \frac{\Delta t}{2\Delta y} \left[ K(t_j - t_k) \left( p_1 y_t(y_i, t_k) - r_1 \left( y_{tt}(y_i, t_k) - \frac{l'(t_k)}{l(t_k)} y_t(y_i, t_k) \right) \right) \right] + 2K'(t_j - t_k)r_1 y_t(y_i, t_k),
\]

\[
C_{i,j}^{2,k} = \frac{\Delta t}{\Delta y^2} \left[ - K(t_j - t_k) \left( r_1 y_t^2(y_i, t_k) + \frac{p_2}{\beta l^2(t_k)} \right) - K'(t_j - t_k) \frac{r_2}{\beta l^2(t_k)} \right].
\]

Then each tridiagonal block below the diagonal is defined by
\[ M_{j,k} = \begin{bmatrix}
-2C_{1,j}^{2,k} & C_{1,j}^{1,k} + C_{1,j}^{2,k} & \cdots & 0 & 0 \\
-C_{2,j}^{1,k} + C_{2,j}^{2,k} & -2C_{2,j}^{2,k} & \cdots & 0 & 0 \\
0 & \cdots & -C_{m-2,j}^{1,k} + C_{m-2,j}^{2,k} & \cdots & 0 \\
0 & \cdots & 0 & \cdots & -C_{m-1,j}^{1,k} + C_{m-1,j}^{2,k} & -2C_{m-1,j}^{2,k}
\end{bmatrix}, \]

for \( k = 1, 2, \ldots, j - 1 \).

The vector \( \vec{b} \) contains all terms on the right-hand side of (3.9). Then each \( \vec{b}_j \) takes the form

\[
\vec{b}_j = \begin{bmatrix}
q_2 \int_0^{t_j} K(t_j - \tau) u_g(y_1, \tau) \, d\tau + \tau_0 J(t_j) + r_1 K(t_j) \sigma_t(y_1, 0) \\
q_2 \int_0^{t_j} K(t_j - \tau) u_g(y_2, \tau) \, d\tau + \tau_0 J(t_j) + r_1 K(t_j) \sigma_t(y_2, 0) \\
\vdots \\
q_2 \int_0^{t_j} K(t_j - \tau) u_g(y_{m-1}, \tau) \, d\tau + \tau_0 J(t_j) + r_1 K(t_j) \sigma_t(y_{m-1}, 0)
\end{bmatrix}
\]

Again, we solve for \( \sigma(y, t) \), \( u(y, t) \), \( f(t) \) and \( r(t) \) as a fixed point problem. The method for solving for \( u(y, t) \) does not change. Solving for \( \sigma(y, t) \) now requires \( l'(t) \) and \( y_{tt}(y, t) \). To calculate \( y_{tt}(y, t) \), we simply use the first-order centered difference scheme defined in Chapter 2 to take the derivative of \( y_t(y, t) \) with respect to time. In comparison to the pseudocode in Chapter 2, the only changes are the inputs and coefficients in the SolveForSigma algorithm. The Driver algorithm is updated to calculate \( y_{tt}(y, t) \) and \( l'(t) \). Note that in the pseudocode for the SolveForSigma algorithm, the coefficients \( D_1^j \) and \( D_2^j \) are vectors, since \( y_t(y_1 : y_{m-1}) \) is a vector. For this reason they are indexed as such, i.e. \( D_1^j(y_1 : y_{m-1}) = (D_1^j(y_1), D_1^j(y_2), \ldots, D_1^j(y_{m-1})) \).
Algorithm 4: SolveForSigma (Second-Order Model)

**input**: $p_1, p_2, q_1, q_2, r_1, r_2, y_t(y, t), y_{tt}(y, t), u_y(y, t), l(t), \sigma_t(y, 0), \beta, \tau_0, \Delta y, \Delta t$

**output**: $\sigma(y, t)$ for all $(y, t)$ in the unit strip.

for $j = 1 : N$

Define coefficient on first-order derivatives

\[
D_j^1 = \frac{1}{2\Delta y} r_1 K(0)y_t(y_i, t_j) + \frac{\Delta t}{2\Delta y^2} \left( K(0) \left[ p_1 y_t(y_1 : y_{m-1}, t_j) - r_1 (y_{tt}(y_1 : y_{m-1}, t_j) \right) - \frac{\nu(t_j)}{\nu(t_j)} y_t(y_1 : y_{m-1}, t_j) \right) + 2K'(0) r_1 y_t(y_1 : y_{m-1}, t_j) \right) \]

Create submatrix of first-order derivative coefficients

$M_1 = \text{diag}(D_j^1(y_2 : y_{m-1}), 1) - \text{diag}(D_j^1(y_1 : y_{m-2}, 1))$

Define coefficient on second-order derivatives

\[
D_j^2 = -\frac{r_2 K(0)}{\beta^2 (t_j) \Delta y} - \frac{\Delta t}{2\Delta y^2} \left[ \frac{r_2 K'(0)}{\beta^2 (t_j)} + K(0) \left( r_1 y_t(y_1 : y_{m-1}, t_j) + \frac{p_2}{\beta^2 (t_j)} \right) \right] \]

Create submatrix of second-order derivative coefficients

$M_2 = \text{diag}(D_j^2(y_2 : y_{m-1}), -1) - \text{diag}(2D_j^2(y_1 : y_{m-2}), 1) + \text{diag}(D_j^2(y_1 : y_{m-2}, 1))$

Store submatrix in correct block in lower-triangular $A$

$M_{i,j} = I_{M-1} + M_1 + M_2$

for $k = 1 : j$

Define coefficient on first-order derivatives

\[
C_{i,j}^{1,k} = \frac{\Delta t}{2\Delta y} \left[ K(t_j - t_k) \left( p_1 y_t(y_i, t_k) - r_1 (y_{tt}(y_i, t_k) \right) - \frac{\nu(t_k)}{\nu(t_k)} y_t(y_i, t_k) \right) + 2K'(t_j - t_k) r_1 y_t(y_i, t_k) \right] \]

Create submatrix of first-order derivative coefficients

$M_1 = \text{diag}(D_j^1(y_2 : y_{m-2}, 1) - \text{diag}(D_j^1(y_1 : y_{m-1}, -1))$

Define coefficient on second-order derivatives

\[
C_{i,j}^{2,k} = \frac{\Delta t}{2\Delta y} \left[ -K(t_j - t_k) \left( r_1 y_t^2(y_i, t_k) + \frac{p_2}{\beta^2 (t_k)} \right) - K'(t_j - t_k) \frac{r_2}{\beta^2 (t_k)} \right] \]

Create submatrix of second-order derivative coefficients

$M_2 = \text{diag}(D_j^2(y_2 : y_{m-1}, -1) - \text{diag}(2D_j^2(y_1 : y_{m-2}), 1) + \text{diag}(D_j^2(y_1 : y_{m-2}, 1))$

Store submatrix in correct block in lower-triangular $A$

$M_{j,k} = M_1 + M_2$

\[
\bar{b}_j = q_2 \int_0^{t_j} K(t_j - \tau) \frac{u_y(y_1 : y_{m-1}, \tau)}{l(\tau)} d\tau + \tau_0 J(t_j) + r_1 K(t_j) \sigma_t(y_1 : y_{m-1}, 0) \]

Solve $A\bar{\sigma} = \bar{b}$ for $\bar{\sigma}(y, t)$ and reshape to $(m - 1) \times n$ matrix;

Concatenate with zeros for initial/boundary conditions at $y = 0$, $y = 1$ and $t = 0$;
values translate to the coefficients $p$ (3.3) and (3.4) are put into the general form of the constitutive law (equation (3.1)), these elastic coefficients. The parameter values used for each model are given in Table 3.1. When For simplicity we use the same values for each of the viscous coefficients and each of the results below. The Burger model requires two elastic elements and two viscous elements. Using the numerical methods derived above, we solved the system in (3.5) and produced the

3.4 Numerical Results for Second-Order Models

Using the numerical methods derived above, we solved the system in (3.5) and produced the results below. The Burger model requires two elastic elements and two viscous elements. For simplicity we use the same values for each of the viscous coefficients and each of the elastic coefficients. The parameter values used for each model are given in Table 3.1. When (3.3) and (3.4) are put into the general form of the constitutive law (equation (3.1)), these values translate to the coefficients $p_1$, $p_2$, $q_1$, $q_2$, $r_1$, and $r_2$ in Table 3.2.

The displacement $u(y, t)$ does not change with the model. Comparing Figure 3.4 and Figure 2.4, we notice that this plot looks the same for the standard linear model, Burger model, and Kelvin-Voigt connected to a dashpot in series. For this reason, the paths of the

<table>
<thead>
<tr>
<th>Algorithm 5: Driver (Second-Order Model)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>input</strong>: $E_0$, $E_1$, $\mu_0$, $\mu_1$, $r_0$, $L_0$, $T_{final}$, $\beta$, $\tau_0$, $\Delta y$, $\Delta t$</td>
</tr>
<tr>
<td><strong>output</strong>: $\sigma(y, t)$, $u(y, t)$, $l(t)$ for all space and time.</td>
</tr>
<tr>
<td>Define $p_1$, $p_2$, $q_1$, and $q_2$ in terms of $E_0$, $E_1$, $\mu_0$, $\mu_1$;</td>
</tr>
<tr>
<td>Initialize $\sigma(y, t)$, $u(y, t)$, $y_t(y, t)$, $y_{tt}(y, t)$;</td>
</tr>
<tr>
<td><strong>while</strong> $</td>
</tr>
<tr>
<td>Compute $u_y$ and $y_{tt}$;</td>
</tr>
<tr>
<td>Assign $\sigma_{old}(y, t) = \sigma(y, t)$;</td>
</tr>
<tr>
<td>Solve $\sigma(y, t) = SolveForSigma$;</td>
</tr>
<tr>
<td>Compute $\sigma_y(y, t)$;</td>
</tr>
<tr>
<td>$r'(t_0 : t_n) = r_0 + \sigma_y(y_1, t_0 : t_n)/(\beta l(t_0 : t_n))$;</td>
</tr>
<tr>
<td>$f'(t_0 : t_n) = L_0/l(t_0 : t_n) + \sigma_y(y_n, t_0 : t_n)/(\beta l(t_0 : t_n))$;</td>
</tr>
<tr>
<td>$l'(t_0 : t_n) = f'(t_0 : t_n) - r'(t_0 : t_n)$;</td>
</tr>
<tr>
<td><strong>for</strong> $j = 1 : N - 1$ do</td>
</tr>
<tr>
<td><strong>if</strong> $j == 1$ then</td>
</tr>
<tr>
<td>$r(t_{j+1}) = r(t_j) + \Delta t \cdot r'(t_j)$;</td>
</tr>
<tr>
<td>$f(t_{j+1}) = f(t_j) + \Delta t \cdot f'(t_j)$;</td>
</tr>
<tr>
<td><strong>else</strong></td>
</tr>
<tr>
<td>$r(t_{j+1}) = (2\Delta t \cdot r'(t_j) + 4r(t_j) - r(t_{j-1}))/3$;</td>
</tr>
<tr>
<td>$f(t_{j+1}) = (2\Delta t \cdot f'(t_j) + 4f(t_j) - f(t_{j-1}))/3$;</td>
</tr>
<tr>
<td>Update $y_t(y, t) = -(y(f'(t) - r'(t)) + r'(t))/l(t)$;</td>
</tr>
<tr>
<td>Assign $u_{old}(y, t) = u(y, t)$;</td>
</tr>
<tr>
<td>Solve $u(y, t) = SolveForU$;</td>
</tr>
</tbody>
</table>

29
<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\beta$</th>
<th>$\tau_0$</th>
<th>$r_0$</th>
<th>$L_0$</th>
<th>$E_0$</th>
<th>$E_1$</th>
<th>$\mu_0$</th>
<th>$\mu_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burger</td>
<td>5350.4</td>
<td>179.17</td>
<td>0.8</td>
<td>1</td>
<td>10288</td>
<td>5144</td>
<td>53015</td>
<td>53015</td>
</tr>
<tr>
<td>K-V and dashpot in series</td>
<td>5350.4</td>
<td>179.17</td>
<td>0.8</td>
<td>1</td>
<td>10288</td>
<td>n/a</td>
<td>53015</td>
<td>53015</td>
</tr>
</tbody>
</table>

Table 3.1: Parameter values for numerical simulation of second-order models.

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>$p_1$</th>
<th>$q_1$</th>
<th>$r_1$</th>
<th>$p_2$</th>
<th>$q_2$</th>
<th>$r_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burger</td>
<td>15.4593</td>
<td>1</td>
<td>26.5543</td>
<td>53015</td>
<td>0</td>
<td>273191.118</td>
</tr>
<tr>
<td>Kelvin-Voigt and dashpot in series</td>
<td>0.3881</td>
<td>1</td>
<td>0</td>
<td>53015</td>
<td>0</td>
<td>273191.118</td>
</tr>
</tbody>
</table>

Table 3.2: Approximate values for coefficients in (3.1).

Front and rear boundaries look identical to those in the standard linear models. As seen in Figure 3.5, the paths of the Burger model are almost identical to, though slightly ahead of, the paths of the Kelvin-Voigt and dashpot in series. The tic marks on the top and bottom are each 1 unit apart. The steady state length of the cell, seen at the top of Figure 3.5, is about $L_0/r_0 = 1.25$.

Now looking at the plots of $\sigma(y,t)$ for each model in Figure 3.2 and Figure 3.3 we see that the overall magnitude of the stress is smaller for the Burger model. Although the stress is noticeably different between the two models, the steady state speed of the cell is seemingly unaffected. Recalling that

$$r'(t) = r_0 + \frac{\sigma_y(y,t)}{\beta l(t)},$$

these results show that the steady state speed of the cell is dominated by $r_0$, rather than the term containing $\sigma_y(y,t)$. This agrees with our earlier assumption on the steady state speed of the cell. The length of the cell in steady state is approximately 1.25 unit length.
Figure 3.2: Surface plot of $\sigma(y, t)$ for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series.
Figure 3.3: Side view of $\sigma(y, t)$ for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series.
Figure 3.4: Surface plots of the displacement $u(y, t)$ for (a) the Burger model and (b) Kelvin-Voigt and dashpot in series
Figure 3.5: Plot of the front and rear boundaries.
Chapter 4

Models with Spatially Dependent Adhesion Coefficients

In the previous chapters we considered the case where the effective adhesion viscous drag $\beta$ is constant. One may think that the adhesion of the cell to the surface is stronger towards the front of the cell, since the rear of the cell releases from the surface during the retraction stage. Another factor that may affect the viscosity coefficient $\beta$ is the stiffness of the substrate. The stiffness of the substrate beneath the cell would play a role in the adhesion of the cell to the surface via the transmembrane proteins known as integrins [1].

In this chapter, we will explore the differences when $\beta$ is a constant and when $\beta = \beta(x)$. Any model considered thus far can be adjusted so that $\beta = \beta(x)$. Intuition tells us that the stress should increase or decrease proportional to the coefficient $\beta(x)$, in turn changing $\sigma_y(y, t)$. After seeing the results in Chapter 3, we may anticipate very little change in the front and rear paths of the cell, unless there is a drastic change in $\sigma_y(y, t)$ at $y = 0, 1$.

4.1 Derivation of Model Equations

The derivation of this model is almost identical to that in Chapter 3. The only difference is in the derivative of the force balance law,

$$
\varepsilon_t = \left(\frac{\sigma_x(x, t)}{\beta(x)}\right)_x = \frac{\sigma_{xx}(x, t)\beta(x) - \sigma_x(x, t)\beta'(x)}{\beta^2(x)} = \frac{\sigma_{xx}(x, t)}{\beta(x)} - \frac{\sigma_x(x, t)\beta'(x)}{\beta^2(x)}.
$$

Thus, we will find the derivation of the model equations the same, with the exception of this substitution in two instances.
Theorem 4.1. If \( \beta = \beta(x) \), then using the change of variables \( y = (x - r(t))/l(t) \) the partial differential equation

\[
r_1 \sigma_{tt} + p_1 \sigma_t + q_1 (\sigma - \tau_0) = r_2 \varepsilon_{tt} + p_2 \varepsilon_t + q_2 \varepsilon
\]
can be transformed into the integral equation

\[
\sigma(y, t) + 2r_1 K(0)y_t(y, t)\sigma_y(y, t) - r_2 K(0)\left( \frac{\sigma_{yy}(y, t)}{\beta(y)l^2(t)} - \frac{\sigma_y(y, t)\beta'(y)}{\beta^2(y)l^2(t)} \right) + \int_0^t K(t - \tau) \left[ p_1 y_{\tau\tau}(y, \tau) - r_1 \left( y_{\tau\tau}(y, \tau) - \frac{l'(\tau)}{l(\tau)} y_{\tau}(y, \tau) \right) + \frac{p_2 \beta'(y)}{\beta^2(y)l^2(t)} \right] \sigma_y(y, \tau) d\tau \\
+ \int_0^t K'(t - \tau) \left[ 2r_1 y_{\tau}(y, \tau) + \frac{r_2 \beta'(y)}{\beta^2(y)l^2(\tau)} \right] \sigma_y(y, \tau) - \frac{r_2}{\beta l^2(\tau)} \sigma_{yy}(y, \tau) d\tau \\
+ \int_0^t K(t - \tau) \left[ - r_1 y_{\tau}^2(y, \tau) - \frac{p_2}{\beta l^2(\tau)} \right] \sigma_{yy}(y, \tau) d\tau \\
= q_2 \int_0^t K(t - \tau) \frac{u_y(y, \tau)}{l(\tau)} d\tau + \tau_0 J(t) + r_1 K(t)\sigma_t(y, 0),
\]

where

\[
K(t) = \frac{1}{\sqrt{p_1^2 - 4q_1 r_1}} (e^{s_1t} - e^{s_2t}), \quad J(t) = 1 - \frac{p_1}{2\sqrt{p_1^2 - 4q_1 r_1}} (e^{s_1t} - e^{s_2t}) - \frac{1}{2} (e^{s_1t} + e^{s_2t})
\]

\[
s_1 = \frac{-2q_1}{p_1 + \sqrt{p_1^2 - 4q_1 r_1}} \quad \text{and} \quad s_2 = \frac{-2q_1}{p_1 - \sqrt{p_1^2 - 4q_1 r_1}}.
\]

Proof. Beginning with the partial differential equation

\[
r_1 \sigma_{tt} + p_1 \sigma_t + q_1 (\sigma - \tau_0) = r_2 \varepsilon_{tt} + p_2 \varepsilon_t + q_2 \varepsilon,
\]

we substitute \( \varepsilon_t = \sigma_{xx}/\beta(x) \) and \( \varepsilon = u_x \) to get

\[
r_1 \sigma_{tt} + p_1 \sigma_t + q_1 (\sigma - \tau_0) = r_2 \left[ \left( \frac{\sigma_x}{\beta(x)} \right)_t \right] + p_2 \left( \frac{\sigma_x}{\beta(x)} \right)_x + q_2 u_x.
\]

Then we make the change of variables to \( y = (x - r(t))/l(t) \). Let \( \tilde{\beta}(y) = \beta(x) \). For notational convenience, we drop the bar and write \( \beta(y) \) instead of \( \tilde{\beta}(y) \). This results in

\[
r_1 [\sigma_{tt} + \sigma_{ty} y_t + (\sigma_y y_t)] + p_1 (\sigma_t + \sigma_y y_t) + q_1 (\sigma - \tau_0)
\]

\[
= r_2 \left( \frac{\sigma_{yy}}{\beta(y)l^2(t)} - \frac{\sigma_y \beta'(y)}{\beta^2(y)l^2(t)} \right)_t + p_2 \left( \frac{\sigma_{yy}}{\beta(y)l^2(t)} - \frac{\sigma_y \beta'(y)}{\beta^2(y)l^2(t)} \right)_x + q_2 u_y.
\]
where \( \sigma = \sigma(y, t) \), \( u = u(y, t) \), and \( y_t = y_t(y, t) = -(y' + r(t))/l(t) \). Next apply the Laplace transform to both sides. Assuming \( \sigma_{yt} = \sigma_{yt} \), this yields

\[
\begin{align*}
r_1(s^2 \mathcal{L} \{\sigma\} - s\sigma(y, 0) - \sigma_t(y, 0)) + r_1 \mathcal{L} \{\sigma_y y_t\} + r_1 \mathcal{L} \{(\sigma_y y_t)_t\} + p_1(s \mathcal{L} \{\sigma\} - \sigma(y, 0)) \\
+ p_1 \mathcal{L} \{\sigma_y y_t\} + q_1 \mathcal{L} \{\sigma\} = r_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) \\
+ p_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) + q_2 \left( \frac{u_y}{l(t)} \right) + q_1 \frac{\tau_0}{s}.
\end{align*}
\]

The initial condition \( \sigma(y, 0) \) is assumed to be zero, although \( \sigma_t(y, 0) \) is not. Additionally, \( (\sigma_y)_t = \sigma_y + \sigma_{yy} y_t \), so \( \sigma_y = (\sigma_y)_t - \sigma_{yy} y_t \). Now we may rearrange terms and reduce the above equation to get

\[
\begin{align*}
(r_1 s^2 + p_1 s + q_1) \mathcal{L} \{\sigma\} \\
r_1 \mathcal{L} \{(\sigma_y)_t - \sigma_{yy} y_t\} + r_1 \mathcal{L} \{(\sigma_y y_t)_t\} + p_1 \mathcal{L} \{\sigma_y y_t\} - r_1 \sigma_t(y, 0) \\
- r_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) \\
- p_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) \\
= q_2 \left( \frac{u_y}{l(t)} \right) + q_1 \frac{\tau_0}{s}.
\end{align*}
\]

Dividing both sides by \( (r_1 s^2 + p_1 s + q_1) \) and simplifying, we have

\[
\begin{align*}
\mathcal{L} \{\sigma\} + \frac{1}{r_1 s^2 + p_1 s + q_1} \left[ r_1 \mathcal{L} \{(\sigma_y)_t - \sigma_{yy} y_t\} + r_1 \mathcal{L} \{(\sigma_y y_t)_t\} + p_1 \mathcal{L} \{\sigma_y y_t\} - r_1 \sigma_t(y, 0) \\
- r_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) \\
- p_2 \left( \frac{\sigma_y}{\beta(y)^2(t)} - \frac{\sigma_y}{\beta^2(y)^2(t)} \right) \right] \\
= \frac{q_2}{r_1 s^2 + p_1 s + q_1} \left( \frac{u_y}{l(t)} \right) + q_1 \frac{\tau_0}{s(r_1 s^2 + p_1 s + q_1)}.
\end{align*}
\]

When we take the inverse Laplace transform of both sides the result is the sum of convolutions, all with the same kernel \( K(t) \) defined as the inverse Laplace transform of \( 1/(r_1 s^2 + p_1 s + q_1) \). Define \( s_1 \) and \( s_2 \) to be the positive and negative roots of the denominator, respectively. Then the roots can be written equivalently as

\[
s_1 = \frac{-2q_1}{p_1 + \sqrt{p_1^2 - 4q_1 r_1}} \quad s_2 = \frac{-2q_1}{p_1 - \sqrt{p_1^2 - 4q_1 r_1}}.
\]

and the kernel is defined as

\[
K(t) = \mathcal{L}^{-1} \left\{ \frac{1}{r_1 s^2 + p_1 s + q_1} \right\} = \frac{1}{\sqrt{p_1^2 - 4q_1 r_1}} (e^{s_1 t} - e^{s_2 t}).
\]

The function \( J(t) \) on the right-hand side is defined as

\[
J(t) = q_1 \mathcal{L}^{-1} \left\{ \frac{1}{s(r_1 s^2 + p_1 s + q_1)} \right\} = 1 - \frac{p_1}{2 \sqrt{p_1^2 - 4q_1 r_1}} (e^{s_1 t} - e^{s_2 t}) - \frac{1}{2} (e^{s_1 t} + e^{s_2 t})
\]

37
Then taking the inverse Laplace transform of (4.1), we have
\[
\sigma(y, t) + \int_0^t K(t - \tau) \left[ r_1(\sigma_y(y, \tau), y, \tau) - r_1y^2_\tau(y, \tau)\sigma_{yy}(y, \tau) + r_1(\sigma_y(y, \tau)y_\tau(y, \tau)\right]_{\tau} \\
+ p_1y_\tau(y, \tau)\sigma_y(y, \tau) - r_2 \left( \frac{\sigma_{yy}}{\beta(y(\tau))l^2(\tau)} - \frac{\sigma_{y}^\prime\beta(y(\tau))}{\beta^2(\tau)} \right)_{\tau} \\
- p_2 \left( \frac{\sigma_{yy}}{\beta(y)^2(\tau)} - \frac{\sigma_{y}^\prime(y)}{\beta^2(\tau)} \right) d\tau \\
= q_2 \int_0^t K(t - \tau) \left( \frac{u_y(y, \tau)}{\tau} \right) d\tau + \tau_0 J(t) + r_1K(t)\sigma_t(y, 0).
\]

To further simplify (4.2) and avoid time derivatives on \(\sigma(y, \tau)\), we integrate by parts where appropriate. In addition to the integrals simplified in the proof of Theorem 3.1, we have
\[
- r_2 \int_0^t K(t - \tau) \left( \frac{\sigma_{yy}}{\beta(y)^2(\tau)} - \frac{\sigma_{y}^\prime(y)}{\beta^2(\tau)} \right) d\tau \\
= - r_2K(0) \left( \frac{\sigma_{yy}}{\beta(y)^2(\tau)} - \frac{\sigma_{y}^\prime(y)}{\beta^2(\tau)} \right)_{\tau} \\
+ r_2 \int_0^t K(t - \tau) \left( \frac{\sigma_{yy}}{\beta(y)^2(\tau)} - \frac{\sigma_{y}^\prime(y)}{\beta^2(\tau)} \right) d\tau
\]

Then (4.2) simplifies to the final form of the integral equation,
\[
\sigma(y, t) + 2r_1K(0)y_t(y, t)\sigma_y(y, t) - r_2K(0) \left( \frac{\sigma_{yy}(y, t)}{\beta(y)^2(t)} - \frac{\sigma_{y}(y, t)\beta'(y)}{\beta^2(y)^2(t)} \right) \\
+ \int_0^t K(t - \tau) \left[ p_1y_\tau(y, \tau) - r_1 \left( y_{\tau\tau}(y, \tau) - \frac{\tau_\tau'(y, \tau)}{l(\tau)} y_\tau(y, \tau) \right) + \frac{p_2\beta'(y)}{\beta^2(\tau)^2(t)} \right] \sigma_y(y, \tau) d\tau \\
+ \int_0^t K'(t - \tau) \left[ \left(2r_1y_\tau(y, \tau) + \frac{p_2\beta'(y)}{\beta^2(\tau)^2(t)}\right) \sigma(y, \tau) - \frac{r_2\beta'(y)}{\beta^2(\tau)} \sigma_{yy}(y, \tau) \right] d\tau \\
+ \int_0^t K(t - \tau) \left[ - r_1y^2_\tau(y, \tau) - \frac{p_2}{\beta^2(\tau)} \right] \sigma_{yy}(y, \tau) d\tau \\
= q_2 \int_0^t K(t - \tau) \left( \frac{u_y(y, \tau)}{\tau} \right) d\tau + \tau_0 J(t) + r_1K(t)\sigma_t(y, 0).
\]
The proof of the theorem is complete.
4.2 Derivation of Numerical Methods for Models with Variable Adhesion Coefficient

Numerical methods for the variable $\beta$ case are almost identical to those when $\beta$ is constant. We look to build a system of the form (3.10), where each $M_{j,j}$ and $M_{j,k}$ is tri-diagonal. Each block $M_{j,j}$ is defined by

$$M_{j,j} = \begin{bmatrix}
1 - 2D^2_{1,j} & D^1_{1,j} + D^2_{1,j} & \cdots & 0 & 0 \\
-D^1_{2,j} + D^2_{2,j} & 1 - 2D^2_{2,j} & D^1_{2,j} + D^2_{2,j} & \cdots & 0 \\
& \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & -D^1_{m-2,j} + D^2_{m-2,j} & 1 - 2D^2_{m-2,j} & D^1_{m-2,j} + D^2_{m-2,j} \\
0 & 0 & \cdots & -D^1_{m-1,j} + D^2_{m-1,j} & 1 - 2D^2_{m-1,j}
\end{bmatrix},$$

where $j = 1, 2, \ldots, n$, and

$$D^1_{i,j} = \frac{1}{2\Delta y} \left(2r_1K(0)y_i(y_i, t_j) + r_2\frac{\beta'(y_i)}{\beta^2(y_i)l^2(t_j)}\right) + \frac{\Delta t}{4\Delta y} \left[K(0)\left(p_1y_t(y_i, t_j) - r_1\left(y_t(y_i, t_j) - \frac{l'(t_j)}{l(t_j)}y_t(y_i, t_j) + \frac{r_2\beta'(y_i)}{\beta^2(y_i)l^2(t_j)}\right)\right]\right],$$

$$D^2_{i,j} = -\frac{r_2K(0)}{\beta(y_i)l^2(t_j)\Delta y^2} - \frac{\Delta t}{2\Delta y^2} \left[r_2K'(0)\frac{\beta'(y_i)}{\beta(y_i)l^2(t_j)} + K(0)\left(r_1y^2_t(y_i, t_j) + \frac{p_2}{\beta(y_i)l^2(t_j)}\right)\right].$$

Below the main diagonal, each tridiagonal block is defined by

$$M_{j,k} = \begin{bmatrix}
-C^1_{1,j} & C^1_{1,j} + C^2_{1,j} & \cdots & 0 & 0 \\
-C^1_{2,j} + C^2_{2,j} & -2C^2_{2,j} & C^1_{2,j} + C^2_{2,j} & \cdots & 0 \\
& \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & -C^1_{m-2,j} + C^2_{m-2,j} & -2C^2_{m-2,j} & C^1_{m-2,j} + C^2_{m-2,j} \\
0 & 0 & \cdots & -C^1_{m-1,j} + C^2_{m-1,j} & -2C^2_{m-1,j}
\end{bmatrix},$$

where

$$C^1_{i,j} = \frac{\Delta t}{2\Delta y} \left[K(t_j - t_k)\left(p_1y_t(y_i, t_k) - r_1\left(y_t(y_i, t_k) - \frac{l'(t_k)}{l(t_k)}y_t(y_i, t_k) + \frac{p_2\beta'(y_i)}{\beta^2(y_i)l^2(t_k)}\right)\right]\right] + K'(t_j - t_k)\left(2r_1y_t(y_i, t_k) + \frac{r_2\beta'(y_i)}{\beta(y_i)l^2(t_k)}\right),$$

$$C^2_{i,j} = \frac{\Delta t}{\Delta y^2} \left[-K(t_j - t_k)\left(r^2_1y^2_t(y_i, t_k) + \frac{p_2}{\beta l^2(t_k)}\right) - K'(t_j - t_k)\frac{r_2}{\beta(y_i)l^2(t_k)}\right],$$

and $k < j.$

39
As for the right hand side vector, $\vec{b}$, the formulation stays the same. Since $\beta$ is not present on the right hand side in any of the models considered, this must be the case.

The only remaining changes lie in the indexing of $\beta(x)$ when solving for $u(y,t)$, $f'(t)$ and $r'(t)$. The model equations stay the same in each case. All changes to the three algorithms are shown in the pseudocode below.

**Algorithm 6: Driver ($\beta = \beta(x)$)**

**input**: $E_0$, $E_1$, $\mu_0$, $\mu_1$, $r_0$, $L_0$, $T_{final}$, $\beta(y)$, $\tau_0$, $\Delta y$, $\Delta t$

**output**: $\sigma(y,t)$, $u(y,t)$, $l(t)$ for all space and time.

Define $p_1$, $p_2$, $q_1$, and $q_2$ in terms of $E_0$, $E_1$, $\mu_0$, $\mu_1$;

Initialize $\sigma(y,t)$, $u(y,t)$, $y_t(y,t)$, $y_{tt}(y,t)$;

while $||\sigma(y,t) - \sigma_{old}(y,t)|| > Tol$ or $||u(y,t) - u_{old}(y,t)|| > Tol$ do

Compute $u_y$ and $y_{tt}$;

Assign $\sigma_{old}(y,t) = \sigma(y,t)$;

Solve $\sigma(y,t) = \text{SolveForSigma}$;

Compute $\sigma_y(y,t)$;

$r'(t_0 : t_n) = r_0 + \sigma_y(y_1, t_0 : t_n)/(\beta(y_1)l(t_0 : t_n))$;

$f'(t_0 : t_n) = L_0/l(t_0 : t_n) + \sigma_y(y_m, t_0 : t_n)/(\beta(y_m)l(t_0 : t_n))$;

$l'(t_0 : t_n) = f'(t_0 : t_n) - r'(t_0 : t_n)$;

for $j = 1 : N - 1$ do

if $j == 1$ then

$r(t_{j+1}) = r(t_j) + \Delta t \cdot r'(t_j)$;

$f(t_{j+1}) = f(t_j) + \Delta t \cdot f'(t_j)$;

else

$r(t_{j+1}) = (2\Delta t \cdot r'(t_j) + 4r(t_j) - r(t_{j-1}))/3$;

$f(t_{j+1}) = (2\Delta t \cdot f'(t_j) + 4f(t_j) - f(t_{j-1}))/3$;

end if

Update $y_t(y,t) = -(y(f'(t) - r'(t)) + r'(t))/l(t)$;

Assign $u_{old}(y,t) = u(y,t)$;

Solve $u(y,t) = \text{SolveForU}$;
Algorithm 7: SolveForSigma ($\beta = \beta(x)$)

**input**: $p_1, p_2, q_1, q_2, r_1, r_2, y_t(y, t), y_{tt}(y, t), u_y(y, t), l(t), \sigma_t(y, 0), \beta(y), \tau_0, \Delta y, \Delta t$

**output**: $\sigma(y, t)$ for all $(y, t)$ in the unit strip.

Compute $\beta'(y)$;

**for** $j = 1 : N$

Define coefficient on first-order derivatives

$$D_j^1 = \frac{1}{\Delta y} r_1 K(0) y_t(y_i, t_j) + r_2 \frac{\beta'(y_1 : y_{m-1})}{2 \Delta y^2 ((y_1 : y_{m-1})^2(t_j))} + \frac{\Delta t}{4 \Delta y} K(0) \left[ p_1 y_t(y_1 : y_{m-1}, t_j) - r_1 \left( y_t(y_1 : y_{m-1}, t_j - \frac{y'(t_j)}{l(t_j)} y_t(y_1 : y_{m-1}, t_j) \right) \right] + \frac{\Delta t}{4 \Delta y} K'(0) \left( 2 r_1 y_t(y_1 : y_{m-1}, t_j) \right);$$

Create submatrix of first-order derivative coefficients

$M_1 = \text{diag}(D_j^1(y_1 : y_{m-2}, 1)) - \text{diag}(D_j^1(y_2 : y_{m-1}, -1));$

Define coefficient on second-order derivatives

$$D_j^2 = - \frac{r_2 K(0)}{\beta(y_1 : y_{m-1})^2(t_j)} - \frac{\Delta t}{2 \Delta y^2} \left[ \frac{r_2 K'(0)}{\beta(y_1 : y_{m-1})^2(t_j)} + K(0) \left( r_1 y_t^2(y_1 : y_{m-1}, t_j) + \frac{r_2^2 \beta'(y_1 : y_{m-1})}{\beta^2(y_1 : y_{m-1})^2(t_j)} \right) \right];$$

Create submatrix of second-order derivative coefficients

$M_2 = \text{diag}(D_j^2(y_1 : y_{m-2}, 1)) - \text{diag}(2D_j^2(y_1 : y_{m-1})) + \text{diag}(D_j^2(y_2 : y_{m-1}, -1));$

Store submatrix in correct block in lower-triangular $A$

$M_{j,j} = I_{M-1} + M_1 + M_2;

**for** $k = 1 : j$

Define coefficient on first-order derivatives

$$C_{i,j}^{1,k} = \frac{\Delta t}{2 \Delta y} \left[ K(t_j - t_k) \left( p_1 y_t(y_i, t_k) - r_1 \left( y_t(y_i, t_k) - \frac{y'(t_k)}{l(t_k)} y_t(y_i, t_k) \right) \right) + \frac{p_2 \beta'(y_1 : y_{m-1})}{\beta^2(y_1 : y_{m-1})^2(t_k)} \right] + K'(t_j - t_k) \left( 2 r_1 y_t(y_i, t_k) \right);$$

Create submatrix of first-order derivative coefficients

$M_1 = \text{diag}(D_j^1(y_1 : y_{m-2}, 1)) - \text{diag}(D_j^1(y_2 : y_{m-1}, -1));$

Define coefficient on second-order derivatives

$$C_{i,j}^{2,k} = \frac{\Delta t^2}{\Delta y^2} \left[ - K(t_j - t_k) \left( r_1 y_t^2(y_i, t_k) + \frac{r_2}{\beta^2(t_k)} \right) - K'(t_j - t_k) \frac{r_2}{\beta^2(t_k)} \right];$$

Create submatrix of second-order derivative coefficients

$M_2 = \text{diag}(D_j^2(y_1 : y_{m-2}, 1)) - \text{diag}(2D_j^2(y_1 : y_{m-1})) + \text{diag}(D_j^2(y_2 : y_{m-1}, -1));$

Store submatrix in correct block in lower-triangular $A$

$M_{j,k} = M_1 + M_2;$

$$\bar{b}_j = q_2 \int_0^{t_j} \frac{K(t_j - \tau)}{l(\tau)} u_y(y_1 : y_{m-1}, \tau) d\tau + \tau_0 \mathcal{I}(t_j) + r_1 K(t_j) \sigma_t(y_1 : y_{m-1}, 0);$$

Solve $A\vec{\sigma} = \vec{b}$ for $\vec{\sigma}(y, t)$ and reshape to $(m - 1) \times n$ matrix;

Concatenate with zeros for initial/boundary conditions at $y = 0$, $y = 1$ and $t = 0;$

41
Algorithm 8: SolveForU (β = β(x))

\[ \text{input: } y_t(y, t), \sigma_y(y, t), \beta(y), l(t), u_l, u_r, \Delta y, \Delta t \]

\[ \text{output: } u(y, t) \text{ for all } (y, t) \text{ in the unit strip} \]

\[ \text{for } j = 1:n \text{ do} \]
\[ \begin{align*}
\text{Separate coefficients on spatial derivative} \\
a_+ &= \max(y_t(y_0 : y_m, t_j), 0); \\
a_- &= \min(y_t(y_0 : y_m, t_j), 0); \\
\text{Form the matrix } A \\
A &= \text{diag}(1 + \frac{\Delta t}{\Delta y}(a_+ - a_-)) + \text{diag}(\frac{\Delta t}{\Delta y} a_+ - a_-) \\
\text{Form the vector } b \\
b &= u(y_0 : y_m, t_j - 1) + \Delta t \sigma_y(y_0 : y_m, t_j); \\
b(y_0) &= b(y_0) + \frac{\Delta t}{\Delta y} \max(y_t(y_0, t_j), 0) \cdot u_l(t_j); \\
b(y_m) &= b(y_m) - \frac{\Delta t}{\Delta y} \max(y_t(y_m, t_j), 0) \cdot u_r(t_j); \\
\text{Solve } Au = b \text{ for } u(y_0 : y_m, t_j); 
\end{align*} \]

4.3 Numerical Results for Models with Variable Adhesion Coefficient

In this section we compare results where \( \beta = \beta(x) \) and different functions are used for \( \beta(x) \). Two functions for \( \beta(x) \) are a “hump” increase in \( \beta(x) \) at \( x = 4 \) and an oscillating function with period \( \frac{2\pi}{3} \). Plots for the functions are provided in Figure 4.1. The values used in each simulation are provided in Table 4.1 and we let \( \sigma_t(y, 0) = y(1 - y) \).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \beta )</th>
<th>( \tau_0 )</th>
<th>( \sigma_0 )</th>
<th>( \sigma_1 )</th>
<th>( E_0 )</th>
<th>( E_1 )</th>
<th>( \mu_0 )</th>
<th>( \mu_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Linear</td>
<td>5350.4</td>
<td>179.17</td>
<td>0.8</td>
<td>1</td>
<td>10288</td>
<td>10288</td>
<td>53015</td>
<td>n/a</td>
</tr>
<tr>
<td>Burger</td>
<td>5350.4</td>
<td>179.17</td>
<td>0.8</td>
<td>1</td>
<td>10288</td>
<td>5144</td>
<td>53015</td>
<td>53015</td>
</tr>
</tbody>
</table>

Table 4.1: Parameter values for numerical simulation of second-order models.

4.3.1 Numerical Results for the Standard Linear Model

First, we explore the case where the intracellular gel is modeled by the standard linear model. As seen in Figure 4.2 and Figure 4.3, the stress on the cell is greatly affected by the change in \( \beta(x) \). Compared to Figure 2.1, an additional large increase in \( \sigma(y, t) \) occurs when the cell encounters a steep increase in \( \beta(x) \). In each case, we can see that the behavior of the stress is dictated by the effective adhesion coefficient.
If we look at Figure 4.4, we see that the paths of the front and rear boundaries of the cell change very little between the constant $\beta$ and the variable $\beta(x)$ cases. In Figure 4.4, if the graph is above zero, then the boundary is moving slower compared to the constant $\beta$ case, while if the graph is negative, then the boundary is moving faster than the constant $\beta$ case. Note that the difference between the paths in each case is on the order of $10^{-4}$. This means that the position of the boundaries of the cell are almost identical to those in Figure 2.5.

Plots of the displacement $u(y,t)$ are not included, as they are almost identical to those shown in previous chapters.
Figure 4.2: Two views of the surface plot of $\sigma(y, t)$ for standard linear model, where $\beta(x)$ is the “hump” function.
Figure 4.3: Two views of the surface plot of $\sigma(y, t)$ for the standard linear model, where $\beta(x)$ is oscillatory.
Figure 4.4: Plot of $f_{\beta} - f_{\beta(x)}$ (blue) and $r_{\beta} - r_{\beta(x)}$ (green) for the standard linear model.
4.3.2 Numerical Results for the Burger Model

Second, using the same functions for $\beta(x)$, we observe the stress and movement of the cell where the intracellular gel is modeled by the Burger model. We can compare the following plots of $\sigma(y, t)$ below to Figure 3.2.

Comparing Figure 4.5 and Figure 4.6 to the plots of the Burger model with constant $\beta$, we see a large increase in $\sigma(y, t)$ as the cell reaches and crosses the “hump” in $\beta(x)$. In the oscillatory case, we observe oscillations in the stress corresponding to the oscillations in the effective adhesion coefficient. From each case, we observe a direct correlation between increasing substrate stiffness and increasing stress.

Figure 4.7 shows the difference between $f(t)$ and $r(t)$ in the constant $\beta$ case and $f(t)$ and $r(t)$ in each variable $\beta$ case. Again, if the values of the graphs in Figures 4.7 are positive, then the cell is moving faster in the constant case, while negative values of the graphs indicate slower movement when $\beta$ is assumed to be constant. As seen on the vertical axis the differences are small, on the order of $10^{-4}$. Thus, the paths plotted over the full time interval look identical to the paths from the Burger model with $\beta$ held constant in Figure 3.5.

The plots of $u(y, t)$ are not included, as they are almost identical to those shown in previous chapters.
Figure 4.5: Two views of the surface plot of \( \sigma(y,t) \) for the Burger model, where \( \beta(x) \) is the hump function.
Figure 4.6: Two views of the surface plot of $\sigma(y,t)$ for the Burger model, where $\beta(x)$ is oscillatory.
Figure 4.7: Plot of $f_\beta - f_\beta(x)$ (blue) and $r_\beta - r_\beta(x)$ (green) for the Burger model.
Chapter 5

Conclusion

In this thesis, we extended the results of Zheltukhin and Lui in [8] to consider second-order PDEs. In addition, we implemented an integral equation method rather than an explicit finite difference method to solve the model equations. This allows us to avoid higher-order spatial derivatives and simplifies our numerical formulation. Our model resulted in a moving boundary problem, where the PDEs are satisfied between the rear of the cell, \( r(t) \), and the front of the cell, \( f(t) \), for all time. For numerical implementations, we map the domain to the vertical strip \([0, 1] \times \mathbb{R}^+\) by the change of variable \( y = (x - r(t))/(f(t) - r(t)) \). We then discretize space and time in the transformed integral equation, and the result is solving a linear system \( A\vec{\sigma} = \vec{b} \), where \( \vec{\sigma} \) is stress and \( A \) is a block lower-triangular matrix and each block is tridiagonal. Since the displacement \( u(y, t) \) is coupled with \( \sigma(y, t) \), we used a fixed-point approach. In each iteration \( \sigma(y, t) \) and \( u(y, t) \) are updated, eventually converging to a fixed-point within a prescribed tolerance.

We also considered the case where the effective adhesion coefficient \( \beta = \beta(x) \) for a hump function and an oscillatory function. The reason for studying this is to understand how substrate stiffness affects the stress and movement of a cell. Changing the coefficient \( \beta \) is a straightforward way to test the effects of substrate stiffness on cell movement. A more in-depth method involving molecular clutches on the filopodia and elasticity of the compliant substrate has been implemented by Chan and Odde in [3]. Combining our model with the work of Chan and Odde is one idea for future work, although the combination of the time scales in each model may be a challenge.

The results of this thesis allow us to model the cytoplasm of a cell using a wider variety of viscoelastic models including the three cases not studied in [8]. We also studied the effects of a variable coefficient \( \beta \). We found that changes in the constitutive law governing the stress and changes in \( \beta \) affect the stress on the cell, but have no large-scale effects on cell movement. However, on a smaller scale, the changes in front and rear boundaries do depend on the adhesion coefficient \( \beta(x) \).

In our model, the viscous parameters \( \mu \) and the elastic parameters \( E \) are assumed to be constant. In future work we may also assume a inhomogeneous cytoplasm, where the coefficients \( \mu \) and \( E \) depend on the dynamic actin network density, as seen in [8].
Bibliography


Appendix A

Matlab Code

Code for all Matlab functions used in simulations is listed below.

```matlab
function [sigma,ell,f,r,u,betavals] = driver(dy,dt,Tfinal,E0,mu0,betai,r0,L0,
tau0,model,beta_flag)
% Driver for integral equation

if strcmp(model,'Maxwell')
    % Maxwell Model:
    % mu0/E_0 *gamma' + gamma = mu0 eps'
    % Tested
    q1 = 1;
    p2 = mu0; % In Sergey's code, betai is absorbed into p2, but not here
    p1 = mu0/E0;
    q2 = 0;
    r1 = 0;
    r2 = 0;
elseif strcmp(model,'SL')
    % Standard Linear Model:
    % (mu0/E1)*gamma' + gamma = mu0*(1+E2/E1)*eps' + E2*eps
    % Tested
    E1 = E0;
    E2 = E0;
    p1 = mu0/E1;
    q1 = 1;
    p2 = mu0*(1+E2/E1);
    q2 = E2;
    r1 = 0;
    r2 = 0;
elseif strcmp(model,'Model3')
    % Model #3 in Zheltukhin and Lui
    % (mu0/E0)*gamma' + gamma = (mu0*mul1/E0)*eps'' + (mu0+mul1)*eps'
    % Tested
    mul1 = mu0;
```
p1 = mu0/E0;
q1 = 1;
r1 = 0;
p2 = mu0+mu1;
q2 = 0;
r2 = mu0*mu1/E0;

elseif strcmp(model,'Model8')

% Model #3 in Zheltukhin and Lui
%(E0/mu1)*(1+mu0/mu1)*gamma' + gamma = (mu0*mu1/E0)*eps'' + mu1*eps'
% Tested
mu1 = mu0;

p1 = (E0/mu1)*(1+mu0/mu1);
q1 = 1;
r1 = 0;
p2 = mu1;
q2 = 0;
r2 = mu0*mu1/E0;

elseif strcmp(model,'Burger')

% Burger model
%(mu0*mu1/E0)*gamma'' + (mu1/E0+mu1/E1+mu0/E0)*gamma' + gamma = (u0*mu1/(E0*E1))*eps'' + mu1*eps'
% Tested
mu1 = mu0;
E1 = E0;

p1 = mu1/E0 + mu1/E1 + mu0/E0;
q1 = 1;
r1 = 0;
p2 = mu0*mu1/(E0*E1);
q2 = 0;
r2 = mu0*mu1/E0;
end

N = Tfinal/dt+1; % Number of steps in time
M = 1/dy+1;
r = zeros(1,N);
f = [1 zeros(1,N-1)];
ell = f-r; % Initial values for the length of the cell
nell_new = [1 (L0/r0)*ones(1,N-1)];
ellprime = zeros(1,N);

y = (0:dy:1)';
yt = -y*((L0-r0)*ones(1,N)./nell_new)-ones(M,1)*(r0./nell_new);
ytt = spatial_derivative(yt',dt,1)';

u = r0*ones(M,N);
u_new = [zeros(M,1) u(:,2:end)];
sigma = 0;
sigma_old = 1;
sigma.t0 = y.*(1-y);
t = 1;
iter_max=100;
ERR_TOL = 1e-10;
num = numel(u);
while norm(sigma-sigma_old,inf)/num>ERR_TOL || norm(u-u_new,inf)/num>ERR_TOL
    ell = ell_new;
u = u_new;
sigma_old = sigma;

    % Update beta with current values of l(t) and r(t)
betavals = betafunc(y,ell_new,r,betai,beta_flag);

    % Compute spatial derivative of u(y,t)
uy = spatial_derivative(u,dy,1);

    sigma = solve_for_sigma(ell,ellprime,yt,ytt,uy,dy,dt,Tfinal,betavals,p1,p2,q1,q2,r1,r2,tau0,sigma_t0);

    % Compute the length of the cell
    sigma_y = spatial_derivative(sigma,dy,1);

    % Compute rprime for all time
    rprime = r0 + sigma_y(1,:)./(betavals(1,:).*ell);

    % Compute fprime for all time
    fprime = L0./ell + sigma_y(end,:)./(betavals(end,:).*ell);

    ellprime = fprime-rprime;

    % Solve for f and r using second-order Adams-Bashforth
    for j = 1:N-1
        if(j == 1)
            r(j+1) = r(j) + dt*rprime(j); % euler method
            f(j+1) = f(j) + dt*fprime(j); % euler method
        else
            r(j+1) = (2*dt*rprime(j) + 4*r(j) - r(j-1))/3;
            f(j+1) = (2*dt*fprime(j) + 4*f(j) - f(j-1))/3;
        end
    end

    ell_new = f - r;

    % Compute yt and ytt for all y values and each time step
    yt = -y.*(ellprime./ell_new)-ones(M,1)*rprime./ell_new';
    ytt = spatial_derivative(yt',dt,1)';

    t = t+1;
    if t>iter_max
        sigma=sigma';
warning('MATLAB:driver_trapii:iterationLimitExceeded',...  
'Maximum number of iterations reached')
return
end

% Update u using upwind method
u_new = solve_for_u_upwind(u,-rprime*dt,fprime*dt,sigma_y,ell,yt,dt,dy,  
betavals);

if mod(t,10)==0
    disp(['t = ' num2str(t)])
end
end

sigma = sigma';
function sigma = solve_for_sigma(ell, ellprime, yt, yt, uy, dy, dt, Tfinal, ...
    betavals, p1, p2, q1, q2, r1, r2, tau0, sigma, t0)
M = 1/dy + 1;  % Number of steps in space
N = Tfinal/dt + 1;  % Number of steps in time
% Initialize LHS matrix A and RHS vector b
A = zeros((M-2)*(N-1));
b = zeros((M-2)*(N-1),1);
betaprime = spatial_derivative(betavals, dy, 1);  % t derivative of beta
for k = 1:N-1
    % Define coefficients outside integral on first order derivative term
    first_deriv_no_int = (2*r1*yt(2:M-1,k+1) +... 
        r2*betaprime(2:M-1,k+1))./(betavals(2:M-1,k+1).^2*ell(k+1).^2);
    % Define coefficients outside integral on second order derivative term
    second_deriv_no_int = -r2./(betavals(2:end-1,k+1).*ell(k+1).^2);
    % Define the coefficients on the first derivative with kernel K(t)
    first_deriv_K = p1*yt(2:end-1,k+1) - r1*(ytt(2:end-1,k+1) - ... 
        ellprime(k+1)*yt(2:end-1,k+1)/ell(k+1)) + ... 
        p2*betaprime(2:end-1,k+1)./(betavals(2:end-1,k+1).^2*ell(k+1).^2);
    % Define the coefficients on the first derivative with kernel Kprime(t)
    first_deriv_Kprime = 2*r1*yt(2:end-1,k+1) + ... 
        r2*betaprime(2:end-1,k+1)./(betavals(2:end-1,k+1).^2*ell(k+1).^2);
    % Define the coefficients on the second derivative with kernel K(t)
    second_deriv_K = -r1*yt(2:end-1,k+1).^2 - p2./... 
        (betavals(2:end-1,k+1)*ell(k+1).^2);
    % Define the coefficients on the second derivative with kernel Kprime(t)
    second_deriv_Kprime = -r2./(betavals(2:end-1,k+1)*ell(k+1).^2);
    % Define coefficients on first-derivative terms on main diagonal
    D1_j = (1/2*dy)*K(0,p1,q1,r1)*first_deriv_no_int + ... 
        (dt/(4*dy))*K(0,p1,q1,r1)*first_deriv_K +... 
        (dt/(4*dy))*Kprime(0,p1,q1,r1)*first_deriv_Kprime;
    % Assemble first-order portion of block
    M1 = diag(D1_j(1:end-1),1) - diag(D1_j(2:end),-1);
    % Define coefficients on second-derivative terms on main diagonal
    D2_j = (1/dy^2)*K(0,p1,q1,r1)*second_deriv_no_int + ... 
        (dt/(2*dy^2))*K(0,p1,q1,r1)*second_deriv_K +... 
        (dt/(2*dy^2))*Kprime(0,p1,q1,r1)*second_deriv_Kprime;
    % Assemble second-order portion of block
M2 = diag(D2_j(1:end-1),1) - diag(2*D2_j) + diag(D2_j(2:end),-1);

% Assemble full block along main diagonal
A((k-1)*(M-2)+1:k*(M-2),(k-1)*(M-2)+1:k*(M-2)) = eye(M-2) + M1 + M2;

for j = 1:k-1

% Define the coefficients on the first derivative with kernel K(t)
first_deriv_K = p1*yt(2:end-1,j+1) - r1*(ytt(2:end-1,j+1) -
     ellprime(j+1)*yt(2:end-1,j+1)/ell(j+1)) + ...
     p2*betaprime(2:end-1,j+1)./(betavals(2:end-1,j+1).^2*ell(j+1).^2);

% Define the coefficients on the first derivative with kernel Kprime(t)
first_deriv_Kprime = 2*r1*yt(2:end-1,j+1)+ ...
     r2*betaprime(2:end-1,j+1)./(betavals(2:end-1,j+1).^2*ell(j+1).^2);

% Define the coefficients on the second derivative with kernel K(t)
second_deriv_K = -r1*yt(2:end-1,j+1).^2-p2./...
     (betavals(2:end-1,j+1)*ell(j+1)^2);

% Define the coefficients on the second derivative with kernel Kprime(t)
second_deriv_Kprime = -r2./(betavals(2:end-1,j+1)*ell(j+1)^2);

% Define coefficients on first-derivative terms below main diagonal
C1_jk = (dt/(2*dy))*K((k-j)*dt,p1,q1,r1)*first_deriv_K + ...
     (dt/(2*dy))*Kprime((k-j)*dt,p1,q1,r1)*first_deriv_Kprime;

% Assemble first-order portion of block
M1 = diag(C1_jk(1:end-1),1) - diag(C1_jk(2:end),-1);

% Define coefficients on second-derivative terms below main diagonal
C2_jk = (dt/(dy^2))*K((k-j)*dt,p1,q1,r1)*second_deriv_K + ...
     (dt/(dy^2))*Kprime((k-j)*dt,p1,q1,r1)*second_deriv_Kprime;

% Assemble second-order portion of block
M2 = diag(C2_jk(2:end),-1) - diag(2*C2_jk) + diag(C2_jk(1:end-1),1);

% Assemble full block below main diagonal
A((k-1)*(M-2)+1:k*(M-2),(j-1)*(M-2)+1:j*(M-2)) = M1 + M2;
end

% Calculate necessary components for RHS integral computation
t_minus_tau = dt.*(k:-1:0);
K_over_ell = ones(M-2,1)*(K(t_minus_tau,p1,q1,r1)./ell(1:k+1));
RHS_int = q2*K_over_ell.*uy(2:M-1,1:k+1);

% Form RHS vector
b((k-1)+(M-2)+1:k*(M-2)) = tau0*J(k*dt,p1,q1,r1)+dt*trapz(RHS_int,2) + ...
     r1*K(k*dt,p1,q1,r1)*sigma_t0(2:end-1);
% Solve for sigma and reshape to square matrix
sigma = zeros(M,N);
sigma(2:end-1,2:end) = reshape(A\b,M-2,N-1);
end
function u = solve_for_u_upwind(u,u_l,u_r,sigma_y,ell,yt,dt,dy,betavals)

% Solve for u in the equation \( u_t + y_t u_y = \frac{\sigma_y}{\beta \ell} \)
% Upwind method: solve the equation
% \( u_i + (dt/dy) \cdot [a_{\text{plus}} - a_{\text{minus}}] u_i = u_{\text{old}} + dt \cdot \sigma_y/(\beta \ell) \)

n = size(sigma_y,2);

for k = 2:n
    % Define coefficients \( a_{\text{plus}} \) and \( a_{\text{minus}} \) to get correct sign on finite difference terms
    a_plus = max(yt(:,k),0);
    a_minus = min(yt(:,k),0);

    % Form tridiagonal matrix of coefficients
    A = diag(1 + dt*(a_plus-a_minus)/dy) + diag(dt*a_minus(1:end-1)/dy,1) +
        diag(-dt*a_plus(2:end)/dy,-1);

    % Form b vector and apply extrapolated values \( u_{\text{left}} \) and \( u_{\text{right}} \)
    b = u(:,k-1) + dt*sigma_y(:,k)./(betavals(:,k)*ell(k));
    b(1) = b(1) + dt*max(yt(1,k),0)*u_l(k)/dy;
    b(end) = b(end) - dt*min(yt(end,k),0)*u_r(k)/dy;

    % Solve for u
    u(:,k) = A \ b;
end
function y = K(t,p1,q1,r1)
    s1 = -2*q1/(p1+sqrt(p1^2-4*q1*r1));
    s2 = -2*q1/(p1-sqrt(p1^2-4*q1*r1));
    y1 = exp(s1*t)/sqrt(p1^2-4*q1*r1);
    y2 = exp(s2*t)/sqrt(p1^2-4*q1*r1);
    y1(isnan(y1)) = 0;
    y2(isnan(y2)) = 0;
    y = y1-y2;
end

function y = Kprime(t,p1,q1,r1)
    s1 = -2*q1/(p1+sqrt(p1^2-4*q1*r1));
    s2 = -2*q1/(p1-sqrt(p1^2-4*q1*r1));
    y1 = s1*exp(s1*t)/sqrt(p1^2-4*q1*r1);
    y2 = s2*exp(s2*t)/sqrt(p1^2-4*q1*r1);
    y1(isnan(y1)) = 0;
    y2(isnan(y2)) = 0;
    y = y1-y2;
end

function y = J(t,p1,q1,r1)
    s1 = -2*q1/(p1+sqrt(p1^2-4*q1*r1));
    s2 = -2*q1/(p1-sqrt(p1^2-4*q1*r1));
    y1 = p1*exp(s1*t)/(2*sqrt(p1^2-4*q1*r1));
    y2 = p1*exp(s2*t)/(2*sqrt(p1^2-4*q1*r1));
    y3 = exp(s1*t)/2;
    y4 = exp(s2*t)/2;
    y1(isnan(y1)) = 0;
    y2(isnan(y2)) = 0;
    y3(isnan(y3)) = 0;
    y4(isnan(y4)) = 0;
    y = 1-y1+y2-y3-y4;
end
function b = betafunc(y,ell,r,betai,flag)
  % Define x in terms of y
  x = y(:)*(ell(:)')+ones(length(y),1)*(r(:)');
  % Define beta for each x
  switch flag
    case 'constant'
      b = constant(x,betai);
    case 'oscillate'
      b = oscillate(x,betai);
    case 'hump'
      b = hump(x,betai);
  end
end

function y = constant(x,betai)
  y = betai*ones(size(x));
end

function y = hump(x,betai)
  humploc = 4; % origin is moved to this point
  steepness = 2; % larger means more step
  max_height = 2*betai; % size of jump
  min_height = 0.5*betai; % vertical movement of function
  y = (max_height-min_height)./((x - humploc).^2 + steepness) + min_height;
end

function y = oscillate(x,betai)
  osc_per_twopi = 3;
  magnitude = 0.5*betai;
  y = betai+magnitude*sin(osc_per_twopi*x);
end
function z_y = spatial_derivative(z, dy, order)

% given the vector z of size M+1, this program returns the first and second
% derivatives at all grid points using higher-order one-sided difference at
% the end points.

z_y = zeros(size(z));

if order == 1
    z_y(1,:) = -(3*z(1,:) - 4*z(2,:) + z(3,:))./(2*dy);
    z_y(2:end-1,:) = (z(3:end,:) - z(1:end-2,:))./(2*dy);
    z_y(end,:) = (3*z(end,:) - 4*z(end-1,:) + z(end-2,:))./(2*dy);
else
    if order == 2
        z_y(1,:) = (z(1,:) - 2*z(2,:) + z(3,:))./(dy^2);
        z_y(2:end-1,:) = (z(3:end,:) - 2*z(2:end-1,:) + z(1:end-2,:))./(dy^2);
        z_y(end,:) = (z(end,:) - 2*z(end-1,:) + z(end-2,:))./(dy^2);
    end
end