2023

Functional Least Squares Minimization

Kyoung Joo Cox
University Of Alabama At Birmingham

FUNCTIONAL LEAST SQUARES MINIMIZATION

by

KYOUNG JOO COX

IAN W. KNOWLES, COMMITTEE CHAIR
SHANGBING AI
MARIUS N. NKASHAMA
ROGER B. SIDJE
CHENGCUI ZHANG

A DISSERTATION

Submitted to the graduate faculty of the University of Alabama at Birmingham,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

BIRMINGHAM, ALABAMA

2023

FUNCTIONAL LEAST SQUARES MINIMIZATION

KYOUNG JOO COX

APPLIED MATHEMATICS

ABSTRACT

In this thesis, we forecast the trend and price of a selected stock using four new algorithms and a previous mathematical model built from economic 'tendencies'. We use a new variational regularization method to determine the coefficients of the delay differential equation in the previous mathematical model. We also apply our new approach to obtaining initial points for the regularization method as well as a technique for mitigating the exponential growth model error throughout the minimization process of the regularization method. Lastly, we use the newly discovered coefficients together with a new iterative improvement procedure and a knowledge of the stock volatility to greatly improve the price prediction. We compare our model's predictions to actual stock prices, forecasts from a previous model, and predictions from a deep learning model.

DEDICATION

To my parents, Ok Chu Cox, John Kenneth Cox, and Jin Ok Hong, and to my brother, Silven Song Cox, for providing me with continuous support and encouragement throughout all my years of study.

ACKNOWLEDGEMENTS

I want to start by thanking Dr. Ian Knowles, who served as my thesis advisor, for all the time and effort he put into helping me compose this thesis. I am grateful for his consistent support, expert counsel, and unending encouragement during this process.

In addition to my adviser, I am grateful to the other members of my committee: Dr. Shangbing Ai, Dr. Marius Nkashama, Dr. Roger Sidje, and Dr. Chengcui Zhang for their helpful suggestions and assistance.

I also wish to express my gratitude to my parents, my brother, and my sister-in-law for their unwavering support and encouragement. In addition, I want to thank my friends Kevin Campbell, Steven Redolfi, Cameron Mills, Brittany Burdette, Jessica Barnett, and many more who have been there for me.

Finally, I would like to thank the University of Alabama at Birmingham Mathematics Department for their ongoing assistance with both my Masters and Doctorate degrees.

TABLE OF CONTENTS

LIST OF TABLES

Table *Page*

LIST OF FIGURES

Figure *Page*

CHAPTER 1

INTRODUCTION

The predictability of the stock market has long been a source of debate. The supporters of random walk theory[1] assert that the stock market is unpredictable, whereas its opponents, fundamental and technical analysts, believe that the stock market can be predicted by analyzing patterns. Both sides have a plethora of evidence to back up their claims, and the debate continues to this day [**29, 7, 30, 20**].

We believe that while there are undoubtedly random effects present, which do cause the price to "wobble", the broad behavior is basically mechanical and at the base driven by economic forces. The challenge is to isolate these forces in the form of observed relational tendencies (absent reliable economic force laws) between pairs of macro-economic variables. This leads to the need to solve certain inverse problems as we indicate below.

The inverse problem we solve is motivated by Barnett's [**18**, p.46] system of delay differential equations with thirteen different economic variables. Before solving this delay system, whose the solution leads to the trend of Apple's stock, Barnett first solves for the system's coefficients using the least-squares method. Then, Barnett assesses the success of predictions using two metrics: mean absolute percent error (MAPE) and visual inspection. Predictions with less than 10% MAPE are considered successful without further visual inspection, and predictions with greater than 10% MAPE are considered successful if they have similar shapes to actual stock prices. Using both methods yields a success rate of 66%; if the mean absolute percent error is the only measure utilized, Barnett's method yields a success rate of less than 54% [**18**].

[1] closely associated with the supporters of the controversial Efficient Market Hypothesis (EMH), which states that new stock information is immediately reflected in stock pricing

Although Barnett's method produces reasonable predictions, we wish to greatly enhance the forecast accuracy. To accomplish our goal, we employ four new techniques: a new initialization method, an algorithm to mitigate the intrinsic growth defects in the differential equation system itself, the replacement of least squares constants with time varying functions using an inverse problem-solving strategy inspired by Nayak's regularization method [**32**], and finally a new iterative improvement technique using volatility information. Additionally, we recollect the data to forecast closing prices without a stock split, allowing us to compare our results to state-of-the-art Artificial Intelligence (AI) methodologies. Further, to maintain objectivity, we only use mean absolute percent error (MAPE) when measuring our success, and remove the process of visual inspection.

It is important to note that our model makes long-term predictions because the variables used have a long-term impact on Apple's stock price. To be more specific, our model typically predicts one to two months in advance and does not provide effective day-to-day predictions.

The ideas behind our initialization method can also be applied to lower computing costs in other regularization methods. Moreover, the concept of compensating for inherent flaws in models is quite general and can be applied to other mathematical models.

1. The Barnett Model

Barnett's model forecasts Apple's stock trends from November 1996 through February 2019 and predicts Apple's stock prices from August 2018 to December 2018 using Mahato's inverse method for recovering local stock volatility from American option prices [**28**]. The model consists of the following thirteen variables: x_1 represents consumer price index (CPI), x_2 represents inflation rate, x_3 represents Apple's price-earning ratio (P/E ratio), x_4 is federal funds rate, x_5 is Nasdaq-100, x_6 is Apple's earnings, x_7 is Apple's Research and Development (R&D) expenditure, x_8 is Apple's

stock price (AAPL), x_9 represents SPDR Gold Trust (GLD), x_{10} is S&P 500, x_{11} represents Apple's revenue from iPhone sales, x_{12} is consumer confidence index (CCI), and x_{13} is Apple's money spent quarterly on open market buybacks.

Barnett's system of delay differential equations (1.1) is shown below, and the procedure for deriving this system is described in Chapter 6 of Barnett's dissertation [**18**].

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) + c_8 x_{13}(t) \\
x_4'(t) &= c_9 x_1'(t) + c_{10} x_4(t) + c_{11} x_2(t) x_4(t-20) + c_{12} x_8(t) x_4(t) \\
x_5'(t) &= c_{13} x_8(t) + c_{14} x_{10}(t) + c_{15} x_4(t) + c_{16} x_{12}(t) \\
x_6'(t) &= c_{17} x_6(t) + c_{18} x_7(t-17) + c_{19} x_{11}(t) + c_{20} x_{12}(t) \\
x_7'(t) &= c_{21} x_7(t) + c_{22} x_6(t) + c_{23} x_{11}(t) \\
x_8'(t) &= c_{24} x_6(t) + c_{25} x_5'(t) + c_{26} x_8(t) \\
&\quad + c_{27} x_8(t) x_4(t) + c_{28} x_3(t) + c_{29} x_{10}'(t) + c_{30} x_{13}'(t) \\
x_9'(t) &= c_{31} x_{10}(t) + c_{32} x_{12}(t) + c_{33} x_2(t) + c_{34} x_9(t) \\
x_{10}'(t) &= c_{35} x_{10}(t) + c_{36} x_4(t) + c_{37} x_1(t) + c_{38} x_5(t) + c_{39} x_{12}(t) \\
x_{11}'(t) &= c_{40} x_7(t-17) + c_{41} x_{11}(t) + c_{42} x_4(t-20) x_{11}(t) \\
x_{12}'(t) &= c_{43} x_{10}'(t) + c_{44} x_9(t) + c_{45} x_4(t) + c_{46} x_1(t) \\
x_{13}'(t) &= c_{47} x_{10}(t) + c_{48} x_6(t) + c_{49} x_7(t)
\end{aligned}
\tag{1.1}
$$

1.1. Data Collection and Preprocessing. In [**18**], the original data set covers the period from July 1994 to February 2019. It is important to note that Barnett's data for Apple's stock price represents the opening price after a stock split adjustment,

but the stock data that is not Apple's is given without a stock split adjustment. However, we have chosen our stock data uniformly to represent closing stock prices without stock splits, and this allows us to make comparisons with other cutting-edge models; more information on this is in Chapter 4. We now detail the sources and accessibility of the data.

The consumer price index, x_1, and the inflation rate, x_2, are published monthly by the US Bureau of Labor Statistics and the United States Inflation Calculator, respectively [**9, 3**]. The Apple's price-earning ratio, x_3, is calculated every weekday that is not a holiday and is found at YCharts [**15**] and data for federal funds rate or interest rate, x_4, is reported monthly at the Federal Funds Effective Rate [**1**]. The Nasdaq-100, x_5, is an index that covers the top one hundred non-financial businesses listed on the Nasdaq stock exchange, and we obtain this information from Yahoo Finance [**13**]. Both Apple's net income, x_6, and Apple's Research and Development Expenses, x_7, are reported quarterly, and we gather this data from Macrotrends [**4**] and YCharts [**16**] respectively. Further, on every non-holiday weekday, the prices for Apple's shares, x_8, and the SPDR Gold Trust, x_9, are reported and may be viewed at Yahoo Finance [**12, 14**]. Moreover, Apple's revenue from iPhone sales, x_{11}, is disclosed quarterly beginning in September 2007 and can be found at Statista [**8**], and the monthly reported consumer confidence index, x_{12}, can be found at The Organisation for Economic Cooperation and Development (OECD) [**6**]. Finally, Apple's money spent to repurchase shares of its own stock on the open market, x_{13}, can be found at YCharts [**17**]. Since our economic variables' availability varies, we utilize linear interpolation to make up for the days without data. For additional information on linear interpolation, see Barnett's dissertation [**18**, p.49].

The original data set for the American options runs from August to December of 2018. An American option is a style of options contract in which the holder has the right to buy or sell one share of the stock at a specified price K at any time between the time of purchase and a specified future time T. To forecast the price of

Apple's stock, we combine geometric Brownian motion and Mahato's technique for determining local volatility of American option data. To employ Mahato's method, we collect the bid price for the put option from the Nasdaq as shown in Table 1.1, where the strike prices are the closest to the current stock price since those are the most heavily traded [**5**].

Days to 1st expiry	Days to 2nd expiry	Current Stock Price
2	9	133.13
Strike Price	Option Price for 1st Expiry	Option Price for 2nd Expiry
120	0.03	0.2
125	0.15	0.59
130	0.9	1.68
135	3.2	3.9
140	7.25	7.5
145	12.2	12.2

TABLE 1.1. Apple's Put Option Data from January 11, 2023

After preprocessing the data, Barnett forecasts Apple's stock trend and price using the following steps:

(1) Solve for the coefficients of (1.1) using the least squares method.
(2) Predict Apple's stock trend by solving (1.1) using the coefficients found in step (1).
(3) If the bid price for the put option is available, apply Mahato's approach to extract the local volatility and combine it with geometric Brownian motion to forecast stock prices [**31**].

1.2. Solving for the Coefficients. In this section, we describe Barnett's approach to constructing overdetermined systems for the unique least squares coefficients of (1.1).

Let $1 \leq k \leq 7$ and integrate each equation in (1.1) over a fixed subinterval $[t_{k-1}, t_k]$ where each $[t_{k-1}, t_k]$ represents one month. For an example, consider

$$x_8'(t) = c_{24}x_6(t) + c_{25}x_5'(t) + c_{26}x_8(t) + c_{27}x_8(t)x_4(t) + c_{28}x_3(t) + c_{29}x_{10}'(t) + c_{30}x_{13}'(t)$$

Integrating both sides, we get

$$
\begin{aligned}
x_8(t_k) - x_8(t_{k-1}) =& c_{24} \int_{t_{k-1}}^{t_k} x_6(t)dt + c_{25} \int_{t_{k-1}}^{t_k} x_5'(t)dt + c_{26} \int_{t_{k-1}}^{t_k} x_8(t)dt \\
&+ c_{27} \int_{t_{k-1}}^{t_k} x_8(t)x_4(t)dt + c_{28} \int_{t_{k-1}}^{t_k} x_3(t)dt \\
&+ c_{29} \int_{t_{k-1}}^{t_k} x_{10}'(t)dt + c_{30} \int_{t_{k-1}}^{t_k} x_{13}'(t)dt \\
=& c_{24} \int_{t_{k-1}}^{t_k} x_6(t)dt + c_{25} \left(x_5(t_k) - x_5(t_{k-1})\right) + c_{26} \int_{t_{k-1}}^{t_k} x_8(t)dt \\
&+ c_{27} \int_{t_{k-1}}^{t_k} x_8(t)x_4(t)dt + c_{28} \int_{t_{k-1}}^{t_k} x_3(t)dt \\
&+ c_{29} \left(x_{10}(t_k) - x_{10}(t_{k-1})\right) + c_{30} \left(x_{13}(t_k) - x_{13}(t_{k-1})\right)
\end{aligned} \tag{1.2}
$$

Define $A \in \mathbb{R}^{7\times 7}$ by

$$
A = \begin{bmatrix}
\int_{t_0}^{t_1} x_6 & x_5(t_1) - x_5(t_0) & \int_{t_0}^{t_1} x_8 & \int_{t_0}^{t_1} x_8x_4 & \int_{t_0}^{t_1} x_3 & x_{10}(t_1) - x_{10}(t_0) & x_{13}(t_1) - x_{13}(t_0) \\
\int_{t_1}^{t_2} x_6 & x_5(t_2) - x_5(t_1) & \int_{t_1}^{t_2} x_8 & \int_{t_1}^{t_2} x_8x_4 & \int_{t_1}^{t_2} x_3 & x_{10}(t_2) - x_{10}(t_1) & x_{13}(t_2) - x_{13}(t_1) \\
\int_{t_2}^{t_3} x_6 & x_5(t_3) - x_5(t_2) & \int_{t_2}^{t_3} x_8 & \int_{t_2}^{t_3} x_8x_4 & \int_{t_2}^{t_3} x_3 & x_{10}(t_3) - x_{10}(t_2) & x_{13}(t_3) - x_{13}(t_2) \\
\int_{t_3}^{t_4} x_6 & x_5(t_4) - x_5(t_3) & \int_{t_3}^{t_4} x_8 & \int_{t_3}^{t_4} x_8x_4 & \int_{t_3}^{t_4} x_3 & x_{10}(t_4) - x_{10}(t_3) & x_{13}(t_4) - x_{13}(t_3) \\
\int_{t_4}^{t_5} x_6 & x_5(t_5) - x_5(t_4) & \int_{t_4}^{t_5} x_8 & \int_{t_4}^{t_5} x_8x_4 & \int_{t_4}^{t_5} x_3 & x_{10}(t_5) - x_{10}(t_4) & x_{13}(t_5) - x_{13}(t_4) \\
\int_{t_5}^{t_6} x_6 & x_5(t_6) - x_5(t_5) & \int_{t_5}^{t_6} x_8 & \int_{t_5}^{t_6} x_8x_4 & \int_{t_5}^{t_6} x_3 & x_{10}(t_6) - x_{10}(t_5) & x_{13}(t_6) - x_{13}(t_5) \\
\int_{t_6}^{t_7} x_6 & x_5(t_7) - x_5(t_6) & \int_{t_6}^{t_7} x_8 & \int_{t_6}^{t_7} x_8x_4 & \int_{t_6}^{t_7} x_3 & x_{10}(t_7) - x_{10}(t_6) & x_{13}(t_7) - x_{13}(t_6)
\end{bmatrix}
$$

Then, we can express (1.2) as shown below

$$
A \begin{bmatrix} c_{24} \\ c_{25} \\ c_{26} \\ c_{27} \\ c_{28} \\ c_{29} \\ c_{30} \end{bmatrix} = \begin{bmatrix} x_8(t_1) - x_8(t_0) \\ x_8(t_2) - x_8(t_1) \\ x_8(t_3) - x_8(t_2) \\ x_8(t_4) - x_8(t_3) \\ x_8(t_5) - x_8(t_4) \\ x_8(t_6) - x_8(t_5) \\ x_8(t_7) - x_8(t_6) \end{bmatrix} \tag{1.3}
$$

Equation (1.3) can now be solved using the least squares method and we denote the solution $\begin{bmatrix} c_{24} \cdots c_{30} \end{bmatrix}^T$ by $\begin{bmatrix} J_{24} \cdots J_{30} \end{bmatrix}^T$, where $c_i = J_i$ for $24 \leq i \leq 30$. The rest of the coefficients are found in the same manner as described above, and we represent the solution as $J = \begin{bmatrix} J_1 & \cdots & J_{49} \end{bmatrix}^T$ where $c_i = J_i$ for $1 \leq i \leq 49$. We now note that this method always results in an overdetermined system because each equation in (1.1) has at most seven terms. As a result, the coefficients found using this method are unique. Further, since these coefficients are solved on the interval of $[t_0, t_7]$, t_7 is the start time for the delay differential system (1.1).

Remark 1.1. Although utilising more than seven months of data can result in overdetermined systems, Barnett recommends using only seven months of past data because it yields the best outcomes. As a result, the number of past months used to calculate the coefficients is fixed. Moreover, the coefficients obtained by the least squares method are always constant functions of time.

Remark 1.2. In Barnett's code, the order of the terms in the system is altered, and the coefficients are reordered according to the row ordering shown below:

$$\begin{array}{ccccccc} c_1 \cdots c_7 & c_9 & c_{11} & c_{10} & c_{12} \cdots c_{18} & c_{20} & c_{19} \\ c_{21} \cdots c_{25} & c_{28} & c_{27} & c_{26} & c_{31} \cdots c_{34} & c_{36} \cdots c_{39} & c_{35} \\ c_{41} & c_{40} & c_{42} & c_{43} & c_{45} & c_{46} & c_{44} \\ & & c_{47} \cdots c_{49} & c_{29} & c_{30} & c_8 & \end{array}$$

Rearranging the terms does not affect Barnett's code. However, when using our new algorithm, this must be considered. See Chapter 6 for details.

2. Outline of the New Approach

The primary goal of our method is improved the forecasting of Apple's stock prices and trends. To do so, we begin by improving the coefficients for (1.1) as we lift the restriction on the number of past months used. In addition, we discovered

certain flaws in (1.1) and set a few natural restrictions on the coefficients to remedy the problem.

To obtain the coefficients of (1.1), we employ a new initialization technique in a regularization method influenced by Nayak, instead of the least squares method. Regularization methods are used to address ill-posed problems by minimizing a certain functional. One of the best-known methods for solving an ill-posed problem is Tikhonov regularization. However, with Tikhonov regularization, choosing the parameter in the functional is an essential component of the minimization process, and determining the best regularization parameter is a familiar irritation and still an open problem. Even though Nayak's regularization functional appears to be quite similar to Tikhonov's, they differ markedly in that Nayak's regularization discards the regularization parameter and the regularization term is minimized by the same function that minimizes the functional (see Chapter 3 for details). Furthermore, in Chapter 4, we demonstrate the stability of Nayak's regularization method, which has yet to be proven.

We address the significance of starting points for the descent process in Chapter 5. When the linear operator in the ill-posed problem is not injective, we face the issue of having infinitely many solutions, with each solution depending on the starting point. As a result, it is necessary to locate 'good' starting points, and we develop a strategy to always find better initial points than $\mathbf{0}$ or J, where J represents the solution obtained using Barnett's least-squares method.

Now, (1.1) also has exponential growth model errors that cause some of the solutions to be unrealistic. In Chapter 5, we place two requirements on the solutions of (1.1) to overcome this issue and these requirements are used in finding initial points and as stopping criteria for the minimization process.

Lastly, unlike Barnett's method for finding the coefficients of (1.1), our descent method can be applied to any time interval. To highlight the effectiveness of our model, we decided to use one month of past data to calculate the coefficients and

forecast the trend and price of Apple stock for the following two months, and the results are shown in Chapter 6.

3. Neural-Network Based Forecasting Models

To assess the performance of our new method, we chose to compare it to a neural network-based forecasting model that we developed. We used the TensorFlow framework, and Sisodia's model [**35**] served as structural inspiration.

Neural networks are a subsection of a field of study known as artificial intelligence (AI). Artificial intelligence refers to the simulation of human intellect in machines [**24**]. Machine learning is one of the areas of AI and it focuses on simulating human learning while progressively increasing precision by using data and algorithms. A subset of machine learning is neural networks, also known as artificial neural networks (ANN). ANNs are composed of node layers, with each layer connected to the next so that if the output of any node surpasses a certain threshold, the node transmits data to the next layer of nodes. There is an input layer, one or more hidden layers, and an output layer on each node layer. Artificial neural networks with four or more layers constitute what is known as deep learning [**11**].

In the machine learning based model we built, we used a type of ANN called a recurrent neural network (RNN) which uses sequential data or time series data [**10**]. Long Short Term Memory (LSTM) is a form of RNN that does not have vanishing gradients and is commonly utilized in dealing with time series data, speech, and texts. The LSTM makes use of a variety of gates to control how information enters and exits the network. The input gate decides what new information should be added to the network's long-term memory based on the previous hidden state and incoming input data. The forget gate decides which information to remove based on the usefulness of both the new input data and the old hidden state. The output gate decides how much information to reveal by determining the new hidden state using the newly updated cell state, the old hidden state, and the new input data [**21**].

The model we built is a sequential model with seven levels, including the input and the output layer. Other than the output layer, each layer has an LSTM layer and a dropout layer where the dropout layers are employed to address overfitting problems. To avoid confusion, we refer to this deep-learning model we built as the LSTM model and our regularization model as 'our model'. The only data we need for the LSTM model is Apple's closing stock prices, and the length of the training and testing data can be adjusted freely. The training and testing sets are often divided into 90% and 10% of the data, respectively. However, to compare the LSTM model to our model, we restrict the training set to the previous month's data and the testing set to the following two months of data.

CHAPTER 2

Preliminary Theory

In this chapter, we review some of the known theory used in Barnett's method, such as existence and uniqueness of solutions to delay differential equations, stochastic processes, and geometric Brownian motion.

1. Delay Differential Systems

Delay differential equations are a type of differential equation in which some or all of the equation's terms are expressed in earlier times. We can see from this definition that (1.1) is a delay differential equation. Throughout this chapter, we use the theorems and definitions from [**22**] without providing proof to deduce that solutions to our delay differential equation exist and are unique. The proof for these theorems may also be found in Barnett's dissertation [**18**, p.18-26].

Definition 2.1 (Delay Differential Equations)**.** *Let* $x : [t_0 - r, \beta) \to \mathbb{R}^n$ *and* $f : [t_0, \beta) \times D^m \to \mathbb{R}^n$ *for some* $\beta > t_0$ *and some open set* $D \subseteq \mathbb{R}^n$*. Suppose that* $t - r \leq g_i(t) \leq t$ *for* $t \geq t_0$ *and* $i = 1, \cdots, m$ *for some* $r \geq 0$*. The initial value problem*

$$x'(t) = f\left(t, x(g_1(t)), \cdots, x(g_m(t))\right), \qquad x(t) = x_0(t) \text{ for } t_0 - r \leq t \leq t_0, \tag{2.1}$$

is called a bounded delay differential equation.

Definition 2.2. *Let* C *be the set* $C\left([-r, 0], \mathbb{R}^n\right)$ *of all continuous functions mapping* $[-r, 0] \to \mathbb{R}^n$*. Let* $A \subseteq \mathbb{R}^n$*. We define*

$$C_A = C([-r, 0], A)$$

Definition 2.3 (Locally Lipschitz)**.** *The function $F : J \times C_D \to \mathbb{R}^n$ is locally Lipschitzian if for each given $(t, x) \in J \times C_D$ there exists constants $a > 0$ and $b > 0$ such that*

$$\mathcal{E} = ([t - a, t + a] \cap J) \times \{y \in C : \|y - x\|_r \leq b\} \subseteq J \times C_D$$

and F is Lipschitzian on $\mathcal{E}$ and the Lipschitz constant L depend on the point (t, x).

The proofs of the following theorems can be found in Barnett's dissertation [**18**, Chapter 4].

Theorem 2.1 (Uniqueness)**.** *Let $F : [t_0, \beta) \times C_D \to \mathbb{R}^n$ be continuous and locally Lipschitzian. Then, given any $\phi \in C_D$,* (2.1) *has at most one solution on $[t_0 - r, \beta_1)$ for any $\beta_1 \in (t_0, \beta]$.*

Theorem 2.2 (Local Existence)**.** *Let $F : [t_0, \beta) \times C_D \to \mathbb{R}^n$ be continuous and locally Lipschitzian. Then, for each $\phi \in C_D$, the initial value problem given in* (2.1) *has a unique solution on $[t_0 - r, t_0 + \delta)$ for some $\delta > 0$*

We now show existence and uniqueness of our delay differential equation (1.1). For each of the thirteen variables, let $[a_i, b_i]$ represent the domain such that $x_i : [a_i, b_i] \to \mathbb{R}$ for $1 \leq i \leq 13$. Recall that x_4 and x_7 have delays. Further define ψ_4 and ψ_7 as the following:

$$\psi_4 : [a_4, b_4] \to [a_4 - 20, b_4 - 20] : t \mapsto t - 20$$

$$\psi_7 : [a_7, b_7] \to [a_7 - 17, b_7 - 17] : t \mapsto t - 17$$

Further, define x_{4d} and x_{7d} such that

$$x_{4d} : [a_4, b_4] \to \mathbb{R} : s \mapsto x_4(\psi_4(s))$$

$$x_{7d} : [a_7, b_7] \to \mathbb{R} : s \mapsto x_7(\psi_7(s))$$

Let $\beta > t_0$ and $D \subseteq \mathbb{R}$ be some open set. Define $f : [t_0, \beta) \times D^{15} \to \mathbb{R}^{13} : (t, x_1, \cdots, x_{13}, x_{4d}, x_{7d}) \mapsto \mathbb{R}^{13}$ such that:

$$f\left(t, x_1, \cdots, x_{13}, x_{4d}, x_{7d}\right) = \begin{bmatrix} c_1 x_2 x_1 \\ c_2 x_2 + c_3 x_{4d} x_2 + c_4 x_2 \\ c_5 x_2 + c_6 x_6 + x_7 x_8 + c_8 x_{13} \\ c_9 c_1 x_1 x_2 + c_{10} x_4 + c_{11} x_2 x_{4d} + c_{12} x_8 x_4 \\ c_{13} x_8 + c_{14} x_{10} + c_{15} x_4 + c_{16} x_{12} \\ c_{17} x_6 + c_{18} x_{7d} + c_{19} x_{11} + c_{20} x_{12} \\ c_{21} x_7 + c_{22} x_6 + c_{23} x_{11} \\ c_{24} x_6 + c_{25} c_{13} x_8 + c_{25} c_{14} x_{10} + c_{25} c_{15} x_4 + ... \\ ...c_{25} c_{16} x_{12} + c_{26} x_8 + c_{27} x_8 x_4 + c_{28} x_3 + ... \\ ...c_{29} c_{35} x_{10} + c_{29} c_{36} x_4 + c_{29} c_{37} x_1 + ... \\ ...c_{29} c_{38} x_5 + c_{29} c_{39} x_{12} + c_{30} c_{47} x_{10} + ... \\ ...c_{30} c_{48} x_6 + c_{30} c_{49} x_7 \\ c_{31} x_{10} + c_{32} x_{12} + c_{33} x_2 + c_{34} x_9 \\ c_{35} x_{10} + c_{36} x_4 + c_{37} x_1 + c_{38} x_5 + c_{39} x_{12} \\ c_{40} x_{7d} + x_{41} x_{11} + c_{42} x_{4d} x_{11} \\ c_{43} x_{35} x_{10} + c_{43} c_{36} x_4 + c_{43} c_{37} x_1 + ... \\ ...c_{43} c_{38} x_5 + c_{43} c_{39} x_{12} + c_{44} x_9 + c_{45} x_4 + c_{46} x_1 \\ c_{47} x_{10} + c_{48} x_6 + c_{49} x_7 \end{bmatrix} \tag{2.2}$$

Notice that (2.2) is a linear combination of products of continuous functions. As a result, we can conclude that (2.2) is continuous. It is also easy to see that f above is continuously differentiable since f has continuous first partial derivatives with respect to all of its dependent arguments (each component of each derivative is continuous). Since (2.2) is continuously differentiable, we see that it is locally Lipschitzian. Therefore, the unique solution to our delay differential equation (1.1) exists [**18**].

2. About dde23

We use the MATLAB command **dde23** to solve our system of delay differential equations. As discussed in the previous section, each solution $\begin{bmatrix} x_1 & \cdots & x_{13} \end{bmatrix}^T \in$

$\mathcal{L}^2(\Omega)^{13}$ exists and is unique. Furthermore, x_8 of the solution represents our model stock price.

The **dde23** command only considers the system of delay differential equations of the form

$$y'(t) = f\left(t, y(t), y(t-\tau_1), \cdots, y(t-\tau_k)\right) \tag{2.3}$$

where $a \leq t \leq b$. Here, $\tau_1, \cdots, \tau_k$ are the constant delays such that $\tau_i > 0$ for all i.

It should also be noted that **dde23** does not accept vectorized functions as a history function. Thus, a history function must be represented symbolically. Let $T = \max_{1\leq i\leq k} \tau_i$. Recall that Barnett uses prior seven months of information $[t_{k-1}, t_k]$, where $1 \leq k \leq 7$ and each $[t_{k-1}, t_k]$ represents one month, to find the coefficients of (1.1). We define a history function

$$\phi : [a-T, a] \to \mathbb{R} : t \mapsto \Big[x_1(t_7), \cdots, x_{13}(t_7)\Big]^T$$

such that $y(t) = \phi(t)$ for $t \leq a$ [**34**].

3. Geometric Brownian Motion Stock Model

In this section, we briefly outline the concepts and definitions of geometric Brownian motion and stochastic processes; for more information, Barnett's dissertation [**18**, Chapter 3]. Let $(\Omega, \mathcal{F}, \mathbb{P})$ represent the probability space where Ω is the sample space, $\mathcal{F}$ is a sigma algebra of Ω, and $\mathbb{P}$ is a probability measure on $(\Omega, \mathcal{F})$ such that $\mathbb{P}(\Omega) = 1$.

Definition 2.4 (Random Variable)**.** *If $X : \Omega \to \mathbb{R}$ is a measurable function, then X is a random variable.*

Definition 2.5 (Stochastic Process)**.** *Let $X_t : \Omega \to \mathbb{R} : \omega \mapsto X_t(\omega)$. We define $\{X_t : t \in [0, \infty)\}$ as a stochastic process.*

Definition 2.6 (Stochastic Differential Equation)**.** *A stochastic differential equation is a differential equation with a stochastic process in one or more terms* [**37**].

Definition 2.7. *Brownian motion is a collection of random variables* $B = \{B_t\}_{t\geq 0}$ *defined on* $(\Omega, \mathcal{F}, \mathbb{P})$ *and satisfying the following properties.*

(1) *For* $0 \leq t_0 < t_1 < \cdots < t_n$ *the increments* $B_{t_n} - B_{t_{n-1}}, \cdots, B_{t_1} - B_{t_0}$ *are independent.*

(2) *For all* $s < t$*, the distribution of* $B_t - B_s$ *is* $\mathcal{N}(0, t-s)$*. In other words, the increments* $B_t - B_s$ *are normally distributed with mean* 0 *and variance* $t - s$*.*

(3) *For almost every* $\omega \in \Omega$*, the function* $t \mapsto B_t(\omega)$ *is continuous on* $[0, \infty)$*.*

Further, if $B_0 = 0$*, we use the term standard Brownian motion.*

Geometric Brownian motion is a stochastic process with the property that the logarithm of the randomly varying quantity follows a Brownian motion with drift (the average rate of growth of the asset price) [**36**]. In 1965, Paul Samuelson proposed a stochastic differential equation that models stock prices using geometric Brownian motion, as shown below: [**33**]

$$dS(t) = \mu(S(t), t)S(t)dt + \sigma(S(t), t)S(t)dB(t). \tag{2.4}$$

Here μ measures the drift or the average rate of growth of the stock price, $\sigma(S, t)$ is the volatility or the measure of an asset's price swing around the mean price [**25**], $B(t)$ is Brownian motion, and $S(t)$ is the stock price at time t.

To recover the average rate of growth of the stock price, μ, we consider (2.4) without the Brownian motion term as shown below:

$$dS(t) = \mu(S(t), t)S(t)dt. \tag{2.5}$$

We must note that the stock price, $S(t)$ in (2.4), is continuous but not differentiable. However, the solution x_8 to our differential equation (1.1) which is an

estimated stock trend is indeed differentiable. Hence, we now re-write (2.5) as $dx_8(t) \approx \mu(S(t),t)x_8(t)dt$, or $\dfrac{x_8'(t)}{x_8(t)} \approx \mu(S(t),t)$.

Mahato's method, described in [**31**], is used to recover the local volatility $\sigma(S,t)$. After recovering μ and σ, we can now solve (2.4) using the **gbm** command in MATLAB. Note here that the solution to (2.4) is a stochastic process. To determine the predicted stock price, we solve the equation 500 times and average the resulting stochastic processes. This mean is used to forecast the stock price.

CHAPTER 3

Inverse Problems

As stated in Remark 1.1, the coefficients obtained using the least squares method are restricted to be constant functions of time. In this chapter, we consider alternative methods for computing the coefficients of our differential equation (1.1), that allow them to be non-constant functions of time. Following a brief introduction to the notion of inverse problems, we discuss some popular methods for solving inverse problems, and then our approach, which is motivated by Nayak's method [**32**].

1. Background Information on Inverse Problems

Jacques Hadamard argued that a mathematical model of physical phenomena is considered well-posed if a solution exists, the solution is unique, and the solution changes continuously with respect to the initial conditions. In light of this, we consider a mathematical problem to be **well-posed** if it meets the criteria listed below [**32**]:

(1) **Existence:** For all (suitable) data, there exists a solution of the problem (in an appropriate sense).

(2) **Uniqueness:** There is at most one solution to the problem.

(3) **Stability:** The solution depends continuously on the data.

A problem that is NOT well-posed is called **ill-posed**. In mathematical modeling, the problem of ill-posedness can often be found in solving inverse problems. In 1976, Keller wrote in [**27**]:

> "We call two problems inverses of one another if the formulation of each involves all or part of the solution of the other. Often, for historical reasons, one of the two problems has been studied extensively for some time, while the other is newer and not so well

understood. In such cases, the former problem is called the direct problem, while the latter is called the inverse problem."

Keller serves as an inspiration for how we identify an inverse problem, insofar as we view inverse problems as determining causes (unknown) that have caused an observed effect (known). Consider the following equation:

$$T\varphi = g. \tag{3.1}$$

We say that (3.1) is an inverse problem if an operator T and the data g are given, and we are looking to find the function φ.

2. Ill-posed Problems

In this section, we prove that compact operators T lead to ill-posed problems, and derive the need for regularization methods. Though we have Hadamard's definition of ill-posedness, we use Theorem 3.3 below to identify an ill-posed problem with a compact operator. Theorem 3.3 results from the Moore-Penrose inverse being a densely defined unbounded linear operator, which is shown using the singular value decomposition of compact operators. The content covered in this section is inspired by the lecture notes of Kekkonen and Korolev [**26**] as well as Nayak's dissertation [**32**].

Throughout this section, $\mathbf{H}$ and $\mathbf{H}_i$ for $i \in \mathbb{N}$ represent Hilbert spaces and we consider a general inverse problem

$$T\varphi = g \tag{3.2}$$

where $T : \mathbf{H}_1 \to \mathbf{H}_2$ is a linear bounded operator. We should note here that (3.2) may not be solvable for all $g \in \mathbf{H}_2$ if $\mathcal{R}(T)$ is not dense in $\mathbf{H}_2$ and if $g \notin \overline{\mathcal{R}(T)}$ then we must look for a solution $\varphi \in \mathbf{H}_1$ such that $T\varphi$ is the closest to g. Cases like these motivate the definition below of a least-squares solution.

Definition 3.1 (Least-Squares Solution)**.** *An element $\varphi \in \mathbf{H}_1$ is called a least-squares solution of $T\varphi = g$ if*

$$\|T\varphi - g\|_{\mathbf{H}_2} = \inf\left\{\|T\psi - g\|_{\mathbf{H}_2} : \psi \in \mathbf{H}_1\right\}$$

In general, the least-squares solution for any given $g \in \mathbf{H}_2$ may not exist because $\mathcal{R}(T)$ is not always closed in $\mathbf{H}_2$. To be precise, the least-squares solution exists if and only if $g \in \mathcal{R}(T) \oplus \mathcal{R}(T)^{\perp}$. Even if the least-squares solution exists, the inverse problem solution may be non-unique if $\mathcal{N}(T) \neq \{0\}$. In the latter case, we denote $\mathbb{L}$ as the set of all least-squares solution to $T\varphi = g$ and to determine the 'best solution', we use the following definition.

Definition 3.2 (Minimal-Norm Solution)**.** *φ is the minimal-norm solution of $T\varphi = g$ if*

$$\|\varphi\|_{\mathbf{H}_1} = \inf\left\{\|\psi\|_{\mathbf{H}_1} : \psi \in \mathbb{L}\right\}$$

We denote this minimal norm solution by $\varphi^{\dagger}$.

2.1. Moore-Penrose Inverse. In the case where $\mathbb{L} \neq \emptyset$, we know that a unique minimal-norm solution exists and this can be computed using the Moore-Penrose pseudo-inverse (proof for this statement can be found in [**26**, p.18]).

Definition 3.3 (Moore-Penrose Inverse)**.** *Let $T : \mathbf{H}_1 \to \mathbf{H}_2$ and let $\hat{T} := T|_{\mathcal{N}(T)^{\perp}}$. The Moore-Penrose inverse $T^{\dagger}$ is defined as the unique linear extension of $\hat{T}^{-1}$ to*

$$\mathcal{D}(T^{\dagger}) := \mathcal{R}(T) \oplus \mathcal{R}(T)^{\perp}$$

with

$$\mathcal{N}(T^{\dagger}) = \mathcal{R}(T)^{\perp}$$

The following statements are important properties of the Moore-Penrose inverse, and details about them can be found in [**26**]. The Moore-Penrose inverse is well-defined

and is continuous if and only if $\mathcal{R}(T)$ is closed. Further, it satisfies the following equations:

(1) $TT^\dagger T = T$

(2) $T^\dagger T T^\dagger = T^\dagger$

(3) $T^\dagger T = I - P_{\mathcal{N}(A)}$

(4) $TT^\dagger = P_{\overline{\mathcal{R}(A)}}|_{\mathcal{D}(T^\dagger)}$

where $P_{\mathcal{N}(T)}$ and $P_{\overline{\mathcal{N}(T)}}$ represents the orthogonal projections on $\mathcal{N}(T)$ and $\overline{\mathcal{N}(T)}$, respectively. Lastly, as mentioned above, for each $g \in \mathcal{D}(T^\dagger)$, the minimal-norm solution can be found using the Moore-Penrose inverse via $\varphi^\dagger = T^\dagger g$.

2.2. SVD of Compact Operators. Let $\mathcal{K}(\mathbf{H}_1, \mathbf{H}_2)$ denote the space of compact operators. For compact operators, we can find a singular value decomposition, which is similar to that in finite dimensional linear algebra. The proofs of the following theorems can be found in [**26**, p.21-23].

Theorem 3.1. *Let $T \in \mathcal{K}(\mathbf{H}_1, \mathbf{H}_2)$. Then there exists*

(1) *a not-necessarily infinite null sequence $\{\sigma_j\}_{j\in\mathbb{N}}$ with $\sigma_1 \geq \cdots > 0$,*

(2) *an orthonormal basis $\{\psi_j\}_{j\in\mathbb{N}} \subset \mathbf{H}_1$ of $\mathcal{N}(T)^\perp$,*

(3) *an orthonormal basis $\{\eta_j\}_{j\in\mathbb{N}} \subset \mathbf{H}_2$ of $\overline{\mathcal{R}(T)}$ with*

$$A\psi_j = \sigma_j \eta_j \qquad A^*\eta_j = \sigma_j \psi_j \qquad \text{for all } j \in \mathbb{N}.$$

Moreover, for all $\varphi \in \mathbf{H}_1$,

$$T\varphi = \sum_{j=1}^{\infty} \sigma_j \langle \varphi, \psi_j \rangle_{\mathbf{H}_1} \eta_j. \tag{3.3}$$

The sequence $\{(\sigma_j, \psi_j, \eta_j)\}$ is called the SVD of T. Further,

$$T^* g = \sum_{j=1}^{\infty} \sigma_j \langle g, \eta_j \rangle_{\mathbf{H}_2} \psi_j. \tag{3.4}$$

With this theorem, Kekkonen and Korolev are able to represent the Moore-Penrose inverse of T using the SVD of T.

Theorem 3.2. *Let $T \in \mathcal{K}(\mathbf{H}_1, \mathbf{H}_2)$ with singular system $\{(\sigma_j, \psi_j, \eta_j)\}_{j\in\mathbb{N}}$ and $g \in \mathcal{D}(T^\dagger)$. Then the Moore-Penrose inverse of T can be written as*

$$T^\dagger g = \sum_{j=1}^{\infty} \sigma_j^{-1} \langle g, \eta_j \rangle_{\mathbf{H}_2} \psi_j. \tag{3.5}$$

From this theorem, we can deduce that the SVD of $T^\dagger$ is $\left\{(\sigma_j^{-1}, \psi_j, \eta_j)_{j\in\mathbb{N}}\right\}$, and since $\varphi^\dagger = T^\dagger g$, the equation (3.5) only makes sense if the series converges. However, since $\{\sigma_j\}_{j\in\mathbb{N}}$ is a decreasing sequence, we can see that the Moore-Penrose inverse is unbounded if $\mathcal{R}(T)$ is infinite dimensional which implies that (3.5) may not converge. Thus, we introduce a convergence criterion for the series called the *Picard criterion* where, if the criteria are met, then the minimal-norm solution exists.

Definition 3.4 (Picard Criterion)**.** *We say that the data g satisfies the Picard criterion if*

$$\left\|T^\dagger g\right\|_{\mathbf{H}_1}^2 = \sum_{j=1}^{\infty} \frac{|\langle g, \eta_j \rangle_{\mathbf{H}_2}|^2}{\sigma_j^2} < \infty \tag{3.6}$$

In other words, we know that the minimal-norm solution exists if the coefficients $\langle g, \eta_j \rangle_{\mathbf{H}_2}$ decay sufficiently fast relative to σ_j. In addition, it is clear that the instability of an inverse problem is determined by the rate of singular value decay. Now, consider a normalized sequence $\{\eta_i\}$. Then, we see that for each η_j,

$$\left\|T^\dagger \eta_j\right\|_{\mathbf{H}_1} = \frac{1}{\sigma_j}$$

and $\lim_{j\to\infty} \frac{1}{\sigma_j} = \infty$, which establishes the unboundedness of the generalized inverse $T^\dagger$ for a compact T. This is the motivation behind the following theorem which makes rigorous the widely held, but vague, view that 'inverse problems are usually ill-posed'.

Theorem 3.3 (Ill-posedness)**.** *Let $T : \mathbf{H}_1 \to \mathbf{H}_2$ be compact with $\dim(\mathcal{R}(T)) = \infty$. Then the corresponding inverse problem $T\varphi = g$ is ill-posed* [**32**, p.37].

2.3. Regularization. Recall our general inverse problem $T\varphi = g$ where

$T : \mathbf{H}_1 \to \mathbf{H}_2$, $g \in \mathbf{H}_2$ and $\varphi \in \mathbf{H}_1$. Let g_δ represent noisy data with $\|g_\delta - g\|_{\mathbf{H}_2} \leq \delta$ for some $\delta \in \mathbb{R}$. With a well-posed problem, we generally expect $T^\dagger g_\delta$ to converge to $T^\dagger g$ as $\delta \to 0$. We saw in the previous subsection that the Moore-Penrose inverse is unbounded for compact operators T with $\dim(\mathcal{R}(T)) = \infty$. Hence, in the case of ill-posed problems, convergence usually does not happen, and we must replace $T^\dagger$ with a family of well-posed and bounded operators. This process is called regularization. In layman's terms, regularization is the process of re-formulating an ill-posed problem into a well-posed problem, which is used to obtain a stable approximation of the solution. For the rest of this subsection, we briefly summarize Chapter 3 - 5 of Nayak's dissertation [**32**, p.40-75].

Let $T : \mathbf{H}_1 \to \mathbf{H}_2$ be a bounded linear operator. A family of $\{R_\alpha\}_{\alpha>0}$ of continuous operators is called a *regularization* of $T^\dagger$ if

$$R_\alpha g \to T^\dagger g = \varphi^\dagger$$

for all $g \in \mathcal{D}(T^\dagger)$ as $\alpha \to 0$.

We now derive a specific regularization by constructing a family of continuous operators. Let $\{(\sigma_i, \psi_i, \eta_i)\}$ be the SVD of T and $g_\alpha : \mathbb{R}_+ \to \mathbb{R}_+$ be a bounded function with the property that $\lim_{\alpha\to 0} g_\alpha(\sigma_i) = \frac{1}{\sigma_i}$ for all σ_i. Further, let $R_\alpha : \mathbf{H}_2 \to \mathbf{H}_1$ be defined as

$$R_\alpha \eta := \sum_{i=1}^{\infty} g_\alpha(\sigma_i)\psi_i \langle \eta, \eta_i \rangle_{\mathbf{H}_2} \tag{3.7}$$

Since g_α is bounded, we know that for all $\sigma \in \mathbb{R}_+$, $g_\alpha(\sigma) \leq C_\alpha$ for some constant $C_\alpha < \infty$. Therefore, as Nayak has shown in his dissertation [**32**, p.40], we see that R_α is bounded, which implies the continuity of R_α. Thus, $\{R_\alpha\}$ is a family of continuous operators for all $\eta \in \mathcal{D}(T^\dagger)$ and for $\eta \notin \mathcal{D}(T^\dagger)$, we must expect $\|R_\alpha \eta\|_{\mathbf{H}_1} \to \infty$ due to the unboundedness of the generalized inverse. Now for noisy data g_δ with

$\|g_\delta - g\|_{\mathbf{H}_2} \leq \delta$, we have

$$\begin{aligned} \left\|R_\alpha g_\delta - T^\dagger g\right\|_{\mathbf{H}_1} &\leq \|R_\alpha g_\delta - R_\alpha g\|_{\mathbf{H}_1} + \left\|R_\alpha g - T^\dagger g\right\|_{\mathbf{H}_1} \\ &\leq \delta \|R_\alpha\|_{\mathbf{H}_1} + \left\|R_\alpha g - T^\dagger g\right\|_{\mathbf{H}_1}. \end{aligned} \tag{3.8}$$

Notice here that $\delta \|R_\alpha\|_{\mathbf{H}_1}$ does not stay bounded as $\alpha \to 0$ while the second term converges to zero as $\alpha \to 0$ which implies the pointwise convergence of R_α to $T^\dagger$ on $\mathcal{D}(T)$ for $g \in \mathcal{D}(T^\dagger)$. Therefore, one must choose α carefully in order to achieve a convergent regularization and the strategy of choosing α is called a parameter choice rule.

Definition 3.5. *A parameter choice rule is a function*

$$\begin{aligned} \alpha : \mathbb{R}^+ \times \mathbf{H}_2 &\to \mathbb{R}^+ \\ (\delta, g_\delta) &\mapsto \alpha(\delta, g_\delta) \end{aligned} \tag{3.9}$$

which can be classified into three different classes:

(1) *a-priori parameter choice rules, if α depends only on δ;*

(2) *a-posteriori parameter choice rules, if α depends on both δ and g_δ;*

(3) *heuristic parameter choice rules if α depends only on g_δ.*

The following definition of regularization is motivated by finding $\alpha(\delta, g_\delta)$ such that $R_{\alpha(\delta, g_\delta)} g_\delta \to T^\dagger g$ as $\delta \to 0$.

Definition 3.6 (Convergent Regularization)**.** *A family $\{R_\alpha\}_{\alpha \in I}$ where $I \subset \mathbb{R}^+$ of continuous (not necessarily linear) operators is called a convergent regularization for $T^\dagger$ if for all $g \in \mathcal{D}(T^\dagger)$ there exists a parameter choice rule $\alpha(\delta, g_\delta)$ such that*

$$\limsup_{\delta \to 0} \left\{ \left\|R_{\alpha(\delta, g_\delta)} g_\delta - T^\dagger g\right\|_{\mathbf{H}_1} : g_\delta \in \mathbf{H}_2, \|g_\delta - g\|_{\mathbf{H}_2} \leq \delta \right\} = 0 \tag{3.10}$$

holds, and

$$\limsup_{\delta \to 0} \left\{ \alpha(\delta, g_\delta) : g_\delta \in \mathbf{H}_2, \|g_\delta - g\|_{\mathbf{H}_2} \leq \delta \right\} = 0 \tag{3.11}$$

For a specific $g \in \mathcal{D}(T^\dagger)$, the pair (R_α, α), where $\alpha : \mathbb{R}^+ \times \mathbf{H}_2 \to I$, is called a convergent regularization method of $T\varphi = g$ if (3.10) *and* (3.11) *hold.*

Therefore, in a convergent regularization method, there must exist $\left(R_{\alpha(\delta,g_\delta)}, \alpha(\delta, g_\delta)\right)$ such that for all $g_\delta \in \mathbf{H}_2$ there exists $g \in \mathcal{D}(T^\dagger)$ with $\|g_\delta - g\|_{\mathbf{H}_2} \leq \delta$ implies $R_{\alpha(\delta,g_\delta)} g_\delta \to T^\dagger g$ as $\delta \to 0$. Although the noise level and the noisy data are the only factors considered in this definition of regularization, it is important to keep in mind that in most real-world studies, we also lack access to the exact operator T. Thus, for practical experiments, the parameter choice rule would depend on $\delta, \eta, g_\delta, T_\eta$ where T_η is some approximation of the exact operator with

$$\|T - T_\eta\|_{\mathcal{L}(\mathbf{H}_1, \mathbf{H}_2)} \leq \eta \tag{3.12}$$

Listed below are three basic properties of regularization methods and details can be found on pages 44 - 46 of [**32**].

Proposition 3.1. *Let $R_\alpha : \mathbf{H}_2 \to \mathbf{H}_1$ for all $\alpha > 0$ be a family of continuous operators. Then, $\{R_\alpha\}$ is a regularization for $T^\dagger$ if $R_\alpha \to T^\dagger$ pointwise on $\mathcal{D}(T^\dagger)$ as $\alpha \to 0$. In particular, in this case, for every $g \in \mathcal{D}(T^\dagger)$ there exists an a-priori parameter choice rule α such that (R_α, α) is a convergent regularization method for $T\varphi = g$.*

Proposition 3.2. *Let $T \in \mathcal{L}(\mathbf{H}_1, \mathbf{H}_2)$ and $\{R_\alpha\}$ be a linear regularization for $T^\dagger$ and define for $\xi \in \mathbf{H}_2$*

$$\xi_\alpha := R_\alpha \xi \tag{3.13}$$

Then for $g \in \mathcal{D}(T^\dagger)$

$$g_\alpha \xrightarrow{\alpha \to 0} g^\dagger \tag{3.14}$$

where $g^\dagger := T^\dagger g$, $g_\alpha = R_\alpha g$ and for $g \notin \mathcal{D}(T^\dagger)$

$$\|g_\alpha\|_{\mathbf{H}_1} \xrightarrow{\alpha \to 0} +\infty \tag{3.15}$$

if the linear regularization operators $\{R_\alpha\}$ *satisfy the following bound*

$$\sup_{\alpha>0}\left\{\|TR_\alpha\|_{\mathbf{H}_1}\right\}<\infty \tag{3.16}$$

Proposition 3.3. *Let* $\{R_\alpha\}$ *be a linear regularization for* $T^\dagger$ *with an a-priori parameter choice rule* $\alpha=\alpha(\delta)$*. Then* (R_α,α) *is a convergent regularization method, for every* $g\in\mathcal{D}(T^\dagger)$ *if and only if*

$$\lim_{\delta\to 0}\alpha(\delta)=0 \tag{3.17}$$

and

$$\lim_{\delta\to 0}\delta\left\|R_{\alpha(\delta)}\right\|_{\mathbf{H}_1}=0 \tag{3.18}$$

Three of the best-known regularization methods constructed with these propositions in mind are the Tikhonov regularization method, the Truncated Singular Value Decomposition method, and the Lavrentiev regularization method and the examples of these can be found in [**32**, p.47-49]. We note here that the Tikhonov and Lavrentiev regularization methods are equivalent to solving certain corresponding equations, and the solutions of these equations are obtained by minimizing a corresponding functional. This method of minimizing a corresponding functional is called the 'variational regularization method'. The functional in variational regularization has the form of

$$G_\alpha(\psi):=F(T\psi,g)+\alpha J(\psi) \tag{3.19}$$

where $\xi\in\mathbf{H}_2$, $F:\mathbf{H}_2\times\mathbf{H}_2\to\mathbb{R}\cup\{+\infty\}$, and $J:\mathbf{H}_1\to\mathbb{R}\cup\{+\infty\}$. Here, F and J are functionals and $T\in\mathcal{L}(\mathbf{H}_1,\mathbf{H}_2)$ is continuous and $\alpha>0$ is a constant. The functional F is used to calculate the difference between $T\psi$ and the measured data g, whereas the functional J is a regularization term used to impose certain regularity conditions on the unknown ψ.

3. Tikhonov Regularization

As was already noted, one of the most popular variational regularization techniques for solving ill-posed inverse problems is Tikhonov regularization. Tikhonov regularization method aims to minimize the functional

$$GT_\alpha(\psi) := \|T\psi - g\|^2_{\mathbf{H}_2} + \alpha J(\psi) \tag{3.20}$$

where $T \in \mathcal{L}(\mathbf{H}_1, \mathbf{H}_2)$, $J : \mathbf{H}_1 \to \mathbb{R}^+ \cup \{+\infty\}$, and

$$F : \mathbf{H}_2 \times \mathbf{H}_2 \to \mathbb{R} \cup \{\infty\} : (T\psi, g) \mapsto \|T\psi - g\|^2_{\mathbf{H}_2}$$

Now, for a general J, we cannot expect $g_\alpha := \min_{\psi \in \mathbf{H}_1} GT(\psi) \xrightarrow{\alpha \to 0} g^\dagger$ for a given $g \in \mathbf{H}_2$. Hence, we generalize the definition of least-square solution to justify calling the minimization of (3.20) a regularization.

Definition 3.7. *An element $g^* \in \mathbf{H}_1$ is called the J-minimizing solution of $T\varphi = g$ if*

$$J(g^*) = \inf \{J(\psi) : \psi \in \mathbb{L}\}. \tag{3.21}$$

An analysis of the convergence and convergence rates of the standard Tikhonov regularization can be found in Chapter 4 of Nayak's dissertation [**32**], and the extension for the generalized Tikhonov regularization is left as an exercise to the reader.

4. Nayak's Regularization

Consider the general form of the minimizing functional in the Tikhonov regularization method shown below

$$\|T\psi - g_\delta\|^2 + \alpha \|L(\psi - \psi_0)\|^2, \tag{3.22}$$

where L is the regularization operator and $\alpha > 0$ is the regularization parameter, $\|g - g_\delta\| \leq \delta$, and ψ_0 is an initial guess. The first term is referred to as a fitting term, and it ensures the recovered solution fits the given data well. The second term is

referred to as a smoothing term or regularization term, and it provides some level of smoothness to the inverse recovery. The regularization parameter α is used to balance between smoothing and fitting in the inverse recovery. The usual process of Tikhonov regularization is as follows:

(1) Covert an ill-posed problem into a family of well-posed problems.
(2) Choose α.
(3) Choose an appropriate ψ_0.
(4) Minimize the corresponding functional to approximate the solution.

As we see here, the result of minimization depends on the regularization parameter α, and further, the function that minimizes the functional is not necessarily the minimizer of the regularization term unless it is the trivial solution.

In order to eliminate this parameter dependence, Nayak proposed using a variant of the Tikhonov functional that is parameter free, which vastly simplifies the regularization process. To see this, we first introduce the following notations which are used throughout this section:

- All functions are real-valued and defined on a closed and bounded interval $[a, b] \subset \mathbb{R}$.
- $\mathcal{L}^p[a, b] := (\mathcal{L}^p, \|\cdot\|_{\mathcal{L}^p}, [a, b])$ for $1 \leq p < \infty$ represents the usual Banach space of p-integrable functions on $[a, b]$.
- $\mathcal{L}^\infty[a, b] := (\mathcal{L}^\infty, \|\cdot\|_{\mathcal{L}^\infty}, [a, b])$ contains the essentially bounded measurable functions.
- $\mathcal{H}^q[a, b] = W^{2,q}([a, b]) := (\mathcal{H}^q, \|\cdot\|_{\mathcal{H}^q}, [a, b])$ contains all the functions which have first q weak derivatives square integrable
  ($f, f', \cdots, f^{(q)} \in \mathcal{L}^2([a, b])$, with weak differentiation).
- $\mathcal{H}_0^q[a, b]$ contains all $f \in \mathcal{H}^q([a, b])$ such that f "vanishes" at the boundary.
- $\mathcal{L}^2([a, b])$ and $\mathcal{H}^q([a, b])$ are Hilbert spaces with inner products denoted by $(\cdot, \cdot)_{\mathcal{L}^2}$ and $(\cdot, \cdot)_{\mathcal{H}^q}$.

Now, let $T : \mathcal{D}_T \to \mathcal{L}^2$ be a linear bounded operator where $\mathcal{D}_T \subset \mathcal{L}^2$, and $g \in R(T)$ be a known function defined on $[a, b] \subset \mathbb{R}$. We consider the inverse problem

$$T\varphi = g. \tag{3.23}$$

Let $u(t) = \int_t^b \int_a^\eta g(\xi)d\xi d\eta - \frac{b-t}{b-a} \int_a^b \int_a^\eta g(\xi)d\xi d\eta$ so that $g = -u''$ and $u(a) = 0$ and $u(b) = 0$. Thus, (3.23) can be reformulated as: find φ that satisfies

$$T\varphi = -u''. \tag{3.24}$$

Further, for a given $\psi \in \mathcal{L}^2(\Omega)$, u_ψ is the solution of the boundary value problem below:

$$\begin{aligned} -u''_\psi &= T\psi, \\ u_\psi(a) &= u(a) = 0, \\ u_\psi(b) &= u(b) = 0. \end{aligned} \tag{3.25}$$

Then the solution of the inverse problem $T\varphi = g$ is approximated by the minimizer of the functional

$$G(\psi) = \|T\psi - g\|^2_{\mathcal{L}^2} + \left\|u'_\psi - u'\right\|^2_{\mathcal{L}^2}, \tag{3.26}$$

where $L\psi = u'_\psi$, and $L\psi_0 = u'$, using the Tikhonov operator L from (3.22).

In other words, to find the solution of $T\varphi = g$, we find

$$\varphi = \arg\min_\psi \left\{ \|T\psi - g\|^2_{\mathcal{L}^2} + \|L(\psi - \psi_0)\|^2_{\mathcal{L}^2} \right\}$$

using the following generic descent algorithm:

(1) Initialize $\psi_0 = \mathbf{0}$.

(2) Find the gradient.

(3) Update ψ_0.

(4) Continue until it meets the terminating criteria.

Remark 3.4. Nayak's functional (3.26), unlike other regularization functionals, is convex and offers a unique global minimum when the bounded linear operator T is injective. In addition to previous regularization functionals, Nayak's regularization operator L is related to the operator T without the involvement of any external parameters, hence the minimization is independent of the regularization parameter. Moreover, unlike other regularization techniques, the function that minimizes the functional term also minimizes the regularization term, giving us an advantage in settings with a high level of noise. Furthermore, we use the Neuberger gradient rather than the $\mathcal{L}^2$ gradient to improve the descent rate and efficiency of recovery, as well as to offer flexibility at the boundaries.

The following are important properties of the functional G and the proofs of these can be found in Nayak's dissertation [**32**, p.125-127].

1. G is convex.

2. The first Gâteaux derivative at $\psi \in \mathcal{L}^2[a,b]$ in $h \in \mathcal{L}^2[a,b]$ direction is given by

$$G'(\psi)[h] = -2\int_a^b (Th)(u - u_\psi + g - T\psi)(x)dx. \tag{3.27}$$

3. $\mathcal{L}^2$-gradient of G at ψ is given by

$$\nabla^{\psi}_{\mathcal{L}^2} G = -2T^*(u - u_\psi + g - T\psi). \tag{3.28}$$

4. The second Gâteaux derivative at $\psi \in \mathcal{L}^2[a,b]$ with $h,k \in \mathcal{L}^2[a,b]$ is given by

$$G''(\psi)[h,k] = -2\left\langle \Delta^{-1}(Th) + Th, Tk \right\rangle_{\mathcal{L}^2}. \tag{3.29}$$

5. For fixed $\psi, h \in \mathcal{L}^2[a,b]$, in $\mathcal{H}^1([a,b])$,

$$\lim_{\epsilon \to 0} u_{\psi+\epsilon h} = u_\psi. \tag{3.30}$$

6. $-\Delta$ is a positive operator in $\mathcal{L}^2[a,b]$.

7. G is bounded

$$\left(1+\frac{1}{\lambda_1}\right)^{-1}\|u-u_\psi\|^2_{\mathcal{H}^2} \le G(\psi) \le \|u-u_\psi\|^2_{\mathcal{H}^2}\,, \tag{3.31}$$

where $\lambda_1 > 0$ is the smallest eigenvalue of the positive operator $-\Delta$.

8. The Neuberger gradient is found by solving a boundary value problem. Let $y = \nabla^{\psi}_{\mathcal{H}^1} G$. Then the Neuberger gradient is the solution of the following boundary value problem

$$\begin{aligned} -y'' + y &= \nabla^{\psi}_{\mathcal{L}^2} G, \\ y'y|^b_a &= 0. \end{aligned} \tag{3.32}$$

One can choose from three different boundary conditions: Dirichlet Neuberger gradient, Neumann Neuberger gradient, or mixed Neuberger gradient.

Further, Nayak has shown that for the sequence $\{\psi_m\}$ found using the descent algorithm, the corresponding sequence $\{G(\psi_m)\}$ converges to zero [**32**, p.87,126,131-132]. Conversely, we have from [**32**, p.132] which describes the convergence of the sequence $\{\psi_m\}$ to φ in $\mathcal{L}^2(\Omega)$.

Theorem 3.5. *Suppose that $\{\psi_m\}$ is any sequence of $\mathcal{L}^2$-functions such that the sequence $\{G(\psi_m)\}$ tends to zero. Then $\{\psi_m\}$ converges weakly to φ in $\mathcal{L}^2([a,b])$ and $\{u_{\psi_m}\}$ converges strongly to u in $\mathcal{H}^2([a,b])$. Moreover, the sequence $\{g_m := T\psi_m\}$ converges strongly to g in $\mathcal{L}^2([a,b])$.*

Remark 3.6. In the preceding, we made the assumption that g is in the range of T. In this dissertation, we do not cover the case where g is not in the range of T, which would need further studies using the Moore-Penrose inverses.

CHAPTER 4

Stability

In this chapter, we prove in two parts our modified version of the stability theorems [**32**, p.133] that Nayak has proposed: first for $\hat{G}(\psi) = \left\|u'_\psi - u'\right\|^2_{\mathcal{L}^2}$ and secondly for $G(\psi) = \|T\psi - g\|^2_{\mathcal{L}^2} + \left\|u'_\psi - u'\right\|^2_{\mathcal{L}^2}$. Throughout this section, $\Omega = [a, b] \subset \mathbb{R}$, $\mathcal{H}^q(\Omega)$ contains all the functions which have first q weak derivatives square integrable, and $\mathcal{H}^1_0(\Omega)$ contains all $f \in \mathcal{H}^q(\Omega)$ such that f "vanishes" at the boundary.

1. The General Setup

Let $\{\psi_m\} \subset \mathcal{L}^2(\Omega)$. For the original u, we have that u_{ψ_m} is the solution of the following boundary problem

$$\begin{aligned} T\psi_m &= -u''_{\psi_m} \\ u_{\psi_m}(a) &= u_{\psi_m}(b) = 0. \end{aligned} \tag{4.1}$$

Hence, $u'_{\psi_m}(t)$ can be found by integrating both sides as such:

$$u'_{\psi_m}(t) = u'_{\psi_m}(a) - \int_a^t T\psi_m(\xi)d\xi. \tag{4.2}$$

Now to solve for $u'_{\psi_m}(a)$, we first integrate both sides again.

$$u_{\psi_m}(t) - u_{\psi_m}(a) = (t-a)u'_{\psi_m}(a) - \int_a^t \int_a^\eta T\psi_m(\xi)d\xi d\eta. \tag{4.3}$$

Then, plugging in $t = b$ and using the fact that $u_{\psi_m}(a) = u_{\psi_m}(b) = 0$, we get

$$u'_{\psi_m}(a) = \frac{1}{b-a}\int_a^b \int_a^\eta T\psi_m(\xi)d\xi d\eta. \tag{4.4}$$

Hence, we see that

$$u_{\psi_m}(t) = \frac{t-a}{b-a}\int_a^b \int_a^\eta T\psi_m(\xi)d\xi d\eta - \int_a^t \int_a^\eta T\psi_m(\xi)d\xi d\eta.$$

Let $\tilde{g} \in \mathcal{R}(T)$ be perturbed data with $\|g - \tilde{g}\|_{\mathcal{L}^2} < \delta$ for all sufficiently small $\delta > 0$. Following (3.24) we have that $T\tilde{\varphi} = -\tilde{u}'' = \tilde{g}$ where $\tilde{u}(t) = \int_t^b \int_a^\eta \tilde{g}(\xi)d\xi d\eta - \frac{b-t}{b-a}\int_a^b \int_a^\eta \tilde{g}(\xi)d\xi d\eta$ and $\tilde{u}_{\psi_m}$ is the solution of the following boundary problem

$$\begin{aligned} T\psi_m &= -\tilde{u}''_{\psi_m}, \\ \tilde{u}_{\psi_m}(a) &= \tilde{u}(a) = 0, \\ \tilde{u}_{\psi_m}(b) &= \tilde{u}(b) = 0. \end{aligned} \tag{4.5}$$

We notice here that for each ψ_m, $-u''_{\psi_m} = -\tilde{u}''_{\psi_m} = T\psi_m$ and following a similar process to the above, $\tilde{u}'_{\psi_m} = u'_{\psi_m}$ and $\tilde{u}_{\psi_m} = u_{\psi_m}$.

2. Necessary Components to Prove Stability

In this subsection, we prove a theorem and a lemma needed to show the stability of Nayak's method. Let $\hat{G}(\psi) = \left\|u'_\psi - u'\right\|^2_{\mathcal{L}^2}$ and $\hat{G}_{\tilde{u}}(\psi) = \left\|\tilde{u}'_\psi - \tilde{u}'\right\|^2_{\mathcal{L}^2} = \left\|u'_{\psi_m} - \tilde{u}'\right\|^2_{\mathcal{L}^2}$. Note that the following theorem cannot be found in Nayak's dissertation.

Theorem 4.1. *Let $g, \tilde{g} \in \mathcal{H}^1(\Omega)$ such that $\|g - \tilde{g}\|_{\mathcal{L}^2} < \delta$ for $0 < \delta \leq 1$. Then, for some constant C, the corresponding $u, \tilde{u} \in \mathcal{H}^3_0(\Omega)$ satisfies*

$$\|u - \tilde{u}\|_{\mathcal{H}^1} < C\delta^{1/2}. \tag{4.6}$$

Proof. Recall that

$$u(t) = \int_t^b \int_a^\eta g(\xi)d\xi d\eta - \frac{b-t}{b-a}\int_a^b \int_a^\eta g(\xi)d\xi d\eta,$$

$$u'(t) = -\int_a^t g(\eta)d\eta + \frac{1}{b-a}\int_a^b \int_a^\eta g(\xi)d\xi d\eta,$$

$$\tilde{u}(t) = \int_t^b \int_a^\eta \tilde{g}(\xi)d\xi d\eta - \frac{b-t}{b-a}\int_a^b \int_a^\eta \tilde{g}(\xi)d\xi d\eta,$$

$$\tilde{u}'(t) = -\int_a^t \tilde{g}(\eta)d\eta + \frac{1}{b-a}\int_a^b \int_a^\eta \tilde{g}(\xi)d\xi d\eta.$$

Then, by Minkowski's inequality, we get

$$\begin{aligned}
\|u-\tilde{u}\|_{\mathcal{H}^1}^2 &= \|u-\tilde{u}\|_{\mathcal{L}^2}^2 + \|u'-\tilde{u}'\|_{\mathcal{L}^2}^2 \\
&= \left\| \int_t^b \int_a^\eta g(\xi) - \tilde{g}(\xi)d\xi d\eta + \frac{b-t}{b-a}\int_a^b \int_a^\eta \tilde{g}(\xi) - g(\xi)d\xi d\eta \right\|_{\mathcal{L}^2}^2 \\
&\qquad + \left\| \int_a^t \tilde{g}(\eta) - g(\eta)d\eta + \frac{1}{b-a}\int_a^b \int_a^\eta g(\xi) - \tilde{g}(\xi)d\xi d\eta \right\|_{\mathcal{L}^2}^2 \\
&\leq \left(\left\| \int_t^b \int_a^\eta |g(\xi) - \tilde{g}(\xi)|\, d\xi d\eta \right\|_{\mathcal{L}^2} + \left\| \frac{b-t}{b-a}\int_a^b \int_a^\eta |\tilde{g}(\xi) - g(\xi)|\, d\xi d\eta \right\|_{\mathcal{L}^2} \right)^2 \\
&\quad + \left(\left\| \int_a^t |\tilde{g}(\eta) - g(\eta)|\, d\eta \right\|_{\mathcal{L}^2} + \left\| \frac{1}{b-a}\int_a^b \int_a^\eta |g(\xi) - \tilde{g}(\xi)|\, d\xi d\eta \right\|_{\mathcal{L}^2} \right)^2 .
\end{aligned}$$

We now consider each term separately.

$$\begin{aligned}
\left\| \frac{1}{b-a}\int_t^b \int_a^\eta |g(\xi) - \tilde{g}(\xi)|\, d\xi d\eta \right\|_{\mathcal{L}^2} &= \sqrt{\int_a^b \left| \frac{1}{b-a}\int_t^b \int_a^\eta |g(\xi) - \tilde{g}(\xi)|\, d\xi d\eta \right|^2 dt} \\
&= \frac{1}{b-a}\sqrt{\int_a^b \left(\int_t^b \int_a^\eta |g(\xi) - \tilde{g}(\xi)|\, d\xi d\eta \right)^2 dt} \\
&\quad \text{by applying Hölder, we get} \\
&\leq \frac{1}{b-a}\sqrt{\int_a^b \left(\int_t^b \sqrt{b-a}\sqrt{\int_a^\eta |g(\xi) - \tilde{g}(\xi)|^2\, d\xi d\eta} \right)^2 dt} \\
&< \frac{1}{b-a}\sqrt{\delta^2(b-a)\int_a^b \left(\int_t^b d\eta \right)^2 dt} \\
&\leq \delta(b-a).
\end{aligned}$$

Using similar strategies, we get the following results and leave the detailed algebra for the readers.

$$\begin{aligned}
\left\| \frac{b-t}{b-a} \int_a^b \int_a^\eta \tilde{g}(\xi) - g(\xi) d\xi d\eta \right\|_{\mathcal{L}^2} &< \delta \frac{(b-a)^2}{\sqrt{3}} \\
\left\| \int_a^t \tilde{g}(\eta) - g(\eta) d\eta \right\|_{\mathcal{L}^2} &< \delta^{1/2} (b-a) \qquad (4.7) \\
\left\| \frac{1}{b-a} \int_a^b \int_a^\eta g(\xi) - \tilde{g}(\xi) d\xi d\eta \right\|_{\mathcal{L}^2} &< \delta (b-a).
\end{aligned}$$

Hence,

$$\begin{aligned}
\|u - \tilde{u}\|_{\mathcal{H}^1}^2 &< \left(\delta (b-a)^2 + \delta \frac{(b-a)^2}{\sqrt{3}} \right)^2 + \left(\delta^{1/2} (b-a)^{3/4} + \delta (b-a) \right)^2 \\
&\leq \delta^2 \left((b-a)^2 + \frac{(b-a)^2}{\sqrt{3}} \right)^2 + \delta (b-a)^{7/2} \qquad (4.8) \\
&\leq \delta \left((b-a)^{7/2} + \left((b-a)^2 + \frac{(b-a)^2}{\sqrt{3}} \right)^2 \right).
\end{aligned}$$

Therefore, we see that $\|u - \tilde{u}\|_{\mathcal{H}^1} < C\delta^{1/2}$ for some constant C. □

In the following lemma, we show that $G_{\tilde{u}}$ is bounded below.

Lemma 4.2. *$G_{\tilde{u}}(\psi)$ is bounded below for all $\psi \in \mathcal{L}^2(\Omega)$.*

Proof. Let $v = \tilde{u} - \tilde{u}_\psi$ and observe that $v \in W_0^{2,2}(\Omega)$. Further, let $\{a_i\}$ be an orthonormal basis of eigenfunctions in $\mathcal{L}^2(\Omega)$ for the positive operator $-\Delta$ corresponding to the eigenvalues $\{\lambda_i\}$ where $\lambda_1 \leq \lambda_2 \leq \cdots$ which are bounded below by the positive constant λ_1. Then,

$$
\begin{aligned}
\hat{G}_{\tilde{u}}(\psi) &= \left\|\tilde{u}'_{\psi} - \tilde{u}'\right\|^2_{\mathcal{L}^2} \\
&= \|\nabla v\|^2_{\mathcal{L}^2} \\
&= \langle \nabla v, \nabla v \rangle_{\mathcal{L}^2} \\
&= \langle -\Delta v, v \rangle_{\mathcal{L}^2} \\
&= \left\langle -\Delta \sum c_i a_i, v \right\rangle_{\mathcal{L}^2} \\
&= \left\langle \sum c_i \lambda_i a_i, v \right\rangle_{\mathcal{L}^2} \\
&\geq \lambda_1 \langle v, v \rangle_{\mathcal{L}^2} .
\end{aligned} \tag{4.9}
$$

Therefore, we see that $\hat{G}_{\tilde{u}}(\psi) = \|\nabla v\|^2_{\mathcal{L}^2} \geq \lambda_1 \|v\|^2_{\mathcal{L}^2}$. Thus,

$$
\|\nabla v\|^2_{\mathcal{L}^2} + \lambda_1 \|\nabla v\|^2_{\mathcal{L}^2} \geq \lambda_1 \|v\|^2_{\mathcal{L}^2} + \lambda_1 \|\nabla v\|^2_{\mathcal{L}^2} = \lambda_1 \|v\|^2_{\mathcal{H}^1} ,
$$

or,

$$
(1 + \lambda_1) \|\nabla v\|^2_{\mathcal{L}^2} \geq \lambda_1 \|v\|^2_{\mathcal{H}^1} .
$$

So that $\hat{G}_{\tilde{u}}(\psi) = \|\nabla v\|^2_{\mathcal{L}^2} \geq \dfrac{\lambda_1}{1 + \lambda_1} \|v\|^2_{\mathcal{H}^1}$. □

3. Proof of Stability Part I

Our primary goal for this section is to prove the stability of Nayak's method using $\hat{G}(\psi) = \left\|u'_{\psi} - u'\right\|^2_{\mathcal{L}^2}$. In the following theorem, we show that if a sequence $\{\psi_m\}$ converges weakly to $\tilde{\varphi}$ in $\mathcal{L}^2(\Omega)$ then it also approximates φ weakly in $\mathcal{L}^2$. In other words, if $\hat{G}_{\tilde{u}}(\psi_m)$ is small then $\hat{G}_u(\psi_m)$ is also small.

Theorem 4.3. *[Stability] Suppose for a sequence of functions $\{\psi_m\} \subset \mathcal{L}^2(\Omega)$, the corresponding sequence $\left\{\hat{G}_{\tilde{u}}(\psi_m)\right\} \subset \mathbb{R}$ where $\hat{G}_{\tilde{u}}(\psi_m) = \left\|\tilde{u}'_{\psi_m} - \tilde{u}'\right\|^2_{\mathcal{L}^2}$ converges to zero, where the functional $\hat{G}_{\tilde{u}}$ is formed based on $\tilde{u}$ and $\Omega = [a, b] \subset \mathbb{R}$. Then a u such that $\|u - \tilde{u}\|_{\mathcal{H}^1} < \delta$ for $0 < \delta \leq 1$, there exists a $M(\delta) \in \mathbb{N}$ sch that for all $m \geq M(\delta)$, $\hat{G}_u(\psi_m) < \delta$ where $\hat{G}_u$ is the functional based on u.*

PROOF.

$$\begin{aligned}\sqrt{\hat{G}_u(\psi_m)} &= \left\|u'_{\psi_m} - u'\right\|_{\mathcal{L}^2} \\ &\leq \left\|u'_{\psi_m} - \tilde{u}'\right\|_{\mathcal{L}^2} + \|\tilde{u}' - u'\|_{\mathcal{L}^2} \\ &= \left\|\tilde{u}'_{\psi_m} - \tilde{u}'\right\|_{\mathcal{L}^2} + \|\tilde{u}' - u'\|_{\mathcal{L}^2} \text{ since } \tilde{u}'_{\psi_m} = u'_{\psi_m}.\end{aligned}$$

By hypothesis, since $\hat{G}_{\tilde{u}}(\psi_m) \to 0$, we see that $\left\|\tilde{u}'_{\psi_m} - \tilde{u}'\right\|_{\mathcal{L}^2} \to 0$ as $m \to \infty$. Further, since $\|u - \tilde{u}\|_{\mathcal{H}^1} < \delta$, we see that $\|\tilde{u}' - u'\|_{\mathcal{L}^2} < \delta$. □

Theorem 4.4. *Suppose the function $\tilde{u}$ is a perturbation of the function u and $\|u - \tilde{u}\|_{\mathcal{H}^1} < \delta$ where $0 < \delta \leq 1$. Let $\varphi, \tilde{\varphi} \in \mathcal{L}^2(\Omega)$ satisfy $T\varphi = -u''$ and $T\tilde{\varphi} = -\tilde{u}''$ and let $\hat{G}_u(\varphi) = 0$. Then,*

$$0 \leq \hat{G}_{\tilde{u}}(\varphi) < \delta. \tag{4.10}$$

PROOF.

$$\begin{aligned}\hat{G}_{\tilde{u}}(\varphi) &= \left\|\tilde{u}'_{\varphi} - \tilde{u}'\right\|^2_{\mathcal{L}^2}, \\ &= \left\|u'_{\varphi} - \tilde{u}'\right\|^2_{\mathcal{L}^2} \text{ since } \tilde{u}'_{\varphi} = u'_{\varphi}, \\ &= \|u' - \tilde{u}'\|^2_{\mathcal{L}^2}.\end{aligned}$$

Hence, $\hat{G}_{\tilde{u}}(\varphi) < \delta$. □

4. Proof of Stability Part II

We now show the stability of Nayak's method for

$$G(\psi) = \|T\psi - g\|^2_{\mathcal{L}^2} + \left\|u'_{\psi} - u'\right\|^2_{\mathcal{L}^2}.$$

Let $\tilde{g}$ be the perturbed g so that $\|g - \tilde{g}\|_{\mathcal{L}^2} < \delta$ for a $0 < \delta \leq 1$. Then, the corresponding functional is

$$G_{\tilde{u}}(\psi) = \|T\psi - \tilde{g}\|^2_{\mathcal{L}^2} + \left\|\tilde{u}'_{\psi} - \tilde{u}'\right\|^2_{\mathcal{L}^2},$$

where $-\tilde{u}'' = \tilde{g}$ with $\tilde{u}(a) = \tilde{u}(b) = 0$ as shown above.

We now prove following two theorems to show the stability of Nayak's method when $T : \mathcal{L}^2(\Omega) \to \mathcal{L}^2(\Omega)$.

Theorem 4.5. *Let g be a given data and $\tilde{g}$ be a perturbed data such that $\|g - \tilde{g}\|_{\mathcal{L}^2} < \delta$ for $0 < \delta \leq 1$. Let φ be the minimizer of G such that $G(\varphi) = 0$. Then,*

$$G_{\tilde{u}}(\varphi) < C\delta, \tag{4.11}$$

for some constant C.

PROOF.

$$\begin{aligned} G_{\tilde{u}}(\varphi) &= \|T\varphi - \tilde{g}\|^2_{\mathcal{L}^2} + \left\|\tilde{u}'_\varphi - \tilde{u}'\right\|^2_{\mathcal{L}^2} \\ &= \|g - \tilde{g}\|^2_{\mathcal{L}^2} + \left\|\tilde{u}'_\varphi - \tilde{u}'\right\|^2_{\mathcal{L}^2} \\ &< \delta^2 + \left\|\tilde{u}'_\varphi - \tilde{u}'\right\|^2_{\mathcal{L}^2} \qquad \text{since } \|g - \tilde{g}\|_{\mathcal{L}^2} \leq \delta \\ &< \delta^2 + c^2\delta \text{ for some constant } c \text{ by Theorem 4.1} \\ &\leq \delta(1 + c^2). \end{aligned}$$

□

Theorem 4.6. *Let g be given data and $\tilde{g}$ be perturbed data such that $\|g - \tilde{g}\|_{\mathcal{L}^2} < \delta$ for $0 < \delta \leq 1$. Further, let $\{\psi_m\} \subset \mathcal{L}^2(\Omega)$ such that $G_{\tilde{u}}(\psi_m) \to 0$. Then, there exists a positive number $M(\delta)$ such that for all $m \geq M(\delta)$, $G(\psi_m) < C\delta$ for some constant C.*

PROOF. Notice that $G_{\tilde{u}}(\psi_m) \to 0$ implies that $T\psi_m$ converges to $\tilde{g}$ strongly in $\mathcal{L}^2$. Hence, for large m, we see that

$$\|T\psi_m - g\|^2_{\mathcal{L}^2} \leq \left(\|T\psi_m - \tilde{g}\|_{\mathcal{L}^2} + \|g - \tilde{g}\|_{\mathcal{L}^2}\right)^2 < \delta^2.$$

Therefore,

$$
\begin{aligned}
G(\psi_m) &= \|T\psi - g\|_{\mathcal{L}^2}^2 + \left\|u'_\psi - u'\right\|_{\mathcal{L}^2}^2 \\
&< \delta^2 + \left\|u'_\psi - u'\right\|_{\mathcal{L}^2}^2 \\
&< \delta^2 + c^2\delta \text{ by Theorem 4.1 for some constant } c \\
&\leq \delta(1 + c^2).
\end{aligned} \tag{4.12}
$$

Hence, we see that $G(\psi_m) < C\delta$ for some constant C. □

Extension of the stability proofs to our problem can be found in Chapter 5.

CHAPTER 5

A New Algorithm for Stock Prediction

Our primary objective is to identify better coefficients for the system (1.1) in order to increase the accuracy of Apple's stock trend and price predictions. To better predict Apple's stock trend, we use a new strategy for generating a starting point for the descent process, as well as a way for compensating for inherent flaws in the initial model (1.1) where it has exponential growth model errors that cause some of the solutions to be unrealistic. Furthermore, we apply a descent method inspired by Nayak to obtain non-constant coefficients using a better starting point. Additionally, where information on volatility is available, we incorporate a new iterative improvement technique to more accurately estimate the stock prices. To assess our success, we compare our results to the LSTM model and Barnett's results.

1. Notation

We use the following notations throughout this chapter. The value of the variable v increments by one on the first day of each month starting on July 1, 1994 where $v = 1$ represents July 1, 1994. The domain of our data is $\mathcal{T} = \{\tau : 1 \leq \tau \leq 10479\}$, where $\tau = 1$ represents July 1, 1994, and $\tau = 10479$ represents February 28, 2023. Let $[a, b] \subset \mathcal{T}$. For $1 \leq p < \infty$, $\mathcal{L}^p[a, b]$ denotes the Banach space of p-integrable functions on $[a, b]$ and $\mathcal{L}^\infty[a, b]$ represents the essentially bounded measurable functions. We denote the Sobolev space which contains all functions which have square integrable weak derivatives up to order q on $[a, b]$ as $\mathcal{H}^q[a, b]$. Further, the space $\mathcal{H}_0^q[a, b]$ contains all functions in $\mathcal{H}^q$ that vanish at the boundary.

2. Our Data

Most research on stock price forecasting predicts closing prices rather than opening prices. Since Barnett uses opening prices, we re-gather our data so that we can contrast our findings with those of other studies. Notice here that we do not consider stock splits, whereas Barnett is factoring stock splits into account for Apple's stock prices. Further, using the same sources as in Section 1 of Chapter 1, we collect data from 2018 to February 2023 to broaden the comparison period.

As mentioned in Chapter 1, the data is linearly interpolated down to the daily level, so that $x_1, \cdots, x_{13} \in C^0$. To estimate derivative terms such as x_1', x_{10}', and x_{13}' appearing in certain differential equations, we use MATLAB's **gradient** function [**2**], so that for $A \in \mathbb{R}^n$,

$$\textbf{gradient}(A)_i = \begin{cases} \dfrac{1}{2}(A_{i+1} - A_{i-1}) & \text{for } 1 < i < n \\ A_2 - A_1 & \text{for } i = 1 \\ A_n - A_{n-1} & \text{for } i = n \end{cases}$$

More precisely, $\textbf{gradient}(A)$ is calculated by using one sided differences at the ends and central differences elsewhere. Notice there is no need for explicit division by the time step since the time step is one (day) in our data. The **pchip** function in MATLAB can be used to interpolate data so that all functions are in C^1. However, because stock price interpolation need not be differentiable everywhere, we use linear interpolation and estimate the derivatives.

Recall that the data availability varies. GLD information is only available after December 1, 2004, and iPhone revenue information is only accessible as of October 1, 2007. Starting January 1, 2013, information about Apple's buybacks is available; however, buybacks are only included in the PE ratio equation from January 1, 2013 to June 1, 2014. Beginning on June 1, 2014, all data should be available. Recalling how v is defined in section 1, we divide our timeline into the following five sections:

- $v < 126$; Before December 1, 2004, there was no information on GLD, the iPhone, or buybacks.
- $126 \leq v < 160$; Between December 1, 2004 and September 30, 2007, there was no information on the iPhone or buybacks.
- $160 \leq v < 223$; Between October 1, 2007, and December 31, 2012, there was no information about buybacks.
- $223 \leq v < 240$; Between January 1, 2013 and May 31, 2014, information about buybacks was not included in Apple's stock equation.
- $v \geq 240$; Starting June 1, 2014, all the information was available.

3. Set Up

Recall Barnett's system of delay differential equation:

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) + c_8 x_{13}(t) \\
x_4'(t) &= c_9 x_1'(t) + c_{10} x_4(t) + c_{11} x_2(t) x_4(t-20) + c_{12} x_8(t) x_4(t) \\
x_5'(t) &= c_{13} x_8(t) + c_{14} x_{10}(t) + c_{15} x_4(t) + c_{16} x_{12}(t) \\
x_6'(t) &= c_{17} x_6(t) + c_{18} x_7(t-17) + c_{19} x_{11}(t) + c_{20} x_{12}(t) \\
x_7'(t) &= c_{21} x_7(t) + c_{22} x_6(t) + c_{23} x_{11}(t) \\
x_8'(t) &= c_{24} x_6(t) + c_{25} x_5'(t) + c_{26} x_8(t) \\
&\quad + c_{27} x_8(t) x_4(t) + c_{28} x_3(t) + c_{29} x_{10}'(t) + c_{30} x_{13}'(t) \\
x_9'(t) &= c_{31} x_{10}(t) + c_{32} x_{12}(t) + c_{33} x_2(t) + c_{34} x_9(t) \\
x_{10}'(t) &= c_{35} x_{10}(t) + c_{36} x_4(t) + c_{37} x_1(t) + c_{38} x_5(t) + c_{39} x_{12}(t) \\
x_{11}'(t) &= c_{40} x_7(t-17) + c_{41} x_{11}(t) + c_{42} x_4(t-20) x_{11}(t) \\
x_{12}'(t) &= c_{43} x_{10}'(t) + c_{44} x_9(t) + c_{45} x_4(t) + c_{46} x_1(t) \\
x_{13}'(t) &= c_{47} x_{10}(t) + c_{48} x_6(t) + c_{49} x_7(t)
\end{aligned}
\tag{5.1}
$$

By integrating both sides on $[a,b] \subset \mathcal{T}$ we get the following:

$$
\begin{bmatrix}
\int_a^t c_1 x_2(\xi) x_1(\xi) d\xi \\
\int_a^t c_2 x_1(\xi) + c_3 x_4(\xi - 20) x_2(\xi) + c_4 x_2(\xi) d\xi \\
\int_a^t c_5 x_2(\xi) + c_6 x_6(\xi) + c_7 x_8(\xi) + c_8 x_{13}(\xi) d\xi \\
\int_a^t c_9 x_1'(\xi) + c_{10} x_4(\xi) + c_{11} x_2(\xi) x_4(\xi - 20) + c_{12} x_8(\xi) x_4(\xi) d\xi \\
\int_a^t c_{13} x_8(\xi) + c_{14} x_{10}(\xi) + c_{15} x_4(\xi) + c_{16} x_{12}(\xi) d\xi \\
\int_a^t c_{17} x_6(\xi) + c_{18} x_7(\xi - 17) + c_{19} x_{11}(\xi) + c_{20} x_{12}(\xi) d\xi \\
\int_a^t c_{21} x_7(\xi) + c_{22} x_6(\xi) + c_{23} x_{11}(\xi) d\xi \\
\int_a^t c_{24} x_6(\xi) + c_{25} x_5'(\xi) + c_{26} x_8(\xi) + c_{27} x_8(\xi) x_4(\xi) \\
+ c_{28} x_3(\xi) + c_{29} x_{10}'(\xi) + c_{30} x_{13}'(\xi) d\xi \\
\int_a^t c_{31} x_{10}(\xi) + c_{32} x_{12}(\xi) + c_{33} x_2(\xi) + c_{34} x_9(\xi) d\xi \\
\int_a^t c_{35} x_{10}(\xi) + c_{36} x_4(\xi) + c_{37} x_1(\xi) + c_{38} x_5(\xi) + c_{39} x_{12}(\xi) d\xi \\
\int_a^t c_{40} x_7(\xi - 17) + c_{41} x_{11}(\xi) + c_{42} x_4(\xi - 20) x_{11}(\xi) d\xi \\
\int_a^t c_{43} x_{10}'(\xi) + c_{44} x_9(\xi) + c_{45} x_4(\xi) + c_{46} x_1(\xi) d\xi \\
\int_a^t c_{47} x_{10}(\xi) + c_{48} x_6(\xi) + c_{49} x_7(\xi) d\xi
\end{bmatrix}
= \begin{bmatrix} x_1(t) - x_1(a) \\ \vdots \\ x_{13}(t) - x_{13}(a) \end{bmatrix} \tag{5.2}
$$

We now define $T : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{13}$ as below:

$$
(T\varphi)(t) = T\left(\begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_{49} \end{bmatrix}\right)(t) = \begin{bmatrix}
\int_a^t \varphi_1 x_2(\xi) x_1(\xi) d\xi \\
\int_a^t \varphi_2 x_1(\xi) + \varphi_3 x_4(\xi - 20) x_2(\xi) + \varphi_4 x_2(\xi) d\xi \\
\int_a^t \varphi_5 x_2(\xi) + \varphi_6 x_6(\xi) + \varphi_7 x_8(\xi) + \varphi_8 x_{13}(\xi) d\xi \\
\int_a^t \varphi_9 x_1'(\xi) + \varphi_{10} x_4(\xi) + \varphi_{11} x_2(\xi) x_4(\xi - 20) + \varphi_{12} x_8(\xi) x_4(\xi) d\xi \\
\int_a^t \varphi_{13} x_8(\xi) + \varphi_{14} x_{10}(\xi) + \varphi_{15} x_4(\xi) + \varphi_{16} x_{12}(\xi) d\xi \\
\int_a^t \varphi_{17} x_6(\xi) + \varphi_{18} x_7(\xi - 17) + \varphi_{19} x_{11}(\xi) + \varphi_{20} x_{12}(\xi) d\xi \\
\int_a^t \varphi_{21} x_7(\xi) + \varphi_{22} x_6(\xi) + \varphi_{23} x_{11}(\xi) d\xi \\
\int_a^t \varphi_{24} x_6(\xi) + \varphi_{25} x_5'(\xi) + \varphi_{26} x_8(\xi) + \varphi_{27} x_8(\xi) x_4(\xi) \\
+ \varphi_{28} x_3(\xi) + \varphi_{29} x_{10}'(\xi) + \varphi_{30} x_{13}'(\xi) d\xi \\
\int_a^t \varphi_{31} x_{10}(\xi) + \varphi_{32} x_{12}(\xi) + \varphi_{33} x_2(\xi) + \varphi_{34} x_9(\xi) d\xi \\
\int_a^t \varphi_{35} x_{10}(\xi) + \varphi_{36} x_4(\xi) + \varphi_{37} x_1(\xi) + \varphi_{38} x_5(\xi) + \varphi_{39} x_{12}(\xi) d\xi \\
\int_a^t \varphi_{40} x_7(\xi - 17) + \varphi_{41} x_{11}(\xi) + \varphi_{42} x_4(\xi - 20) x_{11}(\xi) d\xi \\
\int_a^t \varphi_{43} x_{10}'(\xi) + \varphi_{44} x_9(\xi) + \varphi_{45} x_4(\xi) + \varphi_{46} x_1(\xi) d\xi \\
\int_a^t \varphi_{47} x_{10}(\xi) + \varphi_{48} x_6(\xi) + \varphi_{49} x_7(\xi) d\xi
\end{bmatrix} \tag{5.3}
$$

By letting $g : [a,b] \to (\mathcal{L}^2[a,b])^{13} : t \mapsto \begin{bmatrix} x_1(t) - x_1(a) \\ \vdots \\ x_{13}(t) - x_{13}(a) \end{bmatrix}$, we can see that (5.2) can now be written as $T\mathbf{c} = g$ where $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_{49} \end{bmatrix}^T$.

4. Investigating T

In this section, we demonstrate that our $T : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{13}$ is compact and that $\dim(\mathcal{R}(T)) = \infty$. As a result of Theorem 3.3, this shows that our inverse problem $T\mathbf{c} = g$ is ill-posed. To prove the compactness of our T, we use Theorem 5.1 and Theorem 5.2 listed below, the proof of Theorem 5.2 can be found in [**19**, p.168].

Definition 5.1 (Hilbert-Schmidt Operators)**.** *Let $A : \mathbf{H}_1 \to \mathbf{H}_2$ be a operator on separable Hilbert spaces $\mathbf{H}_1$ and $\mathbf{H}_2$. A is a Hilbert-Schmidt operator if*

$$\sum_{k=1}^{\infty} \|Ae_k\|_{\mathbf{H}_2}^2 < \infty \tag{5.4}$$

for some (and hence, for all) $\{e_k\}$ orthonormal basis for $\mathbf{H}_1$. [**23**]

We also use the following two theorems.

Theorem 5.1. *On a separable Hilbert space every Hilbert-Schmidt operator is compact* [**23**].

Theorem 5.2 (Hilbert-Schmidt)**.** *Let X be a locally compact space endowed with a positive Borel measure, and assume that $\mathcal{L}^2(X)$ is a separable Hilbert space. Let $K \in \mathcal{L}^2(X \times X)$. Then the operator*

$$(Bf)(x) = \int_X K(x,y) f(y) dy \tag{5.5}$$

is a compact operator on $\mathcal{L}^2(X)$. Such a square-integrable kernel K is called a Hilbert-Schmidt kernel, and such an integral operator is called a Hilbert-Schmidt operator [**19**].

We now note that $(\mathcal{L}^2[a,b])^n$ is a separable Hilbert spaces for all $n \in \mathbb{N}$ and $[a,b]$ is a locally compact space endowed with Lebesgue measure. Let $\psi \in (\mathcal{L}^2[a,b])^{49}$. Then,

$$(T\psi)(t) = \begin{bmatrix} \int_a^t \psi_1 x_2(\xi) x_1(\xi) d\xi \\ \int_a^t \psi_2 x_1(\xi) + \psi_3 x_4(\xi - 20) x_2(\xi) + \psi_4 x_2(\xi) d\xi \\ \int_a^t \psi_5 x_2(\xi) + \psi_6 x_6(\xi) + \psi_7 x_8(\xi) + \psi_8 x_{13}(\xi) d\xi \\ \int_a^t \psi_9 x_1'(\xi) + \psi_{10} x_4(\xi) + \psi_{11} x_2(\xi) x_4(\xi - 20) + \psi_{12} x_8(\xi) x_4(\xi) d\xi \\ \int_a^t \psi_{13} x_8(\xi) + \psi_{14} x_{10}(\xi) + \psi_{15} x_4(\xi) + \psi_{16} x_{12}(\xi) d\xi \\ \int_a^t \psi_{17} x_6(\xi) + \psi_{18} x_7(\xi - 17) + \psi_{19} x_{11}(\xi) + \psi_{20} x_{12}(\xi) d\xi \\ \int_a^t \psi_{21} x_7(\xi) + \psi_{22} x_6(\xi) + \psi_{23} x_{11}(\xi) d\xi \\ \int_a^t \psi_{24} x_6(\xi) + \psi_{25} x_5'(\xi) + \psi_{26} x_8(\xi) + \psi_{27} x_8(\xi) x_4(\xi) \\ + \psi_{28} x_3(\xi) + \psi_{29} x_{10}'(\xi) + \psi_{30} x_{13}'(\xi) d\xi \\ \int_a^t \psi_{31} x_{10}(\xi) + \psi_{32} x_{12}(\xi) + \psi_{33} x_2(\xi) + \psi_{34} x_9(\xi) d\xi \\ \int_a^t \psi_{35} x_{10}(\xi) + \psi_{36} x_4(\xi) + \psi_{37} x_1(\xi) + \psi_{38} x_5(\xi) + \psi_{39} x_{12}(\xi) d\xi \\ \int_a^t \psi_{40} x_7(\xi - 17) + \psi_{41} x_{11}(\xi) + \psi_{42} x_4(\xi - 20) x_{11}(\xi) d\xi \\ \int_a^t \psi_{43} x_{10}'(\xi) + \psi_{44} x_9(\xi) + \psi_{45} x_4(\xi) + \psi_{46} x_1(\xi) d\xi \\ \int_a^t \psi_{47} x_{10}(\xi) + \psi_{48} x_6(\xi) + \psi_{49} x_7(\xi) d\xi \end{bmatrix} \tag{5.6}$$

Now let $M \in (\mathcal{L}^2[a,b])^{13 \times 49}$

$$M = \begin{bmatrix} x_2 x_1 & 0_2 \cdots 0_{49} & & & & & & & \\ 0_1 & x_2 & x_{4d} x_2 & x_2 & 0_5 \cdots 0_{49} & & & & \\ 0_1 \cdots 0_4 & x_2 & x_6 & x_8 & x_{13} & 0_9 \cdots 0_{49} & & & \\ 0_1 \cdots 0_8 & x_1' & x_4 & x_2 x_{4d} & x_8 x_4 & 0_{13} \cdots 0_{49} & & & \\ 0_1 \cdots 0_{12} & x_8 & x_{10} & x_4 & x_{12} & 0_{17} \cdots 0_{49} & & & \\ 0_1 \cdots 0_{16} & x_6 & x_{7d} & x_{11} & x_{12} & 0_{21} \cdots 0_{49} & & & \\ 0_1 \cdots 0_{20} & x_7 & x_6 & x_{11} & 0_{24} \cdots 0_{49} & & & & \\ 0_1 \cdots 0_{23} & x_6 & x_5' & x_8 & x_8 x_4 & x_3 & x_{10}' & x_{13}' & 0_{31} \cdots 0_{49} \\ 0_1 \cdots 0_{30} & x_{10} & x_{12} & x_2 & x_9 & 0_{35} \cdots 0_{49} & & & \\ 0_1 \cdots 0_{34} & x_{10} & x_4 & x_1 & x_5 & x_{12} & 0_{40} \cdots 0_{49} & & \\ 0_1 \cdots 0_{39} & x_{7d} & x_{11} & x_{4d} x_{11} & 0_{43} \cdots 0_{49} & & & & \\ 0_1 \cdots 0_{42} & x_{10}' & x_9 & x_4 & x_1 & 0_{47} & 0_{48} & 0_{49} & \\ 0_1 \cdots 0_{46} & x_{10} & x_6 & x_7 & & & & & \end{bmatrix} \tag{5.7}$$

where 0_k represents 0 in the kth column of a particular row of M. Then, we can see that $(T\psi)(t) = \int_a^t (M\psi)(\xi)d\xi$. Let $\{e_k\}$ be the standard basis for $\mathbb{R}^{49}$ and $\{u_n\}$ be the standard basis for $\mathbb{R}^{13}$. We define the following:

$$P_j : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{49} : \psi \mapsto e_j^T \psi e_j,$$

$$G_i : (\mathcal{L}^2[a,b])^{13} \to (\mathcal{L}^2[a,b]) : \rho \mapsto u_i^T \rho,$$

$$\Gamma_{i,j} : (\mathcal{L}^2[a,b])^{49} \to \mathcal{L}^2[a,b] : \psi \mapsto G_i(T(P_j(\psi))),$$

$$K_{i,j} : [a,b] \times [a,b] \to \mathcal{L}^2[a,b] := (t,\xi) \mapsto \begin{cases} M_{i,j}(\xi) & \text{if } a \le \xi \le t \\ 0 & \text{if } t < \xi \le b \end{cases}.$$

Notice here, $K_{i,j} \in \mathcal{L}^2([a,b] \times [a,b])$ and

$$\Gamma_{i,j}(\psi) = \int_a^t M_{i,j}\psi_j(\xi)d\xi = \int_{[a,b]} K_{i,j}(t,\xi)\psi_j(\xi)d\xi.$$

Now, define an operator $S_{i,j} : \mathcal{L}^2[a,b] \to \mathcal{L}^2[a,b] : \phi \mapsto \int_{[a,b]} K_{i,j}(t,\xi)\phi(\xi)d\xi$.
Using these, we first show that $\dim(\mathcal{R}(T)) = \infty$.

Theorem 5.3. *Let $T : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{13}$ be as defined above, and assume $x_1, \cdots, x_{13}$ are zero only at finitely many places. Then, $dim(\mathcal{R}(T)) = \infty$*

Proof. Denote $(T_1\psi)(t) = \int_a^t M_{(1,:)}(\xi)\psi(\xi)d\xi$. Then,

$$\begin{aligned}(T_1\psi)(t) &= \int_a^t M_{(1,:)}(\xi)\psi(\xi)d\xi \\ &= \int_a^t \sum_{j=1}^{49} M_{1,j}(\xi)\psi_i(\xi)d\xi \\ &= \int_a^t M_{1,1}(\xi)\psi_1(\xi)d\xi \\ &= (S_{1,1}\psi_1)(t).\end{aligned}$$

To show that $\dim(\mathcal{R}(T)) = \infty$, we only need to show that $\dim(\mathcal{R}(T_1)) = \infty$. Let $\{d_i\}_{i=1}^{\infty}$ be a basis of $\mathcal{L}^2[a,b]$.

Let $m_1, \cdots, m_n \in \mathbb{R}$. Then,

$$\begin{aligned} m_1T_1(d_1) + \cdots + m_nT_1(d_n) &= \int_a^t m_1d_1(\xi)x_1(\xi)x_2(\xi) + \cdots + m_nd_n(\xi)x_1(\xi)x_2(\xi)d\xi \\ &= \int_a^t \left(m_1d_1(\xi) + \cdots + m_nd_n(\xi)\right) x_1(\xi)x_2(\xi)d\xi \end{aligned}$$

Since x_1 and x_2 are non-zero almost everywhere, we see that in order for $m_1T_1(d_1) + \cdots + m_nT_1(d_n) = 0$ for all t, $m_1d_1(\xi) + \cdots + m_nd_n(\xi) = 0$ a.e.. Since $d_1, \cdots, d_n$ are linearly independent, we see that $m_1 = \cdots = m_n = 0$. Therefore, we see that $T(d_1), \cdots, T(d_n)$ are linearly independent.

From here, we can see that $\{T_1(d_i)\}_{i=1}^{\infty}$ is a set of linearly independent vectors in $\mathcal{R}(T_1)$. Therefore, $\dim(\mathcal{R}(T_1)) = \infty$ which implies that $\dim(\mathcal{R}(T)) = \infty$.

□

Theorem 5.4. *T is compact.*

PROOF. Applying Hilbert-Schmidt Theorem, we see that each operator $S_{i,j}$ is a compact operator on $\mathcal{L}^2[a,b]$ and a Hilbert-Schmidt operator as defined in Theorem 5.2. Let $\{v_k\}$ be an orthonormal basis of $(\mathcal{L}^2)^{49}$. Then,

$$\begin{aligned} \sum_{k=1}^{\infty} \|Tv_k\|^2_{(\mathcal{L}^2)^{49}} &= \sum_{k=1}^{\infty} \langle Tv_k, Tv_k\rangle_{(\mathcal{L}^2)^{13}} \\ &= \sum_{k=1}^{\infty}\sum_{j=1}^{49}\sum_{m=1}^{49} \langle T(P_j(v_k)), T(P_m(v_k))\rangle_{(\mathcal{L}^2)^{13}} \\ &= \sum_{k=1}^{\infty}\sum_{i=1}^{13}\sum_{j=1}^{49}\sum_{r=1}^{13}\sum_{m=1}^{49} \langle G_i(T(P_j(v_k))), G_r(T(P_m(v_k)))\rangle_{\mathcal{L}^2} \\ &= \sum_{k=1}^{\infty} \sum_{\substack{1\le j\le 49\\ 1\le i\le 13}} \sum_{\substack{1\le m\le 49\\ 1\le r\le 13}} \langle \Gamma_{i,j}(v_k), \Gamma_{r,m}(v_k)\rangle_{\mathcal{L}^2} \\ &= \sum_{k=1}^{\infty} \sum_{\substack{1\le j\le 49\\ 1\le i\le 13}} \sum_{\substack{1\le m\le 49\\ 1\le r\le 13}} \langle S_{i,j}(v_k)_j, S_{r,m}(v_k)_m\rangle_{\mathcal{L}^2} \end{aligned}$$

Applying Cauchy-Schwarz inequality, we see that

$$\sum_{k=1}^{\infty} \|Tv_k\|^2_{(\mathcal{L}^2)^{49}} \leq \sum_{k=1}^{\infty} \sum_{\substack{1\leq j\leq 49\\ 1\leq i\leq 13}} \sum_{\substack{1\leq m\leq 49\\ 1\leq r\leq 13}} \|S_{i,j}(v_k)_j\|_{\mathcal{L}^2} \|S_{r,m}(v_k)_m\|_{\mathcal{L}^2}.$$

Now, using the fact that $uz \leq \frac{1}{2}(u^2+z^2)$ we obtain

$$\sum_{k=1}^{\infty} \|Tv_k\|^2_{(\mathcal{L}^2)^{49}} \leq \frac{1}{2} \sum_{\substack{1\leq j\leq 49\\ 1\leq i\leq 13}} \sum_{\substack{1\leq m\leq 49\\ 1\leq r\leq 13}} \sum_{k=1}^{\infty} \left(\|S_{i,j}(v_k)_j\|^2_{\mathcal{L}^2} + \|S_{r,m}(v_k)_m\|^2_{\mathcal{L}^2} \right). \tag{5.8}$$

Since each $S_{i,j}$ and $S_{r,m}$ is a Hilbert-Schmidt operator, we can see that $\sum_{k=1}^{\infty} \|S_{i,j}(v_k)_j\|^2 < \infty$ and $\sum_{k=1}^{\infty} \|S_{r,m}(v_k)_m\|^2 < \infty$ for each i, j, k, and m. Therefore,

$$\sum_{k=1}^{\infty} \|Tv_k\|^2_{(\mathcal{L}^2)^{49}} < \infty$$

and hence, T is a compact operator. □

Given that T is a compact operator and $\dim(\mathcal{R}(T)) = \infty$, we can see that our inverse problem $T\mathbf{c} = g$ is ill-posed by Theorem 3.3. As a result, we must solve the inverse problem using an iterative variational regularization method, and availing ourselves of its many advantages (see Remark 3.4) we choose to apply the Nayak regularization method.

5. Application of Nayak's Regularization Method

In this section, we apply Nayak's regularization method to our inverse problem

$$T\mathbf{c} = g \tag{5.9}$$

where $T : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{13}$, $g : [a,b] \to (\mathcal{L}^2[a,b])^{13}$, and $c \in (\mathcal{L}^2[a,b])^{49}$ are as described in section 3.

Using a similar set up to the above, we now let $u : [a,b] \to (\mathcal{L}^2[a,b])^{13}$ be defined by

$$u(t) = \int_t^b \int_a^\eta g(\xi)d\xi d\eta - \frac{b-t}{b-a}\int_a^b \int_a^\eta g(\xi)d\xi d\eta, \tag{5.10}$$

so that $g = -u''$ and $u(a) = u(b) = 0$. Therefore, our inverse problem can now be reformulated as

$$T\mathbf{c} = -u''. \tag{5.11}$$

With this, we can also see that $u' : [a,b] \to (\mathcal{L}^2[a,b])^{13}$ is defined as

$$u'(t) = -\int_a^t g(\xi)d\xi + \frac{1}{b-a}\int_a^b \int_a^\eta g(\xi)d\xi d\eta. \tag{5.12}$$

We now define $u_\psi : [a,b] \to (\mathcal{L}^2[a,b])^{13}$ as the solution of the boundary value problem below:

$$\begin{aligned} T\psi &= -u_\psi'', \\ u_\psi(a) &= u(a) = 0, \\ u_\psi(b) &= u(b) = 0. \end{aligned} \tag{5.13}$$

To solve this, we first integrate both sides on $[a,b]$,

$$u_\psi'(t) = u_\psi'(a) - \int_a^t (T\psi)(\xi)d\xi.$$

Integrating both sides again and using the fact that $u_\psi(a) = 0$, we now get

$$u_\psi(t) = (t-a)u_\psi'(a) - \int_a^t \int_a^\eta (T\psi)(\xi)d\xi d\eta. \tag{5.14}$$

Further, notice that at $t = b$,

$$u_\psi(b) = (b-a)u_\psi'(a) - \int_a^b \int_a^\eta (T\psi)(\xi)d\xi d\eta,$$

$$u_\psi'(a) = \frac{1}{b-a}\int_a^b \int_a^\eta (T\psi)(\xi)d\xi d\eta, \qquad \text{since } u_\psi(b) = 0.$$

From this, we can now see that $u'_\psi : [a,b] \to (\mathcal{L}^2[a,b])^{13}$ is defined as

$$
\begin{aligned}
u'_\psi(t) &= u'_\psi(a) - \int_a^t (T\psi)(\xi)d\xi, \\
u'_\psi(t) &= \frac{1}{b-a}\int_a^b \int_a^\eta (T\psi)(\xi)d\xi d\eta - \int_a^t (T\psi)(\xi)d\xi,
\end{aligned} \tag{5.15}
$$

and $u_\psi : [a,b] \to (\mathcal{L}^2[a,b])^{13}$ is defined as

$$
u_\psi(t) = \frac{t-a}{b-a}\int_a^b \int_a^\eta (T\psi)(\xi)d\xi d\eta - \int_a^t \int_a^\eta (T\psi)(\xi)d\xi d\eta. \tag{5.16}
$$

Given these, we can now see that the functional which we are minimizing for Nayak's regularization method is

$$
G : (\mathcal{L}^2[a,b])^{49} \to \mathbb{R} : \psi \mapsto \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u'_\psi - u'\right\|^2_{(\mathcal{L}^2)^{13}}.
$$

The following theorem is an extension of Nayak's Theorem 7.1 [**32**, p.125] whose proof can be routinely extended.

Theorem 5.5. *Let $T : (\mathcal{L}^2[a,b])^{49} \to (\mathcal{L}^2[a,b])^{13}$ be a bounded linear compact operator and $g : [a,b] \to (\mathcal{L}^2[a,b])^{13}$.*

(1) $$\left(1 + \frac{1}{\lambda_1}\right)^{-1} \|u - u_\psi\|^2_{(\mathcal{H}^2)^{13}} \le G(\psi) \le \|u - u_\psi\|^2_{(\mathcal{H}^2)^{13}}$$
where $\lambda_1 > 0$ is the first eigenvalue of the positive operator $-\Delta$

(2) *For fixed $\psi, h \in (\mathcal{L}^2[a,b])^{49}$*

$$
\lim_{\epsilon \to 0} u_{\psi + \epsilon h} = u_\psi \tag{5.17}
$$

in $(\mathcal{H}^1[a,b])^{13}$.

(3) *The first Gâteaux derivative at $\psi \in (\mathcal{L}^2[a,b])^{49}$ in $h \in (\mathcal{L}^2[a,b])^{49}$ direction is given by*

$$
G'(\psi)[h] = -2\left\langle Th, u - u_\psi + g - T\psi \right\rangle_{(\mathcal{L}^2[a,b])^{13}}. \tag{5.18}
$$

Further, the $\mathcal{L}^2$-gradient of G at ψ is given by

$$\nabla^{\psi}_{\mathcal{L}^2} G = -2T^*(u - u_\psi + g - T\psi). \tag{5.19}$$

(4) *The second Gâteaux derivative at $\psi \in (\mathcal{L}^2[a,b])^{49}$ with $h, k \in (\mathcal{L}^2[a,b])^{49}$ is given by*

$$G''(\psi)[h,k] = 2\left\langle -\Delta^{-1}(Th) + Th, Tk \right\rangle_{(\mathcal{L}^2[a,b])^{13}}. \tag{5.20}$$

We now observe that G is convex but not strictly convex. Let $h \in (\mathcal{L}^2[a,b])^{49}$. Then,

$$\begin{aligned} G''(\psi)[h,h] &= 2\left\langle -\Delta^{-1}(Th) + Th, Th \right\rangle_{(\mathcal{L}^2[a,b])^{13}} \\ &= 2\left\langle -\Delta^{-1}(Th), Th \right\rangle_{(\mathcal{L}^2[a,b])^{13}} + 2\left\langle Th, Th \right\rangle_{(\mathcal{L}^2[a,b])^{13}} \\ &= 2\left\langle y, -\Delta y \right\rangle_{(\mathcal{L}^2[a,b])^{13}} + 2\left\langle -\Delta y, -\Delta y \right\rangle_{(\mathcal{L}^2[a,b])^{13}}, \end{aligned}$$

where $-\Delta y = Th$ and $y \in (\mathcal{H}^2_0(\Omega))^{13}$. Since $-\Delta$ is a positive operator on $(\mathcal{H}^2_0)^{13}$, we see that $G''(\psi)[h,h] = 0$ if and only if $Th = 0$. Therefore, $G''(\psi) = 0$ for all $h \in \mathrm{Ker}(T)$ since $\mathrm{Ker}(T) \neq \{0\}$. Hence, $G''(\psi)$ is a positive semi-definite quadratic form. Thus, our functional does not guarantee a unique global minimizer.

We now show the stability of Nayak's method for our problem, using the notations defined below.

$$T_i : (\mathcal{L}^2[a,b])^{49} \to \mathcal{L}^2[a,b] : \psi \mapsto \int_a^t M_{(i,:)}(\xi)\psi(\xi)d\xi, \tag{5.21}$$

$$u_i' : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto e_i^T u' = \frac{1}{b-a}\int_a^b \int_a^\eta g_i(\xi)d\xi d\eta - \int_a^t g_i(\xi)d\xi, \tag{5.22}$$

$$(u_\psi')_i : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto e_i^T u_\psi' = \frac{1}{b-a}\int_a^b \int_a^\eta T_i\psi(\xi)d\xi d\eta - \int_a^t T_i\psi(\xi)d\xi. \tag{5.23}$$

Using these notations, we can now express (5.34) as

$$\begin{aligned} G(\psi) &= \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u'_\psi - u'\right\|^2_{(\mathcal{L}^2)^{13}} \\ &= \sum_{i=1}^{13} \left(\|T_i\psi - g_i\|^2_{\mathcal{L}^2} + \left\|(u'_\psi)_i - u'_i\right\|^2_{\mathcal{L}^2} \right). \end{aligned} \tag{5.24}$$

Now, expressing each term of G associated with i as below

$$G_i(\psi) = \|T_i\psi - g_i\|^2_{\mathcal{L}^2} + \left\|(u'_\psi)_i - u'_i\right\|^2_{\mathcal{L}^2}, \tag{5.25}$$

we see that

$$G(\psi) = \sum_{i=1}^{13} G_i(\psi). \tag{5.26}$$

Theorem 5.6. *Let $g, \tilde{g} \in (\mathcal{H}^1(\Omega))^{13}$ such that $\|g - \tilde{g}\|_{(\mathcal{L}^2)^{13}} < \delta$ for $0 < \delta \leq 1$. Then, for some constant C, the corresponding $u, \tilde{u} \in (\mathcal{H}^3_0(\Omega))^{13}$ satisfies*

$$\|u - \tilde{u}\|_{(\mathcal{H}^1)^{13}} < C\delta. \tag{5.27}$$

Proof.

$$\begin{aligned} \|u - \tilde{u}\|_{(\mathcal{H}^1)^{13}} &= \sqrt{\|u - \tilde{u}\|^2_{(\mathcal{L}^2)^{13}} + \|u' - \tilde{u}'\|^2_{(\mathcal{L}^2)^{13}}} \\ &= \sqrt{\sum_{i=1}^{13} \left(\|u_i - \tilde{u}_i\|^2_{\mathcal{L}^2} + \|u'_i - \tilde{u}'_i\|^2_{\mathcal{L}^2} \right)} \\ &= \sqrt{\sum_{i=1}^{13} \|u_i - \tilde{u}_i\|^2_{\mathcal{H}^1}}. \end{aligned}$$

Now since $\|g - \tilde{g}\|_{(\mathcal{L}^2)^{13}} \leq \delta$, we know that $\|g_i - \tilde{g}_i\|_{\mathcal{L}^2} \leq \delta$ for all $1 \leq i \leq 13$. Then, by Theorem 4.1, we see $\|u_i - \tilde{u}_i\|_{\mathcal{H}^1} \leq C_i\delta$ for all i where C_i are constants. Let C_M represent $\max(C_i)$. Then,

$$\begin{aligned} \|u - \tilde{u}\|_{(\mathcal{H}^1)^{13}} &< \sqrt{\sum_{i=1}^{13} C_i^2\delta^2} \\ &\leq \sqrt{13} C_M \delta. \end{aligned}$$

Therefore, we see that $\|u - \tilde{u}\|_{(\mathcal{H}^1)^{13}} < C\delta$. □

Theorem 5.7. *Let $g \in (\mathcal{L}^2(\Omega))^{13}$ be given data and $\tilde{g}$ be perturbed data such that $\|g - \tilde{g}\|_{(\mathcal{L}^2)^{13}} < \delta$ for $0 < \delta \leq 1$. Further, let $\{\psi_m\} \subset (\mathcal{L}^2(\Omega))^{49}$ such that $G_{\tilde{u}}(\psi_m) \to 0$. Then, there exists a positive number $M(\delta)$ such that for all $m \geq M(\delta)$, $G(\psi_m) < C\delta$ for some constant C.*

PROOF. Recall that $G_{\tilde{u}}(\psi_m) = \|T\psi_m - \tilde{g}\|^2_{(\mathcal{L}^2)^{13}} + \left\|\tilde{u}'_{\psi_m} - \tilde{u}'\right\|^2_{(\mathcal{L}^2)^{13}}$. Since $G_{\tilde{u}}(\psi_m) \to 0$, we see that $T\psi_m$ converges to $\tilde{g}$ in $(\mathcal{L}^2)^{13}$ and each $T_i\psi_m$ converges to $\tilde{g}_i$ in $\mathcal{L}^2$. Hence, for large m, we see that

$$\begin{aligned}
\|T\psi_m - g\|^2_{(\mathcal{L}^2)^{13}} &= \sum_{i=1}^{13} \|T_i\psi_m - g_i\|^2_{\mathcal{L}^2} \\
&\leq \sum_{i=1}^{13} \left(\|T_i\psi_m - \tilde{g}_i\|_{\mathcal{L}^2} + \|g_i - \tilde{g}_i\|_{\mathcal{L}^2}\right)^2 \\
&< 13\delta^2.
\end{aligned}$$

Therefore,

$$\begin{aligned}
G(\psi_m) &= \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u'_\psi - u'\right\|^2_{(\mathcal{L}^2)^{13}} \\
&\leq \delta^2 + C\delta^2 \text{ by Theorem 5.6} \\
&\leq (1 + C)\delta^2 \\
&\leq (1 + C)\delta.
\end{aligned}$$

□

Now, recall that the solution of our inverse problem is found by obtaining

$$\varphi = \arg\min_{\psi} \left\{ \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u'_\psi - u'\right\|^2_{(\mathcal{L}^2)^{13}} \right\}. \tag{5.28}$$

When minimizing our G, we follow the descent process using the Neuberger gradient since it allows flexibility at the boundaries. Now, the Neuberger gradient, $\nabla^{\psi}_{\mathcal{H}^1} G$, is

found by solving the boundary value problem below:

$$
\begin{aligned}
-(\nabla^{\psi}_{\mathcal{H}^1} G)'' + \nabla^{\psi}_{\mathcal{H}^1} G &= \nabla^{\psi}_{\mathcal{L}^2} G, \\
(\nabla^{\psi}_{\mathcal{H}^1} G)'(\nabla^{\psi}_{\mathcal{H}^1} G)|_a^b &= 0.
\end{aligned} \tag{5.29}
$$

As we see, to obtain the $\mathcal{H}^1$ gradient, we need $\nabla^{\psi}_{\mathcal{L}^2} G = -2T^*(u - u_\psi + g - T\psi)$. Therefore, we now derive $T^* : (\mathcal{L}^2[a,b])^{13} \to (\mathcal{L}^2[a,b])^{49}$ such that

$$
\langle T\psi, \rho \rangle_{(\mathcal{L}^2)^{13}} = \langle \psi, T^* \rho \rangle_{(\mathcal{L}^2)^{49}},
$$

for $\psi \in (\mathcal{L}^2[a,b])^{49}$ and $\rho \in (\mathcal{L}^2[a,b])^{13}$.

Let $M \in (\mathcal{L}^2[a,b])^{13\times 49}$ be as defined in (5.7). Then,

$$
\begin{aligned}
\langle T\psi, \rho \rangle_{(\mathcal{L}^2)^{13}} &= \sum_{i=1}^{13} \left\langle \int_a^t M_{(i,:)}(\xi)\psi(\xi)d\xi, \rho_i \right\rangle_{\mathcal{L}^2} \\
&= \sum_{i=1}^{13} \left[-\int_t^b \rho_i(\xi)d\xi \cdot \int_a^t M_{(i,:)}(\xi)\psi(\xi)d\xi \right]_a^b + \left\langle M_{(i,:)}(\xi)\psi(\xi), \int_t^b \rho_i(\xi)d\xi \right\rangle_{\mathcal{L}^2} \\
&= \sum_{i=1}^{13} \left\langle M_{(i,:)}(\xi)\psi(\xi), \int_t^b \rho_i(\xi)d\xi \right\rangle_{\mathcal{L}^2} \\
&= \left\langle \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \\ \psi_7 \\ \psi_8 \\ \psi_9 \\ \psi_{10} \\ \psi_{11} \\ \psi_{12} \end{bmatrix}, \begin{bmatrix} x_1(t)x_2(t)\int_t^b \rho_1(\xi)d\xi \\ x_1(t)\int_t^b \rho_2(\xi)d\xi \\ x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_3(\xi)d\xi \\ x_6(t)\int_t^b \rho_3(\xi)d\xi \\ x_8(t)\int_t^b \rho_3(\xi)d\xi \\ x_{13}(t)\int_t^b \rho_3(\xi)d\xi \\ x_1'(t)\int_t^b \rho_4(\xi)d\xi \\ x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \end{bmatrix} \right\rangle_{(\mathcal{L}^2)^{12}} +
\end{aligned}
$$

$$+\left\langle \begin{bmatrix} \psi_{13} \\ \psi_{14} \\ \psi_{15} \\ \psi_{16} \\ \psi_{17} \\ \psi_{18} \\ \psi_{19} \\ \psi_{20} \\ \psi_{21} \\ \psi_{22} \\ \psi_{23} \\ \psi_{24} \\ \psi_{25} \\ \psi_{26} \\ \psi_{27} \\ \psi_{28} \\ \psi_{29} \\ \psi_{30} \end{bmatrix}, \begin{bmatrix} x_8(t)\int_t^b \rho_5(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\ x_4(t)\int_t^b \rho_5(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\ x_6(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_6(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_7(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \psi_8(\xi)d\xi \\ x_5'(t)\int_t^b \psi_8(\xi)d\xi \\ x_8(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\ x_3(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\ x_{13}'(t)\int_t^b \rho_8(\xi)d\xi \end{bmatrix} \right\rangle_{(\mathcal{L}^2)^{18}} + \left\langle \begin{bmatrix} \psi_{31} \\ \psi_{32} \\ \psi_{33} \\ \psi_{34} \\ \psi_{35} \\ \psi_{36} \\ \psi_{37} \\ \psi_{38} \\ \psi_{39} \\ \psi_{40} \\ \psi_{41} \\ \psi_{42} \\ \psi_{43} \\ \psi_{44} \\ \psi_{45} \\ \psi_{46} \\ \psi_{47} \\ \psi_{48} \\ \psi_{49} \end{bmatrix}, \begin{bmatrix} x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\ x_2(t)\int_t^b \rho_9(\xi)d\xi \\ x_9(t)\int_t^b \rho_9(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_5(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_{11}(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_4(t-20)x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_9(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_{13}(\xi)d\xi \\ x_6(t)\int_t^b \rho_{13}(\xi)d\xi \\ x_7(t)\int_t^b \rho_{13}(\xi)d\xi \end{bmatrix} \right\rangle_{(\mathcal{L}^2)^{19}}$$

Therefore, $T^* : (\mathcal{L}^2[a,b])^{13} \to (\mathcal{L}^2[a,b])^{49}$ is defined as below:

$$(T^*\rho)(t) = \begin{bmatrix} x_1(t)x_2(t)\int_t^b \rho_1(\xi)d\xi \\ x_1(t)\int_t^b \rho_2(\xi)d\xi \\ x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_3(\xi)d\xi \\ x_6(t)\int_t^b \rho_3(\xi)d\xi \\ x_8(t)\int_t^b \rho_3(\xi)d\xi \\ x_{13}(t)\int_t^b \rho_3(\xi)d\xi \\ x_1'(t)\int_t^b \rho_4(\xi)d\xi \\ x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)\int_t^b \rho_5(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\ x_4(t)\int_t^b \rho_5(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\ x_6(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_6(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_7(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \psi_8(\xi)d\xi \\ x_5'(t)\int_t^b \psi_8(\xi)d\xi \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x_8(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\ x_3(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\ x_{13}'(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\ x_2(t)\int_t^b \rho_9(\xi)d\xi \\ x_9(t)\int_t^b \rho_9(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_5(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_{11}(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_4(t-20)x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_9(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_{13}(\xi)d\xi \\ x_6(t)\int_t^b \rho_{13}(\xi)d\xi \\ x_7(t)\int_t^b \rho_{13}(\xi)d\xi \end{bmatrix}. \tag{5.30}$$

Now that we have T^* defined, recall that the Neuberger gradient $\nabla^{\psi}_{\mathcal{H}^1}G$ is found by solving the boundary value problem below:

$$\begin{aligned} -(\nabla^{\psi}_{\mathcal{H}^1}G)'' + \nabla^{\psi}_{\mathcal{H}^1}G &= \nabla^{\psi}_{\mathcal{L}^2}G, \\ (\nabla^{\psi}_{\mathcal{H}^1}G)'(\nabla^{\psi}_{\mathcal{H}^1}G)\big|_a^b &= 0. \end{aligned} \tag{5.31}$$

Let $h = \nabla^{\psi}_{\mathcal{H}^1}G$ and $L = \nabla^{\psi}_{\mathcal{L}^2}G$. Then see that,

$$\begin{aligned} h - h'' &= L \\ \iff h_k - h_k'' &= L_k, \end{aligned} \tag{5.32}$$

where $1 \leq k \leq 49$. Let $y_{1_k} = h_k$ and $y_{2_k} = h_k'$ s.t.

$$\begin{aligned} y_{1_k}' &= y_{2_k}, \\ y_{2_k}' &= h_k - L_k = y_{1_k} - L_k. \end{aligned} \tag{5.33}$$

We solve (5.33) with Neumann boundary conditions in MATLAB for each h_k with **bvp4c**, as we do not have sufficient information about the behaviors at the endpoints. The end result here is the vector $\nabla^{\psi}_{\mathcal{H}^1}G \in (\mathcal{L}^2[a,b])^{49}$.

With the information presented above, we apply the descent algorithm outlined below to obtain a solution:

(1) Initialize ψ_0 and define needed functions.

(2) Calculate $\nabla^{\psi_m}_{\mathcal{H}^1}G$.

(3) Update $\psi_{m+1} = \psi_m - \alpha\nabla^{\psi_m}_{\mathcal{H}^1}G$.

We now note that since $\mathcal{N}(T) \neq \{0\}$, even if $g \in \mathcal{R}(T) \oplus \mathcal{R}(T)^{\perp}$ and the least-squares solution exists, the solution to our inverse problem will not be unique. Due to this non-uniqueness issue, we cannot ignore the effect of initial ψs. Furthermore, because of the possible flaws in the initial model (1.1) and the data, we cannot anticipate that G converges to zero. As a result, neither the number of iterations nor the actual value of G can serve as our terminating condition for the descent process. In the following

sections, we present our method for determining initial points, and our termination criteria.

Remark 5.8. The α Nayak used can be found in [**32**, p.90]. However, we do not use Nayak's method of finding α. Let $\max_h = \max\left|\nabla^{\psi_0}_{\mathcal{H}^1} G\right|$. Then, we choose $\alpha = 10^k$ where $\max_h \cdot 10^3 < \alpha < \max_h \cdot 10^4$ and $k \in \mathbb{N}$, see Appendix C page 458.

6. The Starting Point Problem

Nayak proposed using the zero vector as the initial point for his regularization method. However, our inverse problem has a non-uniqueness issue and when the descent process starts from $\psi_0 = \mathbf{0}$, it only produces one of the many solutions, which may or may not be optimal. In this section, we explore a way to select more effective starting points.

6.1. Notation. Recall that $x_1, \cdots, x_{13} \in \mathcal{L}^2[a,b]$ are continuous functions on $[a,b] \subset \mathcal{T}$. Further, recall the following equations:

$$G(\psi) = \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u'_\psi - u'\right\|^2_{(\mathcal{L}^2)^{13}}, \tag{5.34}$$

$$u(t) = \int_t^b \int_a^\eta g(\xi)d\xi d\eta - \frac{b-t}{b-a}\int_a^b \int_a^\eta g(\xi)d\xi d\eta, \tag{5.35}$$

$$u_\psi(t) = \frac{t-a}{b-a}\int_a^b \int_a^\eta (T\psi)(\xi)d\xi d\eta - \int_a^t \int_a^\eta (T\psi)(\xi)d\xi d\eta. \tag{5.36}$$

Let $J \in (\mathcal{L}^2[a,b])^{49}$ represent the least-square solution found using Barnett's method and let $M \in (\mathcal{L}^2[a,b])^{13\times 49}$ be as defined in (5.7). We also denote the elements of g as g_i for all $1 \le i \le 13$ since $g \in (\mathcal{L}^2[a,b])^{13}$. Recall the following notations for all $1 \le i \le 13$ from Section 5.

$$T_i : (\mathcal{L}^2[a,b])^{49} \to \mathcal{L}^2[a,b] : \psi \mapsto \int_a^t M_{(i,:)}(\xi)\psi(\xi)d\xi, \tag{5.37}$$

$$u_i' : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto e_i^T u' = \frac{1}{b-a}\int_a^b \int_a^\eta g_i(\xi)d\xi d\eta - \int_a^t g_i(\xi)d\xi, \tag{5.38}$$

$$(u_\psi')_i : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto e_i^T u_\psi' = \frac{1}{b-a}\int_a^b \int_a^\eta T_i\psi(\xi)d\xi d\eta - \int_a^t T_i\psi(\xi)d\xi. \tag{5.39}$$

Using these notations, we can express (5.34) as

$$\begin{aligned} G(\psi) &= \|T\psi - g\|^2_{(\mathcal{L}^2)^{13}} + \left\|u_\psi' - u'\right\|^2_{(\mathcal{L}^2)^{13}} \\ &= \sum_{i=1}^{13} \left(\|T_i\psi - g_i\|^2_{\mathcal{L}^2} + \left\|(u_\psi')_i - u_i'\right\|^2_{\mathcal{L}^2} \right). \end{aligned} \tag{5.40}$$

Now, expressing each term of G associated with i as below

$$G_i(\psi) = \|T_i\psi - g_i\|^2_{\mathcal{L}^2} + \left\|(u_\psi')_i - u_i'\right\|^2_{\mathcal{L}^2}, \tag{5.41}$$

we see that

$$G(\psi) = \sum_{i=1}^{13} G_i(\psi). \tag{5.42}$$

Further, we denote the initial $\psi \in (\mathcal{L}^2[a,b])^{49}$ as ψ_0 and a zero vector as $\mathbf{0} \in (\mathcal{L}^2[a,b])^{49}$. Now, define k and m as follows:

$$k : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \int_a^t x_1(s)x_2(s)ds, \tag{5.43}$$

$$m : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \frac{1}{b-a}\int_a^b \int_a^\eta k(s)ds d\eta - \int_a^t k(s)ds. \tag{5.44}$$

Let $\omega \in (\mathcal{L}^2[a,b])^{49}$ be a constant function. Then we can see that

$$T_1\omega(t) = \int_a^t M_{(1,:)}(\xi)\omega(\xi)d\xi = \int_a^t \omega_1(\xi)x_1(\xi)x_2(\xi)d\xi = \omega_1(t)k(t)$$

$$
\begin{aligned}
(u'_\omega)_1(t) &= \frac{1}{b-a}\int_a^b \int_a^\eta T_1\omega(\xi)d\xi d\eta - \int_a^t T_1\omega(\xi)d\xi \\
&= \frac{1}{b-a}\int_a^b \int_a^\eta \omega_1(\xi)k(\xi)d\xi d\eta - \int_a^t \omega_1(\xi)k(\xi)d\xi \\
&= \omega_1(t)m(t).
\end{aligned}
$$

6.2. Necessary Theorems. We say that a ψ_0 is a better starting point if $G(\psi_0) < G(\mathbf{0})$ and $G(\psi_0) < G(J)$. Though the existence of such a ψ_0 may be deduced based on the descent process used in the construction of the decreasing sequence $\{G(\psi_m)\}$, we propose a method of finding a better ψ_0 that is independent of the descent process.

Recall that Barnett solves her equations over fixed subinterval $[t_{k-1}, t_k]$ where each $[t_{k-1}, t_k]$ represents one month for $1 \leq k \leq 7$. Consider the integral of the equation $x_1' = c_1 x_1(t)x_2(t)$ given by

$$
x_1(t) = c_1 e^{\int_1^t x_2(s)dx}.
$$

Since x_1 is approximately linear on each time interval $[t_{k-1}, t_k]$, the only way for c to be a constant function of time is if $x_2(s)$ is approximately $\frac{1}{s}$. Since x_2 is unlikely to satisfy this requirement, we should not expect c_1 in (1.1) to be constant. Therefore, Barnett's least-square solution cannot be a minimizer of G_1. Hence, we assume that $G_1(J) > 0$, noting that this is indeed consistent with all current data.

Theorem 5.9. *Suppose $\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \neq 0$. Then, we can always find a $\psi_0 \in (\mathcal{L}^2[a,b])^{49}$ such that $G(\psi_0) < G(\boldsymbol{0})$ without using the descent process.*

Proof. We first show that we can always find $\psi \in (\mathcal{L}^2[a,b])^{49}$ such that $G_1(\psi) < G_1(\mathbf{0})$ where

$$
\begin{aligned}
G_1(\psi) &= \|T_1\psi - g_1\|_{\mathcal{L}^2}^2 + \left\|(u'_\psi)_1 - u_1'\right\|_{\mathcal{L}^2}^2 = \|\psi_1 k - g_1\|_{\mathcal{L}^2}^2 + \|\psi_1 m - u_1'\|_{\mathcal{L}^2}^2 \\
G_1(\mathbf{0}) &= \|T_1\mathbf{0} - g_1\|_{\mathcal{L}^2}^2 + \|(u'_{\mathbf{0}})_1 - u_1'\|_{\mathcal{L}^2}^2 = \|g_1\|_{\mathcal{L}^2}^2 + \|u_1'\|_{\mathcal{L}^2}^2
\end{aligned} \tag{5.45}
$$

We divide the proof into cases. First, suppose that $\int_a^b k(t)^2 + m(t)^2 dt > 0$ and

$$q = \frac{2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt} > 0.$$

Then, we see that for c with $0 < c < q$,

$$c\int_a^b k(t)^2 + m(t)^2 dt < 2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt,$$

and thus

$$0 < 2\int_a^b g_1(t)k(t)dt + 2\int_a^b u_1'(t)m(t)dt - c\int_a^b k(t)^2 dt - c\int_a^b m(t)^2 dt.$$

Now, by multiplying both sides by c we get

$$0 < 2c\int_a^b g_1(t)k(t)dt + 2c\int_a^b u_1'(t)m(t)dt - c^2\int_a^b k(t)^2 dt - c^2\int_a^b m(t)^2 dt.$$

We now add and subtract $G_1(\mathbf{0}) = \int_a^b g_1(t)^2 dt + \int_a^b u_1'(t)^2 dt$, so that

$$\begin{aligned}
0 &< 2c\int_a^b g_1(t)k(t)dt + 2c\int_a^b u_1'(t)m(t)dt - c^2\int_a^b k(t)^2 dt - c^2\int_a^b m(t)^2 dt \\
&\qquad + \int_a^b g_1(t)^2 dt + \int_a^b u_1'(t)^2 dt - \int_a^b g_1(t)^2 dt - \int_a^b u_1'(t)^2 dt \\
&= G_1(\mathbf{0}) - \Big[\int_a^b (ck(t))^2 - 2cg_1(t)k(t) + g_1(t)^2 dt \\
&\qquad + \int_a^b u_1'(t)^2 - 2cu_1'(t)m(t) + (cm(t))^2 dt\Big] \\
&= G_1(\mathbf{0}) - \Big[\int_a^b (ck(t) - g_1(t))^2 dt + \int_a^b \left(u_1'(t) - cm(t)\right)^2 dt\Big] \\
&= G_1(\mathbf{0}) - \|T_1\mathbf{c} - g_1\|_{\mathcal{L}^2}^2 - \|u_1' - (u_{\mathbf{c}}')_1\|_{\mathcal{L}^2}^2 \text{ where } \mathbf{c} \in (\mathcal{L}^2[a,b])^{49} \text{ with } \mathbf{c}_1 = c \\
&= G_1(\mathbf{0}) - G_1(\mathbf{c})
\end{aligned}$$

Hence, we can see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 \in (0,q)$, $G_1(\mathbf{c}) < G_1(\mathbf{0})$.

Now, suppose $\int_a^b k(t)^2 + m(t)^2 dt > 0$ and

$$q = \frac{2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt} < 0.$$

Then, for c with $q < c < 0$, we see that

$$\begin{aligned}
&2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt < c\int_a^b k(t)^2 + m(t)^2 dt \\
\iff &2c\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt > c^2\int_a^b k(t)^2 + m(t)^2 dt \\
\iff &0 < -c^2\int_a^b k(t)^2 + m(t)^2 dt + 2c\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \\
&= \int_a^b g_1(t)^2 + \int_a^b u_1'(t)^2 dt - \int_a^b g_1(t)^2 dt - \int_a^b u_1'(t)^2 dt \\
&\qquad - c^2\int_a^b k(t)^2 + m(t)^2 dt + 2c\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \\
&= \int_a^b g_1(t)^2 + \int_a^b u_1'(t)^2 dt - \left[\int_a^b c^2 k(t)^2 - 2cg_1(t)k(t) + g_1(t)^2 dt\right. \\
&\qquad \left. + \int_a^b u_1'(t)^2 - 2cu_1'(t)m(t) + c^2 m(t)^2 dt\right] \\
&= G_1(\mathbf{0}) - \left[\int_a^b (ck(t) - g_1(t))^2 dt + \int_a^b (u_1'(t) - cm(t))^2 dt\right] \\
&= G_1(\mathbf{0}) - \|T_1\mathbf{c} - g_1\|_{\mathcal{L}^2}^2 - \|u_1' - (u_{\mathbf{c}}')_1\|_{\mathcal{L}^2}^2 \\
&\qquad \text{where } \mathbf{c} \in (\mathcal{L}^2[a,b])^{49} \text{ with } \mathbf{c}_1 = c, \\
&= G_1(\mathbf{0}) - G_1(\mathbf{c})
\end{aligned}$$

Hence, we see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $c_1 \in (q, 0)$, $G_1(\mathbf{c}) < G_1(\mathbf{0})$.

Lastly, suppose $\int_a^b k(t)^2 + m(t)^2 dt = 0$ and let $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 = c \in \mathbb{R}$. Then,

$$\begin{aligned}
G_1(\mathbf{0}) - G_1(\mathbf{c}) &= -c^2\int_a^b k(t)^2 + m(t)^2 dt + 2c\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \\
&= 2c\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt.
\end{aligned}$$

Therefore, if $\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt < 0$, we see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 < 0$ gives $G_1(\mathbf{c}) < G_1(\mathbf{0})$. Likewise, if $\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt > 0$, we see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 > 0$ gives $G_1(\mathbf{c}) < G_1(\mathbf{0})$.

Hence, we have shown the existence of $\psi \in (\mathcal{L}^2[a,b])^{49}$ with $G_1(\psi) < G_1(\mathbf{0})$ when $\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \neq 0$. Now, let $\rho \in (\mathcal{L}^2[a,b])^{49}$ be a vector of constant functions that satisfies $G_1(\rho) < G_1(\mathbf{0})$. Let $\gamma = e_1^T \rho e_1$ so that $G_1(\rho) = G_1(\gamma)$ where $\{e_i\}$ are the standard basis for $\mathbb{R}^{49}$. Then,

$$\begin{aligned} G(\mathbf{0}) &= \sum_{i=1}^{13} G_i(\mathbf{0}) \\ &> G_1(\rho) + \sum_{i=2}^{13} G_i(\mathbf{0}) \\ &> G_1(\gamma) + \sum_{i=2}^{13} G_i(\mathbf{0}) \\ &= G(\gamma). \end{aligned}$$

Therefore, we can always find $\psi_0 \in (\mathcal{L}^2[a,b])^{49}$ where $G(\psi_0) < G(\mathbf{0})$. □

Barnett's model is designed to predict Apple's stock trend starting on $v = 28$, which is October 1, 1996. To follow Barnett's model as closely as possible, we assume the range of v to starts at 28. However, since we we want to expand the range of prediction to March 31, 2023, the range of v is extended to 344 which is February 1, 2023.

Theorem 5.10. *Let* $v \in \left[28, \cdots, 344\right]$ *and let* $n \in [1, v-1]$. *Suppose* $\int_a^b k(t)^2 + m(t)^2 dt > 0$ *on the interval* $[a,b]$ *where* $a = v - n$ *and* $b = v$ *and suppose*

$$q = \frac{2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt}.$$

Let $J_1 = e_1^T J$ *where* $\{e_i\}$ *is the standard basis of* $\mathbb{R}^{49}$ *and* J *is the least-square solution found on* $[a,b]$ *as explained in Section 1.2. Further, suppose* $J_1 \neq q - J_1$. *Then, we can always find a* $\psi_0 \in (\mathcal{L}^2[a,b])^{49}$ *such that* $G(\psi_0) < G(J)$ *without using the descent process on* $[a,b]$.

PROOF. Suppose $J_1 < q - J_1$. Then, we see that there is $c \in \mathbb{R}$ such that $J_1 < c < q - J_1$. Notice here, $c - J_1 > 0$. Then,

$$c < \frac{2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt} - J_1,$$

and after some rearrangement,

$$(c + J_1)\int_a^b k(t)^2 + m(t)^2 dt < 2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt.$$

Subtracting and multiplying by $c - J_1 > 0$ gives

$$\begin{aligned} 0 > \int_a^b & (c^2 - J_1^2)k(t)^2 + (c^2 - J_1^2)m(t)^2 \\ & - 2(c - J_1)g_1(t)k(t) - 2(c - J_1)u_1'(t)m(t)dt, \end{aligned}$$

and adding and subtracting $\int_a^b g_1(t)^2 + u_1'(t)^2 dt$ we get

$$\begin{aligned} 0 > & \int_a^b c^2k(t)^2 - 2cg_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2cu_1'(t)m(t) + c^2m(t)^2 \\ & - \left[J_1^2k(t)^2 - 2J_1g_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right]dt \\ = & \int_a^b (T_1\mathbf{c}(t) - g_1(t))^2 + (u_1'(t) - (u_{\mathbf{c}}')_1(t))^2 dt \\ & - \int_a^b (T_1J(t) - g_1(t))^2 + (u_1'(t) - (u_J')_1(t))^2 dt, \end{aligned}$$

where $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T \in (\mathcal{L}^2[a,b])^{49}$. Consequently,

$$0 > G_1(\mathbf{c}) - G_1(J).$$

For the $\mathbf{c}$ defined above, we see that $G_1(\mathbf{c}) < G_1(J)$ and by (5.42), we see that $G(\mathbf{c}) < G(J)$. Hence, we see that if $J_1 < q - J_1$ then there is $\mathbf{c}$ such that $G(\mathbf{c}) < G(J)$ and proof for the case where $J_1 > q - J_1$ follows similarly. Therefore, we can always find a $\psi_0 \in (\mathcal{L}^2[a,b])^{49}$ such that $G(\psi_0) < G(J)$. □

Theorem 5.11. *Let* $v \in \left[28, \cdots, 344\right]$ *and let* $n \in [1, v-1]$. *Suppose* $\int_a^b k(t)^2 + m(t)^2 dt = 0$ *on the interval* $[a,b]$ *where* $a = v - n$ *and* $b = v$ *and* $2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \neq 0$. *Let* $J_1 = e_1^T J$ *where* $\{e_i\}$ *is the standard basis of* $\mathbb{R}^{49}$ *and* J *is the least-square solution found on* $[a,b]$, *as explained in Section 1.2. Then, we can always find a* $\psi_0 \in (\mathcal{L}^2[a,b])^{49}$ *such that* $G(\psi_0) < G(J)$ *without using the descent process on* $[a,b]$.

PROOF. Let $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T$ where $c \in \mathbb{R}$. Then,

$$\begin{aligned}
&G_1(\mathbf{c}) - G_1(J) \\
&\quad = \int_a^b c^2k(t)^2 - 2cg_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2cu_1'(t)m(t) + c^2m(t)^2 \\
&\qquad - \left[J_1^2k(t)^2 - 2J_1g_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right] dt \\
&\quad = \int_a^b -2cg_1(t)k(t) - 2c_1u_1'(t)m(t) + 2J_1g_1(t)k(t) + 2J_1u_1'(t)m(t) \\
&\qquad + c^2(k(t)^2 + m(t)^2) - J_1^2(k(t)^2 + m(t)^2)dt \\
&\quad = \int_a^b -2cg_1(t)k(t) - 2c_1u_1'(t)m(t) + 2J_1g_1(t)k(t) + 2J_1u_1'(t)m(t)dt \\
&\quad = \int_a^b 2g_1(t)k(t)(J_1 - c) + 2u_1'(t)m(t)(J_1 - c)dt \\
&\quad = 2(J_1 - c)\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt.
\end{aligned}$$

Hence, if $2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt > 0$ we see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 > J_1$ gives $G_1(\mathbf{c}) < G_1(J)$. Likewise, if $2\int_a^b g_1(t)k(t) + u_1'(t)m(t)dt < 0$ we see that for all $\mathbf{c} \in (\mathcal{L}^2[a,b])^{49}$ with $\mathbf{c}_1 < J_1$ gives $G_1(\mathbf{c}) < G_1(J)$. □

Notice that above theorems use the fact that either $J_1 \neq q - J_1$ or $q \neq 0$. The occurrence of $J_1 = q - J_1$ or $q = 0$ is unlikely, and this is reflected in our data. However, if $J_1 = q - J_1$ or if $q = 0$ and $\int_a^b k(t)^2 + m(t)^2 dt = 0$, there does not exist $c \in \mathbb{R}$ with $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T$ such that $G_1(\mathbf{c}) < G_1(J)$. Further, if $q = 0$ then there does not exist $\mathbf{c} = \begin{bmatrix} c & 0_2 & \cdots & 0_{49} \end{bmatrix}^T$ such that $G_1(\mathbf{c}) < G_1(\mathbf{0})$.

For the first case, let $J_1 = q - J_1$ and let $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T$. Then,

$$\begin{aligned} G_1(\mathbf{c}) - G_1(J) &= \int_a^b c^2k(t)^2 - 2cg_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2cu_1'(t)m(t) + c^2m(t)^2 \\ &\quad - \left[J_1^2k(t)^2 - 2J_1g_1(t)k(t) + g_1(t)^2 + u_1'(t)^2 - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right] dt \\ &= \int_a^b k(t)^2(c^2 - J_1^2) + m(t)^2(c^2 - J_1^2) \\ &\qquad - 2g_1(t)k(t)(c - J_1) - 2u_1'(t)m(t)(c - J_1)dt \\ &= \int_a^b (k(t)^2 + m(t)^2)(c^2 - J_1^2) - 2(c - J_1)(g_1(t)k(t) + u_1'(t)m(t))dt \end{aligned}$$

Now since $J_1 = q - J_1$ implies $J_1 \int_a^b k(t)^2 + m(t)^2 dt = \int_a^b g_1(t)k(t) + u_1'(t)m(t)dt$, we see that

$$\begin{aligned} G_1(\mathbf{c}) - G_1(J) &= \int_a^b (k(t)^2 + m(t)^2)(c^2 - J_1^2) - 2J_1(c - J_1)(k(t)^2 + m(t)^2)dt \\ &= \int_a^b (k(t)^2 + m(t)^2)\left[(c^2 - J_1^2) - 2J_1(c - J_1)\right] dt \\ &= \int_a^b (k(t)^2 + m(t)^2)(c - J_1)^2 dt. \end{aligned}$$

Therefore, $G_1(\mathbf{c}) - G_1(J) \geq 0$. In this case, we take $c = J_1$ to achieve $G_1(\mathbf{c}) = G_1(J)$.

Now for the second case, suppose $q = 0$ and $\int_a^b k(t)^2 + m(t)^2 dt = 0$, then

$$\begin{aligned} G_1(\mathbf{c}) - G_1(J) &= \int_a^b c^2(k(t)^2 + m(t)^2) - J_1^2(k(t)^2 + m(t)^2) \\ &\qquad + -2g_1(t)k(t)(c - J_1) - 2u_1'(t)m(t)(c - J_1)dt \\ &= 0. \end{aligned}$$

Therefore, in this case we take $c = J_1$ to achieve $G_1(\mathbf{c}) = G_1(J)$.

Lastly, suppose $q = 0$ and let $\mathbf{c} = \begin{bmatrix} c & 0_2 & \cdots & 0_{49} \end{bmatrix}^T$ where $c \in \mathbb{R}$. Then,

$$\begin{aligned} G_1(\mathbf{0}) - G_1(\mathbf{c}) &= -c^2 \int_a^b k(t)^2 + m(t)^2 dt + 2c \int_a^b g_1(t)k(t) + u_1'(t)m(t)dt \\ &= -c^2 \int_a^b k(t)^2 + m(t)^2 dt. \end{aligned}$$

Therefore, $G_1(\mathbf{0}) - G_1(\mathbf{c}) \leq 0$. In this case, we take $c = 0$ to achieve $G_1(\mathbf{c}) = G_1(\mathbf{0})$.

Given that, with our data, $J_1 \neq q - J_1$ and $q \neq 0$ for all $[a, b] \subset \mathcal{T}$, we know that $\psi \in (\mathcal{L}^2[a, b])^{49}$ for which $G(\psi) < G(\mathbf{0})$ and $G(\psi) < G(J)$ do exist. Therefore, we seek to devise a generic method for determining such ψ. But first, we must analyze the relationship between our differential equations and the accessibility of our data.

7. Different Timelines

As noted in section 2, we have five different sections in our domain, and this affects the formulation of our differential equation system (1.1). In this section, we define our delay differential equation system according to the timeline, as well as T and T^*.

Recall from section 1 that v denotes the first day of each month starting on July 1, 1994 such that $v = 1$ represents July 1, 1994. For $v < 126$ we do not have data for GLD (SPDR Gold Trust), iPhone, and buybacks. Thus, we replace any places with x_9, x_{11}, and x_{13} with 0. Hence, the differential equation (1.1) is now reformulated as follows:

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) \\
x_4'(t) &= c_8 x_1'(t) + c_9 x_4(t) + c_{10} x_2(t) x_4(t-20) + c_{11} x_8(t) x_4(t) \\
x_5'(t) &= c_{12} x_8(t) + c_{13} x_{10}(t) + c_{14} x_4(t) + c_{15} x_{12}(t) \\
x_6'(t) &= c_{16} x_6(t) + c_{17} x_7(t-17) + c_{18} x_{12}(t) \\
x_7'(t) &= c_{19} x_7(t) + c_{20} x_6(t) \\
x_8'(t) &= c_{21} x_6(t) + c_{22} x_5'(t) + c_{23} x_8(t) + c_{24} x_8(t) x_4(t) + c_{25} x_3(t) + c_{26} x_{10}'(t) \\
x_{10}'(t) &= c_{27} x_{10}(t) + c_{28} x_4(t) + c_{29} x_1(t) + c_{30} x_5(t) + c_{31} x_{12}(t) \\
x_{12}'(t) &= c_{32} x_{10}'(t) + c_{33} x_4(t) + c_{34} x_1(t)
\end{aligned}
\tag{5.46}
$$

We now define $g \in (\mathcal{L}^2[a,b])^{10}$ and $T : (\mathcal{L}^2[a,b])^{34} \to (\mathcal{L}^2[a,b])^{10}$ and $T^* : (\mathcal{L}^2[a,b])^{10} \to (\mathcal{L}^2[a,b])^{34}$.

$$g(t) = \begin{bmatrix} x_1(t) - x_1(a) \\ x_2(t) - x_2(a) \\ x_3(t) - x_3(a) \\ x_4(t) - x_4(a) \\ x_5(t) - x_5(a) \\ x_6(t) - x_6(a) \\ x_7(t) - x_7(a) \\ x_8(t) - x_8(a) \\ x_{10}(t) - x_{10}(a) \\ x_{12}(t) - x_{12}(t) \end{bmatrix} \tag{5.47}$$

$$(T\psi)(t) = \begin{bmatrix} \int_a^t \psi_1(\xi)x_2(\xi)x_1(\xi)d\xi \\ \int_a^t \psi_2(\xi)x_1(\xi) + \psi_3(\xi)x_4(\xi-20)x_2(\xi) + \psi_4(\xi)x_2(\xi)d\xi \\ \int_a^t \psi_5(\xi)x_2(\xi) + \psi_6(\xi)x_6(\xi) + \psi_7(\xi)x_8(\xi) \\ \int_a^t \psi_8(\xi)x_1'(\xi) + \psi_9(\xi)x_4(\xi) + \psi_{10}(\xi)x_2(\xi)x_4(\xi-20) + \cdots \\ \cdots + \psi_{11}(\xi)x_8(\xi)x_4(\xi)d\xi \\ \int_a^t \psi_{12}(\xi)x_8(\xi) + \psi_{13}(\xi)x_{10}(\xi) + \psi_{14}(\xi)x_4(\xi) + \psi_{15}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{16}(\xi)x_6(\xi) + \psi_{17}(\xi)x_7(\xi-17) + \psi_{18}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{19}(\xi)x_7(\xi) + \psi_{20}(\xi)x_6(\xi)d\xi \\ \int_a^t \psi_{21}(\xi)x_6(\xi) + \psi_{22}(\xi)x_5'(\xi) + \psi_{23}(\xi)x_8(\xi) + \\ \psi_{24}(\xi)x_8(\xi)x_4(\xi) + \psi_{25}(\xi)x_3(\xi) + \psi_{26}(\xi)x_{10}'(\xi)d\xi \\ \int_a^t \psi_{27}(\xi)x_{10}(\xi) + \psi_{28}(\xi)x_4(\xi) + \cdots \\ \cdots + \psi_{29}(\xi)x_1(\xi) + \psi_{30}(\xi)x_5(\xi) + \psi_{31}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{32}(\xi)x_{10}'(\xi) + \psi_{33}(\xi)x_4(\xi) + \psi_{34}(\xi)x_1(\xi)d\xi \end{bmatrix} \tag{5.48}$$

$$
(T^*\rho)(t) = \begin{bmatrix} x_1(t)x_2(t)\int_t^b \rho_1(\xi)d\xi \\ x_1(t)\int_t^b \rho_2(\xi)d\xi \\ x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_3(\xi)d\xi \\ x_6(t)\int_t^b \rho_3(\xi)d\xi \\ x_8(t)\int_t^b \rho_3(\xi)d\xi \\ x_1'(t)\int_t^b \rho_4(\xi)d\xi \\ x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)\int_t^b \rho_5(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\ x_4(t)\int_t^b \rho_5(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\ x_6(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\ \vdots \end{bmatrix} \begin{bmatrix} \\ \vdots \\ x_7(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_8(\xi)d\xi \\ x_5'(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\ x_3(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\ x_4(t)\int_t^b \rho_9(\xi)d\xi \\ x_1(t)\int_t^b \rho_9(\xi)d\xi \\ x_5(t)\int_t^b \rho_9(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{10}(\xi)d\xi \end{bmatrix} \tag{5.49}
$$

When $126 \leq v < 160$, we do not have x_{11} and x_{13}. Below is the reformulated differential equation system:

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) \\
x_4'(t) &= c_8 x_1'(t) + c_9 x_4(t) + c_{10} x_2(t) x_4(t-20) + c_{11} x_8(t) x_4(t) \\
x_5'(t) &= c_{12} x_8(t) + c_{13} x_{10}(t) + c_{14} x_4(t) + c_{15} x_{12}(t) \\
x_6'(t) &= c_{16} x_6(t) + c_{17} x_7(t-17) + c_{18} x_{12}(t) \\
x_7'(t) &= c_{19} x_7(t) + c_{20} x_6(t) \\
x_8'(t) &= c_{21} x_6(t) + c_{22} x_5'(t) + c_{23} x_8(t) + c_{24} x_8(t) x_4(t) + c_{25} x_3(t) + c_{26} x_{10}'(t) \\
x_9'(t) &= c_{27} x_{10}(t) + c_{28} x_{12}(t) + c_{29} x_2(t) + c_{30} x_9(t) \\
x_{10}'(t) &= c_{31} x_{10}(t) + c_{32} x_4(t) + c_{33} x_1(t) + c_{34} x_5(t) + c_{35} x_{12}(t) \\
x_{12}'(t) &= c_{36} x_{10}'(t) + c_{37} x_9(t) + c_{38} x_4(t) + c_{39} x_1(t)
\end{aligned} \tag{5.50}
$$

Then, we can now define $g \in (\mathcal{L}^2[a,b])^{11}$, $T : (\mathcal{L}^2[a,b])^{39} \to (\mathcal{L}^2[a,b])^{11}$, and $T^* : (\mathcal{L}^2[a,b])^{11} \to (\mathcal{L}^2[a,b])^{39}$ as follows.

$$
g(t) = \begin{bmatrix}
x_1(t) - x_1(a) \\
x_2(t) - x_2(a) \\
x_3(t) - x_3(a) \\
x_4(t) - x_4(a) \\
x_5(t) - x_5(a) \\
x_6(t) - x_6(a) \\
x_7(t) - x_7(a) \\
x_8(t) - x_8(a) \\
x_9(t) - x_9(a) \\
x_{10}(t) - x_{10}(a) \\
x_{12}(t) - x_{12}(t)
\end{bmatrix} \tag{5.51}
$$

$$
(5.52)\qquad T\psi = \begin{bmatrix}
\int_a^t \psi_1(\xi)x_2(\xi)x_1(\xi)d\xi \\
\int_a^t \psi_2(\xi)x_1(\xi) + \psi_3(\xi)x_4(\xi-20)x_2(\xi) + \psi_4(\xi)x_2(\xi)d\xi \\
\int_a^t \psi_5(\xi)x_2(\xi) + \psi_6(\xi)x_6(\xi) + \psi_7(\xi)x_8(\xi)d\xi \\
\int_a^t \psi_8(\xi)x_1'(\xi) + \psi_9(\xi)x_4(\xi) + \cdots \\
\cdots + \psi_{10}(\xi)x_2(\xi)x_4(\xi-20) + \psi_{11}(\xi)x_8(\xi)x_4(\xi)d\xi \\
\int_a^t \psi_{12}(\xi)x_8(\xi) + \psi_{13}(\xi)x_{10}(\xi) + \psi_{14}(\xi)x_4(\xi) + \psi_{15}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{16}(\xi)x_6(\xi) + \psi_{17}(\xi)x_7(\xi-17) + \psi_{18}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{19}(\xi)x_7(\xi) + \psi_{20}(\xi)x_6(\xi)d\xi \\
\int_a^t \psi_{21}(\xi)x_6(\xi) + \psi_{22}(\xi)x_5'(\xi) + \psi_{23}(\xi)x_8(\xi) + \psi_{24}(\xi)x_8(\xi)x_4(\xi) + \cdots \\
\cdots + \psi_{25}(\xi)x_3(\xi) + \psi_{26}(\xi)x_{10}'(\xi)d\xi \\
\int_a^t \psi_{27}(\xi)x_{10}(\xi) + \psi_{28}(\xi)x_{12}(\xi) + \psi_{29}(\xi)x_2(\xi) + \psi_{30}(\xi)x_9(\xi)d\xi \\
\int_a^t \psi_{31}(\xi)x_{10}(\xi) + \psi_{32}(\xi)x_4(\xi) + \psi_{33}(\xi)x_1(\xi) + \cdots \\
\cdots + \psi_{34}(\xi)x_5(\xi) + \psi_{35}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{36}(\xi)x_{10}'(\xi) + \psi_{37}(\xi)x_9(\xi) + \psi_{38}(\xi)x_4(\xi) + \psi_{39}(\xi)x_1(\xi)d\xi
\end{bmatrix}
$$

$$
(5.53)\qquad (T^*\rho)(t) = \begin{bmatrix}
x_1(t)x_2(t)\int_t^b \rho_1(\xi)d\xi \\
x_1(t)\int_t^b \rho_2(\xi)d\xi \\
x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\
x_2(t)\int_t^b \rho_2(\xi)d\xi \\
x_2(t)\int_t^b \rho_3(\xi)d\xi \\
x_6(t)\int_t^b \rho_3(\xi)d\xi \\
x_8(t)\int_t^b \rho_3(\xi)d\xi \\
x_1'(t)\int_t^b \rho_4(\xi)d\xi \\
x_4(t)\int_t^b \rho_4(\xi)d\xi \\
x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\
x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \\
x_8(t)\int_t^b \rho_5(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
x_4(t)\int_t^b \rho_5(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\
x_6(t)\int_t^b \rho_6(\xi)d\xi \\
x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\
x_7(t)\int_t^b \rho_7(\xi)d\xi \\
x_6(t)\int_t^b \rho_7(\xi)d\xi \\
x_6(t)\int_t^b \rho_8(\xi)d\xi \\
x_5'(t)\int_t^b \rho_8(\xi)d\xi \\
x_8(t)\int_t^b \rho_8(\xi)d\xi \\
x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\
x_3(t)\int_t^b \rho_8(\xi)d\xi \\
x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\
x_2(t)\int_t^b \rho_9(\xi)d\xi \\
x_9(t)\int_t^b \rho_9(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_1(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_5(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_{10}'(t)\int_t^b \rho_{11}(\xi)d\xi \\
x_9(t)\int_t^b \rho_{11}(\xi)d\xi \\
x_4(t)\int_t^b \rho_{11}(\xi)d\xi \\
x_1(t)\int_t^b \rho_{11}(\xi)d\xi
\end{bmatrix}
$$

We now consider when $160 \leq v < 223$. Recall that during this time, we do not have x_{13}. Hence, we replace the coefficients in the terms with x_{13} with zeros. Below is the reformulated differential equation system:

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) \\
x_4'(t) &= c_8 x_1'(t) + c_9 x_4(t) + c_{10} x_2(t) x_4(t-20) + c_{11} x_8(t) x_4(t) \\
x_5'(t) &= c_{12} x_8(t) + c_{13} x_{10}(t) + c_{14} x_4(t) + c_{15} x_{12}(t) \\
x_6'(t) &= c_{16} x_6(t) + c_{17} x_7(t-17) + c_{18} x_{11}(t) + c_{19} x_{12}(t) \\
x_7'(t) &= c_{20} x_7(t) + c_{21} x_6(t) + c_{22} x_{11}(t) \\
x_8'(t) &= c_{23} x_6(t) + c_{24} x_5'(t) + c_{25} x_8(t) + c_{26} x_8(t) x_4(t) \\
&\quad + c_{27} x_3(t) + c_{28} x_{10}'(t) \\
x_9'(t) &= c_{29} x_{10}(t) + c_{30} x_{12}(t) + c_{31} x_2(t) + c_{32} x_9(t) \\
x_{10}'(t) &= c_{33} x_{10}(t) + c_{34} x_4(t) + c_{35} x_1(t) + c_{36} x_5(t) + c_{37} x_{12}(t) \\
x_{11}'(t) &= c_{38} x_7(t-17) + c_{39} x_{11}(t) + c_{40} x_4(t-20) x_{11}(t) \\
x_{12}'(t) &= c_{41} x_{10}'(t) + c_{42} x_9(t) + c_{43} x_4(t) + c_{44} x_1(t)
\end{aligned}
\tag{5.54}
$$

We now define $g \in (\mathcal{L}^2[a,b])^{12}$, $T : (\mathcal{L}^2[a,b])^{44} \to (\mathcal{L}^2[a,b])^{12}$, and $T^* : (\mathcal{L}^2[a,b])^{12} \to (\mathcal{L}^2[a,b])^{44}$ as below.

$$
g = \begin{bmatrix} x_1(t) - x_1(a) \\ \vdots \\ x_{12}(t) - x_{12}(a) \end{bmatrix} \tag{5.55}
$$

$$(T\psi)(t) = \begin{bmatrix} \int_a^t \psi_1(\xi)x_2(\xi)x_1(\xi)d\xi \\ \int_a^t \psi_2(\xi)x_1(\xi) + \psi_3(\xi)x_4(\xi-20)x_2(\xi) + \psi_4(\xi)x_2(\xi)d\xi \\ \int_a^t \psi_5(\xi)x_2(\xi) + \psi_6(\xi)x_6(\xi) + \psi_7(\xi)x_8(\xi)d\xi \\ \int_a^t \psi_8(\xi)x_1'(\xi) + \psi_9(\xi)x_4(\xi) + \psi_{10}(\xi)x_2(\xi)x_4(\xi-20) + \cdots \\ \cdots + \psi_{11}(\xi)x_8(\xi)x_4(\xi)d\xi \\ \int_a^t \psi_{12}(\xi)x_8(\xi) + \psi_{13}(\xi)x_{10}(\xi) + \psi_{14}(\xi)x_4(\xi) + \psi_{15}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{16}(\xi)x_6(\xi) + \psi_{17}(\xi)x_7(\xi-17) + \cdots \\ \cdots + \psi_{18}(\xi)x_{11}(\xi) + \psi_{19}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{20}(\xi)x_7(\xi) + \psi_{21}(\xi)x_6(\xi) + \psi_{22}(\xi)x_{11}(\xi)d\xi \\ \int_a^t \psi_{23}(\xi)x_6(\xi) + \psi_{24}(\xi)x_5'(\xi) + \psi_{25}(\xi)x_8(\xi) + \cdots \\ \cdots + \psi_{26}(\xi)x_8(\xi)x_4(\xi) + \psi_{27}(\xi)x_3(\xi) + \psi_{28}(\xi)x_{10}'(\xi)d\xi \\ \int_a^t \psi_{29}(\xi)x_{10}(\xi) + \psi_{30}(\xi)x_{12}(\xi) + \psi_{31}(\xi)x_2(\xi) + \psi_{32}(\xi)x_9(\xi) \\ \int_a^t \psi_{33}(\xi)x_{10}(\xi) + \psi_{34}(\xi)x_4(\xi) + \psi_{35}(\xi)x_1(\xi) + \cdots \\ \cdots + \psi_{36}(\xi)x_5(\xi) + \psi_{37}(\xi)x_{12}(\xi)d\xi \\ \int_a^t \psi_{38}(\xi)x_7(\xi-17) + \psi_{39}(\xi)x_{11}(\xi) + \psi_{40}(\xi)x_4(\xi-20)x_{11}(\xi)d\xi \\ \int_a^t \psi_{41}(\xi)x_{10}'(\xi) + \psi_{42}(\xi)x_9(\xi) + \psi_{43}(\xi)x_4(\xi) + \psi_{44}(\xi)x_1(\xi)d\xi \end{bmatrix} \tag{5.56}$$

$$(T^*\rho)(t) = \begin{bmatrix} x_2(t)x_1(t)\int_t^b \rho_1(\xi)d\xi \\ x_1(t)\int_t^b \rho_2(\xi)d\xi \\ x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_2(\xi)d\xi \\ x_2(t)\int_t^b \rho_3(\xi)d\xi \\ x_6(t)\int_t^b \rho_3(\xi)d\xi \\ x_8(t)\int_t^b \rho_3(\xi)d\xi \\ x_1'(t)\int_t^b \rho_4(\xi)d\xi \\ x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \\ x_8(t)\int_t^b \rho_5(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\ x_4(t)\int_t^b \rho_5(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x_6(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_6(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\ x_7(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_7(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_7(\xi)d\xi \\ x_6(t)\int_t^b \rho_8(\xi)d\xi \\ x_5'(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)\int_t^b \rho_8(\xi)d\xi \\ x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\ x_3(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\ x_2(t)\int_t^b \rho_9(\xi)d\xi \\ x_9(t)\int_t^b \rho_9(\xi)d\xi \\ x_{10}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_5(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_{12}(t)\int_t^b \rho_{10}(\xi)d\xi \\ x_7(t-17)\int_t^b \rho_{11}(\xi)d\xi \\ x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_4(t-20)x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\ x_{10}'(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_9(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_4(t)\int_t^b \rho_{12}(\xi)d\xi \\ x_1(t)\int_t^b \rho_{12}(\xi)d\xi \end{bmatrix} \tag{5.57}$$

Lastly, we consider when $223 \leq v < 240$. Here, x_{13} is present in x_3 but not in x_8.

$$
\begin{aligned}
x_1'(t) &= c_1 x_2(t) x_1(t) \\
x_2'(t) &= c_2 x_1(t) + c_3 x_4(t-20) x_2(t) + c_4 x_2(t) \\
x_3'(t) &= c_5 x_2(t) + c_6 x_6(t) + c_7 x_8(t) + c_8 x_{13}(t) \\
x_4'(t) &= c_9 x_1'(t) + c_{10} x_4(t) + c_{11} x_2(t) x_4(t-20) + c_{12} x_8(t) x_4(t) \\
x_5'(t) &= c_{13} x_8(t) + c_{14} x_{10}(t) + c_{15} x_4(t) + c_{16} x_{12}(t) \\
x_6'(t) &= c_{17} x_6(t) + c_{18} x_7(t-17) + c_{19} x_{11}(t) + c_{20} x_{12}(t) \\
x_7'(t) &= c_{21} x_7(t) + c_{22} x_6(t) + c_{23} x_{11}(t) \\
x_8'(t) &= c_{24} x_6(t) + c_{25} x_5'(t) + c_{26} x_8(t) \\
&\quad + c_{27} x_8(t) x_4(t) + c_{28} x_3(t) + c_{29} x_{10}'(t) \\
x_9'(t) &= c_{30} x_{10}(t) + c_{31} x_{12}(t) + c_{32} x_2(t) + c_{33} x_9(t) \\
x_{10}'(t) &= c_{34} x_{10}(t) + c_{35} x_4(t) + c_{36} x_1(t) + c_{37} x_5(t) + c_{38} x_{12}(t) \\
x_{11}'(t) &= c_{39} x_7(t-17) + c_{40} x_{11}(t) + c_{41} x_4(t-20) x_{11}(t) \\
x_{12}'(t) &= c_{42} x_{10}'(t) + c_{43} x_9(t) + c_{44} x_4(t) + c_{45} x_1(t) \\
x_{13}'(t) &= c_{46} x_{10}(t) + c_{47} x_6(t) + c_{48} x_7(t)
\end{aligned}
\tag{5.58}
$$

We now define $g \in (\mathcal{L}^2[a,b])^{13}$, $T : (\mathcal{L}^2[a,b])^{48} \to (\mathcal{L}^2[a,b])^{13}$, and $T^* : (\mathcal{L}^2[a,b])^{13} \to (\mathcal{L}^2[a,b])^{48}$.

$$
g(t) = \begin{bmatrix} x_1(t) - x_1(a) \\ \vdots \\ x_{13}(t) - x_{13}(a) \end{bmatrix} \tag{5.59}
$$

$$
(T\psi)(t) = \begin{bmatrix}
\int_a^t \psi_1(\xi)x_2(\xi)x_1(\xi)d\xi \\
\int_a^t \psi_2(\xi)x_1(\xi) + \psi_3(\xi)x_4(\xi-20)x_2(\xi) + \psi_4(\xi)x_2(\xi)d\xi \\
\int_a^t \psi_5(\xi)x_2(\xi) + \psi_6(\xi)x_6(\xi) + \psi_7(\xi)x_8(\xi) + \psi_8(\xi)x_{13}(\xi)d\xi \\
\int_a^t \psi_9(\xi)x_1'(\xi) + \psi_{10}(\xi)x_4(\xi) + \psi_{11}(\xi)x_2(\xi)x_4(\xi-20) + \cdots \\
\cdots + \psi_{12}(\xi)x_8(\xi)x_4(\xi)d\xi \\
\int_a^t \psi_{13}(\xi)x_8(\xi) + \psi_{14}(\xi)x_{10}(\xi) + \psi_{15}(\xi)x_4(\xi) + \psi_{16}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{17}(\xi)x_6(\xi) + \psi_{18}(\xi)x_7(\xi-17) + \cdots \\
\cdots + \psi_{19}(\xi)x_{11}(\xi) + \psi_{20}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{21}(\xi)x_7(\xi) + \psi_{22}(\xi)x_6(\xi) + \psi_{23}(\xi)x_{11}(\xi)d\xi \\
\int_a^t \psi_{24}(\xi)x_6(\xi) + \psi_{25}(\xi)x_5'(\xi) + \psi_{26}(\xi)x_8(\xi) + \psi_{27}(\xi)x_8(\xi)x_4(\xi) \\
+\psi_{28}(\xi)x_3(\xi) + \psi_{29}(\xi)x_{10}'(\xi)d\xi \\
\int_a^t \psi_{30}(\xi)x_{10}(\xi) + \psi_{31}(\xi)x_{12}(\xi) + \psi_{32}(\xi)x_2(\xi) + \psi_{33}(\xi)x_9(\xi)d\xi \\
\int_a^t \psi_{34}(\xi)x_{10}(\xi) + \psi_{35}(\xi)x_4(\xi) + \psi_{36}(\xi)x_1(\xi) + \cdots \\
\cdots + \psi_{37}(\xi)x_5(\xi) + \psi_{38}(\xi)x_{12}(\xi)d\xi \\
\int_a^t \psi_{39}(\xi)x_7(\xi-17) + \psi_{40}(\xi)x_{11}(\xi) + \psi_{41}(\xi)x_4(\xi-20)x_{11}(\xi)d\xi \\
\int_a^t \psi_{42}(\xi)x_{10}'(\xi) + \psi_{43}(\xi)x_9(\xi) + \psi_{44}(\xi)x_4(\xi) + \psi_{45}(\xi)x_1(\xi)d\xi \\
\int_a^t \psi_{46}(\xi)x_{10}(\xi) + \psi_{47}(\xi)x_6(\xi) + \psi_{48}(\xi)x_7(\xi)d\xi
\end{bmatrix} \tag{5.60}
$$

$$
(T^*\rho)(t) = \begin{bmatrix}
x_1(t)x_2(t)\int_t^b \rho_1(\xi)d\xi \\
x_1(t)\int_t^b \rho_2(\xi)d\xi \\
x_4(t-20)x_2(t)\int_t^b \rho_2(\xi)d\xi \\
x_2(t)\int_t^b \rho_2(\xi)d\xi \\
x_2(t)\int_t^b \rho_3(\xi)d\xi \\
x_6(t)\int_t^b \rho_3(\xi)d\xi \\
x_8(t)\int_t^b \rho_3(\xi)d\xi \\
x_{13}(t)\int_t^b \rho_3(\xi)d\xi \\
x_1'(t)\int_t^b \rho_4(\xi)d\xi \\
x_4(t)\int_t^b \rho_4(\xi)d\xi \\
x_2(t)x_4(t-20)\int_t^b \rho_4(\xi)d\xi \\
x_8(t)x_4(t)\int_t^b \rho_4(\xi)d\xi \\
x_8(t)\int_t^b \rho_5(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_5(\xi)d\xi \\
x_4(t)\int_t^b \rho_5(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_5(\xi)d\xi \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
x_6(t)\int_t^b \rho_6(\xi)d\xi \\
x_7(t-17)\int_t^b \rho_6(\xi)d\xi \\
x_{11}(t)\int_t^b \rho_6(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_6(\xi)d\xi \\
x_7(t)\int_t^b \rho_7(\xi)d\xi \\
x_6(t)\int_t^b \rho_7(\xi)d\xi \\
x_{11}(t)\int_t^b \rho_7(\xi)d\xi \\
x_6(t)\int_t^b \rho_8(\xi)d\xi \\
x_5'(t)\int_t^b \rho_8(\xi)d\xi \\
x_8(t)\int_t^b \rho_8(\xi)d\xi \\
x_8(t)x_4(t)\int_t^b \rho_8(\xi)d\xi \\
x_3(t)\int_t^b \rho_8(\xi)d\xi \\
x_{10}'(t)\int_t^b \rho_8(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_9(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_9(\xi)d\xi \\
x_2(t)\int_t^b \rho_9(\xi)d\xi \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
x_9(t)\int_t^b \rho_9(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_4(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_1(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_5(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_{12}(t)\int_t^b \rho_{10}(\xi)d\xi \\
x_7(t-17)\int_t^b \rho_{11}(\xi)d\xi \\
x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\
x_4(t-20)x_{11}(t)\int_t^b \rho_{11}(\xi)d\xi \\
x_{10}'(t)\int_t^b \rho_{12}(\xi)d\xi \\
x_9(t)\int_t^b \rho_{12}(\xi)d\xi \\
x_4(t)\int_t^b \rho_{12}(\xi)d\xi \\
x_1(t)\int_t^b \rho_{12}(\xi)d\xi \\
x_{10}(t)\int_t^b \rho_{13}(\xi)d\xi \\
x_6(t)\int_t^b \rho_{13}(\xi)d\xi \\
x_7(t)\int_t^b \rho_{13}(\xi)d\xi
\end{bmatrix} \tag{5.61}
$$

Remark 5.12. It should be noted that each T presented above is compact and $\dim(\mathcal{R}(T)) = \infty$; the proof is similar to that for theorems 5.4 and 5.3. Therefore, for each of these cases, we know that the inverse problem we have is ill-posed and we must formulate an iterative variational regularization method.

8. Strategies for Finding a Starting Point

In this section, we discuss an algorithmic approach to finding starting points. Our approach is dependent on the number of equations in our delay differential system as well as the number of terms in each equation. Mindful that the number of terms on the right hand side of our differential equation system varies, we categorize them into three different groups: one term, two terms, or three terms and above on the right hand side. The methods introduced in this section are used to build our algorithm **"Get_Constants_Final.m"** in MATLAB, and the pseudocode is available in Chapter 7.

Remark 5.13. Consider

$$x_5'(t) = c_{13}x_8(t) + c_{14}x_{10}(t) + c_{15}x_4(t) + c_{16}x_{12}(t). \tag{5.62}$$

We say that the term on the right hand side of (5.62) corresponding to the nth coefficient is the term that effects G_5 the least if

$$n = \underset{13\leq i\leq 16}{\arg\min} \left|G_5(\mathbf{0}) - G_5(0.1e_i)\right|,$$

where $\{e_i\}$ are the standard basis for $\mathbb{R}^{49}$. A similar concept applies to each of the other equations in our differential equation system.

Below is the general strategy to find our initial guess ψ_0:

(1) Determine which three terms effect $G_i(\psi)$ the least for all $1 \leq i \leq 13$.

(2) Set the rest of the coefficients to zero if $G_i(\mathbf{0}) < G_i(J)$ and set the rest of the coefficients to corresponding components of J if $G_i(J) \leq G_i(\mathbf{0})$.

(3) If $G_i(\mathbf{0}) < G_i(J)$, choose two of the coefficients from (1) and assign values in the interval $[-c, c]$ where $c \in (0, 3]$. If $G_i(J) \leq G_i(\mathbf{0})$, set the coefficients chosen to values in the interval $[e_j^T J - c, e_j^T J + c]$, where $c \in (0, 3]$ and $j \in [1, 49]$ corresponds to the components chosen. Then we find the third coefficient.

8.1. One Term. The first case we consider is when our system has an equation, such as $x_1'(t) = c_1 x_1(t) x_2(t)$, with only one term on the right hand side. In this case, we cannot use the general strategy described above. This subsection describes how to obtain a better initial guess in this situation. Keep in mind that our goal is to find ψ_0 such that $G_1(\psi_0) < G_1(J)$ and $G_1(\psi_0) < G_1(\mathbf{0})$.

Consider first the case when $G_1(\mathbf{0}) < G_1(J)$. Then, let $\mathbf{c} = ce_1 \in (\mathcal{L}^2[a, b])^{49}$ where $c \in \mathbb{R}$. We desire to find $c \in \mathbb{R}$ such that $G_1(\mathbf{c}) < G_1(\mathbf{0})$. Recall the following definitions from Section 6.1:

$$\begin{aligned}
G_1(\psi) &= \|T_1\psi - g_1\|_{\mathcal{L}^2}^2 + \left\|u_1' - (u_\psi')_1\right\|_{\mathcal{L}^2}^2, \\
k(t) &= \int_a^t x_1(s)x_2(s)ds, \\
m(t) &= \frac{1}{b-a}\int_a^b \int_a^\eta k(s)ds d\eta - \int_a^t k(s)ds, \\
T_1\psi(t) &= \psi_1(t)k(t), \\
(u_\psi')_1(t) &= \psi_1(t)m(t).
\end{aligned}$$

Consider

$$\begin{aligned}
G_1(\mathbf{c}) - G_1(\mathbf{0}) &= \|T_1\mathbf{c} - g_1\|_{\mathcal{L}^2}^2 + \|u_1' - (u_\mathbf{c}')_1\|_{\mathcal{L}^2}^2 - \|g_1\|_{\mathcal{L}^2}^2 - \|u_1'\|_{\mathcal{L}^2}^2 \\
&= \|ck - g_1\|_{\mathcal{L}^2}^2 + \|u_1' - cm\|_{\mathcal{L}^2}^2 - \|g_1\|_{\mathcal{L}^2}^2 - \|u_1'\|_{\mathcal{L}^2}^2 \\
&= \int_a^b c^2k(t)^2 - 2ck(t)g_1(t) + g_1(t)^2 + u_1'(t)^2 - 2cu_1'(t)m(t) + c^2m(t)^2 \\
&\qquad\qquad - g_1(t)^2 - u_1'(t)^2 dt \\
&= c^2\int_a^b k(t)^2 + m(t)^2 dt + c\int_a^b -2k(t)g_1(t) - 2u_1'(t)m(t)dt.
\end{aligned}$$

Now, we want to find c so that $G_1(\mathbf{c}) - G_1(\mathbf{0}) < 0$. Hence, $c \neq 0$ and for such a c we require

$$c^2 \int_a^b k(t)^2 + m(t)^2 dt + c \int_a^b -2k(t)g_1(t) - 2u_1'(t)m(t)dt < 0.$$

In other words,

$$c^2 \int_a^b k(t)^2 + m(t)^2 dt < 2c \int_a^b k(t)g_1(t) + u_1'(t)m(t)dt.$$

Note that if $\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt = 0$, then no such c exists and we set $c = 0$ to achieve $G_1(\mathbf{c}) = G_1(\mathbf{0})$.

First, suppose $\int_a^b k(t)^2 + m(t)^2 dt = 0$ and $\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt \neq 0$. Then, we choose c so that

$$0 < 2c \int_a^b k(t)g_1(t) + u_1'(t)m(t)dt.$$

Now suppose $\int_a^b k(t)^2 + m(t)^2 dt > 0$. If $\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt > 0$, then choose c which satisfy

$$0 < c < \frac{2\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt}.$$

and if $\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt < 0$, then choose c which satisfy

$$0 > c > \frac{2\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt}.$$

Consider now the case when $G_1(J) < G_1(\mathbf{0})$ and let $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T$. We now desire to find $c \in \mathbb{R}$ such that $G_1(\mathbf{c}) < G_1(J)$.

$$\begin{aligned} G_1(\mathbf{c}) - G_1(J) &= \|T_1\mathbf{c} - g_1\|_{\mathcal{L}^2}^2 + \|u_1' - (u_{\mathbf{c}}')_1\|_{\mathcal{L}^2}^2 - \|T_1 J - g_1\|_{\mathcal{L}^2}^2 - \|u_1' - (u_J')_1\|_{\mathcal{L}^2}^2 \\ &= \|ck - g_1\|_{\mathcal{L}^2}^2 + \|u_1' - cm\|_{\mathcal{L}^2}^2 - \|J_1 k - g_1\|_{\mathcal{L}^2}^2 - \|u_1' - J_1 m\|_{\mathcal{L}^2}^2 \end{aligned}$$

$$= \int_a^b c^2k(t)^2 - 2ck(t)g_1(t) + g_1(t)^2 + u_1'(t)^2 - 2cu_1'(t)m(t) + c^2m(t)^2$$
$$- \left[J_1^2k(t)^2 - 2J_1k(t)g_1(t) + g_1(t)^2 + u_1'(t)^2 - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right] dt$$
$$= c^2 \int_a^b k(t)^2 + m(t)^2 dt + c \int_a^b -2k(t)g_1(t) - 2u_1'(t)m(t)dt$$
$$+ \int_a^b - \left[J_1^2k(t)^2 - 2J_1k(t)g_1(t) - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right] dt$$

Let $B_1 = \int_a^b k(t)^2 + m(t)^2 dt$, $B_2 = \int_a^b -2k(t)g_1(t) - 2u_1'(t)m(t)dt$, and

$$B_3 = \int_a^b - \left[J_1^2k(t)^2 - 2J_1k(t)g_1(t) - 2J_1u_1'(t)m(t) + J_1^2m(t)^2\right] dt.$$

Then,

$$G_1(\mathbf{c}) - G_1(J) = B_1c^2 + B_2c + B_3 \tag{5.63}$$

where the constants $B_1, B_2, B_3 \in \mathbb{R}$ are all known.

First, suppose $B_1 > 0$. In this case, in order for $G_1(\mathbf{c}) - G_1(J) < 0$, the discriminant of (5.63), $B_2^2 - 4B_1B_3$, must be greater than zero. If $B_2^2 - 4B_1B_3 \leq 0$, then we let $c = J_1$ to obtain $G_1(\mathbf{c}) = G_1(J)$, and we rely on finding $G_i(\mathbf{c}) < G_i(J)$ for $2 \leq i \leq 13$ so that $G(\mathbf{c}) < G(J)$. Now, if $B_2^2 - 4B_1B_3 > 0$, let α_1 and α_2 be the roots of (5.63) such that $\alpha_1 < \alpha_2$. Then, we choose $c \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. If $B_2 \neq 0$, then we choose $c < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c = J_1$.

In summary,

Case 1: $G_1(\mathbf{0}) < G_1(J)$. Let $\mathbf{c} = \begin{bmatrix} c & 0_2 & \cdots & 0_{49} \end{bmatrix}^T$ where $c \in \mathbb{R}$.

Sub-case 1: If $\int_a^b k(t) + g_1(t) + u_1'(t)m(t)dt = 0$ then let $c = 0$.

Sub-case 2: Suppose $\int_a^b k(t) + g_1(t) + u_1'(t)m(t)dt \neq 0$ and $\int_a^b k(t)^2 + m(t)^2 dt = 0$.
Then, choose c so that $0 < 2c \int_a^b k(t)g_1(t) + u_1'(t)m(t)dt$.

Sub-case 3: Suppose $\int_a^b k(t) + g_1(t) + u_1'(t)m(t)dt > 0$ and $\int_a^b k(t)^2 + m(t)^2 dt > 0$. Then choose c to satisfy

$$0 < c < \frac{2\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt}.$$

Sub-case 4: Suppose $\int_a^b k(t) + g_1(t) + u_1'(t)m(t)dt > 0$ and $\int_a^b k(t)^2 + m(t)^2 dt < 0$. Then choose c to satisfy

$$0 > c > \frac{2\int_a^b k(t)g_1(t) + u_1'(t)m(t)dt}{\int_a^b k(t)^2 + m(t)^2 dt}.$$

Case 2: $G_1(J) < G(\mathbf{0})$. Let $\mathbf{c} = \begin{bmatrix} c & J_2 & \cdots & J_{49} \end{bmatrix}^T$ where $c \in \mathbb{R}$.
Let $B_1 = \int_a^b k(t)^2 + m(t)^2 dt$, $B_2 = \int_a^b -2k(t)g_1(t) - 2u_1'(t)m(t)dt$, and

$$B_3 = \int_a^b - \left[J_1^2 k(t)^2 - 2J_1 k(t)g_1(t) - 2J_1 u_1'(t)m(t) + J_1^2 m(t)^2\right] dt.$$

Sub-case 1: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 \leq 0$ then $c = J_1$.

Sub-case 2: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 > 0$ then let α_1 and α_2 be the roots such that $\alpha_1 < \alpha_2$. Choose $c \in (\alpha_1, \alpha_2)$.

Sub-case 3: If $B_1 = 0$ and $B_2 \neq 0$, then we choose $c < \frac{-B_3}{B_2}$.

Sub-case 4: If $B_1 = B_2 = 0$ and $B_3 < 0$ then choose any $c \in \mathbb{R}$.

Sub-case 5: If $B_1 = B_2 = 0$ and $B_3 > 0$ then $c = J_1$.

8.2. Two Terms. For $v < 159$, our system has $x_7'(t) = \psi_1 x_7(t) + \psi_2 x_6(t)$ and we are also unable to employ the general strategy here. Keep in mind that we are looking for two distinct coefficients. For situations like these, we employ the next method:

(1) If $G_i(\mathbf{0}) < G_i(J)$, set one of the coefficients as a value in $[-z, z]$ where $z \in (0, 3]$. If $G_i(J) \leq G_i(\mathbf{0})$, set one of the the coefficients as a value in $[e_j^T J - z, e_j^T J + z]$ where $z \in (0, 3]$ and $j \in [1, 49]$ is an index corresponding to the component chosen.

(2) Solve for the other coefficient.

(3) Repeat this process by setting the other coefficient as a constant.

Here, we detail the process of solving for the other coefficient using the notations developed below.

$$k_1 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \int_a^t x_7(s)ds$$

$$k_2 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \int_a^t x_6(s)ds$$

$$m_1 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \frac{1}{b-a}\int_a^b \int_a^\eta k_1(s)dsd\eta - \int_a^t k_1(s)ds$$

$$m_2 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \frac{1}{b-a}\int_a^b \int_a^\eta k_2(s)dsd\eta - \int_a^t k_2(s)ds$$

Now, let $\omega \in (\mathcal{L}^2[a,b])^{49}$ be a vector of constant functions. Then,

$$T_7\omega(t) = \int_a^t M_{(7,:)}(\xi)\omega(\xi)d\xi = \int_a^t \omega_{21}x_7(\xi) + \omega_{22}x_6(\xi)d\xi = \omega_{21}k_1(t) + \omega_{22}k_2(t)$$

$$(u'_\omega)_7(t) = \frac{1}{b-a}\int_a^b \int_a^\eta T_7\omega(\xi)d\xi d\eta - \int_a^t T_7\omega(\xi)d\xi = \omega_{21}m_1(t) + \omega_{22}m_2(t).$$

Using these notations, we can write G_7 as follows:

$$\begin{aligned} G_7(\psi) &= \|\psi_{21}k_1 + \psi_{22}k_2 - g_7\|^2_{\mathcal{L}^2} + \|\psi_{21}m_1 + \psi_{22}m_2 - u'_7\|^2_{\mathcal{L}^2} \\ &= \int_a^b \psi_{21}^2k_1^2 + \psi_{22}^2k_2^2 + g_7^2 + 2\psi_{21}\psi_{22}k_1k_2 - 2\psi_{21}k_1g_7 - 2\psi_{22}k_2g_7 \\ &\quad + \psi_{21}^2m_1^2 + \psi_{22}^2m_2^2 + (u'_7)^2 + 2\psi_{21}\psi_{22}m_1m_2 - 2\psi_{21}m_1(u'_7) - 2\psi_{22}m_2(u'_7). \end{aligned}$$

Consider first the case $G_7(\mathbf{0}) < G_7(J)$ and let

$$\mathbf{c} = \begin{bmatrix} 0_1 & \cdots & 0_{20} & c_1 & c_2 & 0_{23} & \cdots & 0_{49} \end{bmatrix}^T$$

where $c_1, c_2 \in \mathbb{R}$. We desire to find c_1, c_2 such that $G_7(\mathbf{c}) < G_7(\mathbf{0})$. First, we choose $c_2 \in [-z, z]$ where $z \in (0, 3]$ and we solve for c_1.

$$\begin{aligned} G_7(\mathbf{c}) - G_7(\mathbf{0}) = \int_a^b & c_1^2k_1^2 + c_2^2k_2^2 + g_7^2 + 2c_1c_2k_1k_2 - 2c_1k_1g_7 - 2c_2k_2g_7 \\ & + c_1^2m_1^2 + c_2^2m_2^2 + (u_7')^2 + 2c_1c_2m_1m_2 - 2c_1m_1(u_7') - 2c_2m_2(u_7') \\ & - g_7^2 - (u_7')^2 dt \\ = c_1^2 \int_a^b & k_1^2 + m_1^2 dt + c_1 \int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt \\ & + \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7')dt. \end{aligned}$$

Let $B_1 = \int_a^b k_1^2 + m_1^2 dt$, $B_2 = \int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt$ and $B_3 = \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7')dt$. Then,

$$G_7(\mathbf{c}) - G_7(\mathbf{0}) = B_1c_1^2 + B_2c_1 + B_3. \tag{5.64}$$

Similar to the discussion in the one-term case, first suppose $B_1 > 0$. In this case, if $B_2^2 - 4B_1B_3 \leq 0$, then we set $c_1 = 0$ and if $B_2^2 - 4B_1B_3 > 0$, we let α_1 and α_2 be the roots of (5.64) such that $\alpha_1 < \alpha_2$. Then, we choose $c_1 \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. In this case, if $B_2 \neq 0$, then choose $c_1 < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c_1 = 0$. We now choose $c_1 \in [-z, z]$ where $z \in (0, 3]$ and we solve for c_2 using the similar process as described above.

Now consider the case $G_7(J) < G_7(\mathbf{0})$ and let

$$\mathbf{c} = \begin{bmatrix} J_1 & \cdots & J_{20} & c_1 & c_2 & J_{23} & \cdots & J_{49} \end{bmatrix}$$

where $c_1, c_2 \in \mathbb{R}$. We desire to find c_1, c_2 such that $G_7(\mathbf{c}) < G_7(J)$. First, we choose $c_2 \in [J_{22} - z, J_{22} + z]$ where $z \in (0, 3]$ and we solve for c_1.

$$
\begin{aligned}
G_7(\mathbf{c}) - G_7(J) = \int_a^b\ & c_1^2k_1^2 + c_2^2k_2^2 + g_7^2 + 2c_1c_2k_1k_2 - 2c_1k_1g_7 - 2c_2k_2g_7 \\
& + c_1^2m_1^2 + c_2^2m_2^2 + (u_7')^2 + 2c_1c_2m_1m_2 - 2c_1m_1(u_7') - 2c_2m_2(u_7') \\
& - \Big[J_{21}^2k_1^2 + J_{22}^2k_2^2 + g_7^2 + 2J_{21}J_{22}k_1k_2 - 2J_{21}k_1g_7 - 2J_{22}k_2g_7 \\
& + J_{21}^2m_1^2 + J_{22}^2m_2^2 + (u_7')^2 + 2J_{21}J_{22}m_1m_2 - 2J_{21}m_1(u_7') - 2J_{22}m_2(u_7')\Big]dt \\
= c_1^2 & \int_a^b k_1^2 + m_1^2dt + c_1\int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt \\
+ & \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7') \\
& - \Big[J_{21}^2k_1^2 + J_{22}^2k_2^2 + 2J_{21}J_{22}k_1k_2 - 2J_{21}k_1g_7 - 2J_{22}k_2g_7 \\
& + J_{21}^2m_1^2 + J_{22}^2m_2^2 + 2J_{21}J_{22}m_1m_2 - 2J_{21}m_1(u_7') - 2J_{22}m_2(u_7')\Big]dt
\end{aligned}
$$

Let $B_1 = \int_a^b k_1^2 + m_1^2dt$, $B_2 = \int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt$, and $B_3 = \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7') - \Big[J_{21}^2k_1^2 + J_{22}^2k_2^2 + 2J_{21}J_{22}k_1k_2 - 2J_{21}k_1g_7 - 2J_{22}k_2g_7 + J_{21}^2m_1^2 + J_{22}^2m_2^2 + 2J_{21}J_{22}m_1m_2 - 2J_{21}m_1(u_7') - 2J_{22}m_2(u_7')\Big]dt$.

Similar to above, first suppose $B_1 > 0$. In this case, if $B_2^2 - 4B_1B_3 \leq 0$, then we set $c_1 = J_{21}$ and if $B_2^2 - 4B_1B_3 > 0$, let α_1 and α_2 be the roots of the quadratic such that $\alpha_1 < \alpha_2$. Then, we choose $c_1 \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. In this case, if $B_2 \neq 0$, then choose $c_1 < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c_1 \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c_1 = J_{21}$. Repeat this process by choosing $c_1 \in [J_{21} - z, J_{21} + z]$ where $z \in (0, 3]$ and solve for c_2.

In summary,

Case 1: $G_7(\mathbf{0}) < G_7(J)$. Let $\mathbf{c} = \begin{bmatrix} 0_1 & \cdots & 0_{20} & c_1 & c_2 & 0_{23} & \cdots & 0_{49} \end{bmatrix}^T$ where $c_1, c_2 \in \mathbb{R}$. Choose $c_2 \in [-z, z]$ where $z \in (0, 3]$. Let $B_1 = \int_a^b k_1^2 + m_1^2dt$, $B_2 = \int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt$ and $B_3 = \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7')dt$.

Sub-case 1: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 \leq 0$ then $c_1 = 0$.

Sub-case 2: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 > 0$ then let α_1 and α_2 be the roots such that $\alpha_1 < \alpha_2$. Choose $c_1 \in (\alpha_1, \alpha_2)$.

Sub-case 3: If $B_1 = 0$ and $B_2 \neq 0$, then we choose $c_1 < \frac{-B_3}{B_2}$.

Sub-case 4: If $B_1 = B_2 = 0$ and $B_3 < 0$ then choose any $c_1 \in \mathbb{R}$.

Sub-case 5: If $B_1 = B_2 = 0$ and $B_3 > 0$ then $c_1 = 0$.

Repeat this process by choosing $c_1 \in [-z, z]$ and solve for c_2.

Case 2: $G_7(J) < G_7(\mathbf{0})$. Let $\mathbf{c} = \begin{bmatrix} J_1 & \cdots & J_{20} & c_1 & c_2 & J_{23} & \cdots & J_{49} \end{bmatrix}^T$ where $c_1, c_2 \in \mathbb{R}$. Choose $c_2 \in [J_{22} - z, J_{22} + z]$ where $z \in (0, 3]$. Let $B_1 = \int_a^b k_1^2 + m_1^2 dt$, $B_2 = \int_a^b 2c_2k_1k_2 - 2k_1g_7 + 2c_2m_1m_2 - 2m_1(u_7')dt$, and $B_3 = \int_a^b c_2^2k_2^2 - 2c_2k_2g_7 + c_2^2m_2^2 - 2c_2m_2(u_7') - \Big[J_{21}^2k_1^2 + J_{22}^2k_2^2 + 2J_{21}J_{22}k_1k_2 - 2J_{21}k_1g_7 - 2J_{22}k_2g_7 + J_{21}^2m_1^2 + J_{22}^2m_2^2 + 2J_{21}J_{22}m_1m_2 - 2J_{21}m_1(u_7') - 2J_{22}m_2(u_7')\Big]dt$.

Sub-case 1: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 \leq 0$ then $c_1 = J_{21}$.

Sub-case 2: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 > 0$ then let α_1 and α_2 be the roots such that $\alpha_1 < \alpha_2$. Choose $c_1 \in (\alpha_1, \alpha_2)$.

Sub-case 3: If $B_1 = 0$ and $B_2 \neq 0$, then we choose $c_1 < \frac{-B_3}{B_2}$.

Sub-case 4: If $B_1 = B_2 = 0$ and $B_3 < 0$ then choose any $c_1 \in \mathbb{R}$.

Sub-case 5: If $B_1 = B_2 = 0$ and $B_3 > 0$ then $c_1 = J_{21}$.

Repeat this process by choosing $c_1 \in [J_{21} - z, J_{21} + z]$ and solve for c_2.

8.3. Three Terms or Above. If an equation from our system has three or more terms, we can use the general strategy outlined below. In this section, we choose to work with $x_5'(t) = \psi_{13}x_8(t) + \psi_{14}x_{10}(t) + \psi_{15}x_4(t) + \psi_{16}x_{12}(t)$ as an example, noting that the method can be applied to the rest of the equations as needed.

We first define k_i and m_i for $1 \leq i \leq 4$ as we have above. Notice that the number of k_i and m_i is equal to the number of terms in the equation.

$$k_1 : [a, b] \to \mathcal{L}^2[a, b] : t \mapsto \int_a^t x_8(s)ds$$

$$k_2 : [a, b] \to \mathcal{L}^2[a, b] : t \mapsto \int_a^t x_{10}(s)ds$$

$$k_3 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \int_a^t x_4(s)ds$$

$$k_4 : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \int_a^t x_{12}(s)ds$$

$$m_i : [a,b] \to \mathcal{L}^2[a,b] : t \mapsto \frac{1}{b-a}\int_a^b \int_a^\eta k_i(s)dsd\eta - \int_a^t k_i(s)ds$$

Now, let $\omega \in (\mathcal{L}^2[a,b])^{49}$ be a vector of constant functions. Then,

$$T_5\omega(t) = \int_a^t M_{(5,:)}(\xi)\omega(\xi)d\xi = \omega_{13}k_1(t) + \omega_{14}k_2(t) + \omega_{15}k_3(t) + \omega_{16}k_4(t)$$

$$\begin{aligned}(u'_\omega)_5(t) &= \frac{1}{b-a}\int_a^b \int_a^\eta T_5\omega(\xi)d\xi d\eta - \int_a^t T_5\omega(\xi)d\xi \\ &= \omega_{13}m_1(t) + \omega_{14}m_2(t) + \omega_{15}m_3(t) + \omega_{16}m_4(t)\end{aligned}$$

Using these notations, we have

$$\begin{aligned}G_5(\psi) &= \|T_5\psi - g_5\|^2_{\mathcal{L}^2} + \left\|u'_5 - (u'_\psi)_5\right\|^2_{\mathcal{L}^2} \\ &= \int_a^b \psi_{13}^2k_1^2 + \psi_{14}^2k_2^2 + \psi_{15}^2k_3^2 + \psi_{16}^2k_4^2 + g_5^2 + 2\psi_{13}\psi_{14}k_1k_2 + 2\psi_{13}\psi_{15}k_1k_3 \\ &+ 2\psi_{13}\psi_{16}k_1k_4 - 2\psi_{13}k_1g_5 + 2\psi_{14}\psi_{15}k_2k_3 + 2\psi_{14}\psi_{16}k_2k_4 - 2\psi_{14}k_2g_5 \\ &+ 2\psi_{15}\psi_{16}k_3k_4 - 2\psi_{15}k_3g_5 - 2\psi_{16}k_4g_5 \\ &+ \psi_{13}^2m_1^2 + \psi_{14}^2m_2^2 + \psi_{15}^2m_3^2 + \psi_{16}^2m_4^2 + (u'_5)^2 + 2\psi_{13}\psi_{14}m_1m_2 + 2\psi_{13}\psi_{15}m_1m_3 \\ &+ 2\psi_{13}\psi_{16}m_1m_4 - 2\psi_{13}m_1u'_5 + 2\psi_{14}\psi_{15}m_2m_3 + 2\psi_{14}\psi_{16}m_2m_4 - 2\psi_{14}m_2u'_5 \\ &+ 2\psi_{15}\psi_{16}m_3m_4 - 2\psi_{15}m_3u'_5 - 2\psi_{16}m_4u'_5dt\end{aligned}$$

As a first case, suppose $G_5(\mathbf{0}) < G_5(J)$ and let

$$\mathbf{c} = \begin{bmatrix}0_1 & \cdots & 0_{12} & c_1 & c_2 & c_3 & c_4 & 0_{17} & \cdots & 0_{49}\end{bmatrix}^T$$

where $c_i \in \mathbb{R}$ for all $1 \leq i \leq 4$. We first determine which three terms effect G_5 the least. First, suppose the first three terms in x'_5 is affected the least and set the coefficient of the remaining term as zero, $c_4 = J_0$. Now, choose $c_2, c_3 \in [-z, z]$ where $z \in (0, 3]$ and

solve for c_1.

$$\begin{aligned} G_5(\mathbf{c}) - G_5(\mathbf{0}) = \int_a^b & c_1^2k_1^2 + c_2^2k_2^2 + c_3^2k_3^2 + g_5^2 + 2c_1c_2k_1k_2 + 2c_1c_3k_1k_3 \\ & - 2c_1k_1g_5 + 2c_2c_3k_2k_3 - 2c_2k_2g_5 - 2c_3k_3g_5 \\ & + c_1^2m_1^2 + c_2^2m_2^2 + c_3^2m_3^2 + (u_5')^2 + 2c_1c_2m_1m_2 + 2c_1c_3m_1m_3 \end{aligned}$$

$$\begin{aligned} & - 2c_1m_1u_5' + 2c_2c_3m_2m_3 - 2c_2m_2u_5' - 2c_3m_3u_5' - g_5^2 - (u_5')^2dt \\ & = c_1^2\int_a^b k_1^2 + m_1^2dt \\ & + c_1\int_a^b 2c_2k_1k_2 + 2c_3k_1k_3 - 2k_1g_5 + 2c_2m_1m_2 + 2c_3m_1m_3 - 2m_1u_5'dt \\ & + \int_a^b c_2^2k_2^2 + c_3^2k_3^2 + 2c_2c_3k_2k_3 - 2c_2k_2g_5 - 2c_3k_3g_5 \\ & \qquad + c_2^2m_2^2 + c_3^2m_3^2 + 2c_2c_3m_2m_3 - 2c_2m_2u_5' - 2c_3m_3u_5'dt \end{aligned}$$

Let $B_1 = \int_a^b k_1^2 + m_1^2$, $B_2 = \int_a^b 2c_2k_1k_2 + 2c_3k_1k_3 - 2k_1g_5 + 2c_2m_1m_2 + 2c_3m_1m_3 - 2m_1u_5'dt$, and $B_3 = \int_a^b c_2^2k_2^2 + c_3^2k_3^2 + 2c_2c_3k_2k_3 - 2c_2k_2g_5 - 2c_3k_3g_5 + c_2^2m_2^2 + c_3^2m_3^2 + 2c_2c_3m_2m_3 - 2c_2m_2u_5' - 2c_3m_3u_5'dt$. Then,

$$G_5(\mathbf{c}) - G_5(\mathbf{0}) = B_1c_1^2 + B_2c_1 + B_3 \tag{5.65}$$

First suppose $B_1 > 0$. In this case, if $B_2^2 - 4B_1B_3 \leq 0$, then we set $c_1 = 0$ and if $B_2^2 - 4B_1B_3 > 0$, we let α_1 and α_2 be the roots of (5.65) such that $\alpha_1 < \alpha_2$. Then, we choose $c_1 \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. Here, if $B_2 \neq 0$, then choose $c_1 < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c_1 \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c_1 = 0$.

We now repeat this process by choosing $c_1, c_3 \in [-z, z]$ where $z \in (0, 3]$ and solve for c_2 and then repeat the process again for $c_1, c_2 \in [-z, z]$ where $z \in (0, 3]$ and solve for c_3. Keep in mind that this case is done by assuming that the first three terms in x_5' are affected the least. The cases where the other three terms are affected the least are solved similarly.

In the case where $G_5(J) < G_5(\mathbf{0})$, let

$$\mathbf{c} = \begin{bmatrix} J_1 & \cdots & J_{12} & c_1 & c_2 & c_3 & c_4 & J_{17} & \cdots & J_{49} \end{bmatrix}^T$$

where $c_i \in \mathbb{R}$ for all $1 \leq i \leq 4$. We first determine which three terms effect G_5 the least. For this argument, we suppose the first three terms in x_5' is affected the least and set the coefficient of the remaining term as the corresponding component of J, $c_4 = J_{16}$. In the cases where the other three terms are affected the least are solved similarly. Further, choose $c_2 \in [J_{14} - z, J_{14} + z]$ and $c_3 \in [J_{15} - z, J_{15} + z]$ where $z \in (0, 3]$ and solve for c_1.

$$\begin{aligned}
&G_5(\mathbf{c}) - G_5(J) \\
&\quad = \int_a^b c_1^2k_1^2 + c_2^2k_2^2 + c_3^2k_3^2 + c_4^2k_4^2 + g_5^2 + 2c_1c_2k_1k_2 + 2c_1c_3k_1k_3 \\
&\qquad + 2c_1c_4k_1k_4 - 2c_1k_1g_5 + 2c_2c_3k_2k_3 + 2c_2c_4k_2k_4 - 2c_2k_2g_5 \\
&\qquad + 2c_3c_4k_3k_4 - 2c_3k_3g_5 - 2c_4k_4g_5 \\
&\qquad + c_1^2m_1^2 + c_2^2m_2^2 + c_3^2m_3^2 + c_4^2m_4^2 + (u_5')^2 + 2c_1c_2m_1m_2 + 2c_1c_3m_1m_3 \\
&\qquad + 2c_1c_4m_1m_4 - 2c_1m_1u_5' + 2c_2c_3m_2m_3 + 2c_2c_4m_2m_4 - 2c_2m_2u_5' \\
&\qquad + 2c_3c_4m_3m_4 - 2c_3m_3u_5' - 2c_4m_4u_5' \\
&\qquad - \Big[J_{13}^2k_1^2 + J_{14}^2k_2^2 + J_{15}^2k_3^2 + J_{16}^2k_4^2 + g_5^2 + 2J_{13}J_{14}k_1k_2 + 2J_{13}J_{15}k_1k_3 \\
&\qquad + 2J_{13}J_{16}k_1k_4 - 2J_{13}k_1g_5 + 2J_{14}J_{15}k_2k_3 + 2J_{14}J_{16}k_2k_4 - 2J_{14}k_2g_5 \\
&\qquad + 2J_{15}J_{16}k_3k_4 - 2J_{15}k_3g_5 - 2J_{16}k_4g_5 + J_{13}^2m_1^2 \\
&\qquad + J_{14}^2m_2^2 + J_{15}^2m_3^2 + J_{16}^2m_4^2 + (u_5')^2 + 2J_{13}J_{14}m_1m_2 + 2J_{13}J_{15}m_1m_3 \\
&\qquad + 2J_{13}J_{16}m_1m_4 - 2J_{13}m_1u_5' + 2J_{14}J_{15}m_2m_3 + 2J_{14}J_{16}m_2m_4 \\
&\qquad - 2J_{14}m_2u_5' + 2J_{15}J_{16}m_3m_4 - 2J_{15}m_3u_5' - 2J_{16}m_4u_5'\Big]dt \\
&\quad = c_1^2\int_a^b k_1^2 + m_1^2dt \\
&\qquad + c_1\int_a^b 2c_2k_1k_2 + 2c_3k_1k_3 + 2c_4k_1k_4 - 2k_1g_5 + 2c_2m_1m_2 \\
&\qquad\qquad + 2c_3m_1m_3 + 2c_4m_1m_4 - 2m_1u_5'dt
\end{aligned}$$

$$
\begin{aligned}
&+ \int_a^b c_2^2k_2^2 + c_3^2k_3^2 + c_4^2k_4^2 + 2c_2c_3k_2k_3 + 2c_2c_4k_2k_4 - 2c_2k_2g_5 \\
&+ 2c_3c_4k_3k_4 - 2c_3k_3g_5 - 2c_4k_4g_5 + c_2^2m_2^2 + c_3^2m_3^2 + c_4^2m_4^2 + 2c_2c_3m_2m_3 \\
&+ 2c_2c_4m_2m_4 - 2c_2m_2u_5' + 2c_3c_4m_3m_4 - 2c_3m_3u_5' - 2c_4m_4u_5' \\
&- \Big[J_{13}^2k_1^2 + J_{14}^2k_2^2 + J_{15}^2k_3^2 + J_{16}^2k_4^2 + 2J_{13}J_{14}k_1k_2 + 2J_{13}J_{15}k_1k_3 \\
&+ 2J_{13}J_{16}k_1k_4 - 2J_{13}k_1g_5 + 2J_{14}J_{15}k_2k_3 + 2J_{14}J_{16}k_2k_4 - 2J_{14}k_2g_5 \\
&+ 2J_{15}J_{16}k_3k_4 - 2J_{15}k_3g_5 - 2J_{16}k_4g_5 + J_{13}^2m_1^2 \\
&+ J_{14}^2m_2^2 + J_{15}^2m_3^2 + J_{16}^2m_4^2 + 2J_{13}J_{14}m_1m_2 + 2J_{13}J_{15}m_1m_3 \\
&+ 2J_{13}J_{16}m_1m_4 - 2J_{13}m_1u_5' + 2J_{14}J_{15}m_2m_3 + 2J_{14}J_{16}m_2m_4 \\
&- 2J_{14}m_2u_5' + 2J_{15}J_{16}m_3m_4 - 2J_{15}m_3u_5' - 2J_{16}m_4u_5'\Big]dt
\end{aligned}
$$

In this case, if $B_2^2 - 4B_1B_3 \leq 0$, then we set $c_1 = 0$ and if $B_2^2 - 4B_1B_3 > 0$, we let α_1 and α_2 be the roots of (5.65) such that $\alpha_1 < \alpha_2$. Then, we choose $c_1 \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. Here, if $B_2 \neq 0$, then choose $c_1 < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c_1 \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c_1 = 0$.

First suppose $B_1 >$. In this case if the discriminant of the quadratic is less than or equal to zero, then we set $c_1 = J_{13}$ and if the discriminant is strictly greater than zero, let α_1 and α_2 be the roots of the quadratic such that $\alpha_1 < \alpha_2$. Then we choose $c_1 \in (\alpha_1, \alpha_2)$.

Now suppose $B_1 = 0$. If $B_2 \neq 0$ then choose $c_1 < \frac{-B_3}{B_2}$. If $B_2 = 0$ and $B_3 < 0$, then choose any $c_1 \in \mathbb{R}$. Lastly, if $B_2 = 0$ and $B_3 \geq 0$, then there does not exist such c and we set $c_1 = J_{13}$. We now repeat this process by choose $c_1 \in [J_{13} - z, J_{13} + z]$ and $c_3 \in [J_{15} - z, J_{15} + z]$ where $z \in (0, 3]$ and solve for c_2 and then repeat the process again for $c_1 \in [J_{13} - z, J_{13} + z]$ and $c_2 \in [J_{14} - z, J_{15} + z]$ where $z \in (0, 3]$ and solve for c_3.

In summary, suppose that first three terms in the x_5' equation effects G_5 the least.

Case 1: $G_5(\mathbf{0}) < G_5(J)$. Let $\mathbf{c} = \begin{bmatrix} 0_1 & \cdots & 0_{12} & c_1 & c_2 & c_3 & c_4 & 0_{17} & \cdots & 0_{49} \end{bmatrix}^T$ where $c_1, c_2, c_3, c_4 \in \mathbb{R}$. Choose $c_2, c_3 \in [-z, z]$ where $z \in (0, 3]$.

Sub-case 1: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 \leq 0$ then $c_1 = 0$.

Sub-case 2: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 > 0$ then let α_1 and α_2 be the roots such that $\alpha_1 < \alpha_2$. Choose $c_1 \in (\alpha_1, \alpha_2)$.

Sub-case 3: If $B_1 = 0$ and $B_2 \neq 0$, then we choose $c_1 < \frac{-B_3}{B_2}$.

Sub-case 4: If $B_1 = B_2 = 0$ and $B_3 < 0$ then choose any $c_1 \in \mathbb{R}$.

Sub-case 5: If $B_1 = B_2 = 0$ and $B_3 > 0$ then $c_1 = 0$.

Repeat this process by choosing $c_1, c_3 \in [-z, z]$ and solve for c_2 and then repeat this process by choosing $c_1, c_2 \in [-z, z]$ and solve for c_3.

Case 2: $G_5(J) < G_5(\mathbf{0})$. Let $\mathbf{c} = \begin{bmatrix} J_1 & \cdots & J_{12} & c_1 & c_2 & c_3 & c_4 & J_{17} & \cdots & J_{49} \end{bmatrix}^T$ where $c_1, c_2, c_3, c_4 \in \mathbb{R}$. Choose $c_2 \in [J_{14} - z, J_{14} + z]$ and $c_2 \in [J_{15} - z, J_{15} + z]$ where $z \in (0, 3]$.

Sub-case 1: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 \leq 0$ then $c_1 = J_{13}$.

Sub-case 2: If $B_1 > 0$ and $B_2^2 - 4B_1B_3 > 0$ then let α_1 and α_2 be the roots such that $\alpha_1 < \alpha_2$. Choose $c_1 \in (\alpha_1, \alpha_2)$.

Sub-case 3: If $B_1 = 0$ and $B_2 \neq 0$, then we choose $c_1 < \frac{-B_3}{B_2}$.

Sub-case 4: If $B_1 = B_2 = 0$ and $B_3 < 0$ then choose any $c_1 \in \mathbb{R}$.

Sub-case 5: If $B_1 = B_2 = 0$ and $B_3 > 0$ then $c_1 = J_{13}$.

Repeat this process by choosing $c_1 \in [J_{13} - z, J_{13} + z]$ and $c_3 \in [J_{15} - z, J_{15} + z]$ and solve for c_2 and then repeat this process by choosing $c_1 \in [J_{13} - z, J_{13} + z]$ and $c_2 \in [J_{14} - z, J_{14} + z]$ and solve for c_3.

Following the methods outlined above, we did indeed obtain $G(\mathbf{c}) < G(J)$ and $G(\mathbf{c}) < G(\mathbf{0})$ in every run.

9. Counter the Bad Model

Based on the theorems provided in Section 6, we see that there are infinitely many ψ_0 such that $G(\psi_0) < G(0)$ and $G(\psi_0) \leq G(J)$ which produces many solutions

to our delay differential equation. Now, due to the inherent flaws in the system (1.1), we discovered that some of these ψs caused the solutions to (1.1) to become exponentially large with solutions taking values over an unacceptably large range. To address this issue, we impose an ad hoc restriction as follows.

We first define Max_v. Let v be as defined in Section 1 and let

$$\lambda_v = \max_{v-1 \le t \le v} x_8(t) - \min_{v-1 \le t \le v} x_8(t)$$

$$\Lambda_v = \max_{v-2 \le t \le v} x_8(t) - \min_{v-2 \le t \le v} x_8(t)$$

where x_8 denotes the price of Apple stock. Then,

$$Max_v = \max_{j \le v} \{\lambda_j, \Lambda_j\}$$

For convenience, we set $[t_{-1}, t_0]$ as the history period for our delay differential equation, and set $[t_0, t_1]$ as the time period in which the solution to our delay differential equation is defined.

Let y_ψ represent the solution to our delay differential equation found using ψ and $y_{8,\psi}$ represent the solution for the equation for x_8. Further, let $x_8(v)$ represent the value of the stock price at v. Then, we say that a ψ is a Good-initial-ψ if it meets the following two conditions:

(1) $y_{8,\psi} \ge 0$

(2) $y_{8,\psi} \in [x_8(v) - \beta Max_v, x_8(v) + \beta Max_v]$ for some $\beta \in (0, 3)$

These conditions rule out the ψs that induce solutions with unreasonable range and are also used as stopping criteria in our minimization process.

10. Iterative Improvements Using Volatility

Recall that we can predict the price of Apple's stock using American option price data [**18**, p.84-86]. Although Barnett has previously given us some volatility information obtained from Apple option price data from July to December of 2018, we

gather additional information to broaden the testing window from January to March of 2023.

A brief summary of this strategy is as follows. Suppose we have computed the coefficients of (1.1) for $[t_{-1}, t_0] \subset \mathcal{T}$ and obtained the solution of (1.1) using these coefficients on $[t_0, t_1] \subset \mathcal{T}$. Let $y \in (\mathcal{L}^2[t_0, t_1])^{13}$ be the solution to (1.1), $y' \in (\mathcal{L}^2[t_0, t_1])^{13}$ be the estimated derivatives of y, and let $y_8 \in \mathcal{L}^2[t_0, t_1]$ represent the solution of the equation for x_8. Further, let $\sigma(S, t)$ represent a recovered local volatility where S represents a stock price. Then, the drift or average rate of growth of the stock price μ can be estimated by:

$$\mu(S, t) \approx \frac{y_8'}{y_8}.$$

Now, using the drift and the local volatility found above, we solve the stochastic differential equation

$$dS(t) = \mu(S(t), t)S(t)dt + \sigma(S(t), t)S(t)dB(t), \tag{5.66}$$

using the **gbm** command in MATLAB. Following Barnett, we solve the stochastic differential equation 500 times, and average the resulting stochastic processes. For later use, let F represent this average of the resulting stochastic processes, which becomes our forecast for the stock price.

10.1. Iterative Improvements. Though the process described above can already produce a better estimation of stock prices, we want to improve the result even further. Hence, we iterate this process on $[t_0, t_1]$. We describe the process of the first iteration below.

Let $X = \begin{bmatrix} y_1 & \cdots & y_7 & F & y_9 & \cdots & y_{13} \end{bmatrix}^T \in (\mathcal{L}^2[t_0, t_1])^{13}$. We now use X as our data points to solve (1.1) on $[t_0, t_1]$. In other words, we solve

$$
\begin{aligned}
y_1'(t) &= c_1 y_2(t) y_1(t) \\
y_2'(t) &= c_2 y_1(t) + c_3 y_4(t-20) y_2(t) + c_4 y_2(t) \\
y_3'(t) &= c_5 y_2(t) + c_6 y_6(t) + c_7 y_8(t) + c_8 y_{13}(t) \\
y_4'(t) &= c_9 y_1'(t) + c_{10} y_4(t) + c_{11} y_2(t) y_4(t-20) + c_{12} F(t) y_4(t) \\
y_5'(t) &= c_{13} F(t) + c_{14} y_{10}(t) + c_{15} y_4(t) + c_{16} y_{12}(t) \\
y_6'(t) &= c_{17} y_6(t) + c_{18} y_7(t-17) + c_{19} y_{11}(t) + c_{20} y_{12}(t) \\
y_7'(t) &= c_{21} y_7(t) + c_{22} y_6(t) + c_{23} y_{11}(t) \\
F'(t) &= c_{24} y_6(t) + c_{25} y_5'(t) + c_{26} F(t) \\
&\quad + c_{27} F(t) y_4(t) + c_{28} y_3(t) + c_{29} y_{10}'(t) + c_{30} y_{13}'(t) \\
y_9'(t) &= c_{31} y_{10}(t) + c_{32} y_{12}(t) + c_{33} y_2(t) + c_{34} y_9(t) \\
y_{10}'(t) &= c_{35} y_{10}(t) + c_{36} y_4(t) + c_{37} y_1(t) + c_{38} y_5(t) + c_{39} y_{12}(t) \\
y_{11}'(t) &= c_{40} y_7(t-17) + c_{41} y_{11}(t) + c_{42} y_4(t-20) y_{11}(t) \\
y_{12}'(t) &= c_{43} y_{10}'(t) + c_{44} y_9(t) + c_{45} y_4(t) + c_{46} y_1(t) \\
y_{13}'(t) &= c_{47} y_{10}(t) + c_{48} y_6(t) + c_{49} y_7(t)
\end{aligned}
\tag{5.67}
$$

Once the coefficients of the system (5.67) are found using the descent method, we then solve the delay differential equation using $X(t_0)$ as the history vector. As a result, we obtain $y^{[2]} \in (\mathcal{L}^2[t_0, t_1])^{13}$ as the solution. Further, we find $F^{[2]}$, the second forecast to the stock price, as described above using $y^{[2]}$. It should be noted that this iterative procedure can be repeated numerous times. For consistency, we typically perform two iterations on $[t_0, t_1]$.

11. General Strategy

We briefly outline our method in this section. Recall that our goal is to forecast the trend in Apple's stock prices, and for specified months with volatility available,

we forecast actual stock values by iteratively solving the equation (5.66). To do so, we employ two distinct strategies: one to forecast the trend alone and the other to forecast both the trend and the prices.

11.1. Forecasting Trend Alone. To forecast the trend in Apple's stock prices, we use the steps listed below.

Step 1: Find up to 10 Good-initial ψs using **"Get_Constants_Final.m"**

Step 2: Calculate $G(\psi)$ and $G_8(\psi)$ for each Good-initial ψ found from **Step 1**. Select up to five ψs with the lowest $G(\psi)$s and $G_8(\psi)$s. Then record the overlapping ψ(s). **"Get_Constants_Which_One.m"**

Step 3: Minimize G using the gradient descent method with the Neumann gradient for 300 iterations. Here, we use each ψ from **Step 3** as a starting point. **"H1_All_Psi.m"**

Step 4: Use the resulting ψ(s) from **Step 3** to solve the DDE. Average the solutions and select the minimized ψ which gives you the solution which is closest to the average **"Apply_KJ_C_to_predict_good_psi.m"**

11.2. Forecasting Trend and the Price. We can utilize volatility data to forecast stock prices for specific time periods. We implement the steps outlined below within these time-frames.

Round 1: On $[t_{-1}, t_0]$

Step 1: Find up to 10 Good-initial ψs using **"Get_Constants_Final.m"**

Step 2: Calculate $G(\psi)$ and $G_8(\psi)$ for each Good-initial ψ found from **Step 1**. Select up to five ψs with the lowest $G(\psi)$s and $G_8(\psi)$s. Then record the overlapping ψ(s). **"Get_Constants_Which_One.m"**

Step 3: Minimize G using the gradient descent method with the Neumann gradient for 300 iterations. Here, we use each ψ from **Step 3** as a starting point. **"H1_All_Psi.m"**

Step 4: Use the resulting ψ(s) from **Step 3** to solve the DDE. Average the solutions and select the minimized ψ which gives you the solution which is closest to the average **"Apply_KJ_C_to_predict_good_psi.m"**

Step 5: Get 1000 stock predictions with the chosen ψ from Step 5. Then calculate the average of the predictions and pick the one that is closest to the average. "**Apple_Stock_Pred_Guess.m**"

Round 2: Iterative Improvement On $[t_0, t_1]$

Step 1: Find up to 10 Good-initial ψs using **"Get_Constants_Final.m"** on $[t_0, t_1]$ Here, we take the DDE solutions for $x_1, \cdots, x_7$ and $x_9, \cdots, x_{13}$. For x_8, we take the Geometric Brownian Motion

Step 2: Calculate $G(\psi)$ and $G_8(\psi)$ for each Good-initial ψ found from **Step 1**. Select up to five ψs with the lowest $G(\psi)$s and $G_8(\psi)$s. Then record the overlapping ψ(s). **"Get_Constants_Which_One.m"**

Step 3: Minimize G using the gradient descent method with the Neumann gradient for 300 iterations. Here, we use each ψ from **Step 3** as a starting point. **"H1_All_Psi.m"**

Step 4: Use the resulting ψ(s) from **Step 3** to solve the DDE. Average the solutions and select the minimized ψ which gives you the solution which is closest to the average **"Apply_KJ_C_to_predict_good_psi.m"**

Step 5: Get 1000 stock predictions with the chosen ψ from Step 5. Then calculate the average of the predictions and pick the one that is closest to the average. "**Apple_Stock_Pred_Guess.m**"

Round 3: Iterative Improvement On $[t_0, t_1]$

Step 1: Find up to 10 Good-initial ψs using **"Get_Constants_Final.m"** on $[t_0, t_1]$ Here, we take the DDE solutions for $x_1, \cdots, x_7$ and $x_9, \cdots, x_{13}$. For x_8, we take the Geometric Brownian Motion

Step 2: Calculate $G(\psi)$ and $G_8(\psi)$ for each Good-initial ψ found from **Step 1**. Select up to five ψs with the lowest $G(\psi)$s and $G_8(\psi)$s. Then record the overlapping ψ(s). **"Get_Constants_Which_One.m"**

Step 3: Minimize G using the gradient descent method with the Neumann gradient for 300 iterations. Here, we use each ψ from **Step 3** as a starting point. **"H1_All_Psi.m"**

Step 4: Use the resulting ψ(s) from **Step 3** to solve the DDE. Average the solutions and select the minimized ψ which gives you the solution which is closest to the average **"Apply_KJ_C_to_predict_good_psi.m"**

- The $y_8(\psi)$ found using this ψ gives us our final trend.

Step 5: Get 1000 stock predictions with the chosen ψ from Step 5. Then calculate the average of the predictions and pick the one that is closest to the average. This is our final price prediction. "**Apple_Stock_Pred_Guess.m**"

12. LSTM Networks

This section provides a quick introduction to deep learning and a description of the deep learning model patterned after [**35**] we developed to forecast Apple's stock price.

As discussed in Chapter 1 Section 3, artificial neural networks (ANN) are modeled after the human brain to uncover complicated relationships and identify patterns in data. Each ANN consist of an input layer, one or more hidden layers, and an output layer where each layer consists of one or more nodes, each of which is linked to one or more nodes in the following layer. We say that an ANN is deep learning if there are two or more hidden layers. Furthermore, each node in the hidden layer sends information to the nodes in the next layer if its output, calculated using mathematical functions, exceeds a specified threshold.

The computations used in the hidden states determine the patterns and relationships discovered by a model. As one would expect, the majority of relationships

and patterns cannot be completely explained by the linear transformation of the inputs alone and require the integration of some non-linearity. To introduce nonlinearity, we apply activation functions such as the sigmoid function or the *tanh* function as the outermost function in each node, where the inner function is usually a linear combination of inputs and weights. After each iteration, the weights are updated using the minimized loss function to reflect the relative significance of each input. The purpose of a loss function is to measure the quality of an algorithm; the smaller the loss function's output, the better the algorithm. Furthermore, the loss function, which is dependent on the weights, is minimized using the gradient descent method.

A recurrent neural network (RNN) is a form of ANN that is designed to process sequential data. A single hidden layer makes up a vanilla RNN, and the output from a previous time step serves as the input for the next time step. As mentioned above, on each node of the hidden layer, we apply a sigmoid function to a linear combination of the weights and input. This method can be represented as numerous layers of neural networks to reveal that the same weight matrix is used at every time step in one cycle of forward propagation. It should be noted that forward propagation refers to the process of going from the input layer to the output layer, whereas backward propagation represents the process of traveling from the output layer to the input layer. We use backpropagation to calculate the gradient at the first time step, and since the weight matrix is constant throughout forward propagation, there are numerous factors of weight matrix involved in computing the gradient at the first time step. As a result, we encounter the vanishing gradient problem if the weight matrix's largest singular value is smaller than one. In this instance, the weights cannot be updated, and the model's performance suffers.

The long short term memory (LSTM) network is a type of RNN that was developed to solve the vanishing gradient issue. Just like RNN, the vanilla LSTM network has a single layer of hidden network. However, unlike RNN, each node of the hidden layer of the LSTM is made of three gates: the forget gate, the input gate,

and the output gate. The forget gate decides which information to remove based on the usefulness of both the new input data and the old hidden state. The output gate decides how much information to reveal by determining the new hidden state using the newly updated cell state, the old hidden state, and the new input data [**21**].

12.1. Our LSTM Model. To prevent confusion, we refer to the regularization model as 'our model' and the deep learning model we constructed as the LSTM model. The LSTM model is built using a sequential model from TensorFlow and has three major parts: data preprocessing, building and training the model, and making predictions with the model.

To preprocess the data, we upload Apple's stock information from [**12**], utilize the Pandas package to separate the prospective training set from the potential testing set, then normalize our data using the scikit-learn package. Let training_set_scaled represent the scaled closing prices of Apple's stock and let v represent the first day of the month where we want to start the prediction such that $v + 1$ represent the first day of the following month. Recall that unlike other deep learning models, which are trained on 90% of the data, we want to limit the training set to the previous month's data to allow for comparison with our model. We now prepare our training set of X_train and y_train from $v - 1$ to v to train our model. Let $i \in [v - 1, v)$ represent each available data point between $v - 1$ and v. Then, each component of X_train is a vector of closing prices on $[i - 30, i - 1]$ and the corresponding y_train element is the closing price at i.

In order to construct our sequential model, we use the TensorFlow framework. We use five hidden layers, each of which has LSTM networks and a dropout regularization; the dropout layers are used to address overfitting issues. Further, for the loss function we use the mean squared error and to minimize the loss function we use a modified version of stochastic gradient descent called the Adam optimization algorithm. Once the LSTM model is built as described, we fit the LSTM model with X_train and Y_train.

Lastly, as we have for our model, we predict two months of Apple's stock prices using the LSTM model. Let X_test represent the scaled test data where each element is a vector of closing prices on $[j-30, j-1]$ where $j \in [v, v+2)$. We then use X_test in the prediction function of the LSTM model we built to forecast the stock prices on $[v, v+2)$ and undo the scaling to accurately represent the result.

Because we are only using one month of data as the training period, the LSTM model predictions are used as a trend prediction. We should note that, if we train the LSTM model with more than a few months of data, the forecasts begin to resemble stock prices. In the latter case, we may compare those outcomes to our stock price forecasts.

CHAPTER 6

Results

In this section, we compare the results of our method to Barnett's method and the LSTM model. It should be noted that the LSTM model results are regarded as a forecast for Apple's stock trend for the reasons explained in Section 12. Although the number of months necessary to determine the coefficient in Barnett's model is fixed at seven, both our model and the LSTM model have fully adjustable training periods. However, in order to demonstrate the efficacy of our model, we chose to use one month as a training period and forecast Apple's stock trend for two months.

We make a total of 317 predictions for the Apple stock trend, where the starting date of the first prediction is October 1, 1996, and the starting date of the last prediction is February 1, 2023. To be consistent with Barnett, we use the mean absolute percentage error (MAPE) to assess the accuracy of the forecasts. Let $F, A \in \mathbb{R}^n$ where F is the forecast array and A is the actual data. To calculate MAPE, we use the built in MATLAB function **mape** and the input for **mape** is (F, A).

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{A_i - F_i}{A_i} \right|$$

Note that MAPE does not determine whether the model properly predicts the direction of the underlying stock; rather, it evaluates the degree of separation between the actual stock price and the forecast. This is the rationale behind Barnett's decision to use visual inspection as a measurement technique. However, to maintain objectivity, we solely utilize MAPE to assess the quality of our model.

Using American options data, we make a total of seven forecasts for Apple's stock price. These forecasts begin on July 1, August 1, September 1, October 1, and

November 1, 2018, and January 1 and February 1, 2023. Remember that the stock price forecast is the average of the 500 solutions to the stochastic differential equation. Notice that each run provides a different prediction to the user. To remove the user's control, we repeat the process 1000 times and select the prediction that is closest to the average of the predictions. This additional step resulted in more consistent and objective outcomes. Additionally, utilizing this forecast, we implement our iterative improvements twice as described in Section 10. Furthermore, to observe the behavior of our model under additional iterations, we ran twelve iterations of the iterative improvement algorithm on a few samples.

To be precise, we consider a prediction *successful* if the MAPE is less than 10%, and *extremely successful* if the MAPE is less than 5%. The table 6.1 and the table 6.2 below shows the overall comparison of the forecasts made for Apple's stock trend and price respectively.

	Less than 5% of error	Less than 10% of error
Our Method	166	279
Barnett's Method	85	186
LSTM	73	206

TABLE 6.1. Overall Comparison for Apple's Stock Trend

	Less than 5% of error	Less than 10% of error
Our Method	2	6
Barnett's Method	0	3

TABLE 6.2. Overall Comparison for Apple's Stock Price

For the trend forecasts, our model outperformed Barnett's model 92.33% of the time and outperformed the LSTM model 93.93% of the time. Further, for the price forecasts, our model outperformed Barnett's model 100% of the time. It should be noted that we do not compare our method to the LSTM model here because the LSTM model's predictions are only used to compare stock trends.

We observed the following results when we conducted fifteen iterations of the iterative improvement method on predictions starting at $v = 292$ and $v = 344$: the MAPE of the trend and the actual stock price are lower than Barnett's on each iteration,

and the value of the MAPE fluctuates on each iteration. The non-convergence of the solutions is expected since the results we obtain are dependent on the constraints we impose to counteract the poor model and method of selecting good initial values. Nonetheless, the fact that our model consistently outperformed Barnett's suggests that the new methods we implemented are effective in generating convincing results.

Some of the outcomes are shown below. In Section 1, we show a few examples where our method outperformed both Barnett's model and the LSTM model in predicting the trend of Apple's stock prices. Then, in Section 2, we offer a few instances where our model did not predict the stock trend as accurately as the other two. In Section 3, we present the results of all stock price forecasts produced using local volatility data and our new iterative improvement method. Lastly, in Section 4, we show the progression of the predicted trends and the stock price forecasts throughout the twelve iterations of iterative improvement methods.

1. Trend Predictions: Our Method Outperforms All

Comparing the predicted trend to Apple's stock price for 02/01/1997 − 03/31/1997

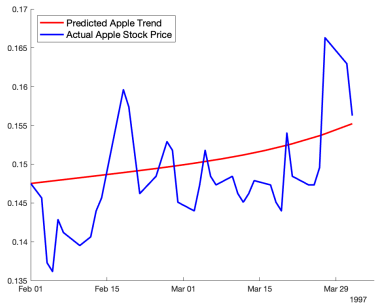

FIGURE 6.1. Our Method

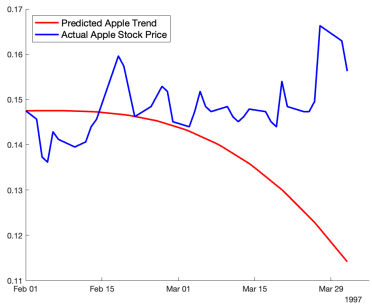

FIGURE 6.2. Barnett

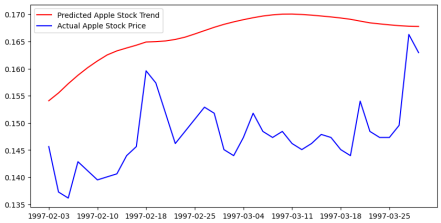

FIGURE 6.3. LSTM

	MAPE
Our Method	3.41%
Barnett's Method	8.0648%
LSTM	12.5637%

TABLE 6.3. MAPE

Comparing the predicted trend to Apple's stock price for 03/01/01 − 04/30/01

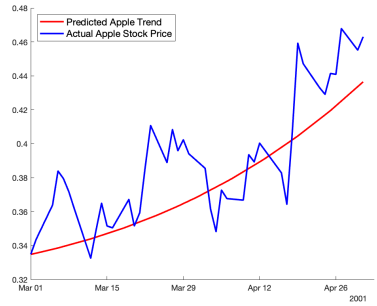

FIGURE 6.4. Our Method

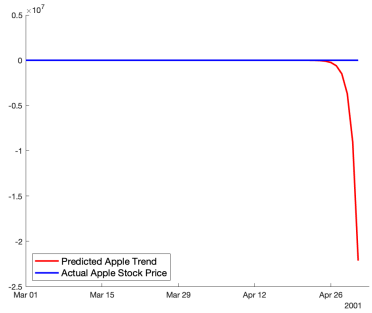

FIGURE 6.5. Barnett

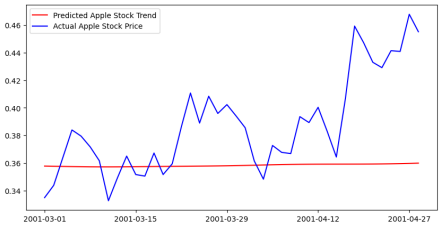

FIGURE 6.6. LSTM

	MAPE
Our Method	5.0929%
Barnett's Method	131060000%
LSTM	8.1581%

TABLE 6.4. MAPE

Comparing the predicted trend to Apple's stock price for 05/01/04 – 06/30/04

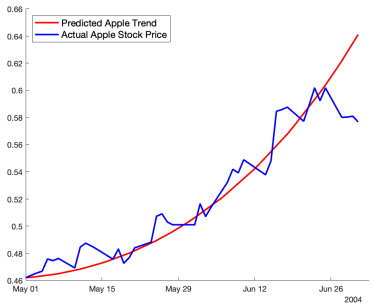

FIGURE 6.7. Our Method

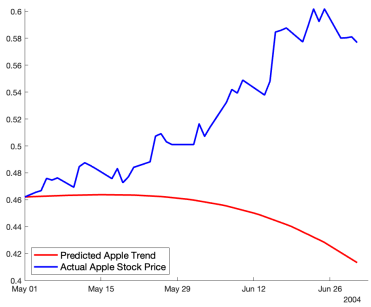

FIGURE 6.8. Barnett

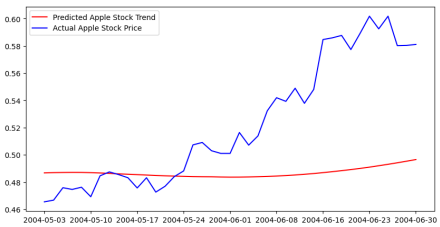

FIGURE 6.9. LSTM

	MAPE
Our Method	1.9712%
Barnett's Method	12.4260%
LSTM	7.5119%

TABLE 6.5. MAPE

Comparing the predicted trend to Apple's stock price for 07/01/05 – 08/31/05

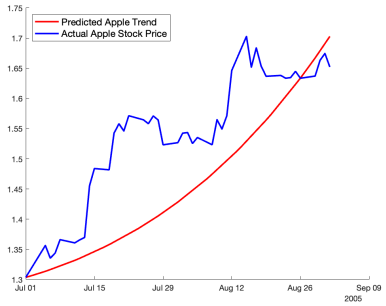

FIGURE 6.10. Our Method

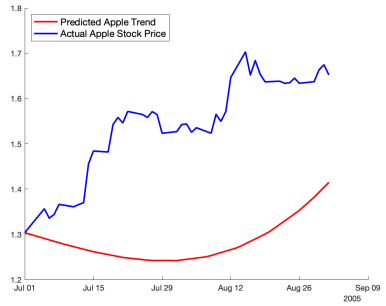

FIGURE 6.11. Barnett

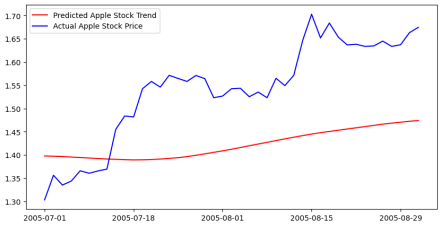

FIGURE 6.12. LSTM

	MAPE
Our Method	5.5043%
Barnett's Method	16.0120%
LSTM	8.5791%

TABLE 6.6. MAPE

Comparing the predicted trend to Apple's stock price for 05/01/06 – 06/30/06

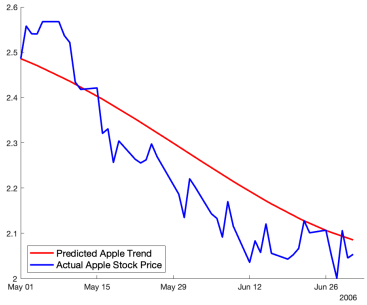

FIGURE 6.13. Our Method

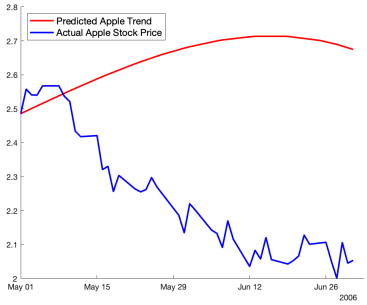

FIGURE 6.14. Barnett

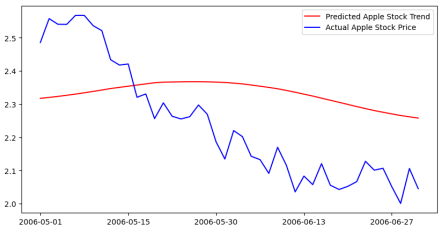

FIGURE 6.15. LSTM

	MAPE
Our Method	3.3047%
Barnett's Method	19.5700%
LSTM	8.0573%

TABLE 6.7. MAPE

Comparing the predicted trend to Apple's stock price for 10/01/06 – 11/30/06

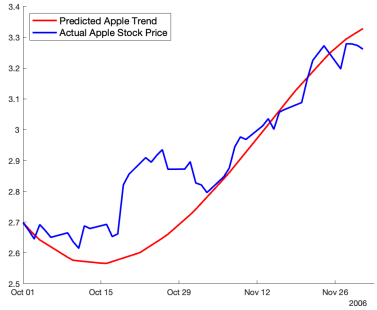

FIGURE 6.16. Our Method

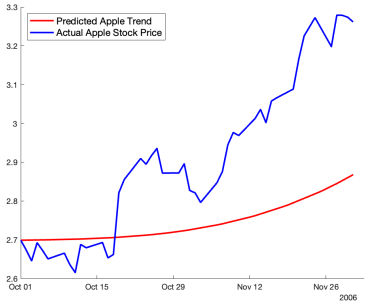

FIGURE 6.17. Barnett

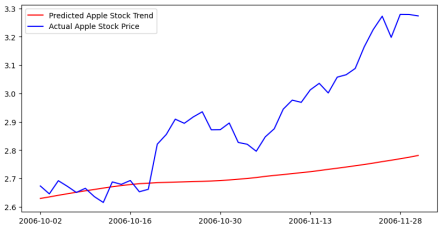

FIGURE 6.18. LSTM

	MAPE
Our Method	3.031%
Barnett's Method	5.9932%
LSTM	6.6070%

TABLE 6.8. MAPE

Comparing the predicted trend to Apple's stock price for 09/01/08 – 10/31/08

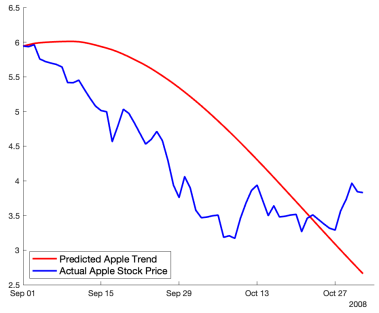

FIGURE 6.19. Our Method

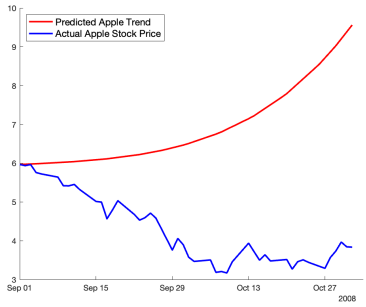

FIGURE 6.20. Barnett

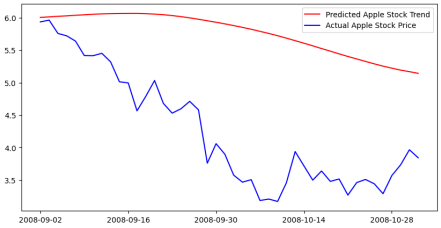

FIGURE 6.21. LSTM

	MAPE
Our Method	19.2644%
Barnett's Method	71.4530%
LSTM	39.8936%

TABLE 6.9. MAPE

Comparing the predicted trend to Apple's stock price for 10/01/13 – 11/30/13

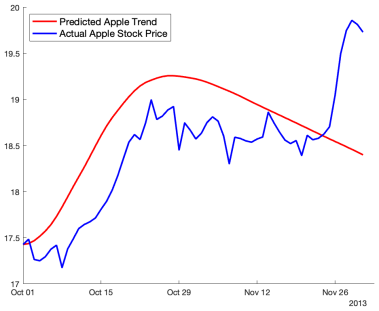

FIGURE 6.22. Our Method

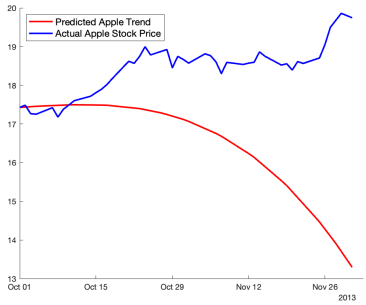

FIGURE 6.23. Barnett

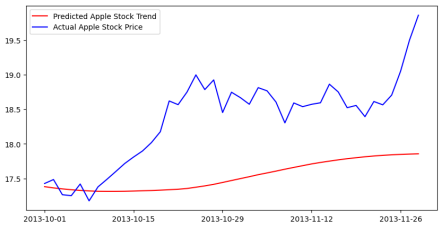

FIGURE 6.24. LSTM

	MAPE
Our Method	2.608%
Barnett's Method	10.431%
LSTM	4.4868%

TABLE 6.10. MAPE

Comparing the predicted trend to Apple's stock price for 10/01/14 – 11/30/14

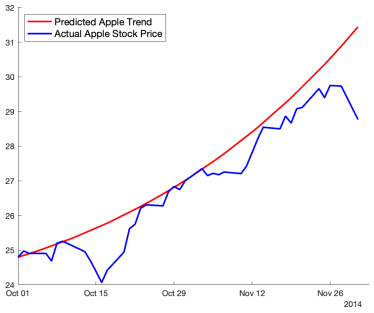

FIGURE 6.25. Our Method

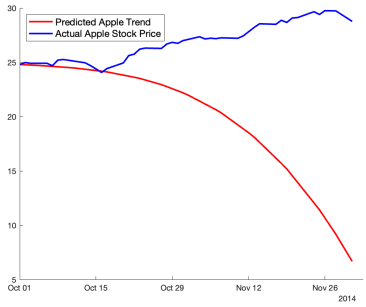

FIGURE 6.26. Barnett

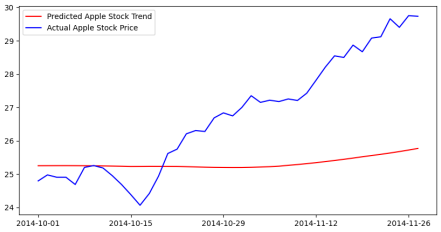

FIGURE 6.27. LSTM

	MAPE
Our Method	2.0504%
Barnett's Method	24.5900%
LSTM	6.1066%

TABLE 6.11. MAPE

Comparing the predicted trend to Apple's stock price for 01/01/16 – 02/30/16

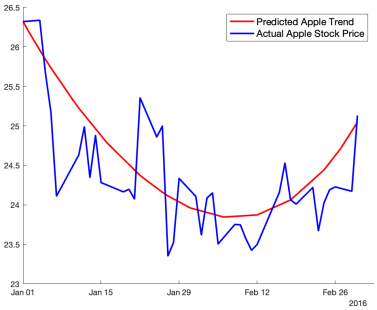

FIGURE 6.28. Our Method

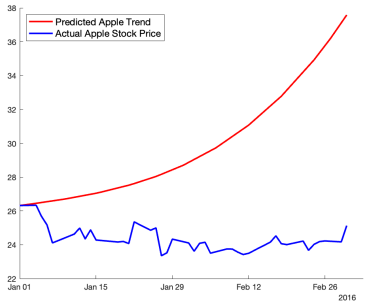

FIGURE 6.29. Barnett

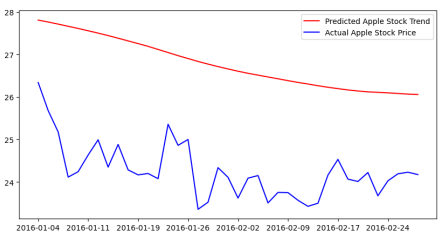

FIGURE 6.30. LSTM

	MAPE
Our Method	1.7123%
Barnett's Method	22.6700%
LSTM	10.3944%

TABLE 6.12. MAPE

Comparing the predicted trend to Apple's stock price for 10/01/16 – 11/30/16

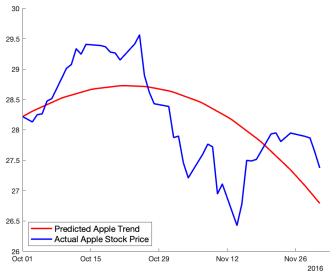

FIGURE 6.31. Our Method

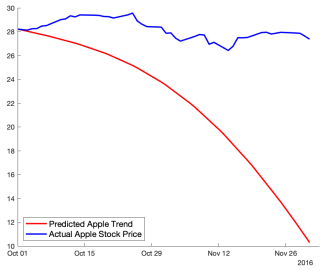

FIGURE 6.32. Barnett

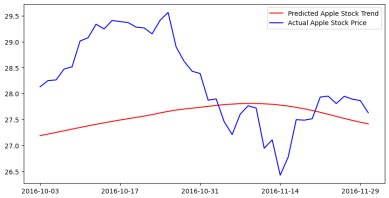

FIGURE 6.33. LSTM

	MAPE
Our Method	2.1310%
Barnett's Method	22.1210%
LSTM	3.2162%

TABLE 6.13. MAPE

Comparing the predicted trend to Apple's stock price for 03/01/19 – 04/30/19

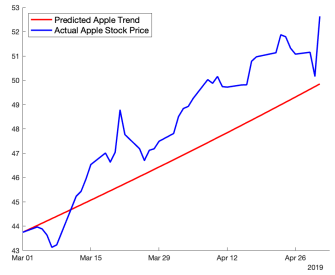

FIGURE 6.34. Our Method

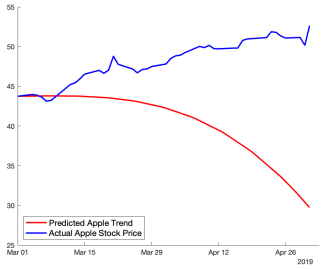

FIGURE 6.35. Barnett

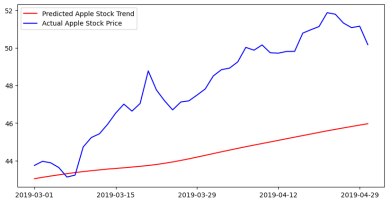

FIGURE 6.36. LSTM

	MAPE
Our Method	3.0706%
Barnett's Method	15.5340%
LSTM	7.3767%

TABLE 6.14. MAPE

Comparing the predicted trend to Apple's stock price for 04/01/19 – 05/31/19

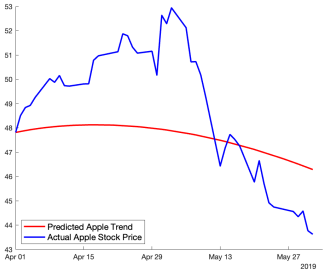

FIGURE 6.37. Our Method

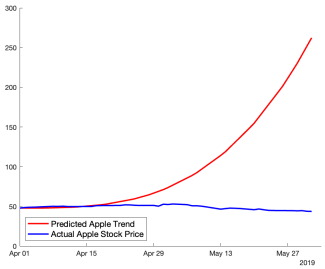

FIGURE 6.38. Barnett

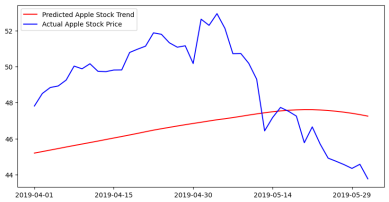

FIGURE 6.39. LSTM

	MAPE
Our Method	4.3117%
Barnett's Method	113.6900%
LSTM	6.7518%

TABLE 6.15. MAPE

Comparing the predicted trend to Apple's stock price for 12/01/19 – 01/31/20

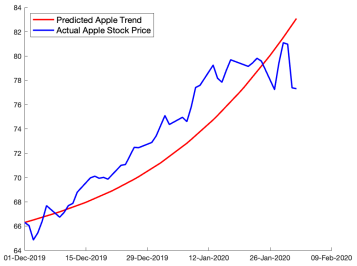

FIGURE 6.40. Our Method

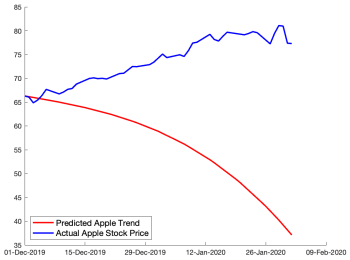

FIGURE 6.41. Barnett

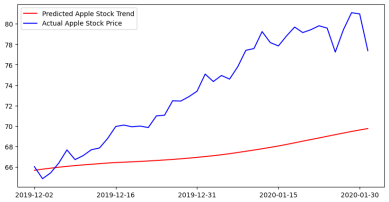

FIGURE 6.42. LSTM

	MAPE
Our Method	2.6454%
Barnett's Method	22.5170%
LSTM	8.2299%

TABLE 6.16. MAPE

Comparing the predicted trend to Apple's stock price for 01/01/21 – 02/28/21

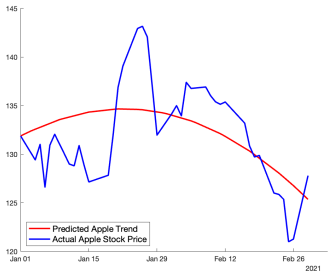

FIGURE 6.43. Our Method

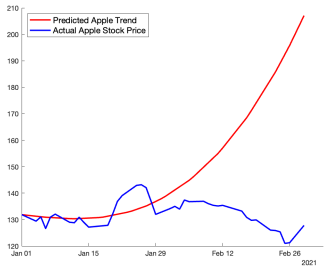

FIGURE 6.44. Barnett

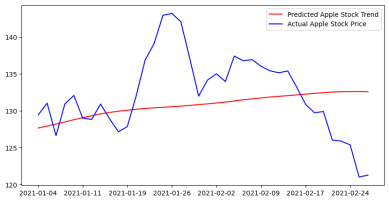

FIGURE 6.45. LSTM

	MAPE
Our Method	2.5659%
Barnett's Method	14.8280%
LSTM	3.4271%

TABLE 6.17. MAPE

Comparing the predicted trend to Apple's stock price for 08/01/22 – 09/30/22

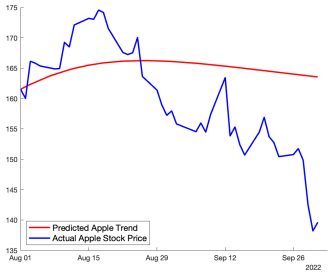

FIGURE 6.46. Our Method

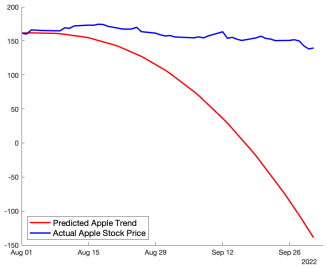

FIGURE 6.47. Barnett

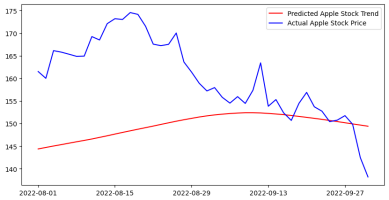

FIGURE 6.48. LSTM

	MAPE
Our Method	5.1733%
Barnett's Method	58.0380%
LSTM	6.9522%

TABLE 6.18. MAPE

2. Trend Predictions: Our Method did not outperform

Comparing the predicted trend to Apple's stock price for 03/01/97 – 04/30/97

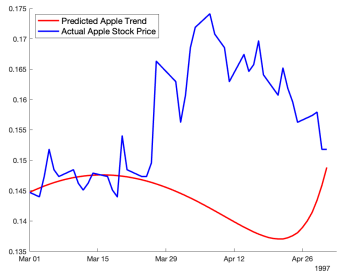

FIGURE 6.49. Our Method

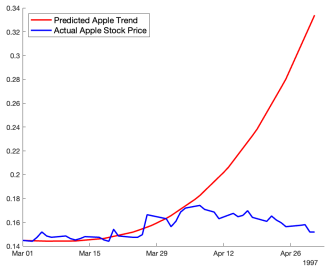

FIGURE 6.50. Barnett

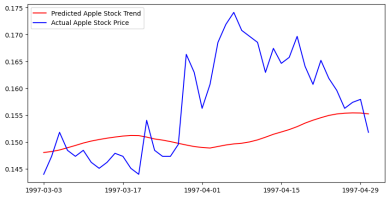

FIGURE 6.51. LSTM

	MAPE
Our Method	8.3561%
Barnett's Method	22.4640%
LSTM	5.1280%

TABLE 6.19. MAPE

Comparing the predicted trend to Apple's stock price for 08/01/97 – 09/30/97

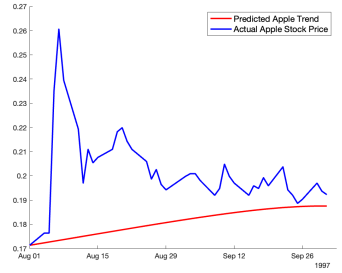

FIGURE 6.52. Our Method

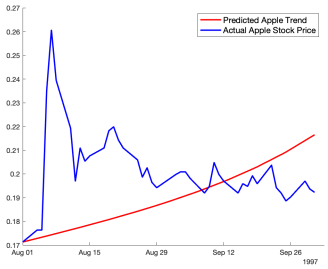

FIGURE 6.53. Barnett

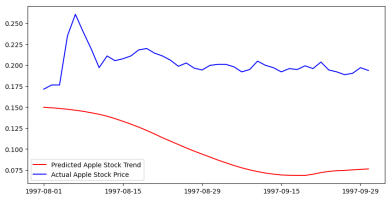

FIGURE 6.54. LSTM

	MAPE
Our Method	9.6750%
Barnett's Method	8.5805%
LSTM	49.8095%

TABLE 6.20. MAPE

Comparing the predicted trend to Apple's stock price for 08/01/06 – 09/30/06

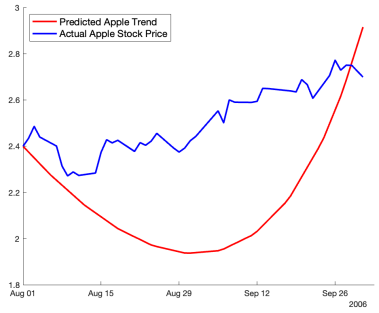

FIGURE 6.55. Our Method

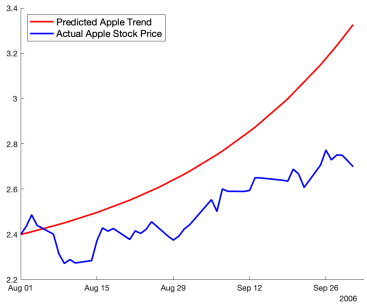

FIGURE 6.56. Barnett

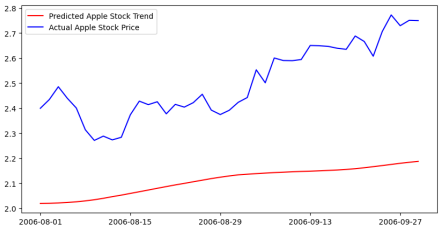

FIGURE 6.57. LSTM

	MAPE
Our Method	13.7721%
Barnett's Method	9.1850%
LSTM	15.5275%

TABLE 6.21. MAPE

Comparing the predicted trend to Apple's stock price for 11/01/06 – 21/31/06

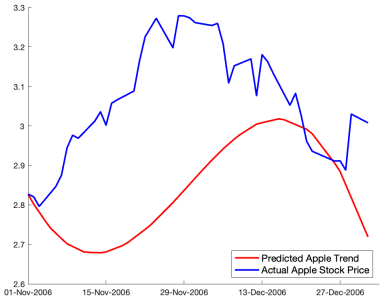

FIGURE 6.58. Our Method

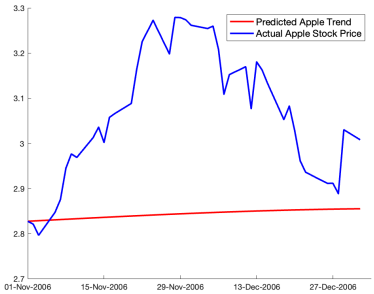

FIGURE 6.59. Barnett

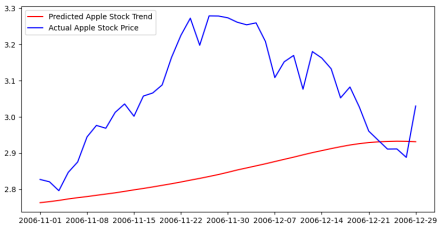

FIGURE 6.60. LSTM

	MAPE
Our Method	7.3535%
Barnett's Method	7.1366%
LSTM	7.0330%

TABLE 6.22. MAPE

Comparing the predicted trend to Apple's stock price for 12/01/09 – 01/31/10

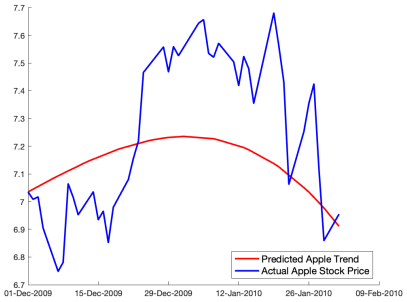

FIGURE 6.61. Our Method

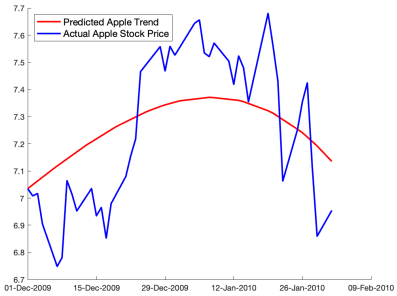

FIGURE 6.62. Barnett

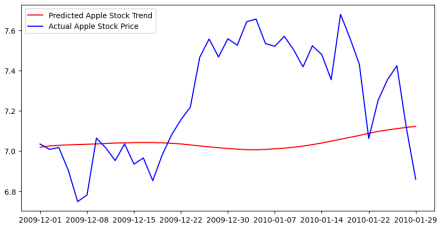

FIGURE 6.63. LSTM

	MAPE
Our Method	3.1484%
Barnett's Method	2.6027%
LSTM	3.8142%

TABLE 6.23. MAPE

Comparing the predicted trend to Apple's stock price for 06/01/17 – 07/31/17

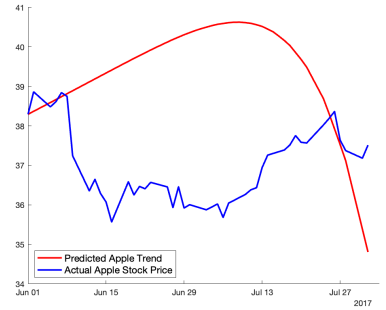

FIGURE 6.64. Our Method

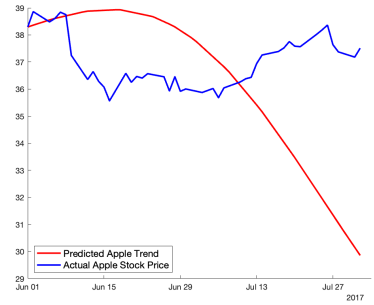

FIGURE 6.65. Barnett

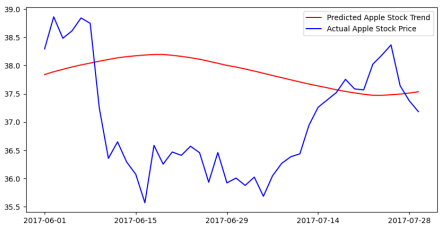

FIGURE 6.66. LSTM

	MAPE
Our Method	7.1837%
Barnett's Method	7.0596%
LSTM	3.2204%

TABLE 6.24. MAPE

Comparing the predicted trend to Apple's stock price for 08/01/19 – 09/30/19

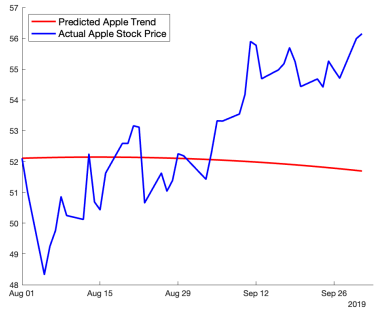

FIGURE 6.67. Our Method

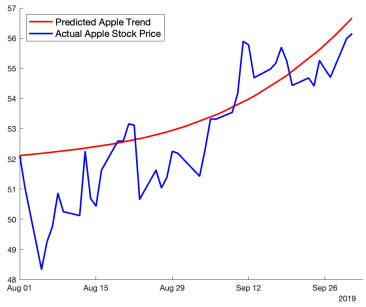

FIGURE 6.68. Barnett

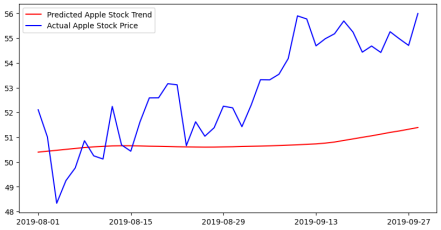

FIGURE 6.69. LSTM

	MAPE
Our Method	3.5193%
Barnett's Method	2.1060%
LSTM	4.2847%

TABLE 6.25. MAPE

Comparing the predicted trend to Apple's stock price for 02/01/22 – 03/31/22

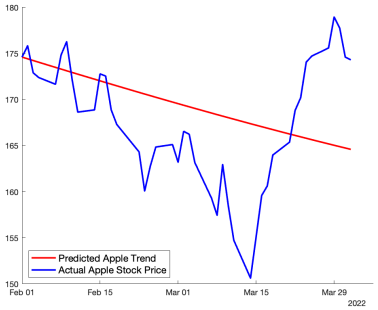

FIGURE 6.70. Our Method

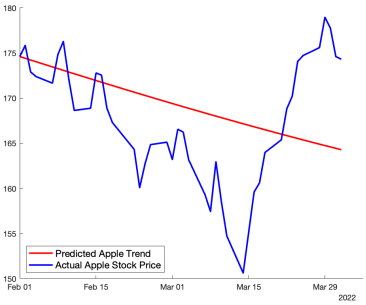

FIGURE 6.71. Barnett

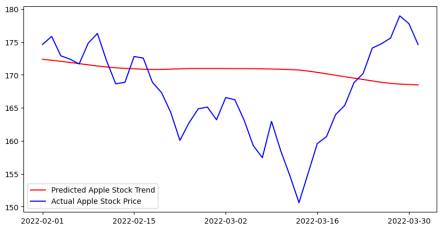

FIGURE 6.72. LSTM

	MAPE
Our Method	3.6558%
Barnett's Method	3.6245%
LSTM	3.8633%

TABLE 6.26. MAPE

3. Price Predictions Using Local Volatility

Comparing the predicted trend to Apple's stock price for 07/01/18 − 08/31/18

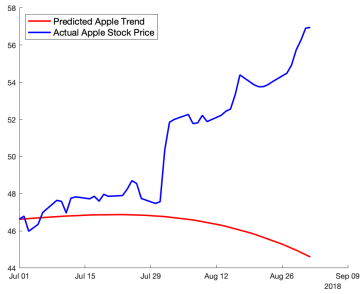

FIGURE 6.73. Barnett's Trend Prediction, MAPE= 7.8874%

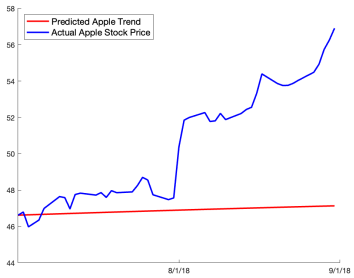

FIGURE 6.74. Our Trend Prediction in Round 1, MAPE= 4.3421%

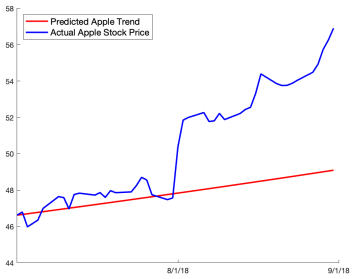

FIGURE 6.75. Our Trend Prediction in Round 2, MAPE= 1.7216%

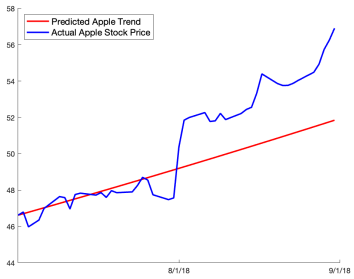

FIGURE 6.76. Our Trend Prediction in Round 3, MAPE= 2.9737%

Comparing the predicted price to Apple's stock price for 07/01/18 − 08/31/18

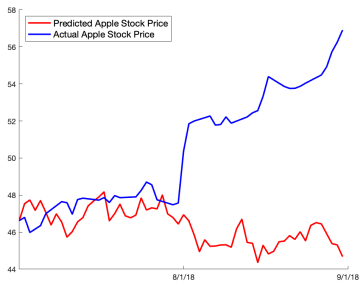

FIGURE 6.77. Barnett's Price Prediction, MAPE= 8.2471%

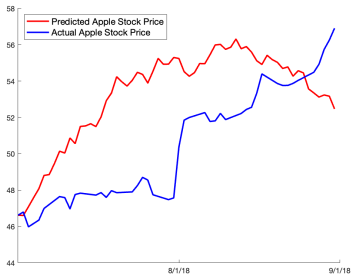

FIGURE 6.78. Our Price Prediction in Round 1, MAPE= 4.4218%

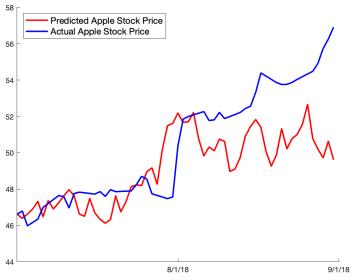

FIGURE 6.79. Our Price Prediction in Round 2, MAPE= 4.7431%

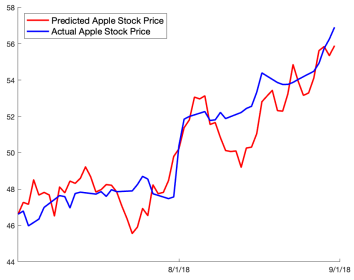

FIGURE 6.80. Our Price Prediction in Round 3, MAPE= 2.7033%

Comparing the predicted trend to Apple's stock price for 08/01/18 – 09/30/18

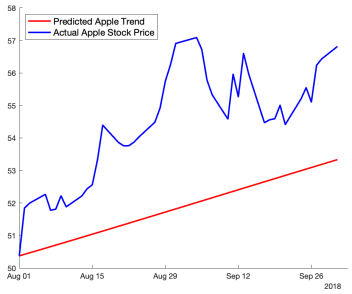

FIGURE 6.81. Barnett's Trend Prediction, MAPE= 4.7774%

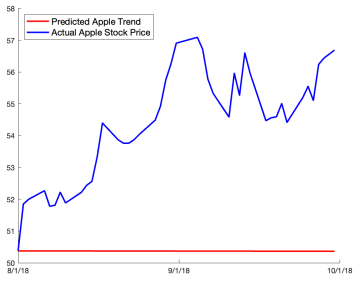

FIGURE 6.82. Our Trend Prediction in Round 1, MAPE= 7.4536%

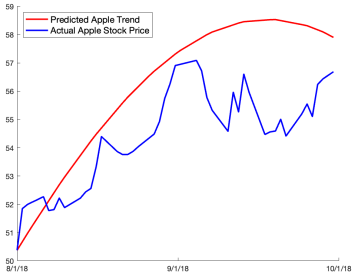

FIGURE 6.83. Our Trend Prediction in Round 2, MAPE= 3.2743%

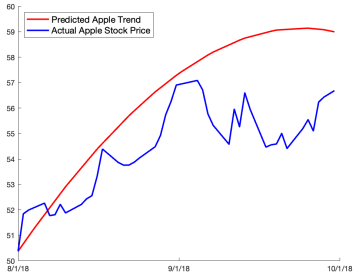

FIGURE 6.84. Our Trend Prediction in Round 3, MAPE= 3.7930%

Comparing the predicted price to Apple's stock price for 08/01/18 – 09/30/18

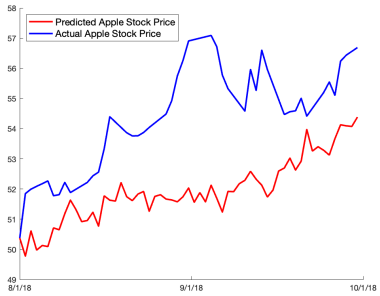

FIGURE 6.85. Barnett's Price Prediction, MAPE= 4.5264%

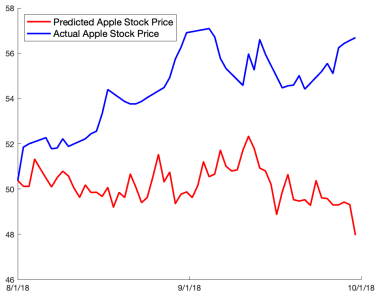

FIGURE 6.86. Our Price Prediction in Round 1, MAPE= 7.8243%

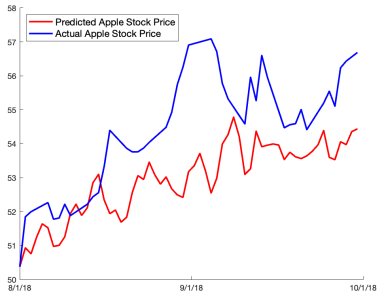

FIGURE 6.87. Our Price Prediction in Round 2, MAPE= 2.8032%

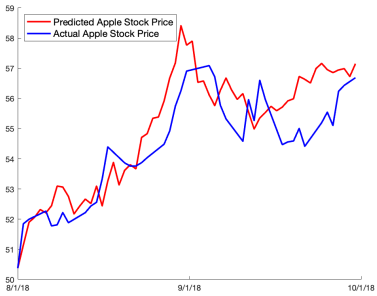

FIGURE 6.88. Our Price Prediction in Round 3, MAPE= 1.5813%

Comparing the predicted trend to Apple's stock price for 09/01/18 – 10/31/18

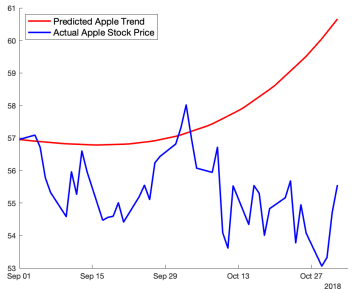

FIGURE 6.89. Barnett's Trend Prediction, MAPE= 4.2858%

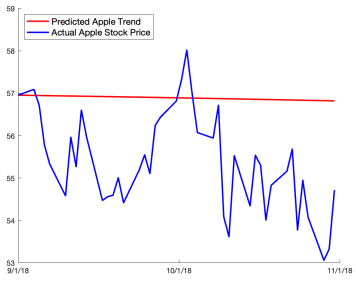

FIGURE 6.90. Our Trend Prediction in Round 1, MAPE= 2.8753%

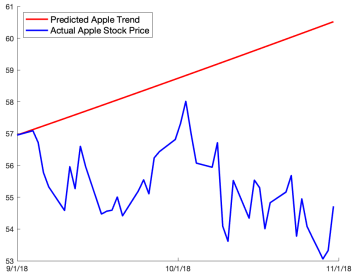

FIGURE 6.91. Our Trend Prediction in Round 2, MAPE= 6.1441%

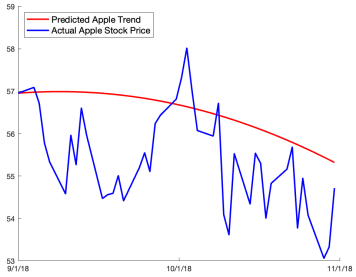

FIGURE 6.92. Our Trend Prediction in Round 3, MAPE= 2.2079%

Comparing the predicted price to Apple's stock price for 09/01/18 – 10/31/18

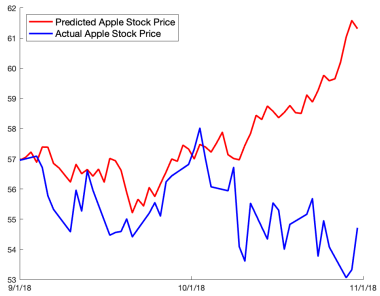

FIGURE 6.93. Barnett's Price Prediction, MAPE= 4.2121%

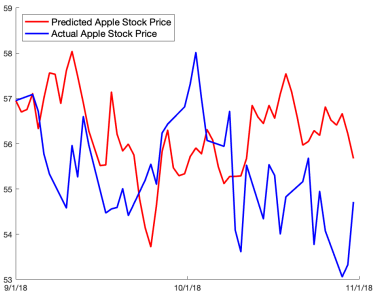

FIGURE 6.94. Our Price Prediction in Round 1, MAPE= 2.4876%

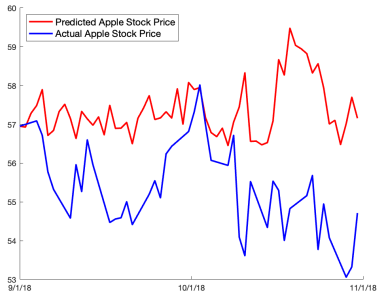

FIGURE 6.95. Our Price Prediction in Round 2, MAPE= 3.6387%

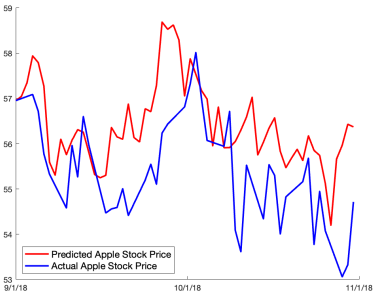

FIGURE 6.96. Our Price Prediction in Round 3, MAPE= 2.0186%

Comparing the predicted trend to Apple's stock price for 10/01/18 – 11/30/18

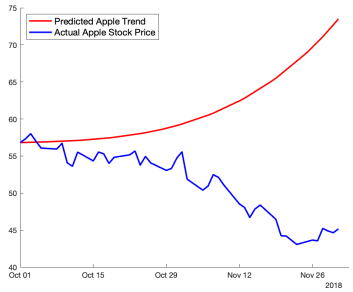

FIGURE 6.97. Barnett's Trend Prediction, MAPE= 21.0989%

FIGURE 6.98. Our Trend Prediction in Round 1, MAPE= 19.3004%

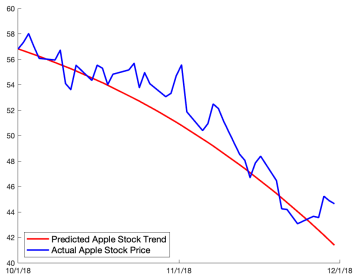

FIGURE 6.99. Our Trend Prediction in Round 2, MAPE= 2.7302%

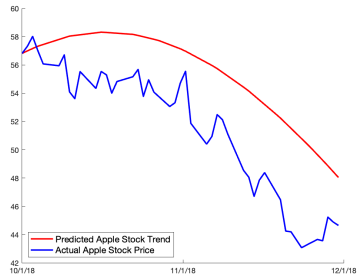

FIGURE 6.100. Our Trend Prediction in Round 3, MAPE= 8.1499%

Comparing the predicted price to Apple's stock price for 10/01/18 – 11/30/18

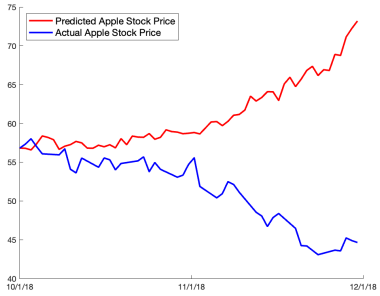

FIGURE 6.101. Barnett's Price Prediction, MAPE= 21.0670%

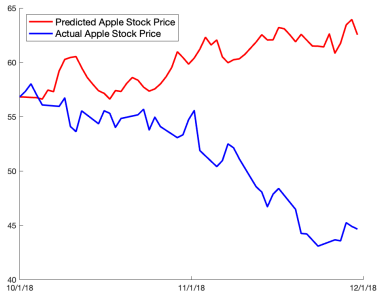

FIGURE 6.102. Our Price Prediction in Round 1, MAPE= 18.2078%

FIGURE 6.103. Our Price Prediction in Round 2, MAPE= 2.2345%

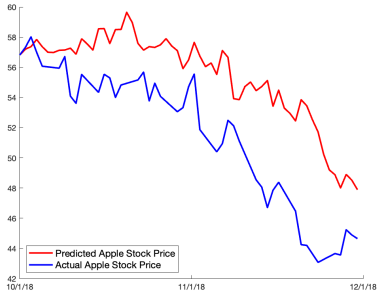

FIGURE 6.104. Our Price Prediction in Round 3, MAPE= 8.0997%

Comparing the predicted trend to Apple's stock price for 11/01/18 – 12/31/18

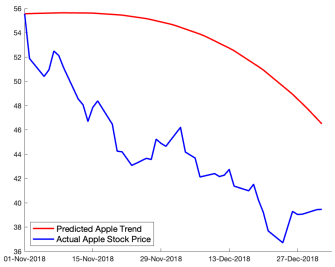

FIGURE 6.105. Barnett's Trend Prediction, MAPE= 21.3037%

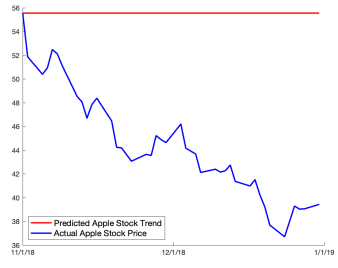

FIGURE 6.106. Our Trend Prediction in Round 1, MAPE= 26.8098%

FIGURE 6.107. Our Trend Prediction in Round 2, MAPE= 14.9107%

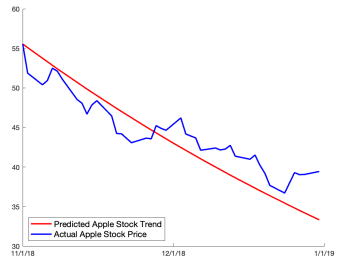

FIGURE 6.108. Our Trend Prediction in Round 3, MAPE= 5.8448%

Comparing the predicted price to Apple's stock price for 11/01/18 – 12/31/18

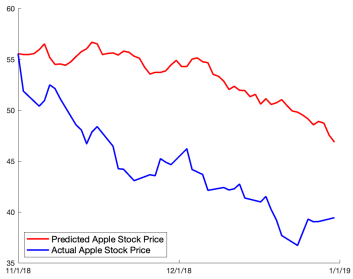

FIGURE 6.109. Barnett's Price Prediction, MAPE= 20.9704%

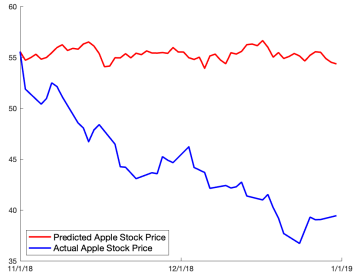

FIGURE 6.110. Our Price Prediction in Round 1, MAPE= 26.1985%

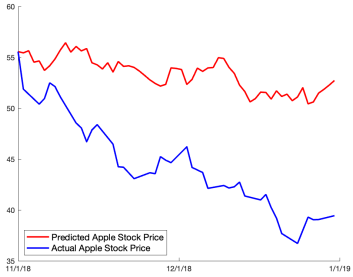

FIGURE 6.111. Our Price Prediction in Round 2, MAPE= 21.4799%

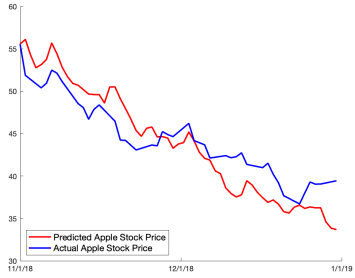

FIGURE 6.112. Our Price Prediction in Round 3, MAPE= 5.6771%

Comparing the predicted trend to Apple's stock price for 01/01/23 − 02/28/23

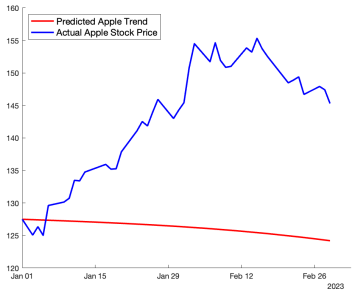

FIGURE 6.113. Barnett's Trend Prediction, MAPE= 11.4581%

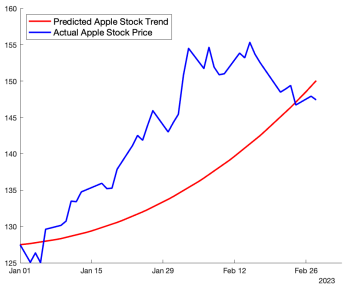

FIGURE 6.114. Our Trend Prediction in Round 1, MAPE= 5.3385%

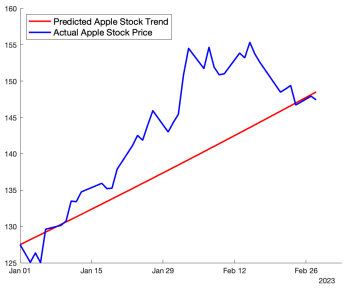

FIGURE 6.115. Our Trend Prediction in Round 2, MAPE= 3.6851%

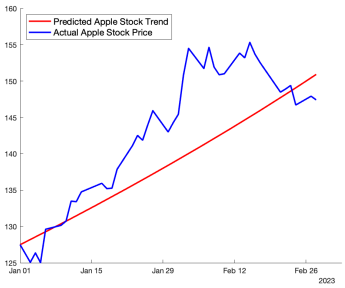

FIGURE 6.116. Our Trend Prediction in Round 3, MAPE= 3.4080%

Comparing the predicted price to Apple's stock price for 01/01/23 − 02/28/23

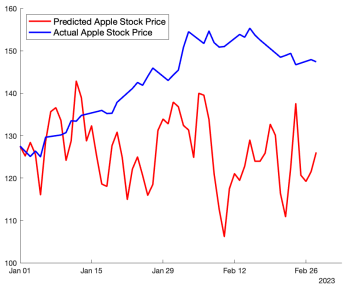

FIGURE 6.117. Barnett's Price Prediction, MAPE= 11.9323%

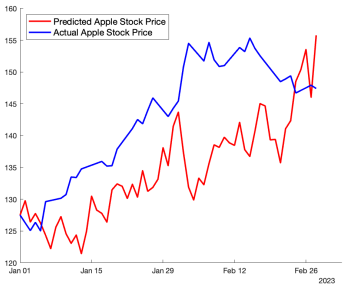

FIGURE 6.118. Our Price Prediction in Round 1, MAPE= 6.3508%

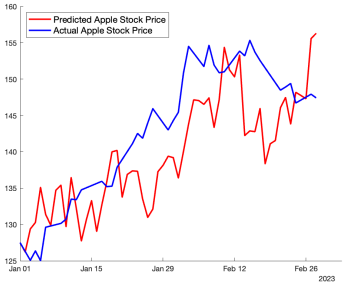

FIGURE 6.119. Our Price Prediction in Round 2, MAPE= 3.7201%

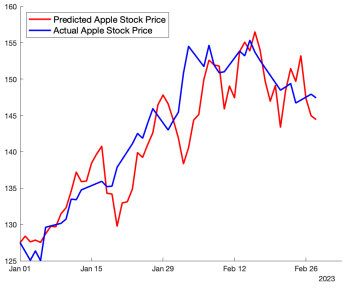

FIGURE 6.120. Our Price Prediction in Round 3, MAPE= 2.1084%

Comparing the predicted trend to Apple's stock price for 02/01/23 − 03/31/23

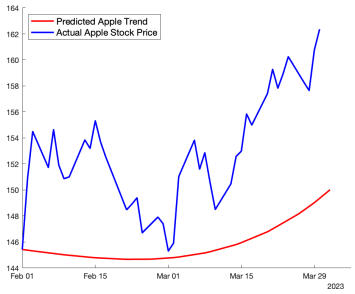

FIGURE 6.121. Barnett's Trend Prediction, MAPE= 4.6824%

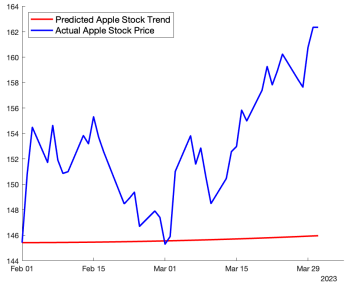

FIGURE 6.122. Our Trend Prediction in Round 1, MAPE= 4.7972%

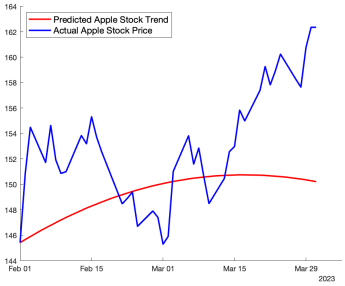

FIGURE 6.123. Our Trend Prediction in Round 2, MAPE= 2.9682%

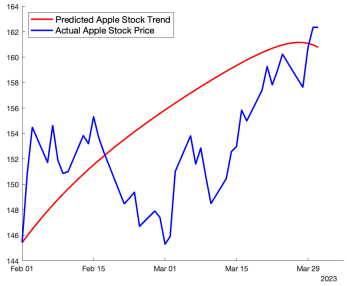

FIGURE 6.124. Our Trend Prediction in Round 3, MAPE= 2.8974%

Comparing the predicted price to Apple's stock price for 02/01/23 − 03/31/23

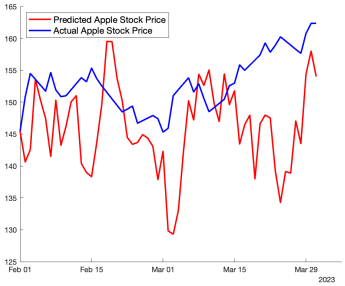

FIGURE 6.125. Barnett's Price Prediction, MAPE= 5.3794%

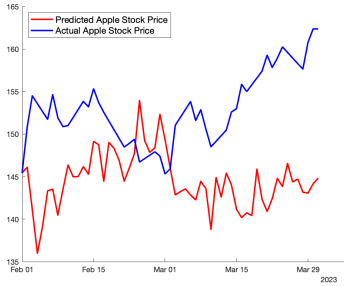

FIGURE 6.126. Our Price Prediction in Round 1, MAPE= 5.8635%

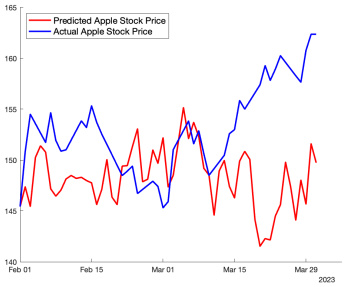

FIGURE 6.127. Our Price Prediction in Round 2, MAPE= 3.7329%

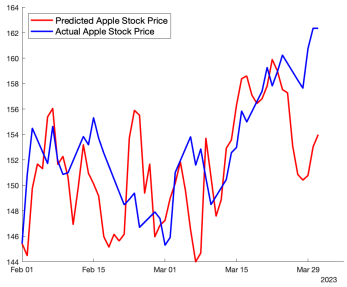

FIGURE 6.128. Our Price Prediction in Round 3, MAPE= 2.3297%

4. Twelve Iterations on Iterative Improvement Method

Trends on 10/1/18 - 11/31/18

FIGURE 6.129. No Iteration, MAPE= 19.3004%

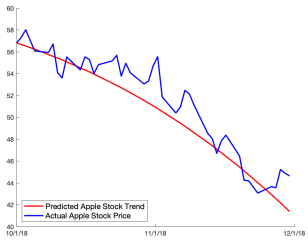

FIGURE 6.130. 1st Iteration, MAPE= 2.7302%

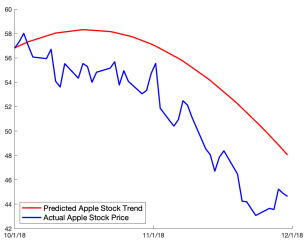

FIGURE 6.131. 2nd Iteration, MAPE= 8.1499%

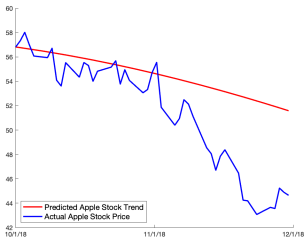

FIGURE 6.132. 3rd Iteration, MAPE= 6.8533%

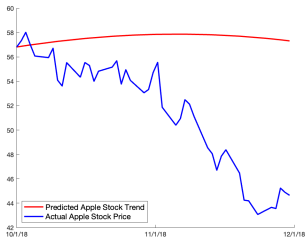

FIGURE 6.133. 4th Iteration, MAPE= 13.0803%

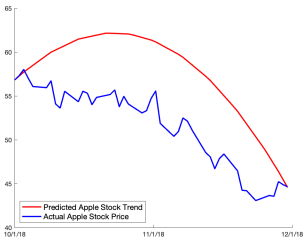

FIGURE 6.134. 5th Iteration, MAPE= 11.9754%

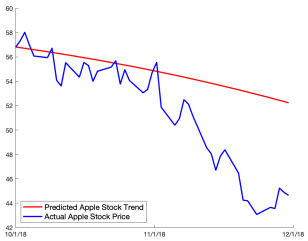

FIGURE 6.135. 6th Iteration, MAPE= 7.3630%

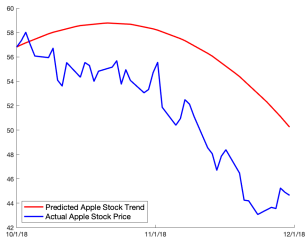

FIGURE 6.136. 7th Iteration, MAPE= 10.5799%

FIGURE 6.137. 8th Iteration, MAPE= 12.3594%

FIGURE 6.138. 9th Iteration, MAPE= 11.5477%

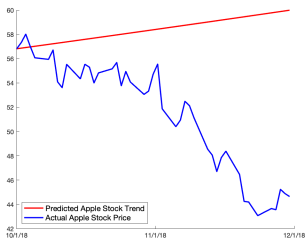

FIGURE 6.139. 10th Iteration, MAPE= 14.8899%

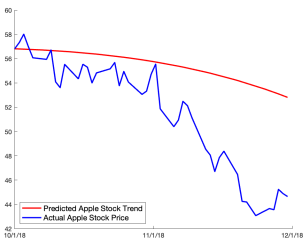

FIGURE 6.140. 11th Iteration, MAPE= 8.6404%

Price 10/1/18-11/31/18

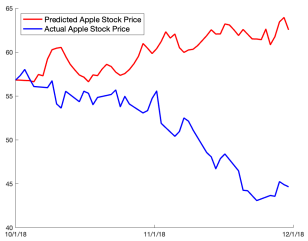

FIGURE 6.141. No Iteration, MAPE= 18.2078%

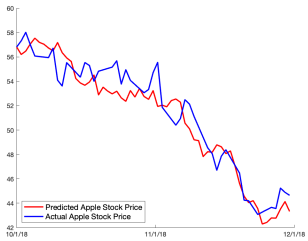

FIGURE 6.142. 1st Iteration, MAPE= 2.2345%

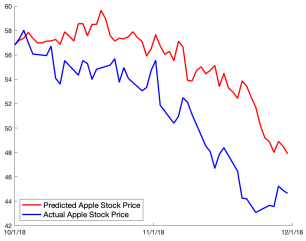

FIGURE 6.143. 2nd Iteration, MAPE= 8.0997%

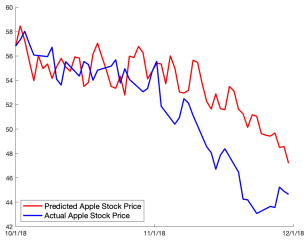

FIGURE 6.144. 3rd Iteration, MAPE= 5.9778%

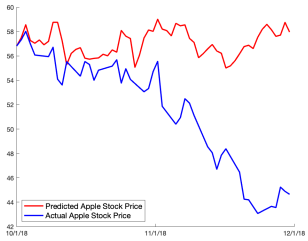

FIGURE 6.145. 4th Iteration, MAPE= 12.0738%

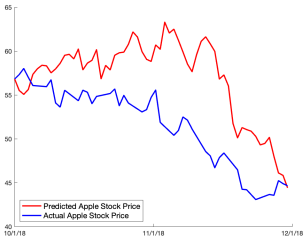

FIGURE 6.146. 5th Iteration, MAPE= 11.2112%

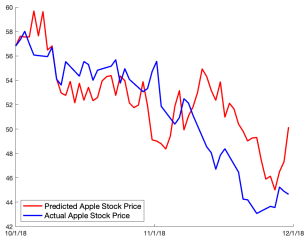

FIGURE 6.147. 6th Iteration, MAPE= 5.3583%

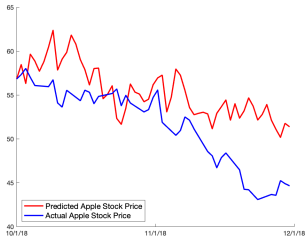

FIGURE 6.148. 7th Iteration, MAPE= 8.6343%

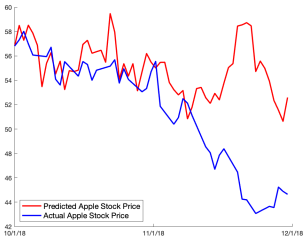

FIGURE 6.149. 8th Iteration, MAPE= 8.2745%

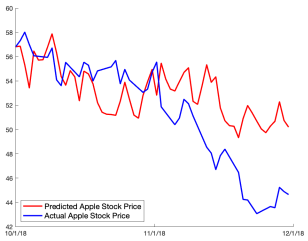

FIGURE 6.150. 9th Iteration, MAPE= 6.5177%

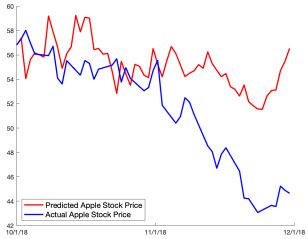

FIGURE 6.151. 10th Iteration, MAPE= 8.7052%

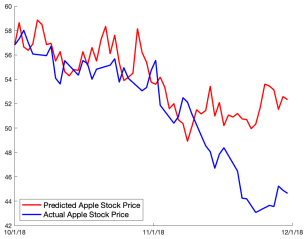

FIGURE 6.152. 11th Iteration, MAPE= 6.3152%

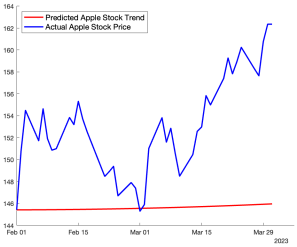

FIGURE 6.153. No Iteration, MAPE= 4.7972%

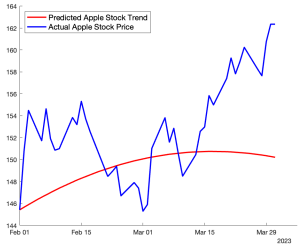

FIGURE 6.154. 1st Iteration, MAPE= 2.9682%

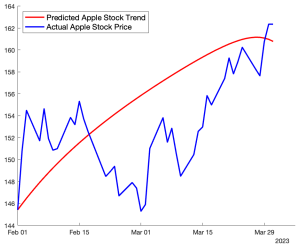

FIGURE 6.155. 2nd Iteration, MAPE= 2.8974%

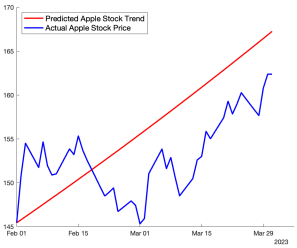

FIGURE 6.156. 3rd Iteration, MAPE= 3.6137%

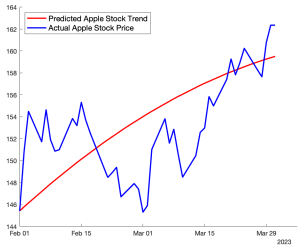

FIGURE 6.157. 4th Iteration, MAPE= 2.3491%

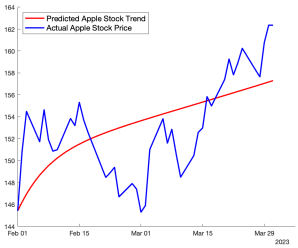

FIGURE 6.158. 5th Iteration, MAPE= 2.1357%

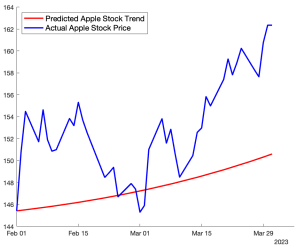

FIGURE 6.159. 6th Iteration, MAPE= 3.5824%

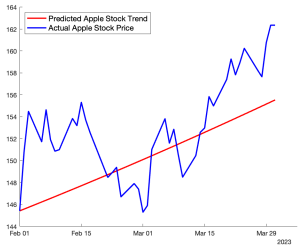

FIGURE 6.160. 7th Iteration, MAPE= 2.3863%

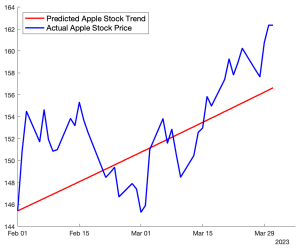

FIGURE 6.161. 8th Iteration, MAPE= 2.2239%

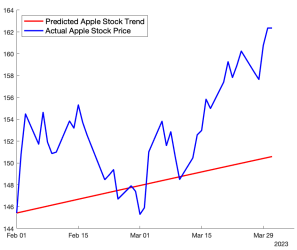

FIGURE 6.162. 9th Iteration, MAPE= 3.3632%

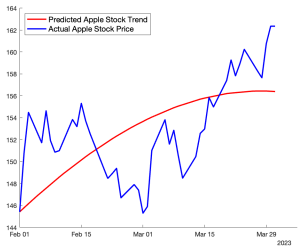

FIGURE 6.163. 10th Iteration, MAPE= 2.4119%

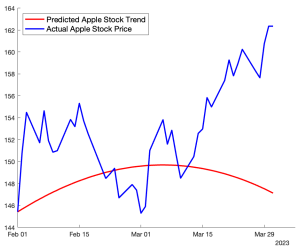

FIGURE 6.164. 11th Iteration, MAPE= 3.4086%

Price 02/01/23 - 03/31/23

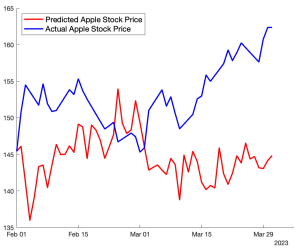

FIGURE 6.165. No Iteration, MAPE= 5.8635%

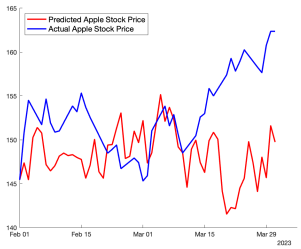

FIGURE 6.166. 1st Iteration, MAPE= 3.7329%

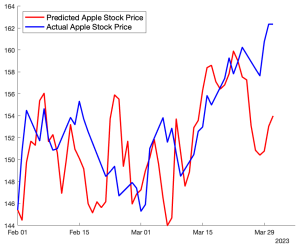

FIGURE 6.167. 2nd Iteration, MAPE= 2.3297%

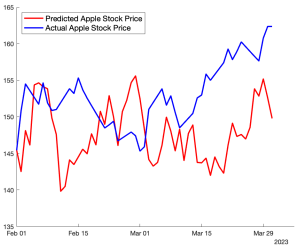

FIGURE 6.168. 3rd Iteration, MAPE= 4.3541%

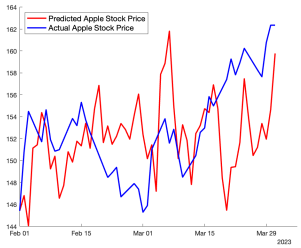

FIGURE 6.169. 4th Iteration, MAPE= 2.7508%

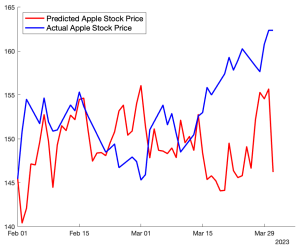

FIGURE 6.170. 5th Iteration, MAPE= 3.4530%

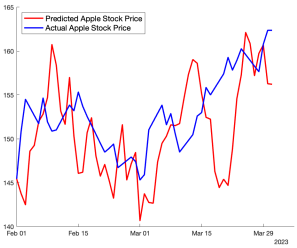

FIGURE 6.171. 6th Iteration, MAPE= 2.9491%

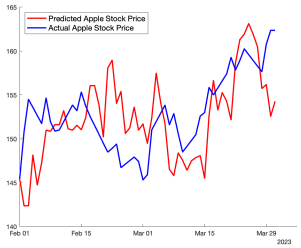

FIGURE 6.172. 7th Iteration, MAPE= 2.5813%

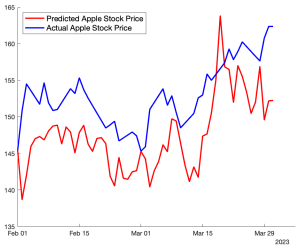

FIGURE 6.173. 8th Iteration, MAPE= 3.7173%

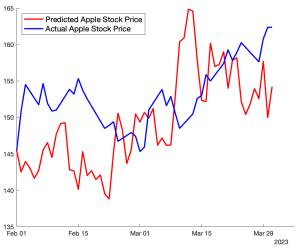

FIGURE 6.174. 9th Iteration, MAPE= 4.1301%

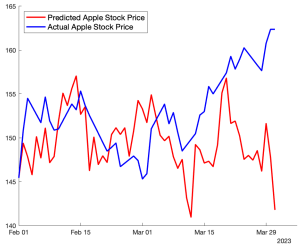

FIGURE 6.175. 10th Iteration, MAPE= 3.4040%

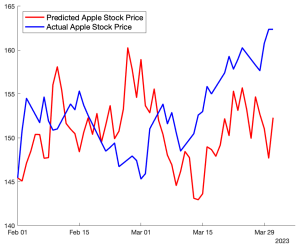

FIGURE 6.176. 11th Iteration, MAPE= 3.4709%

CHAPTER 7

Algorithms

This chapter contains pseudo-codes for our algorithms, while the actual programs may be found in our Appendix. Our model has five main algorithms and many distinct functions. While it would be straightforward to integrate all seven algorithms into one, we decided to keep them distinct so that we could better understand each method.

The five main algorithms are as follows. **Get_Constants_Final.m** gets ten initial ψs where $G(\psi) < G(\mathbf{0})$ and $G(\psi) < G(J)$. **Get_Constants_Which_one.m** decides which of the ten ψs used to solve the differential equation. **H1_All_Psis.m** minimizes $G(\psi)$ using the ψs recovered from "Get_Constants_Which_one.m". **Apply_KJ_C_to_predict_good_psi.m** calculates the averages of the solution to the differential equations using the minimized ψs then picks the ψ that gives the solution closest to the average. **Apple_Stock_Pred_Guess.m** calculates price predictions.

The functions we created are **boundary.m**, **delay_buy_kj.m**, **J_to_KJ.m**, **KJ_to_J.m**, **T_reg_data.m**, **T_star_data.m**, **u.m**, **u_psi.m**, **u_psi_p.m**, **T_reg_xi.m**, and **get_max_diff.m**. The **boundary.m** is used to obtain the Neuberger gradient and **delay_buy_kj.m** is the function handle which is used in MATLAB command **dde23** to solve the delay differential equation. As shown in Chapter 5 Section 3, the number of coefficients in our differential equation system fluctuates depending on the availability of data. However, for coding convenience, we want our **dde23** setup to remain consistent throughout all time periods. As a result, we leave the coefficient at zero for the terms with missing information. For $v < 126$, we locate the zeros as follows:

$$\begin{bmatrix} c_1 \cdots c_7 & 0 & c_8 \cdots c_{17} & 0 & c_{18} \cdots c_{20} & 0 & c_{21} \cdots c_{26} & 0 \\ 0 & 0 & 0 & 0 & c_{27} \cdots c_{31} & 0 & 0 & 0 \\ c_{32} & 0 & c_{33} & c_{34} & 0 & 0 & 0 & \end{bmatrix}.$$

For $126 \leq v < 159$, we insert zeros as indicated below:

$$\begin{bmatrix} c_1 \cdots c_7 & 0 & c_8 \cdots c_{17} & 0 & c_{18} \cdots c_{20} & 0 & c_{21} \cdots c_{26} & 0 \\ c_{27} \cdots c_{35} & 0 & 0 & 0 & c_{36} \cdots c_{39} & 0 & 0 & 0 \end{bmatrix}.$$

For $159 \leq v < 228$, we fill in the rest with zeros as follows:

$$\begin{bmatrix} c_1 \cdots c_7 & 0 & c_8 \cdots c_{28} & 0 & c_{29} \cdots c_{44} & 0 & 0 & 0 \end{bmatrix}.$$

For $228 \leq v < 240$,

$$\begin{bmatrix} c_1 \cdots c_{29} & 0 & c_{30} \cdots c_{48} \end{bmatrix}.$$

This process is done through the function **KJ_to_J.m**. During each time period, we set up T and T^* accordingly and this is done through **T_reg_data.m** and **T_star_data.m**. The similarly named functions such as **T_reg_x1.m** are the T and T^* functions for each equation. Further, during each time period, we calculate $G(J)$ and to do so, we must convert J into the correct form. As mentioned in Remark 1.2, the way Barnett coded the coefficients is different than that is shown in (1.1). This reordering of Barnett's coefficients is done through **J_to_KJ.m**. Lastly, u, u_ψ and u_{ψ_m} are calculated using the functions **u.m**, **u_psi.m**, and **u_psi_p.m** respectively.

Remark 7.1. There are two adjustable parameters in **"Get_Constant_Final.m"**: z and bound. The bound is βMax_v for some $\beta \in (0, 3)$, and $z \in (0, 3]$ determines the range of the coefficients selected in the process of finding the initial points. Each iteration of **"Get_Constant_Final.m"** includes solving the delay differential equation. Setting $z = 0.1$ and $\beta = 2$ produced adequate results in most cases, but there are some cases where this method may require an excessive amount of time, when $z = 0.1$

is used. In the latter case, we recommend adjusting z to 1 or 0.01. As for the bound, we suggest leaving $\beta = 2$. However, one can adjust this parameter.

Algorithm 1 Get_Constants_Final.m Refer to Chapter 5 Section 8

Calculate $G(J)$, $G(\mathbf{0})$, Max_v.
Set $z \in (0, 3]$ and for the bound set $\beta \in (0, 3)$.
for i=1:13 **do**
    if $G_i(\mathbf{0}) < G_i(J)$ **then**
        Step 1: Find three terms which effects $G(\psi_i)$ the least.
        Step 2: Set the rest of the coefficients as zeros then find the coefficients.
        while $j < 20$ **do**
            Choose $c_2, c_3 \in [-z, z]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 1$ with smallest $G(\psi 1)$.
        while $j < 20$ **do**
            Choose $c_1, c_3 \in [-z, z]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 2$ with smallest $G(\psi 2)$.
        while $j < 20$ **do**
            Choose $c_1, c_2 \in [-z, z]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 6$ with smallest $G(\psi 3)$.
    end if
    if $G_i(J) \leq G_i(\mathbf{0})$ **then**
        Step 1: Find three terms which effects $G(\psi_i)$ the least.
        Step 2: Set the rest of the coefficients as corresponding elements of J then find the coefficients
        while $j < 20$ **do**
            Choose $c_2, c_3 \in [-z + e_k^T J, z + e_k^T J]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 4$ with smallest $G_i(\psi 4)$.
        while $j < 20$ **do**
            Choose $c_1, c_3 \in [-z + e_k^T J, z + e_k^T J]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 5$ with smallest $G_i(\psi 5)$.
        while $j < 20$ **do**
            Choose $c_1, c_2 \in [-z + e_k^T J, z + e_k^T J]$ and calculate B_1, B_2, and B_3.
        end while
        Choose the $\psi 6$ with smallest $G_i(\psi 6)$.
    end if
    From $\{\psi 1, \psi 2, \psi 3, \psi 4, \psi 5, \psi 6\}$ choose the one with smallest $G_i(\psi i)$
end for

Algorithm 2 Get_Constants_Which_one.m

1: Get the initial psis found in **Get_Constants_Final.m**
2: Calculate $G_8(\psi)$s and get indices for lowest five
3: Calculate $G(\psi)s$ and get the indices for lowest five
4: Print the indices of overlapping ψs

Algorithm 3 H1_All_Psis.m

1: Import the indicated psis from **Get_Constants_Which_one.m**
2: Set $z \in (0, 3]$ and for the bound set $\beta \in (0, 3)$
3: **for** i=1:amount of psis **do**
4: Use one of the psis imported
5: Calculate T, u, u_ψ, u'_ψ the $\mathcal{L}^2$ gradient, and the Neuberger gradient using **T_reg_data.m**, **u.m**, **u_psi.m**, **u_psi_p.m**, **T_star_data.m**, and **boundary.m** respectively.
6: Initialize α
7: **while** $k < 300$ **do**
8: $\psi_{k+1} = \psi_k - \alpha \nabla^{\psi_k}_{\mathcal{H}^1} G$
9: **end while**
10: **end for**

Algorithm 4 Apply_KJ_C_to_predict_good_psi.m

1: Import the minimized psis from **H1_All_Psis.m**
2: Solve the delay differential equations using the imported psis as the coefficients.
3: Find the average of the solutions.
4: Choose the psi which gives the solution closest to the average of the solutions.

Algorithm 5 Apple_Stock_Pred_Guess.m

1: Import the chosen psi from **Apply_KJ_C_to_predict_good_psi.m**
2: $i = 0$
3: $S = [\,]$
4: **while** $i < 1000$ **do**
5: Solve the Stochastic Differential equation using **gbm** command in MATLAB 500 times then average the solutions.
6: Append this solution to the solutions set S
7: **end while**
8: Average the solutions in S and find the one closest to the average. This is our stock price prediction.

CHAPTER 8

Conclusions and Future Directions

Our primary goal is to improve our prediction of Apple's stock price and trend by acquiring better coefficients for (1.1). Because our inverse problem is ill-posed, we use Nayak's regularization method to find the coefficients. However, since Nayak's regularization is designed for ill-posed problems involving injective operators, we must modify our techniques and theories to accommodate our non-injective operator. We do indeed follow Nayak's proposed method for updating ψ using the Neuberger gradient, and we use Nayak's functional as a measure of goodness throughout our research. But since the functional G corresponding to our operator is just convex and not strictly convex, we obtain non-unique minimizers that depend on the starting points of the decent process. Furthermore, in many cases, the effects of the flaws in (1.1) caused the solutions to be outside of an acceptable range. To address these issues, we use our new initialization approach, as well as a mechanism for countering faults in our mathematical model, to obtain improved coefficients for (1.1).

With our new initialization process for the descent process and our remedy for the flaws in (1.1), our method outperformed Barnett's model 92% of the time and outperformed the LSTM model 93% of the time when predicting Apple's stock trend. In addition, by applying the iteration improvement technique with the aforementioned methods, our model outperformed Barnett's model 100% times when predicting Apple's stock prices. Though our findings remain promising, we believe building a better underlying mathematical model to describe each economic variable would yield even better predictions.

REFERENCES

[1] Federal Reserve Economic Data. Federal Funds Effective Rate. https://fred.stlouisfed.org/series/FEDFUNDS.

[2] Gradient. https://www.mathworks.com/help/matlab/ref/gradient.html.

[3] Historical Inflation Rates: 1914-2023. U.S. Inflation Calculator. https://www.usinflationcalculator.com/inflation/historical-inflation-rates/.

[4] Macrotrends, Apple Net Income. https://www.macrotrends.net/stocks/charts/AAPL/apple/net-income.

[5] Nasdaq. Apple (AAPL) Option Chain. https://www.nasdaq.com/market-activity/stocks/aapl/option-chain.

[6] OECD. Consumer Confidence Index (CCI). https://data.oecd.org/leadind/consumer-confidence-index-cci.htm.

[7] Random vs. Non-random Walk. https://school.stockcharts.com/doku.php?id=overview%3Arandom_walk_theory.

[8] Statista. Apple's iPhone Revenue from 3rd Quarter 2007 to 4th Quarter 2022. https://www.statista.com/statistics/263402/apples-iphone-revenue-since-3rd-quarter-2007/.

[9] U.S. Bureau of Labor Statistics. Consumer Price Index Historical Tables for U.S. City Average. https://www.bls.gov/regions/mid-atlantic/data/consumerpriceindexhistorical_us_table.htm.

[10] What are Recurrent Neural Networks? https://www.ibm.com/topics/recurrent-neural-networks.

[11] What is Machine Learning? https://www.ibm.com/topics/machine-learning.

[12] Yahoo! Finance. Apple Inc. Analysis. https://finance.yahoo.com/quote/AAPL/history?period1=1199145600&period2=1673913600&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true.

[13] Yahoo! Finance. Nasdaq-100 Index. https://finance.yahoo.com/quote/%5ENDX/history?period1=1199145600&period2=1674000000&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true.

[14] Yahoo! Finance. SPDR Gold Shares (GLD). https://finance.yahoo.com/quote/GLD/history?period1=1196467200&period2=1674000000&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true.

[15] YCharts. Apple PE Ratio. https://ycharts.com/companies/AAPL/pe_ratio.

[16] YCharts. Apple Research and Development Expanse. https://ycharts.com/companies/AAPL/r_and_d_expense.

[17] YCharts. Apple Stock Buybacks. https://ycharts.com/companies/AAPL/stock_buyback.

[18] Jessica A. Barnett. *Modeling Stock Prices with Differential Equations*. PhD thesis, University of Alabama at Birmingham, 2019.

[19] Daniel Bump. *Automorphic Forms and Representations*. Number 55. Cambridge university press, 1998.

[20] John Y. Campbell, Andrew Wen-Chuan Lo, and Archie Craig MacKinlay. *The Econometrics of Financial Markets*. Princeton University Press, 2011.

[21] Rian Dolphin. LSTM Networks: A Detailed Explanation. https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9, Dec 2021.

[22] Rodney D. Driver. *Ordinary and Delay Differential Equations*, volume 20. Springer-Verlag, 1977.

[23] John M Erdman. Functional Analysis and Operator Algebras: An Introduction. *Version October*, 4, 2015.

[24] Jake Frankenfield. Artificial Intelligence: What it is and How it is Used. https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp, Jan 2023.

[25] Adam Hayes. Volatility: Meaning in Finance and How it Works with Stocks. https://www.investopedia.com/terms/v/volatility.asp, Jan 2023.

[26] Hanne Kekkonen and Yury Korolev. Inverse Problems (Michaelmas 2019). http://www.damtp.cam.ac.uk/research/cia/inverse-problems-michaelmas-2019.

[27] Joseph B. Keller. Inverse Problems. *The American Mathematical Monthly*, 83(2):107–118, 1976.

[28] Ian Knowles and Ajay Mahato. The Inverse Volatility Problem for American Options. *Discrete & Continuous Dynamical Systems-Series S*, 13(12), 2020.

[29] Paul Kosakowski. Financial Markets: Random, Cyclical or Both? https://www.investopedia.com/articles/financial-theory/09/markets-cyclical-vs-random.asp, Aug 2022.

[30] Andrew W Lo and Jiang Wang. Implementing Option Pricing Models when Asset Returns are Predictable. *The Journal of Finance*, 50(1):87–129, 1995.

[31] Ajay Mahato. *The Inverse Volatility Problem for American Options*. PhD thesis, University of Alabama at Birmingham, 2014.

[32] Abinash Nayak. *Inverse Problems, Regularization and Its Applications*. PhD thesis, The University of Alabama at Birmingham, 2019.

[33] Paul A. Samuelson. Rational Theory of Warrant Pricing (with appendix by Henry P. McKean Jr. Selecta). *Industrial management review*, 6(2):13–39, 1965.

[34] Lawrence F Shampine, S Thompson, and J Kierzenka. Solving Delay Differential Equations with dde23. *URL http://www. runet. edu/˜ thompson/webddes/tutorial. pdf*, 2000.

[35] Rahul Sisodia. How to Predict Apple Stock Price. `https://python.plainenglish.io/predicting-apple-stock-price-36f329cda530`, Jan 2022.

[36] Wikipedia. Geometric Brownian Motion — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Geometric%20Brownian%20motion&oldid=1140966364`, 2023. [Online; accessed 20-April-2023].

[37] Wikipedia. Stochastic Differential Equation — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Stochastic%20differential%20equation&oldid=1146698628`, 2023. [Online; accessed 20-April-2023].

Appendix A: Get_Constants_Final.m

Get_Constants_Final.m

```
clc
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code finds 10 possible initial points for                        %
% the descent process.                                                  %
% Please read Chapter 5 of my dissertation before using                 %
% this code.                                                            %
% The overall idea is finding initial points with                       %
% smaller value of G than G(J) or G(0).                                 %
% Inputs: v, a, b, s, l                                                 %
% Hyperparameters: bound & dd                                           %
% Output: 10 initial constant functions for the descent                 %
%         process                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Warnings                                       %
% 1) You must run Barnett's code first to obtain the                    %
%    coefficients found using the least squares method.                 %
% 2) dde23 may take a while to run at times. If that is                 %
% the case, I suggest stopping the program and                          %
```

```
% re-running it.                                                    %
% 3) Be sure to adjust the Hyperparamters. One may use              %
% the suggested hyperparameters which can be found in               %
% Appendix until getting a good feel for adjusting                  %
% parameters on their own.                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Set Up v, a, b, s, l, and lags                         %
% v : Starting month where v=28 is 10/1/96 and                      %
% v=29=11/1/96 and so on,                                           %
%     The range of v is 28<=v<=344.                                 %
%     Upper range can be extended as a user collects                %
%     more data.                                                    %
% a : Starting month of the past data we want to use.               %
% b : b should always equal to v.                                   %
% s : Step size. Since we are working with data                     %
%   interplated daily,                                              %
%     our step size is always 1. This is used in                    %
%     integral estimation.                                          %
% l : This sets how far into the future is predicted                %
% lags: This is the delays for each variable x1-x13.                %
% We keep it as shown below to follow Barnett's code.               %
% However, if a user desires, this can be changed to                %
% [20,17] only and change the delay_buy_kj accordingly.             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=343;
```

```
a=v-1;
b=v;
s=1;
l=2;
lags=[12,1,1,20,1,12,17,1,1,1,1,1,1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Hyperparameters                                        %
% max_diff: This is Max_v described in Chapter 5                        %
% Section 9. Gets the maximum difference of past stock                  %
% prices in range of one month and two months.                          %
% bound: This is the beta*max_diff described in                         %
%Chapter 5 Section 9. Sets the bound for the forecasts.                 %
% The range is 0<beta<3                                                 %
% dd: This is the z described in Chapter 5 Section 8.2.                 %
%     Sets the range for one or two variables                           %
%       (please read Chapter 5).                                        %
%     The range is 0<dd<=3.                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_diff = get_max_diff(v);
bound = max_diff*2;
dd = .1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Preprocess the data                                 %
% This portion is a modified portion of Barnett's data                  %
% preprocessing process. If a user decides to extend the%
% data, 'mo' must be changed accordingly.                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];

%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;
    1156;1187;1217;1248;1278;1309;1340;1370;1401;1431;
    1462;1493;1522;1553;1583;1614;1644;1675;1706;1736;
```

```
    1767;1797;1828;1859;1887;1918;1948;1979;2009;2040;
    2071;2101;2132;2162;2193;2224;2252;2283;2313;2344;
    2374;2405;2436;2466;2497;2527;2558;2589;2617;2648;
    2678;2709;2739;2770;2801;2831;2862;2892;2923;2954;
    2983;3014;3044;3075;3105;3136;3167;3197;3228;3258;
    3289;3320;3348;3379;3409;3440;3470;3501;3532;3562;
    3593;3623;3654;3685;3713;3744;3774;3805;3835;3866;
    3897;3927;3958;3988;4019;4050;4078;4109;4139;4170;
    4200;4231;4262;4292;4323;4353;4384;4415;4444;4475;
    4505;4536;4566;4597;4628;4658;4689;4719;4750;4781;
    4809;4840;4870;4901;4931;4962;4993;5023;5054;5084;
    5115;5146;5174;5205;5235;5266;5296;5327;5358;5388;
    5419;5449;5480;5511;5539;5570];


%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
```

```
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;



%-------------------------------------------------------
% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector

%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);  % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
```

```
%Interpolated Apple stock price
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price
%and inearly interpolated from there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
vnz3 = find(SandP500 ~= 0);
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
```

```
ir_nz = ir1(ir1 ~= 0);
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
rd = interp1(vnz8, rd_nz, iv, 'linear')';

%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
```

```
vnz12 = find(iphone1 ~= 0);
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting
%on December 1,2007, so now we account for that and start
   all vectors on
%January 1, 2008
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
```

```
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-----------------------------------------------------------
%Complete the same process with the older data files
%-----------------------------------------------------------
%Apple's Net Income
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
```

```
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
%Apple didn't have any
%buybacks prior to 2008 so this variable is 0
buyback_old=zeros(mo1(end),1);
%Apple's EPS
eps_old=csvread('old_eps.csv',1,1);

%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);                    %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);    %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0); %Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
%Interpolated APPL function is above


%The same process is repeated for each variable
```

```
%Apple's Net Income
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
cpi_onz = cpi_old(cpi_old ~= 0);
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';
```

```
%GLD
gld_onz = gld_old(gld_old ~= 0);
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);

%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';

%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';

%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';

%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);
```

```
%Confidence Index
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';

%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special
%cases) instead of January 1,2018, which is where the new
   data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
pe_old=AAPL_old./eps(1:mo1(end)-1); %caculate P/E ratio
   using EPS and stock price
```

```
%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];

mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Set up the time steps                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t= (mo(a):mo(b));
t_size = size(t);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Setting up x_1(t)~x_13(t)                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_1(t)=cpi(mo(a):mo(b),:);
x_2(t)=inflation(mo(a):mo(b),:);
x_3(t)=pe(mo(a):mo(b),:);
x_4(t)=ir(mo(a):mo(b),:);
x_4d(t)=ir(mo(a-20):mo(a-20)+t_size(2)-1,:); % 20 month
   delay
x_5(t)=NASDAQ100(mo(a):mo(b),:);
x_6(t)=rev(mo(a):mo(b),:);
x_7(t)=rd(mo(a):mo(b),:);
x_7d(t)=rd(mo(a-17):mo(a-17)+t_size(2)-1,:); % 17 month
   delay
x_8(t)=AAPL(mo(a):mo(b),:);
x_9(t)=gld(mo(a):mo(b),:);
x_10(t)=sp500(mo(a):mo(b),:);
x_11(t)=iphone(mo(a):mo(b),:);
x_12(t)=conf(mo(a):mo(b),:);
x_13(t)=buyback(mo(a):mo(b),:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set up g                                     %
% See Chapter 5 Section 3 for reference.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = [x_1(t)-x_1(1);x_2(t)-x_2(1);...
    x_3(t)-x_3(1);x_4(t)-x_4(1);...
```

```
    x_5(t)-x_5(1);x_6(t)-x_6(1);...
    x_7(t)-x_7(1);x_8(t)-x_8(1);...
    x_9(t)-x_9(1);x_10(t)-x_10(1);...
    x_11(t)-x_11(1);x_12(t)-x_12(1);...
    x_13(t)-x_13(1)];
g_size=size(g);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Define the hisotry vector for dde23                        %
% See Chapter 2 Section 2 for details.                                %
% We use constant functions as our history vector.                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0(1)=cpi(mo(v));
x0(2)=inflation(mo(v));
x0(3)=pe(mo(v));
x0(4)=ir(mo(v));
x0(5)=NASDAQ100(mo(v));
x0(6)=rev(mo(v));
x0(7)=rd(mo(v));
x0(8)=AAPL(mo(v));
x0(9)=gld(mo(v));
x0(10)=sp500(mo(v));
x0(11)=iphone(mo(v));
x0(12)=conf(mo(v));
x0(13)=buyback(mo(v));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Get Jessica's Coefficients                             %
% J_to_KJ: This function re-orders Barentt's coefficient%
% to the order we see in equation (1.1).                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
J = readmatrix('coeff_forward_buy.txt');
J2 = J_to_KJ(J);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G(J)                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
% This turns Barnett's coefficients to constant functions
   in time.
psi_0 = repmat(J2,[1,t_size(2)]);

% u(x)  Please see the function called "u" (See Chapter 5
   Section 5)
ux=u(t,g,s,mo(a),mo(b));

% Deriving u'(x) is shown in Chapter 5 Section5.
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

% T_psi_0   Please see the function called "T_reg_data"
T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
```

```
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),x_12(t),x_13(t),s);

% u_psi_0   Please see the function called "u_psi"
u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

% u_psi_prime   Please see the function called "u_psi_p"
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
J_0= sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);

fprintf('Jessicas Gpsi: %d\n',J_0);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Getting  G(0)                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
% Buidling the zero function.
psi_0 = zeros(49,t_size(2));

% u'(x)
% Deriving u'(x) is shown in Chapter 5 Section 5.
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

% T_psi_0   Please see the function called "T_reg_data"
```

```
T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),x_12(t),x_13(t),s);

% u_psi_0   Please see the function called "u_psi"
u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

% u_psi_prime Please see the function called "u_psi_p"
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
fprintf('Zero Gpsi: %d\n',G_0);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Getting 10 Initial Psis                  %
% This portion is outlined in the algorithms section and%
% Chapter 5 Section 8.                                    %
% good_psi: This is where we store the psis we find.      %
% Instead of 'm' in Chapter 5 Section 8, we use 'M'.      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
good_psi = [];
count = 0;
while count<10
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_1(psi)=g_1        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
g = x_1(t)-x_1(1);

k = s*cumtrapz(x_1(t).*x_2(t),2);

common2 = s*cumtrapz(k,2);
M = (1/(mo(b)-mo(a)))*s*trapz(common2,2)-common2;

common = s*cumtrapz(g,2);         % Please see the function
    called "f_int"
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Calculate G_1(J)                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(1),[1,t_size(2)]);

T_psi_0=T_reg_x1(psi_0,x_1(t),x_2(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g(1,:)).^2+(u_psi_prime-up).^2,2)
   *s);
% fprintf('G(J_x1): %d\n',G_J);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Calculate G_1(0)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(0,[1,t_size(2)]);

T_psi_0=T_reg_x1(psi_0,x_1(t),x_2(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g(1,:)).^2+(u_psi_prime-up).^2,2)
   *s);
% fprintf('G(0_x1): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Getting the psi                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if G_0<=G_J
    top_in = g.*k+up.*M;
    top = 2*s*trapz(top_in,2);
    bottom_in = k.*k+M.*M;
    bottom = s*trapz(bottom_in,2);
    final = top/bottom;

    if final>0
        c=1;
        G1=[];
        R1=[];
        while c<5
```

```
        r = -0+(final--0).*rand(5000,1);
        psi_0 = repmat(r(1),[1,t_size(2)]);
        R1 = [R1;r];

        T_psi_0=T_reg_x1(psi_0,x_1(t),x_2(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
        g_less = sum(trapz((T_psi_0-g(1,:)).^2+(
           u_psi_prime-up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = R1(index_1);
elseif final<0
    d=1;
    G1=[];
    R1=[];
    while d<5
        r = final+(0-final).*rand(5000,1);
        psi_0 = repmat(r(1),[1,t_size(2)]);
        R1 = [R1;r];

        T_psi_0=T_reg_x1(psi_0,x_1(t),x_2(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
            g_less = sum(trapz((T_psi_0-g(1,:)).^2+(
               u_psi_prime-up).^2,2)*s);
            G1=[G1;g_less];
            d = d+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = R1(index_1);
    end
elseif G_J<G_0
    B1 = s*trapz(k.^2+M.^2,2);
    B2 = s*trapz(-2*g.*k-2*up.*M,2);
    B3 = -G_J+s*trapz(g.^2+up.^2,2);

    if B2^2-4*B1*(B3)<0
        small_1 = G_J;
        best_P1 = J2(1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G1=[];
        P1=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
            P = r2(1);
            P1 = [P1,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x1(psi_0,x_1(t),x_2(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Getting the psi with lowest G(psi)                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
final_psi = repmat(best_P1,[1,t_size(2)]);

T_psi_0=T_reg_x1(final_psi,x_1(t),x_2(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
g_final = sum(trapz((T_psi_0-g(1,:)).^2+(u_psi_prime-up)
   .^2,2)*s);
% fprintf('G(final_x1): %d\n',g_final);

writematrix(final_psi,'final_psi_x1.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_2(psi)=g_2        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
g = x_2(t)-x_2(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Calculate G_2(J)                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(2:4),[1,t_size(2)]);

common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g(1,:)).^2+(u_psi_prime-up).^2,2)
   *s);
% fprintf('G(J_x2): %d\n',G_J);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Calculate G_2(0)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(0,[3,t_size(2)]);

T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x2): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Getting the psis                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                 Fixing c3 and c4 as random                          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    k1 = s*cumtrapz(x_1(t),2);
    k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
    k3 = s*cumtrapz(x_2(t),2);
```

```
M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        P1 = [P1,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Fixing c2 and c4 as random              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

k1 = s*cumtrapz(x_1(t),2);
k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
k3 = s*cumtrapz(x_2(t),2);

common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;



top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
```

```
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);


B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);
```

```
        T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end

    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Fixing c2 and c3 as random                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

k1 = s*cumtrapz(x_1(t),2);
k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
k3 = s*cumtrapz(x_2(t),2);
```

```
M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
        S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G3=[];
        P3=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);c3(1);r2(1)];
            P3 = [P3,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
               );

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];

            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
```

```
elseif

G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    JJ = J2(2:4);
    for h=1
    b3 = JJ(2);
    b4 = JJ(3);
    r1 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
    r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
    c3 = repmat(r1(1),[1,t_size(2)]);
    c4 = repmat(r2(2),[1,t_size(2)]);



    k1 = s*cumtrapz(x_1(t),2);
    k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
    k3 = s*cumtrapz(x_2(t),2);



    M1_in = s*cumtrapz(k1,2);
    M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

    M2_in = s*cumtrapz(k2,2);
    M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

    M3_in = s*cumtrapz(k3,2);
    M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;
```

```
top1_in = -2*c3.*(g.*k2+up.*M2)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(top)<0
    small_1 = G_J;
    best_P1 = repmat(J2(2:4),[1,t_size(2)]);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
            P = [r2(1);c3(1);c4(1)];
            P1 = [P1,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
               );

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Fixing c2 and c4 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    b2 = JJ(1);
    b4 = JJ(3);
    r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
    r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
    c2 = repmat(r1(1),[1,t_size(2)]);
```

```
c4 = repmat(r2(2),[1,t_size(2)]);

k1 = s*cumtrapz(x_1(t),2);
k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
k3 = s*cumtrapz(x_2(t),2);

common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
```

```
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(top)<0
    small_2 = G_J;
    best_P2 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Fixing c2 and c3 as random                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b3 = JJ(2);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

k1 = s*cumtrapz(x_1(t),2);
k2 = s*cumtrapz(x_4d(t).*x_2(t),2);
k3 = s*cumtrapz(x_2(t),2);
```

```
M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c3.*(g.*k2+up.*M2)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(top)<0
    small_3 = G_J;
    best_P3 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
```

```
        S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G3=[];
        P3=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);c3(1);r2(1)];
            P3 = [P3,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x2(psi_0,x_1(t),x_2(t),x_4d(t),s
               );

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Getting the psi with the smallest G(psi)                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
smalls = [small_1;small_2;small_3];
bests = [best_P1,best_P2,best_P3];

[small,index] = mink(smalls,1);
best_psi = repmat(bests(:,index),[1,t_size(2)]);

final_psi = best_psi;
common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x2(final_psi,x_1(t),x_2(t),x_4d(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf(['G(final_x2): %d\n'],G_0);

writematrix(final_psi,'final_psi_x2.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_3(psi)=g_3        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Getting G_3(0)                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_3(t)-x_3(1);

psi_0 = repmat(0,[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),x_13(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);

% fprintf('G(0_x3): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Getting G_3(J)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(5:8),[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),x_13(t),s);
```

```
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);

G_J(isnan(G_J))=1e30;
% fprintf('G(J_x3): %d\n',G_J);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries      %
% changes the G_3(psi) the least.                       %
% Difference: has the absolute difference between G_0   %
% and perturbed psi                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x3(psi_1,x_2(t),x_6(t),x_8(t),x_13(t),s)
       ;

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g(1,:)).^2+(u_psi_prime-
       up).^2,2)*s);
```

```
    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         Get the smallest three terms                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Set K                                                 %
% K: has k_1 through k_4                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(x_2(t),2);
kay2 = s*cumtrapz(x_6(t),2);
kay3 = s*cumtrapz(x_8(t),2);
kay4 = s*cumtrapz(x_13(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we remove the rows with big_index
```

```
% This should leave us with three ks (terms which are
   affected the least)
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set M                                      %
% M: this has m_1, m_2, m_3, and m_4                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m([big_index],:) = [];
M1 = m(1,:);
```

```
M2 = m(2,:);
M3 = m(3,:);

if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %           Fixing c3 and c4 as random                          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
    top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
       c3.*c4.*(k2.*k3+M2.*M3);
    top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

    B1 = s*trapz(k1.^2+M1.^2,2);
    B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
       -2*(g.*k1+up.*M1);
    B2 = s*trapz(B2_in,2);


    if B2^2-4*B1*(-top)<0
        small_1 = G_0;
        best_P1 = zeros(4,1);
    else
```

```
S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [r2(1);c3(1);c4(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,p_in];

    T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
       x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
```

```
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
        p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
```

```
        P2 = [P2,p_in];

        T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
           x_13(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end

    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c2 and c3 as random                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
```

```
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
```

```
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P3 = [P3,p_in];

            T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
               x_13(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_3(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
```

```
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);


elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                 Set K                                               %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(x_2(t),2);
    kay2 = s*cumtrapz(x_6(t),2);
    kay3 = s*cumtrapz(x_8(t),2);
    kay4 = s*cumtrapz(x_13(t),2);
    K = [kay1;kay2;kay3;kay4];

    % Here, we put the three matters the most up front
    k1 = K(small_index(1),:);
    k2 = K(small_index(2),:);
    k3 = K(small_index(3),:);
    k4 = K(big_index(1),:);
    K_new = [k1;k2;k3;k4];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                   Set M                                             %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    m1_in = s*cumtrapz(kay1,2);
```

```
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);



% Set the constants which doesn't change here
JJ = J2(5:8);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c2 and c3 as random                  %
%                Keeping c_nc1 steady                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);


B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(5:8);
```

```
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(r2(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(c3(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P1 = [P1,P];

        T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
           x_13(t),s);
```

```
        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Fixing c1 and c3 as random                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);
```

```
B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;

if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(5:8);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
```

```
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(c1(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(r2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P2 = [P2,P];

    T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
       x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G2 = [G2;g_less];
    c = c+1;
end
[small_2, index_2] = mink(G2,1);
best_P2 = P2(:,index_2);
```

```
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c1 and c2 as random            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
```

```
B3 = s*trapz(B3_in,2)-G_J;



if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(5:8);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(r2(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
```

```
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P3 = [P3,P];

            T_psi_0=T_reg_x3(psi_0,x_2(t),x_6(t),x_8(t),
               x_13(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Getting the psi with the smallest G_3(psi)       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Calculating G_3(best_psi) and saving it                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x3(final_psi,x_2(t),x_6(t),x_8(t),x_13(t),s)
   ;

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x3): %d\n',G_best);

writematrix(final_psi,'final_psi_x3.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                for T_4(psi)=g_4                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Getting G_4(0)                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_4(t)-x_4(1);
```

```
psi_0 = repmat(0,[4,t_size(2)]);
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),x_4d(t),x_8(t)
   ,s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x4): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Getting G_4(J)                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(9:12),[4,t_size(2)]);

T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),x_4d(t),x_8(t)
   ,s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x4): %d\n',G_J);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This is where we are trying to see which entries     %
% changes the G_4(psi) the least.                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
```

```
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x4(psi_1,x_1(t),x_2(t),x_4(t),x_4d(t),
       x_8(t),s);

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);

    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Getting 3 terms which doesn't effect G_4 as much   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
```

```
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Set K                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(gradient(x_1(t)),2);
kay2 = s*cumtrapz(x_4(t),2);
kay3 = s*cumtrapz(x_2(t).*x_4d(t),2);
kay4 = s*cumtrapz(x_8(t).*x_4(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we will remove the rows with big_index
% This should leave us with three ks
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Set M                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;
```

```
m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m(big_index,:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);

if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %             Fixing c3 and c4 as random            %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
    top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
       c3.*c4.*(k2.*k3+M2.*M3);
    top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);
```

```
B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P1 = [P1,p_in];

            T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
               x_4d(t),x_8(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];

            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c4 as random                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for h=1
```

```
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
```

```
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);r2(1);c4(1)];
            psi_0 = zeros(4,t_size(2));
            psi_0(small_index(1),:) = repmat(P(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P2 = [P2,p_in];

            T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
               x_4d(t),x_8(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end
        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Fixing c2 and c3 as random          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S3 = [S1,S2];
S = sort(S3);

c=1;
G3=[];
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);c3(1);r2(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P3 = [P3,p_in];

    T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
       x_4d(t),x_8(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
```

```
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_4(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);

elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                       Set K                                   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(gradient(x_1(t)),2);
    kay2 = s*cumtrapz(x_4(t),2);
    kay3 = s*cumtrapz(x_2(t).*x_4d(t),2);
    kay4 = s*cumtrapz(x_8(t).*x_4(t),2);
    K = [kay1;kay2;kay3;kay4];
```

```
% Here, we put the three matters the most up front
k1 = K(small_index(1),:);
k2 = K(small_index(2),:);
k3 = K(small_index(3),:);
k4 = K(big_index(1),:);
K_new = [k1;k2;k3;k4];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Set M                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);
```

```
% Set the constants which doesn't change here
JJ = J2(9:12);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c3 as random                     %
%                 Keeping c_nc1 steady                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
```

```
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;



if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(9:12);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(r2(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(c3(3),[1,
               t_size(2)]);
            psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P1 = [P1,P];

            T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
               x_4d(t),x_8(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);

            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c1 and c3 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;

if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(9:12);
```

```
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(r2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(c3(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P2 = [P2,P];

        T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
           x_4d(t),x_8(t),s);
```

```
        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c1 and c2 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);
```

```
B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(9:12);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
```

```
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(r2(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P3 = [P3,P];

        T_psi_0=T_reg_x4(psi_0,x_1(t),x_2(t),x_4(t),
           x_4d(t),x_8(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G3 = [G3;g_less];
        c = c+1;
    end
    [small_3, index_3] = mink(G3,1);
    best_P3 = P3(:,index_3);
end
```

```
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_4(psi)                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Calculating G(best_psi) and saving it                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x4(final_psi,x_1(t),x_2(t),x_4(t),x_4d(t),
   x_8(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x4): %d\n',G_best);
writematrix(final_psi,'final_psi_x4.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_5(psi)=g_5                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Getting G_5(0)                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_5(t)-x_5(1);

psi_0 = repmat(0,[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x5): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Getting G_5(J)                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(13:16),[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x5): %d\n',G_J);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This is where we are trying to see which entries     %
%  changes the G_5(psi) the least.                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x5(psi_1,x_4(t),x_8(t),x_10(t),x_12(t),s
       );

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);

    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
```

```
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Getting the three terms                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Set K                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(x_8(t),2);
kay2 = s*cumtrapz(x_10(t),2);
kay3 = s*cumtrapz(x_4(t),2);
kay4 = s*cumtrapz(x_12(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we will remove the rows with big_index
% This should leave us with three ks
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Set M                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m([big_index],:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);

if G_0<=G_J

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %           Fixing c3 and c4 as random                                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c3 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
```

```
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [r2(1);c3(1);c4(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,p_in];

    T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G1 = [G1;g_less];
    c = c+1;
end
[small_1, index_1] = mink(G1,1);
best_P1 = P1(:,index_1);
```

```
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c4 as random       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S3 = [S1,S2];
S = sort(S3);

c=1;
G2=[];
P2=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);r2(1);c4(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P2 = [P2,p_in];

    T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
```

```
            G2 = [G2;g_less];
            c = c+1;
        end
        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c2 and c3 as random                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
```

```
        small_3 = G_0;
        best_P3 = zeros(3,1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G3=[];
        P3=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);c3(1);r2(1)];
            psi_0 = zeros(4,t_size(2));
            psi_0(small_index(1),:) = repmat(P(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P3 = [P3,p_in];

            T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
               x_12(t),s);
```

```
            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_5(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);


elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                   Set K                                   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
kay1 = s*cumtrapz(x_8(t),2);
kay2 = s*cumtrapz(x_10(t),2);
kay3 = s*cumtrapz(x_4(t),2);
kay4 = s*cumtrapz(x_12(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we put the three matters the most up front
k1 = K(small_index(1),:);
k2 = K(small_index(2),:);
k3 = K(small_index(3),:);
k4 = K(big_index(1),:);
K_new = [k1;k2;k3;k4];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Set M                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;
```

```
m = [m1;m2;m3;m4];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);


% Set the constants which doesn't change here
JJ = J2(13:16);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c3 as random                   %
%                 Keeping c_nc1 steady                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
```

```
B2 = s*trapz(B2_in,2);


B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;



if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(13:16);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
            psi_0 = zeros(4,t_size(2));
            psi_0(small_index(1),:) = repmat(r2(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(c2(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(c3(3),[1,
               t_size(2)]);
            psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P1 = [P1,P];

            T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);

            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c1 and c3 as random             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
```

```
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(13:16);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);


        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(r2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(c3(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
```

```
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
            (4,1)];
        P2 = [P2,P];

        T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
            x_12(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
            -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c1 and c2 as random          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
```

```
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(13:16);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);
```

```
c=1;
G3=[];
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(c1(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(r2(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P3 = [P3,P];

    T_psi_0=T_reg_x5(psi_0,x_4(t),x_8(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G3 = [G3;g_less];
```

```
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_5(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G(best_psi) and saving it           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x5(final_psi,x_4(t),x_8(t),x_10(t),x_12(t),s
   );

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x5): %d\n',G_best);
```

```
writematrix(final_psi,'final_psi_x5.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_6(psi)=g_6         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Getting G_6(0)                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_6(t)-x_6(1);

psi_0 = repmat(0,[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x6): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Getting G_6(J)                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(17:20),[1,t_size(2)]);
```

```
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
G_J(isnan(G_J))=1e30;
% fprintf('G(J_x6): %d\n',G_J);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries      %
% changes the G_6(psi) the least.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x6(psi_1,x_6(t),x_7d(t),x_11(t),x_12(t),
       s);
```

```
    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);

    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Pick three terms which effects G_6 the least       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Set K                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(x_6(t),2);
kay2 = s*cumtrapz(x_7d(t),2);
kay3 = s*cumtrapz(x_11(t),2);
kay4 = s*cumtrapz(x_12(t),2);
K = [kay1;kay2;kay3;kay4];
```

```
% Here, we will remove the rows with big_index
% This should leave us with three ks
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set M                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m(big_index,:) = [];
M1 = m(1,:);
```

```
M2 = m(2,:);
M3 = m(3,:);

if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                  Fixing c3 and c4 as random                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
    top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
       c3.*c4.*(k2.*k3+M2.*M3);
    top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

    B1 = s*trapz(k1.^2+M1.^2,2);
    B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
       -2*(g.*k1+up.*M1);
    B2 = s*trapz(B2_in,2);

    if B2^2-4*B1*(-top)<0
        small_1 = G_0;
        best_P1 = zeros(4,1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [r2(1);c3(1);c4(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,p_in];

    T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Fixing c2 and c4 as random                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);
```

```
if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
        p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P2 = [P2,p_in];
```

```
        T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
           x_12(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c3 as random                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);
```

```
B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P3 = [P3,p_in];

            T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    Getting the psi with the smallest G_6(psi)          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
```

```
    final_psi = repmat(bests(:,index),[1,t_size(2)]);


elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                       Set K                                       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(x_6(t),2);
    kay2 = s*cumtrapz(x_7d(t),2);
    kay3 = s*cumtrapz(x_11(t),2);
    kay4 = s*cumtrapz(x_12(t),2);
    K = [kay1;kay2;kay3;kay4];

    % Here, we put the three matters the most up front
    k1 = K(small_index(1),:);
    k2 = K(small_index(2),:);
    k3 = K(small_index(3),:);
    k4 = K(big_index(1),:);
    K_new = [k1;k2;k3;k4];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                       Set M                                       %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    m1_in = s*cumtrapz(kay1,2);
    m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;


    m2_in = s*cumtrapz(kay2,2);
```

```
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);


% Set the constants which doesn't change here
JJ = J2(17:20);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c3 as random                       %
%                Keeping c_nc1 steady                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
```

```
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(17:20);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
```

```
S = sort(S3);


c=1;
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);


    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(r2(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,P];


    T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
       x_12(t),s);


    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c1 and c3 as random        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
```

```
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;

if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(17:20);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
```

```
            psi_0(small_index(1),:) = repmat(c1(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(r2(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(c3(3),[1,
               t_size(2)]);
            psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P2 = [P2,P];

            T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end
        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Fixing c1 and c2 as random           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;
```

```
if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(17:20);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(r2(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P3 = [P3,P];
```

```
            T_psi_0=T_reg_x6(psi_0,x_6(t),x_7d(t),x_11(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_6(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G_6(best_psi) and saving it         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
T_psi_0=T_reg_x6(final_psi,x_6(t),x_7d(t),x_11(t),x_12(t),
   s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );

% fprintf('G(final_x6): %d\n',G_best);
writematrix(final_psi,'final_psi_x6.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_7(psi)=g_7         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G_7(0)                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_7(t)-x_7(1);

psi_0 = repmat(0,[3,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s);
```

```
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x7): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Getting G_7(J)                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(21:23),[3,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x7): %d\n',G_J);


if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %              Fixing c3 and c4 as random                                 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);
```

```
k1 = s*cumtrapz(x_7(t),2);
k2 = s*cumtrapz(x_6(t),2);
k3 = s*cumtrapz(x_11(t),2);


M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
```

```
        small_1 = G_0;
        best_P1 = zeros(3,1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G1=[];
        P1=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [r2(1);c3(1);c4(1)];
            P1 = [P1,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
               );

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];

            c = c+1;
```

```
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c4 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
g = x_2(t)-x_2(1);

r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);
```

```
if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
```

```
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c3 as random            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_3 = G_0;
```

```
    best_P3 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        P3 = [P3,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G3 = [G3;g_less];
        c = c+1;
    end
    [small_3, index_3] = mink(G3,1);
```

```
        best_P3 = P3(:,index_3);
    end
    end

elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    JJ = J2(21:23);
    for h=1
    b3 = JJ(2);
    b4 = JJ(3);
    r1 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
    r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
    c3 = repmat(r1(1),[1,t_size(2)]);
    c4 = repmat(r2(2),[1,t_size(2)]);

    k1 = s*cumtrapz(x_7(t),2);
    k2 = s*cumtrapz(x_6(t),2);
    k3 = s*cumtrapz(x_11(t),2);



    M1_in = s*cumtrapz(k1,2);
    M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

    M2_in = s*cumtrapz(k2,2);
    M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

    M3_in = s*cumtrapz(k3,2);
```

```
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = -2*c3.*(g.*k2+up.*M2)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(top)<0
    small_1 = G_J;
    best_P1 = repmat(J2(21:23),[1,t_size(2)]);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
```

```
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        P1 = [P1,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Fixing c2 and c4 as random        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b4 = JJ(3);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
```

```
r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c4 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(top)<0
    small_2 = G_J;
    best_P2 = J2(21:23);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
```

```
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
           );

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c2 and c3 as random                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
b2 = JJ(1);
b3 = JJ(2);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c3.*(g.*k2+up.*M2)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(top)<0
    small_3 = G_J;
    best_P3 = J2(21:23);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);
```

```
        c=1;
        G3=[];
        P3=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);c3(1);r2(1)];
            P3 = [P3,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x7(psi_0,x_6(t),x_7(t),x_11(t),s
               );

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Getting the psi with the smallest G_7(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
smalls = [small_1;small_2;small_3];
bests = [best_P1,best_P2,best_P3];

[small,index] = mink(smalls,1);
best_psi = repmat(bests(:,index),[1,t_size(2)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Calculating G(best_psi) and saving it           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
final_psi = best_psi;
common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);
T_psi_0=T_reg_x7(final_psi,x_6(t),x_7(t),x_11(t),s);
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(best_x7): %d\n',G_0);
writematrix(final_psi,'final_psi_x7.txt');
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_8(psi)=g_8        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Getting G_8(0)                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_8(t)-x_8(1);

psi_0 = repmat(0,[7,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),x_6(t),x_8(t),
   x_10(t),x_13(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x8): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Getting G_8(0)                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(24:30),[7,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);
```

```
T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),x_6(t),x_8(t),
   x_10(t),x_13(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);

% fprintf('G(J_x8): %d\n',G_J)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries      %
% changes the G_8(psi) the least.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(7,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x8(psi_1,x_3(t),x_4(t),x_5(t),x_6(t),x_8
       (t),x_10(t),x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);
```

```
    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(7,1);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Getting the index for the three terms that affects    %
% G_8 the least                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

BI = ones(7,1);
BI(small_index(1))=0;
BI(small_index(2))=0;
BI(small_index(3))=0;
[get_big, big_index] = maxk(BI,4);
big_index = sort(big_index);


if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                    Set K                                %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
kay1 = s*cumtrapz(x_6(t),2);
gx_5 = gradient(x_5(t));
kay2 = s*cumtrapz(gx_5,2);
kay3 = s*cumtrapz(x_8(t),2);
kay4 = s*cumtrapz(x_8(t).*x_4(t),2);
kay5 = s*cumtrapz(x_3(t),2);
gx_10 = gradient(x_10(t));
kay6 = s*cumtrapz(gx_10,2);
gx_13 = gradient(x_13(t));
kay7 = s*cumtrapz(gx_13,2);
K = [kay1;kay2;kay3;kay4;kay5;kay6;kay7];

% Here, we will remove the rows with big_index
% This should leave us with three ks
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set M                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;
```

```
m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m5_in = s*cumtrapz(kay5,2);
m5 = (1/(mo(b)-mo(a)))*s*trapz(m5_in,2)-m5_in;

m6_in = s*cumtrapz(kay6,2);
m6 = (1/(mo(b)-mo(a)))*s*trapz(m6_in,2)-m6_in;

m7_in = s*cumtrapz(kay7,2);
m7 = (1/(mo(b)-mo(a)))*s*trapz(m7_in,2)-m7_in;

m = [m1;m2;m3;m4;m5;m6;m7];

m([big_index],:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c3 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
r = -dd+(dd--dd).*rand(5000,1);
c3 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
```

```
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
        p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
        P1 = [P1,p_in];

        T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
           x_6(t),x_8(t),x_10(t),x_13(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S3 = [S1,S2];
S = sort(S3);

c=1;
G2=[];
P2=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);r2(1);c4(1)];
    psi_0 = zeros(7,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
    P2 = [P2,p_in];

    T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
       x_6(t),x_8(t),x_10(t),x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
```

```
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Fixing c2 and c3 as random              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);
```

```
if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
        p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
        P3 = [P3,p_in];
```

```
            T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
               x_6(t),x_8(t),x_10(t),x_13(t),s);
            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_8(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);

elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % This is where we are trying to see which entries  %
    % changes the G_8(psi) the least.                   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
M = J2(24:30);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = M(i)+0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x8(psi_1,x_3(t),x_4(t),x_5(t),x_6(t)
       ,x_8(t),x_10(t),x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up
       ).^2,2)*s);

    % Appending the difference
    difference = abs(G_J - g_psi_1);
    Difference = [Difference;difference];

    M=J2(24:30);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Gettig the three index                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);
```

```
BI = ones(7,1);
BI(small_index(1))=0;
BI(small_index(2))=0;
BI(small_index(3))=0;
[get_big, big_index] = maxk(BI,4);
big_index = sort(big_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Set K                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(x_6(t),2);
gx_5 = gradient(x_5(t));
kay2 = s*cumtrapz(gx_5,2);
kay3 = s*cumtrapz(x_8(t),2);
kay4 = s*cumtrapz(x_8(t).*x_4(t),2);
kay5 = s*cumtrapz(x_3(t),2);
gx_10 = gradient(x_10(t));
kay6 = s*cumtrapz(gx_10,2);
gx_13 = gradient(x_13(t));
kay7 = s*cumtrapz(gx_13,2);
K = [kay1;kay2;kay3;kay4;kay5;kay6;kay7];

% Here, we put the three matters the least up front
k1 = K(small_index(1),:);
k2 = K(small_index(2),:);
k3 = K(small_index(3),:);
```

```
k4 = K(big_index(1),:);
k5 = K(big_index(2),:);
k6 = K(big_index(3),:);
k7 = K(big_index(4),:);
K_new = [k1;k2;k3;k4;k5;k6;k7];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Set M                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m5_in = s*cumtrapz(kay5,2);
m5 = (1/(mo(b)-mo(a)))*s*trapz(m5_in,2)-m5_in;

m6_in = s*cumtrapz(kay6,2);
m6 = (1/(mo(b)-mo(a)))*s*trapz(m6_in,2)-m6_in;

m7_in = s*cumtrapz(kay7,2);
```

```
m7 = (1/(mo(b)-mo(a)))*s*trapz(m7_in,2)-m7_in;

m = [m1;m2;m3;m4;m5;m6;m7];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);
M5 = m(big_index(2),:);
M6 = m(big_index(3),:);
M7 = m(big_index(4),:);

% Set the constants which doesn't change here
JJ = J2(24:30);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
c5 = repmat(JJ(big_index(2)),[1,t_size(2)]);
c6 = repmat(JJ(big_index(3)),[1,t_size(2)]);
c7 = repmat(JJ(big_index(4)),[1,t_size(2)]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c2 and c3 as random            %
%              Keeping c_nc1 steady                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
```

```
st2 = b2-dd;
en2 = b2+dd;
st3 = b3-dd;
en3 = b3+dd;
r1 = st2+(en2-st2).*rand(5000,1);
r2 = st3+(en3-st3).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)...
    +2*c5.*(k1.*k5+M1.*M5)+2*c6.*(k1.*k6+M1.*M6)+2*c7
       .*(k1.*k7+M1.*M7)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
       .^2.*(k7.^2+M7.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)...
    +2*c2.*c5.*(k2.*k5+M2.*M5)+2*c2.*c6.*(k2.*k6+M2.*
       M6)...
    +2*c2.*c7.*(k2.*k7+M2.*M7)...
```

```
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)...
    +2*c3.*c6.*(k3.*k6+M3.*M6)+2*c3.*c7.*(k3.*k7+M3.*
       M7)...
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
```

```
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

    psi_0 = zeros(7,t_size(2));
    psi_0(small_index(1),:) = repmat(r2(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
       (2)]);
    psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
       (2)]);
    psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
    P1 = [P1,P];

    T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
       x_6(t),x_8(t),x_10(t),x_13(t),s);
```

```
        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Fixing c1 and c3 as random                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
st1 = b1-dd;
en1 = b1+dd;
st3 = b3-dd;
en3 = b3+dd;
r1 = st1+(en1-st1).*rand(5000,1);
r2 = st3+(en3-st3).*rand(5000,1);

c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);
```

```
B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)...
    +2*c5.*(k2.*k5+M2.*M5)+2*c6.*(k2.*k6+M2.*M6)+2*c7
       .*(k2.*k7+M2.*M7)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
       .^2.*(k7.^2+M7.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)...
    +2*c1.*c5.*(k1.*k5+M1.*M5)+2*c1.*c6.*(k1.*k6+M1.*
       M6)...
    +2*c1.*c7.*(k1.*k7+M1.*M7)...
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)...
    +2*c3.*c6.*(k3.*k6+M3.*M6)+2*c3.*c7.*(k3.*k7+M3.*
       M7)...
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
```

```
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
```

```
        psi_0(small_index(2),:) = repmat(r2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(c3(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
           (2)]);
        psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
           (2)]);
        psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
        P2 = [P2,P];

        T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
           x_6(t),x_8(t),x_10(t),x_13(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G2 = [G2;g_less];
        c = c+1;
    end
```

```
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c1 and c2 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
st1 = b1-dd;
en1 = b1+dd;
st2 = b2-dd;
en2 = b2+dd;
r1 = st1+(en1-st1).*rand(5000,1);
r2 = st2+(en2-st2).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)...
    +2*c5.*(k3.*k5+M3.*M5)+2*c6.*(k3.*k6+M3.*M6)+2*c7
       .*(k3.*k7+M3.*M7)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);
```

```
B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
       .^2.*(k7.^2+M7.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)...
    +2*c1.*c5.*(k1.*k5+M1.*M5)+2*c1.*c6.*(k1.*k6+M1.*
       M6)...
    +2*c1.*c7.*(k1.*k7+M1.*M7)...
    +2*c2.*c4.*(k2.*k4+M2.*M4)+2*c2.*c5.*(k2.*k5+M2.*
       M5)...
    +2*c2.*c6.*(k2.*k6+M2.*M6)+2*c2.*c7.*(k2.*k7+M2.*
       M7)...
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;
```

```
if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(r2(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
           (2)]);
```

```
            psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
               (2)]);
            psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
            P3 = [P3,P];

            T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
               x_6(t),x_8(t),x_10(t),x_13(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);

            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Fixing c2 and c3 as random                          %
%                 Keeping c_nc1 steady                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
b2=0;
b3=0;
st2 = b2-dd;
en2 = b2+dd;
st3 = b3-dd;
en3 = b3+dd;
r1 = st2+(en2-st2).*rand(5000,1);
r2 = st3+(en3-st3).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)...
     +2*c5.*(k1.*k5+M1.*M5)+2*c6.*(k1.*k6+M1.*M6)+2*c7
        .*(k1.*k7+M1.*M7)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
     +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
        .^2.*(k7.^2+M7.^2)...
     +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
        M4)...
```

```
    +2*c2.*c5.*(k2.*k5+M2.*M5)+2*c2.*c6.*(k2.*k6+M2.*
       M6)...
    +2*c2.*c7.*(k2.*k7+M2.*M7)...
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)...
    +2*c3.*c6.*(k3.*k6+M3.*M6)+2*c3.*c7.*(k3.*k7+M3.*
       M7)...
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_4 = G_J;
    best_P4 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
```

```
S = sort(S3);


c=1;
G4=[];
P4=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);


    psi_0 = zeros(7,t_size(2));
    psi_0(small_index(1),:) = repmat(r2(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
       (2)]);
    psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
       (2)]);
    psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
    P4 = [P4,P];
```

```
        T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
           x_6(t),x_8(t),x_10(t),x_13(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G4 = [G4;g_less];
        c = c+1;
    end
    [small_4, index_4] = mink(G4,1);
    best_P4 = P4(:,index_4);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Fixing c1 and c3 as random                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=0;
b3=0;
st1 = b1-dd;
en1 = b1+dd;
st3 = b3-dd;
en3 = b3+dd;
r1 = st1+(en1-st1).*rand(5000,1);
```

```
r2 = st3+(en3-st3).*rand(5000,1);

c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)...
    +2*c5.*(k2.*k5+M2.*M5)+2*c6.*(k2.*k6+M2.*M6)+2*c7
       .*(k2.*k7+M2.*M7)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
       .^2.*(k7.^2+M7.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)...
    +2*c1.*c5.*(k1.*k5+M1.*M5)+2*c1.*c6.*(k1.*k6+M1.*
       M6)...
    +2*c1.*c7.*(k1.*k7+M1.*M7)...
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)...
    +2*c3.*c6.*(k3.*k6+M3.*M6)+2*c3.*c7.*(k3.*k7+M3.*
       M7)...
```

```
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_5 = G_J;
    best_P5 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G5=[];
    P5=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
psi_0 = zeros(7,t_size(2));
psi_0(small_index(1),:) = repmat(c1(1),[1,
   t_size(2)]);
psi_0(small_index(2),:) = repmat(r2(2),[1,
   t_size(2)]);
psi_0(small_index(3),:) = repmat(c3(3),[1,
   t_size(2)]);
psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
   (2)]);
psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
   (2)]);
psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
   (2)]);
psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
   (2)]);
P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
   (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
P5 = [P5,P];

T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
   x_6(t),x_8(t),x_10(t),x_13(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
   -up).^2,2)*s);
```

```
        G5 = [G5;g_less];
        c = c+1;
    end
    [small_5, index_5] = mink(G5,1);
    best_P5 = P5(:,index_5);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c1 and c2 as random               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=0;
b2=0;
st1 = b1-dd;
en1 = b1+dd;
st2 = b2-dd;
en2 = b2+dd;
r1 = st1+(en1-st1).*rand(5000,1);
r2 = st2+(en2-st2).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);
```

```
B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)...
    +2*c5.*(k3.*k5+M3.*M5)+2*c6.*(k3.*k6+M3.*M6)+2*c7
       .*(k3.*k7+M3.*M7)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +c5.^2.*(k5.^2+M5.^2)+c6.^2.*(k6.^2+M6.^2)+c7
       .^2.*(k7.^2+M7.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)...
    +2*c1.*c5.*(k1.*k5+M1.*M5)+2*c1.*c6.*(k1.*k6+M1.*
       M6)...
    +2*c1.*c7.*(k1.*k7+M1.*M7)...
    +2*c2.*c4.*(k2.*k4+M2.*M4)+2*c2.*c5.*(k2.*k5+M2.*
       M5)...
    +2*c2.*c6.*(k2.*k6+M2.*M6)+2*c2.*c7.*(k2.*k7+M2.*
       M7)...
    +2*c4.*c5.*(k4.*k5+M4.*M5)+2*c4.*c6.*(k4.*k6+M4.*
       M6)...
    +2*c4.*c7.*(k4.*k7+M4.*M7)...
    +2*c5.*c6.*(k5.*k6+M5.*M6)+2*c5.*c7.*(k5.*k7+M5.*
       M7)+2*c6.*c7.*(k6.*k7+M6.*M7)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
```

```
    -2*c5.*(g.*k5+up.*M5)-2*c6.*(g.*k6+up.*M6)-2*c7.*(
       g.*k7+up.*M7)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_6 = G_J;
    best_P6 = J2(24:30);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G6=[];
    P6=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
```

```
    psi_0(small_index(3),:) = repmat(r2(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
       (2)]);
    psi_0(big_index(3),:) = repmat(c6(1),[1,t_size
       (2)]);
    psi_0(big_index(4),:) = repmat(c7(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
    P6 = [P6,P];

    T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
       x_6(t),x_8(t),x_10(t),x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);

    G6 = [G6;g_less];
    c = c+1;
end
[small_6, index_6] = mink(G6,1);
best_P6 = P6(:,index_6);
```

```
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c3 and c4 as random             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c3 = repmat(0,[1,t_size(2)]);
c4 = repmat(0,[1,t_size(2)]);

top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_7 = G_0;
    best_P7 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G7=[];
P7=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [r2(1);c3(1);c4(1)];
    psi_0 = zeros(7,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
    P7 = [P7,p_in];

    T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
       x_6(t),x_8(t),x_10(t),x_13(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G7 = [G7;g_less];
        c = c+1;
    end
    [small_7, index_7] = mink(G7,1);
    best_P7 = P7(:,index_7);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c4 as random                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(0,[1,t_size(2)]);
c4 = repmat(0,[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
```

```
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_8 = G_0;
    best_P8 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G8=[];
    P8=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
        p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
        P8 = [P8,p_in];
```

```
            T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
               x_6(t),x_8(t),x_10(t),x_13(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G8 = [G8;g_less];
            c = c+1;
        end
        [small_8, index_8] = mink(G8,1);
        best_P8 = P8(:,index_8);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c3 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c2 = repmat(0,[1,t_size(2)]);
    c3 = repmat(0,[1,t_size(2)]);

    top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
    top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
       c2.*c3.*(k1.*k2+M1.*M2);
```

```
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_9 = G_0;
    best_P9 = zeros(7,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G9=[];
    P9=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        psi_0 = zeros(7,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1);psi_0(5,1);psi_0(6,1);psi_0(7,1)];
            P9 = [P9,p_in];

            T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),
               x_6(t),x_8(t),x_10(t),x_13(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G9 = [G9;g_less];
            c = c+1;
        end
        [small_9, index_9] = mink(G9,1);
        best_P9 = P9(:,index_9);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_8(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3;small_4;small_5;
       small_6;small_7;small_8;small_9];
```

```
    bests = [best_P1,best_P2,best_P3,best_P4,best_P5,
       best_P6,best_P7,best_P8,best_P9];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Calculating G_8(best_psi) and saving it          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x8(final_psi,x_3(t),x_4(t),x_5(t),x_6(t),x_8
   (t),x_10(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x8): %d\n',G_best);

writematrix(final_psi,'final_psi_x8.txt');

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            for T_9(psi)=g_9          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Getting G(0)                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_9(t)-x_9(1);

psi_0 = repmat(0,[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x9): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                Getting G(J)                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_9(t)-x_9(1);

psi_0 = repmat(J2(31:34),[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);
```

```
T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x9): %d\n',G_J);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries      %
% changes the G_9(psi) the least.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x9(psi_1,x_2(t),x_9(t),x_10(t),x_12(t),s
       );

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);
```

```
    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Getting the three terms which effects G_9 the least %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Set K                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
kay1 = s*cumtrapz(x_10(t),2);
kay2 = s*cumtrapz(x_12(t),2);
kay3 = s*cumtrapz(x_2(t),2);
kay4 = s*cumtrapz(x_9(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we will remove the rows with big_index
% This should leave us with three ks
```

```
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Set M                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m([big_index],:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);
```

```
if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %              Fixing c3 and c4 as random           %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
    top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
       c3.*c4.*(k2.*k3+M2.*M3);
    top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

    B1 = s*trapz(k1.^2+M1.^2,2);
    B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
       -2*(g.*k1+up.*M1);
    B2 = s*trapz(B2_in,2);

    if B2^2-4*B1*(-top)<0
        small_1 = G_0;
        best_P1 = zeros(4,1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);
```

```
c=1;
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [r2(1);c3(1);c4(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,p_in];

    T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G1 = [G1;g_less];
```

```matlab
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c4 as random            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
```

```
        small_2 = G_0;
        best_P2 = zeros(4,1);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G2=[];
        P2=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);r2(1);c4(1)];
            psi_0 = zeros(4,t_size(2));
            psi_0(small_index(1),:) = repmat(P(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P2 = [P2,p_in];

            T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
               x_12(t),s);
```

```
        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Fixing c2 and c3 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
```

```
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(P(3),[1,
           t_size(2)]);
```

```
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P3 = [P3,p_in];

            T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_9(psi)      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
```

```
elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                              Set K                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(x_10(t),2);
    kay2 = s*cumtrapz(x_12(t),2);
    kay3 = s*cumtrapz(x_2(t),2);
    kay4 = s*cumtrapz(x_9(t),2);
    K = [kay1;kay2;kay3;kay4];

    % Here, we put the three matters the most up front
    k1 = K(small_index(1),:);
    k2 = K(small_index(2),:);
    k3 = K(small_index(3),:);
    k4 = K(big_index(1),:);
    K_new = [k1;k2;k3;k4];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                              Set M                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    m1_in = s*cumtrapz(kay1,2);
    m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

    m2_in = s*cumtrapz(kay2,2);
    m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

    m3_in = s*cumtrapz(kay3,2);
```

```
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);


% Set the constants which doesn't change here
JJ = J2(31:34);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c2 and c3 as random                         %
%                Keeping c_nc1 steady                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);
```

```
B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(31:34);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
```

```
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(r2(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P1 = [P1,P];

    T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);

    G1 = [G1;g_less];
```

```
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c1 and c3 as random                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
```

```
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;

if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(31:34);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(r2(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(c3(3),[1,
               t_size(2)]);
            psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P2 = [P2,P];

            T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
               x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end
        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c1 and c2 as random                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    b1=JJ(small_index(1));
```

```
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(31:34);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
```

```
S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G3=[];
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(c1(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(r2(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P3 = [P3,P];

    T_psi_0=T_reg_x9(psi_0,x_2(t),x_9(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_9(psi       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G_9(best_psi) and saving it         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x9(final_psi,x_2(t),x_9(t),x_10(t),x_12(t),s
   );

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x9): %d\n',G_best);

writematrix(final_psi,'final_psi_x9.txt');
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          for T_10(psi)=g_10          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G(0)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_10(t)-x_10(1);

psi_0 = repmat(0,[5,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),x_10(t),x_12(
   t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
```

```
% fprintf('G(0_x10): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Getting G(J)                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(35:39),[5,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),x_10(t),x_12(
   t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x10): %d\n',G_J);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries                      %
% changes the G_10(psi) the least.                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(5,1);
Ms = size(M);
Difference = [];
for i=1:Ms(1)
    M(i) = 0.1;
```

```
    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x10(psi_1,x_1(t),x_4(t),x_5(t),x_10(t),
       x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);

    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(5,1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Getting the three terms                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);

BI = ones(5,1);
BI(small_index(1))=0;
BI(small_index(2))=0;
BI(small_index(3))=0;
[get_big, big_index] = maxk(BI,2);
big_index = sort(big_index);
```

```
if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                              Set K                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(x_10(t),2);
    kay2 = s*cumtrapz(x_4(t),2);
    kay3 = s*cumtrapz(x_1(t),2);
    kay4 = s*cumtrapz(x_5(t),2);
    kay5 = s*cumtrapz(x_12(t),2);
    K = [kay1;kay2;kay3;kay4;kay5];

    % Here, we will remove the rows with big_index
    % This should leave us with three ks
    K(big_index,:) = [];
    k1 = K(1,:);
    k2 = K(2,:);
    k3 = K(3,:);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                              Set M                                    %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    m1_in = s*cumtrapz(kay1,2);
    m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

    m2_in = s*cumtrapz(kay2,2);
    m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;
```

```
m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m5_in = s*cumtrapz(kay5,2);
m5 = (1/(mo(b)-mo(a)))*s*trapz(m5_in,2)-m5_in;

m = [m1;m2;m3;m4;m5];

m(big_index,:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c3 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c3 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
```

```
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(5,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        psi_0 = zeros(5,t_size(2));
```

```
            psi_0(small_index(1),:) = repmat(P(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1);psi_0(5,1)];
            P1 = [P1,p_in];

            T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
               x_10(t),x_12(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c4 as random              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(5,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
```

```
G2=[];
P2=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);r2(1);c4(1)];
    psi_0 = zeros(5,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1)];
    P2 = [P2,p_in];

    T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
       x_10(t),x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G2 = [G2;g_less];
    c = c+1;
end
[small_2, index_2] = mink(G2,1);
```

```
    best_P2 = P2(:,index_2);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c3 as random          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(5,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G3=[];
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);c3(1);r2(1)];
    psi_0 = zeros(5,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1)];
    P3 = [P3,p_in];

    T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
       x_10(t),x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_10(psi)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);



elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                   Set K                                   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    kay1 = s*cumtrapz(x_10(t),2);
    kay2 = s*cumtrapz(x_4(t),2);
```

```
kay3 = s*cumtrapz(x_1(t),2);
kay4 = s*cumtrapz(x_5(t),2);
kay5 = s*cumtrapz(x_12(t),2);
K = [kay1;kay2;kay3;kay4;kay5];


% Here, we put the three matters the most up front
k1 = K(small_index(1),:);
k2 = K(small_index(2),:);
k3 = K(small_index(3),:);
k4 = K(big_index(1),:);
k5 = K(big_index(2),:);
K_new = [k1;k2;k3;k4;k5];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Set M                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
```

```
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m5_in = s*cumtrapz(kay5,2);
m5 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-s*cumtrapz(
   m5_in,2);

m = [m1;m2;m3;m4;m5];

M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);
M5 = m(big_index(2),:);



% Set the constants which doesn't change here
JJ = J2(35:39);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
c5 = repmat(JJ(big_index(2)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c3 as random                        %
%                Keeping c_nc1 steady                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
```

```
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)+2*c5.*(k1.*k5+M1.*M5)-2*(g.*k1+
   up.*M1);
B2 = s*trapz(B2_in,2);

B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)+c5.^2.*(k5.^2+M5.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c2.*c5.*(k2.*k5+M2.*M5)...
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)+2*c4.*c5.*(k4.*k5+M4.*M5)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)-2*c5.*(g.*k5+up.*M5)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(35:39);
else
```

```
S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
S3 = [S1,S2];
S = sort(S3);

c=1;
G1=[];
P1=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

    psi_0 = zeros(5,t_size(2));
    psi_0(small_index(1),:) = repmat(r2(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1)];
    P1 = [P1,P];
```

```
        T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
           x_10(t),x_12(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c1 and c3 as random          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);
```

```
B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)+2*c5.*(k2.*k5+M2.*M5)-2*(g.*k2+
   up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)+c5.^2.*(k5.^2+M5.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c1.*c5.*(k1.*k5+M1.*M5)...
    +2*c3.*c4.*(k3.*k4+M3.*M4)+2*c3.*c5.*(k3.*k5+M3.*
       M5)+2*c4.*c5.*(k4.*k5+M4.*M5)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)-2*c5.*(g.*k5+up.*M5)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;



if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(35:39);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);
```

```
c=1;
G2=[];
P2=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

    psi_0 = zeros(5,t_size(2));
    psi_0(small_index(1),:) = repmat(c1(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(r2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(c3(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1);psi_0(5,1)];
    P2 = [P2,P];

    T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
       x_10(t),x_12(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c1 and c2 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b1=JJ(small_index(1));
b2=JJ(small_index(2));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c2 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)+2*c5.*(k3.*k5+M3.*M5)-2*(g.*k3+
   up.*M3);
B2 = s*trapz(B2_in,2);
```

```
B3_in = c2.^2.*(k2.^2+M2.^2)+c1.^2.*(k1.^2+M1.^2)+c4
   .^2.*(k4.^2+M4.^2)+c5.^2.*(k5.^2+M5.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c1.*c5.*(k1.*k5+M1.*M5)...
    +2*c2.*c4.*(k2.*k4+M2.*M4)+2*c2.*c5.*(k2.*k5+M2.*
       M5)+2*c4.*c5.*(k4.*k5+M4.*M5)...
    -2*c2.*(g.*k2+up.*M2)-2*c1.*(g.*k1+up.*M1)-2*c4.*(
       g.*k4+up.*M4)-2*c5.*(g.*k5+up.*M5)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(35:39);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
        psi_0 = zeros(5,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(c2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(r2(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        psi_0(big_index(2),:) = repmat(c5(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1);psi_0(5,1)];
        P3 = [P3,P];

        T_psi_0=T_reg_x10(psi_0,x_1(t),x_4(t),x_5(t),
           x_10(t),x_12(t),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);

        G3 = [G3;g_less];
        c = c+1;
    end
```

```
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_10(psi      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G_10(best_psi) and saving it        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x10(final_psi,x_1(t),x_4(t),x_5(t),x_10(t),
   x_12(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x10): %d\n',G_best);

writematrix(final_psi,'final_psi_x10.txt');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          for T_11(psi)=g_11         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G_11(0)                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_11(t)-x_11(1);

psi_0 = repmat(0,[3,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x11): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G_11                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(40:42),[3,t_size(2)]);

common = s*cumtrapz(g,2);
```

```
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
G_J(isnan(G_J))=1e30;
% fprintf('G(J_x11): %d\n',G_J);


if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %           Fixing c3 and c4 as random                          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    k1 = s*cumtrapz(x_7d(t),2);
    k2 = s*cumtrapz(x_11(t),2);
    k3 = s*cumtrapz(x_4d(t).*x_11(t),2);


    M1_in = s*cumtrapz(k1,2);
    M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;
```

```
M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);
```

```
    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        P1 = [P1,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
           ),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G1 = [G1;g_less];
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Fixing c2 and c4 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
```

```
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
           ),s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c3 as random                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
```

```
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
            P = [c2(1);c3(1);r2(1)];
            P3 = [P3,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
               ),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end

elseif G_J<G_0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    JJ = J2(40:42);
    for h=1
    b3 = JJ(2);
    b4 = JJ(3);
    r1 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
    r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
```

```
c3 = repmat(r1(1),[1,t_size(2)]);
c4 = repmat(r2(2),[1,t_size(2)]);

k1 = s*cumtrapz(x_7d(t),2);
k2 = s*cumtrapz(x_11(t),2);
k3 = s*cumtrapz(x_4d(t).*x_11(t),2);


M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = -2*c3.*(g.*k2+up.*M2)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
```

```
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(top)<0
    small_1 = G_J;
    best_P1 = repmat(J2(40:42),[1,t_size(2)]);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        P1 = [P1,P];
        psi_0 = repmat(P,[1,t_size(2)]);


        T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
           ),s);


        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        c = c+1;
    end
    [small_1, index_1] = mink(G1,1);
    best_P1 = P1(:,index_1);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b4 = JJ(3);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c4 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);
```

```
B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(top)<0
    small_2 = G_J;
    best_P2 = J2(40:42);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
           ),s);
```

```
        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Fixing c2 and c3 as random          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b3 = JJ(2);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c3.*(g.*k2+up.*M2)+g
   .^2+up.^2;
```

```
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(top)<0
    small_3 = G_J;
    best_P3 = J2(40:42);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        P3 = [P3,P];
        psi_0 = repmat(P,[1,t_size(2)]);
```

```
            T_psi_0=T_reg_x11(psi_0,x_4d(t),x_7d(t),x_11(t
               ),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_11(psi)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
smalls = [small_1;small_2;small_3];
bests = [best_P1,best_P2,best_P3];

[small,index] = mink(smalls,1);
best_psi = repmat(bests(:,index),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G_11(best_psi) and saving it        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
final_psi = best_psi;
common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x11(final_psi,x_4d(t),x_7d(t),x_11(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(best_x11): %d\n',G_0);

writematrix(final_psi,'final_psi_x11.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          for T_12(psi)=g_12        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Getting G_12(0)                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_12(t)-x_12(1);

psi_0 = repmat(0,[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);
```

```
T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),x_10(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x12): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Getting G_12(J)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(43:46),[4,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),x_10(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(J_x12): %d\n',G_J);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we are trying to see which entries                      %
% changes the G_12(psi) the least.                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = zeros(4,1);
Ms = size(M);
Difference = [];
```

```
for i=1:Ms(1)
    M(i) = 0.1;

    psi_1 = repmat(M,[1,t_size(2)]);

    T_psi_1=T_reg_x12(psi_1,x_1(t),x_4(t),x_9(t),x_10(t),s
       );

    u_psi_prime = u_psi_p(T_psi_1,mo(a),mo(b),s);
    g_psi_1 = sum(trapz((T_psi_1-g).^2+(u_psi_prime-up)
       .^2,2)*s);

    % Appending the difference
    difference = abs(G_0 - g_psi_1);
    Difference = [Difference;difference];

    M=zeros(4,1);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the three terms.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[get_big, big_index] = maxk(Difference,1);
big_index = sort(big_index);
[get_small, small_index] = mink(Difference,3);
small_index = sort(small_index);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Set K                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gx10 = gradient(x_10(t));
kay1 = s*cumtrapz(gx10,2);
kay2 = s*cumtrapz(x_9(t),2);
kay3 = s*cumtrapz(x_4(t),2);
kay4 = s*cumtrapz(x_1(t),2);
K = [kay1;kay2;kay3;kay4];

% Here, we will remove the rows with big_index
% This should leave us with three ks
K(big_index,:) = [];
k1 = K(1,:);
k2 = K(2,:);
k3 = K(3,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Set M                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
```

```
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];

m([big_index],:) = [];
M1 = m(1,:);
M2 = m(2,:);
M3 = m(3,:);



if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                 Fixing c3 and c4 as random                     %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
    top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
       c3.*c4.*(k2.*k3+M2.*M3);
    top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);
```

```
B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [r2(1);c3(1);c4(1)];
        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(P(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(P(2),[1,
           t_size(2)]);
```

```
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P1 = [P1,p_in];

            T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
               x_10(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c4 as random               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
```

```
c2 = repmat(r(1),[1,t_size(2)]);
c4 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
```

```
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [c2(1);r2(1);c4(1)];
            psi_0 = zeros(4,t_size(2));
            psi_0(small_index(1),:) = repmat(P(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(P(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(P(3),[1,
               t_size(2)]);
            p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P2 = [P2,p_in];

            T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
               x_10(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end

        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Fixing c2 and c3 as random                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
r = -dd+(dd--dd).*rand(5000,1);
c2 = repmat(r(1),[1,t_size(2)]);
c3 = repmat(r(2),[1,t_size(2)]);

top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(4,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
```

```
S3 = [S1,S2];
S = sort(S3);

c=1;
G3=[];
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    P = [c2(1);c3(1);r2(1)];
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(P(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(P(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(P(3),[1,
       t_size(2)]);
    p_in = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P3 = [P3,p_in];

    T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
       x_10(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
```

```
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_12(psi)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);



elseif
G_J<G_0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                          Set K                                %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    gx10 = gradient(x_10(t));
    kay1 = s*cumtrapz(gx10,2);
    kay2 = s*cumtrapz(x_9(t),2);
    kay3 = s*cumtrapz(x_4(t),2);
```

```
kay4 = s*cumtrapz(x_1(t),2);
K = [kay1;kay2;kay3;kay4];


% Here, we put the three matters the most up front
k1 = K(small_index(1),:);
k2 = K(small_index(2),:);
k3 = K(small_index(3),:);
k4 = K(big_index(1),:);
K_new = [k1;k2;k3;k4];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Set M                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1_in = s*cumtrapz(kay1,2);
m1 = (1/(mo(b)-mo(a)))*s*trapz(m1_in,2)-m1_in;

m2_in = s*cumtrapz(kay2,2);
m2 = (1/(mo(b)-mo(a)))*s*trapz(m2_in,2)-m2_in;

m3_in = s*cumtrapz(kay3,2);
m3 = (1/(mo(b)-mo(a)))*s*trapz(m3_in,2)-m3_in;

m4_in = s*cumtrapz(kay4,2);
m4 = (1/(mo(b)-mo(a)))*s*trapz(m4_in,2)-m4_in;

m = [m1;m2;m3;m4];
```

```
M1 = m(small_index(1),:);
M2 = m(small_index(2),:);
M3 = m(small_index(3),:);
M4 = m(big_index(1),:);


% Set the constants which doesn't change here
JJ = J2(43:46);
c4 = repmat(JJ(big_index(1)),[1,t_size(2)]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Fixing c2 and c3 as random                    %
%                 Keeping c_nc1 steady                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2=JJ(small_index(2));
b3=JJ(small_index(3));
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k1.^2+M1.^2,2);

B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c3.*(k1.*k3+M1.*M3)+2*
   c4.*(k1.*k4+M1.*M4)-2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);
```

```
B3_in = c2.^2.*(k2.^2+M2.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c2.*c3.*(k2.*k3+M2.*M3)+2*c2.*c4.*(k2.*k4+M2.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c2.*(g.*k2+up.*M2)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;



if B2^2-4*B1*(B3)<0
    small_1 = G_J;
    best_P1 = J2(43:46);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);


        psi_0 = zeros(4,t_size(2));
```

```
            psi_0(small_index(1),:) = repmat(r2(1),[1,
               t_size(2)]);
            psi_0(small_index(2),:) = repmat(c2(2),[1,
               t_size(2)]);
            psi_0(small_index(3),:) = repmat(c3(3),[1,
               t_size(2)]);
            psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
               (2)]);
            P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
               (4,1)];
            P1 = [P1,P];

            T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
               x_10(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);

            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c1 and c3 as random            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for h=1
b1=JJ(small_index(1));
b3=JJ(small_index(3));
r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c1 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);

B1 = s*trapz(k2.^2+M2.^2,2);

B2_in = 2*c1.*(k1.*k2+M1.*M2)+2*c3.*(k2.*k3+M2.*M3)+2*
   c4.*(k2.*k4+M2.*M4)-2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c3.^2.*(k3.^2+M3.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c3.*(k1.*k3+M1.*M3)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c3.*c4.*(k3.*k4+M3.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c3.*(g.*k3+up.*M3)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;
```

```
if B2^2-4*B1*(B3)<0
    small_2 = G_J;
    best_P2 = J2(43:46);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);

        psi_0 = zeros(4,t_size(2));
        psi_0(small_index(1),:) = repmat(c1(1),[1,
           t_size(2)]);
        psi_0(small_index(2),:) = repmat(r2(2),[1,
           t_size(2)]);
        psi_0(small_index(3),:) = repmat(c3(3),[1,
           t_size(2)]);
        psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
           (2)]);
        P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
           (4,1)];
        P2 = [P2,P];
```

```
            T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
               x_10(t),s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end
        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
    end

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Fixing c1 and c2 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    b1=JJ(small_index(1));
    b2=JJ(small_index(2));
    r1 = (b1-dd)+((b1+dd)-(b1-dd)).*rand(5000,1);
    r2 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
    c1 = repmat(r1(1),[1,t_size(2)]);
    c2 = repmat(r2(2),[1,t_size(2)]);
```

```
B1 = s*trapz(k3.^2+M3.^2,2);

B2_in = 2*c1.*(k1.*k3+M1.*M3)+2*c2.*(k2.*k3+M2.*M3)+2*
   c4.*(k3.*k4+M3.*M4)-2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);

B3_in = c1.^2.*(k1.^2+M1.^2)+c2.^2.*(k2.^2+M2.^2)+c4
   .^2.*(k4.^2+M4.^2)...
    +2*c1.*c2.*(k1.*k2+M1.*M2)+2*c1.*c4.*(k1.*k4+M1.*
       M4)+2*c2.*c4.*(k2.*k4+M2.*M4)...
    -2*c1.*(g.*k1+up.*M1)-2*c2.*(g.*k2+up.*M2)-2*c4.*(
       g.*k4+up.*M4)...
    +g.^2+up.^2;
B3 = s*trapz(B3_in,2)-G_J;


if B2^2-4*B1*(B3)<0
    small_3 = G_J;
    best_P3 = J2(43:46);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(B3)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
```

```
P3=[];
while c<5
    r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
    psi_0 = zeros(4,t_size(2));
    psi_0(small_index(1),:) = repmat(c1(1),[1,
       t_size(2)]);
    psi_0(small_index(2),:) = repmat(c2(2),[1,
       t_size(2)]);
    psi_0(small_index(3),:) = repmat(r2(3),[1,
       t_size(2)]);
    psi_0(big_index(1),:) = repmat(c4(1),[1,t_size
       (2)]);
    P = [psi_0(1,1);psi_0(2,1);psi_0(3,1);psi_0
       (4,1)];
    P3 = [P3,P];

    T_psi_0=T_reg_x12(psi_0,x_1(t),x_4(t),x_9(t),
       x_10(t),s);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

    g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
       -up).^2,2)*s);
    G3 = [G3;g_less];
    c = c+1;
end
[small_3, index_3] = mink(G3,1);
```

```
        best_P3 = P3(:,index_3);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_12(psi      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    smalls = [small_1;small_2;small_3];
    bests = [best_P1,best_P2,best_P3];

    [small,index] = mink(smalls,1);
    final_psi = repmat(bests(:,index),[1,t_size(2)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Calculating G_12(best_psi) and saving it          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T_psi_0=T_reg_x12(final_psi,x_1(t),x_4(t),x_9(t),x_10(t),s
   );

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_best = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
% fprintf('G(final_x12): %d\n',G_best);

writematrix(final_psi,'final_psi_x12.txt');

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          for T_13(psi)=g_13         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Getting G_13(0)                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_13(t)-x_13(1);

psi_0 = repmat(0,[3,t_size(2)]);

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(0_x13): %d\n',G_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Getting G_13(J)                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(J2(47:49),[3,t_size(2)]);

T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),s);
```

```
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);

% fprintf('G(J_x13): %d\n',G_J);

if G_0<=G_J
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %           Fixing c3 and c4 as random                          %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c3 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);

    k1 = s*cumtrapz(x_10(t),2);
    k2 = s*cumtrapz(x_6(t),2);
    k3 = s*cumtrapz(x_7(t),2);



    M1_in = s*cumtrapz(k1,2);
    M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

    M2_in = s*cumtrapz(k2,2);
    M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

    M3_in = s*cumtrapz(k3,2);
    M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;
```

```
top1_in = c3.*(g.*k2+up.*M2)+c4.*(g.*k3+up.*M3);
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_1 = G_0;
    best_P1 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G1=[];
    P1=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
```

```
            P = [r2(1);c3(1);c4(1)];
            P1 = [P1,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
               s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G1 = [G1;g_less];
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
        best_P1 = P1(:,index_1);
    end
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Fixing c2 and c4 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c2 = repmat(r(1),[1,t_size(2)]);
    c4 = repmat(r(2),[1,t_size(2)]);
```

```
top1_in = c2.*(g.*k1+up.*M1)+c4.*(g.*k3+up.*M3);
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(-top)<0
    small_2 = G_0;
    best_P2 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
```

```
            P2 = [P2,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
               s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G2 = [G2;g_less];
            c = c+1;
        end

        [small_2, index_2] = mink(G2,1);
        best_P2 = P2(:,index_2);
    end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Fixing c2 and c3 as random                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    r = -dd+(dd--dd).*rand(5000,1);
    c2 = repmat(r(1),[1,t_size(2)]);
    c3 = repmat(r(2),[1,t_size(2)]);

    top1_in = c2.*(g.*k1+up.*M1)+c3.*(g.*k2+up.*M2);
```

```
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = 2*s*trapz(top1_in,2)-s*trapz(top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
B2 = s*trapz(B2_in,2);



if B2^2-4*B1*(-top)<0
    small_3 = G_0;
    best_P3 = zeros(3,1);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(-top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);


    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        P3 = [P3,P];
        psi_0 = repmat(P,[1,t_size(2)]);
```

```
            T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
               s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
    end

elseif G_J<G_0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for h=1
    b3 = JJ(2);
    b4 = JJ(3);
    r1 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
    r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
    c3 = repmat(r1(1),[1,t_size(2)]);
    c4 = repmat(r2(2),[1,t_size(2)]);

    k1 = s*cumtrapz(x_10(t),2);
```

```
k2 = s*cumtrapz(x_6(t),2);
k3 = s*cumtrapz(x_7(t),2);


M1_in = s*cumtrapz(k1,2);
M1 = (1/(mo(b)-mo(a)))*s*trapz(M1_in,2)-M1_in;

M2_in = s*cumtrapz(k2,2);
M2 = (1/(mo(b)-mo(a)))*s*trapz(M2_in,2)-M2_in;

M3_in = s*cumtrapz(k3,2);
M3 = (1/(mo(b)-mo(a)))*s*trapz(M3_in,2)-M3_in;


top1_in = -2*c3.*(g.*k2+up.*M2)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c3.^2.*(k2.^2+M2.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c3.*c4.*(k2.*k3+M2.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k1.^2+M1.^2,2);
B2_in = 2*c3.*(k1.*k2+M1.*M2)+2*c4.*(k1.*k3+M1.*M3)
   -2*(g.*k1+up.*M1);
B2 = s*trapz(B2_in,2);


if B2^2-4*B1*(top)<0
```

```
        small_1 = G_J;
        best_P1 = repmat(J2(47:49),[1,t_size(2)]);
    else
        S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
        S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
        S3 = [S1,S2];
        S = sort(S3);

        c=1;
        G1=[];
        P1=[];
        while c<5
            r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
            P = [r2(1);c3(1);c4(1)];
            P1 = [P1,P];
            psi_0 = repmat(P,[1,t_size(2)]);

            T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
               s);

            u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

            g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
               -up).^2,2)*s);
            c = c+1;
        end
        [small_1, index_1] = mink(G1,1);
```

```
    best_P1 = P1(:,index_1);
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Fixing c2 and c4 as random                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b4 = JJ(3);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b4-dd)+((b4+dd)-(b4-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c4 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c4.*(g.*k3+up.*M3)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c4.^2.*(k3.^2+M3.^2)+2*
   c2.*c4.*(k1.*k3+M1.*M3);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k2.^2+M2.^2,2);
B2_in = 2*c2.*(k1.*k2+M1.*M2)+2*c4.*(k2.*k3+M2.*M3)
   -2*(g.*k2+up.*M2);
B2 = s*trapz(B2_in,2);
```

```
if B2^2-4*B1*(top)<0
    small_2 = G_J;
    best_P2 = J2(47:49);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G2=[];
    P2=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);r2(1);c4(1)];
        P2 = [P2,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
           s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
        G2 = [G2;g_less];
        c = c+1;
```

```
    end
    [small_2, index_2] = mink(G2,1);
    best_P2 = P2(:,index_2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Fixing c2 and c3 as random                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
b2 = JJ(1);
b3 = JJ(2);
r1 = (b2-dd)+((b2+dd)-(b2-dd)).*rand(5000,1);
r2 = (b3-dd)+((b3+dd)-(b3-dd)).*rand(5000,1);
c2 = repmat(r1(1),[1,t_size(2)]);
c3 = repmat(r2(2),[1,t_size(2)]);


top1_in = -2*c2.*(g.*k1+up.*M1)-2*c3.*(g.*k2+up.*M2)+g
   .^2+up.^2;
top2_in = c2.^2.*(k1.^2+M1.^2)+c3.^2.*(k2.^2+M2.^2)+2*
   c2.*c3.*(k1.*k2+M1.*M2);
top = -G_J+s*trapz(top1_in+top2_in,2);

B1 = s*trapz(k3.^2+M3.^2,2);
B2_in = 2*c2.*(k1.*k3+M1.*M3)+2*c3.*(k2.*k3+M2.*M3)
   -2*(g.*k3+up.*M3);
```

```
B2 = s*trapz(B2_in,2);

if B2^2-4*B1*(top)<0
    small_3 = G_J;
    best_P3 = J2(47:49);
else
    S1 = (-B2+sqrt(B2^2-4*B1*(top)))/(2*B1);
    S2 = (-B2-sqrt(B2^2-4*B1*(top)))/(2*B1);
    S3 = [S1,S2];
    S = sort(S3);

    c=1;
    G3=[];
    P3=[];
    while c<5
        r2 = S(1)+(S(2)-S(1)).*rand(10000,1);
        P = [c2(1);c3(1);r2(1)];
        P3 = [P3,P];
        psi_0 = repmat(P,[1,t_size(2)]);

        T_psi_0=T_reg_x13(psi_0,x_6(t),x_7(t),x_10(t),
           s);

        u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);

        g_less = sum(trapz((T_psi_0-g).^2+(u_psi_prime
           -up).^2,2)*s);
```

```
            G3 = [G3;g_less];
            c = c+1;
        end
        [small_3, index_3] = mink(G3,1);
        best_P3 = P3(:,index_3);
    end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getting the psi with the smallest G_13(psi)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
smalls = [small_1;small_2;small_3];
bests = [best_P1,best_P2,best_P3];

[small,index] = mink(smalls,1);
best_psi = repmat(bests(:,index),[1,t_size(2)]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculating G_13(best_psi) and saving it        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
final_psi = best_psi;
common = s*cumtrapz(g(1,:),2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_x13(final_psi,x_6(t),x_7(t),x_10(t),s);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
G_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s);
% fprintf('G(best_x13): %d\n',G_0);

writematrix(final_psi,'final_psi_x13.txt');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Putting T_1 through T_13 together                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
g = [x_1(t)-x_1(1);x_2(t)-x_2(1);...
    x_3(t)-x_3(1);x_4(t)-x_4(1);...
    x_5(t)-x_5(1);x_6(t)-x_6(1);...
    x_7(t)-x_7(1);x_8(t)-x_8(1);...
    x_9(t)-x_9(1);x_10(t)-x_10(1);...
    x_11(t)-x_11(1);x_12(t)-x_12(1);...
    x_13(t)-x_13(1)];
g_size=size(g);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Putting together the psi      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p1 = readmatrix('final_psi_x1.txt');
p2 = readmatrix('final_psi_x2.txt');
p3 = readmatrix('final_psi_x3.txt');
p4 = readmatrix('final_psi_x4.txt');
p5 = readmatrix('final_psi_x5.txt');
```

```
p6 = readmatrix('final_psi_x6.txt');
p7 = readmatrix('final_psi_x7.txt');
p8 = readmatrix('final_psi_x8.txt');
p9 = readmatrix('final_psi_x9.txt');
p10 = readmatrix('final_psi_x10.txt');
p11 = readmatrix('final_psi_x11.txt');
p12 = readmatrix('final_psi_x12.txt');
p13 = readmatrix('final_psi_x13.txt');

final_psi = [p1;p2;p3;p4;p5;p6;p7;p8;p9;p10;p11;p12;p13];
writematrix(final_psi,'better_than_5.txt');
psi_0 = final_psi;

ux=u(t,g,s,mo(a),mo(b));

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t),x_5(t),...
    x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t),
       x_12(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
```

```
auto_g = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*s
   );
fprintf('Final Gpsi: %d\n',auto_g);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             Using the psi to solve the DDE              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c0 = readmatrix('better_than_5.txt');
c1 = c0(:,end);
c=KJ_to_J(c1);
writematrix(c,'forward_kj.txt');

%-------------------------------------------------------
%Solving the DDE system

A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare DDE resuts
    to
time=linspace(v,v+l,length(A));
tspan=[v,v+l]; %time for which the DDE is solved--in
   months
sol=dde23(@delay_buy_kj,lags, x0,tspan);      %solve DDE
y=interp1(sol.x,sol.y(8,:),time,'linear'); %linearly
   interpolate sol of stock price down to days
```

```
der=interp1(sol.x,sol.yp(8,:),time, 'linear'); %lineaerly
   interpolate derivative down to days

    if min(y)<x0(8)-bound
    elseif min(y)<0
    elseif max(y)>x0(8)+bound
    else
        fprintf('This one is good G(psi):%d\n',auto_g);
        good_psi = [good_psi,final_psi(:,1)];
        fprintf('Count = %d\n',count)
        writematrix(final_psi,'better_psi.txt');
        count=count+1;
        fprintf('_____________________\n');
    end

end

writematrix(good_psi,'round1_good_psis.txt');
```

Appendix B: Get_Constants_Which_one.m

Get_Constants_Which_one.m

```
clc
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code will determine which psis to use by                         %
%calculating G(psi) and G_8(psi).                                       %
% Input: v, a, b, s, l                                                  %
% Output: Maximum of five initial psis to use for the                   %
% descent process                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Warnings                                           %
% 1) You must set v, a, b, s, and l as was in                           %
%   "Get_Constants_Final.m"                                             %
% 2) If this algorithm does not return any overlapping                  %
%   output, plese re-run "Get_Constants_Final.m" and                    %
%   re-run this program.                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Set v, a, b, s, l, and lags                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=343;
a=v-1;
b=v;
s=1;
l=2;
lags=[12,1,1,20,1,12,17,1,1,1,1,1,1];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Preprocess the data                               %
% This portion is a modified portion of Barnett's data                 %
% preprocessing process. If a user decides to extend the               %
% data, 'mo' must be changed accordingly.                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
```

```
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];

%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;1156;
    1187;1217;1248;1278;1309;1340;1370;1401;1431;1462;
    1493;1522;1553;1583;1614;1644;1675;1706;1736;1767;
    1797;1828;1859;1887;1918;1948;1979;2009;2040;2071;
    2101;2132;2162;2193;2224;2252;2283;2313;2344;2374;
    2405;2436;2466;2497;2527;2558;2589;2617;2648;2678;
    2709;2739;2770;2801;2831;2862;2892;2923;2954;2983;
    3014;3044;3075;3105;3136;3167;3197;3228;3258;3289;
    3320;3348;3379;3409;3440;3470;3501;3532;3562;3593;
    3623;3654;3685;3713;3744;3774;3805;3835;3866;3897;
    3927;3958;3988;4019;4050;4078;4109;4139;4170;4200;
    4231;4262;4292;4323;4353;4384;4415;4444;4475;4505;
    4536;4566;4597;4628;4658;4689;4719;4750;4781;4809;
```

```
    4840;4870;4901;4931;4962;4993;5023;5054;5084;5115;
    5146;5174;5205;5235;5266;5296;5327;5358;5388;5419;
    5449;5480;5511;5539;5570];

%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
```

```
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;


% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector

%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);   % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
%Interpolated Apple stock price
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price and inearly interpolated from
   there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
```

```
vnz3 = find(SandP500 ~= 0);
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
ir_nz = ir1(ir1 ~= 0);
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
```

```
rd = interp1(vnz8, rd_nz, iv, 'linear')';

%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
vnz12 = find(iphone1 ~= 0);
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
```

```
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting on December 1,2007, so now we
   account for that and start all vectors on January 1,
   2008
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-----------------------------------------------------
%Complete the same process with the older data files
%-----------------------------------------------------
```

```
%Apple's Net Income
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
 %Apple didn't have anybuybacks prior to 2008 so this
    variable is 0 pple's EPS
buyback_old=zeros(mo1(end),1);

eps_old=csvread('old_eps.csv',1,1);
```

```
%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);                %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);  %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0);%Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
%Interpolated APPL function is above


%The same process is repeated for each variable

%Apple's Net Income
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
```

```
cpi_onz = cpi_old(cpi_old ~= 0);
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';

%GLD
gld_onz = gld_old(gld_old ~= 0);
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);

%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';

%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
```

```
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';

%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';

%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);

%Confidence Index
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';

%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special cases) instead of January 1,2018,
   which is where the new data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
```

```
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
%caculate P/E ratio using EPS and stock price
pe_old=AAPL_old./eps(1:mo1(end)-1);

%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];
```

```
mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month

end


t= (mo(a):mo(b));
t_size = size(t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Setting up x_1(t)~x_13(t)                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_1(t)=cpi(mo(a):mo(b),:);
x_2(t)=inflation(mo(a):mo(b),:);
x_3(t)=pe(mo(a):mo(b),:);
x_4(t)=ir(mo(a):mo(b),:);
x_4d(t)=ir(mo(a-20):mo(a-20)+t_size(2)-1,:); % 20 month
   delay
x_5(t)=NASDAQ100(mo(a):mo(b),:);
x_6(t)=rev(mo(a):mo(b),:);
x_7(t)=rd(mo(a):mo(b),:);
x_7d(t)=rd(mo(a-17):mo(a-17)+t_size(2)-1,:); % 17 month
   delay
x_8(t)=AAPL(mo(a):mo(b),:);
x_9(t)=gld(mo(a):mo(b),:);
```

```
x_10(t)=sp500(mo(a):mo(b),:);
x_11(t)=iphone(mo(a):mo(b),:);
x_12(t)=conf(mo(a):mo(b),:);
x_13(t)=buyback(mo(a):mo(b),:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Calculate G_8(psi)s                                     %
% G8s: This holds all G_8(psi) values                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = x_8(t)-x_8(1);

p = readmatrix('round1_good_psis.txt');

All_P=p(24:30,:);
ps = size(p);

G8s =[];
for i=1:ps(2)
    psi_0 = repmat(All_P(:,i),[1,t_size(2)]);

    common = s*cumtrapz(g,2);
    up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

    T_psi_0=T_reg_x8(psi_0,x_3(t),x_4(t),x_5(t),x_6(t),x_8
       (t),x_10(t),x_13(t),s);
```

```
    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
    G_J = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
       s);
    G8s = [G8s;G_J];
    fprintf('G(x8): %d\n',G_J)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Getting the lowest 5 G_8(psi) values and its index   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[C,D] = mink(G8s,5);
fprintf('Lowest = %d\n',D);

g = [x_1(t)-x_1(1);x_2(t)-x_2(1);...
    x_3(t)-x_3(1);x_4(t)-x_4(1);...
    x_5(t)-x_5(1);x_6(t)-x_6(1);...
    x_7(t)-x_7(1);x_8(t)-x_8(1);...
    x_9(t)-x_9(1);x_10(t)-x_10(1);...
    x_11(t)-x_11(1);x_12(t)-x_12(1);x_13(t)-x_13(1)];
g_size=size(g);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Calculate G(psi)s                                      %
% Gs: This holds all G(psi) values                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gs = [];
for i = 1:ps(2)
```

```
    psi_0 = p(:,i);

    ux=u(t,g,s,mo(a),mo(b));

    common = s*cumtrapz(g,2);
    up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

    T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t),x_5(t),...
         x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
            ),x_12(t),x_13(t),s);

    u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

    u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
    auto_g = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up)
       .^2,2)*s);
    fprintf('G(psi)=%d\n',auto_g);
    Gs = [Gs,auto_g];
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Getting the lowest 5 G(psi) values and its index    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[C2,D2] = mink(Gs,5);
fprintf('Lowest = %d\n',D2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Define the hisotry vector for dde23                         %
% See Chapter 2 Section 2 for details.                                   %
% We use constant functions as our history vector.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0(1)=cpi(mo(v));
x0(2)=inflation(mo(v));
x0(3)=pe(mo(v));
x0(4)=ir(mo(v));
x0(5)=NASDAQ100(mo(v));
x0(6)=rev(mo(v));
x0(7)=rd(mo(v));
x0(8)=AAPL(mo(v));
x0(9)=gld(mo(v));
x0(10)=sp500(mo(v));
x0(11)=iphone(mo(v));
x0(12)=conf(mo(v));
x0(13)=buyback(mo(v));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Solving the DDE and saving the solutions                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ys = [];
for i=1:ps(2)
    c0 = p(:,i);
    c = KJ_to_J(c0);
    writematrix(c,'forward_kj.txt');
```

```
    %-----------------------------------------------------
    A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare DDE
       resuts to
    time=linspace(v,v+l,length(A));
    tspan=[v,v+l]; %time for which the DDE is solved--in
       months
    sol=dde23(@delay_buy_kj,lags, x0,tspan);      %solve
       DDE
    y=interp1(sol.x,sol.y(8,:),time,'linear'); %linearly
       interpolate sol of stock price down to days

    ys = [ys,y'];
    der=interp1(sol.x,sol.yp(8,:),time, 'linear'); %
       lineaerly interpolate derivative down to days
end


smalls = [D';D2];
disp(smalls)

[val,pos]=intersect(D',D2);

fprintf('Use these psis: %d\n',val);
```

Appendix C: H1_All_Psis.m

H1_All_Psis.m

```
clc
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code applies the gradient descent method to the  %
% initial psis found using "Get_Constants_Which_one.m". %
% Input: v,s, a, b, s, l, and psis suggested by         %
% "Get_Constants_Which_one.m                            %
% Output: minimized psis which are now non-constant     %
% functions of time.                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Warnings                          %
% 1) Don't panic if the program stops after certain     %
% iterations. It is designed to stop if the solution    %
% goes out of the range.                                %
% 2) There are maximum of five descent process that     %
% could happen. If one has less than five psis, this    %
% program will stop after the set amount. It may look   %
% like an error but it is not.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Set v, a, b, s, l, lags, max_diff, and bound      %
% Each one of these should be equal to the values from  %
%   "Get_Constants_Final.m".                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=343;
a=v-1;
b=v;
s=1;
l=2;
lags=[12,1,1,20,1,12,17,1,1,1,1,1,1];
max_diff = get_max_diff(v);
bound = max_diff;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input the psis suggested from                         %
%"Get_Constants_Which_one.m"                            %
% The psis suggested should go into "good_psis" as shown%
% below.                                                %
% For example if "5 and 10" was suggested, we enter     %
%               good_psis = [good_psis_0(:,[5,10])];    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
good_psis_0 = readmatrix('round1_good_psis.txt');
good_psis = [good_psis_0(:,[4])];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Preprocess the data                %
% This portion is a modified portion of Barnett's data  %
% preprocessing process. If a user decides to extend the%
% data, 'mo' must be changed accordingly.               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];
```

```
%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;1156;
    1187;1217;1248;1278;1309;1340;1370;1401;1431;1462;
    1493;1522;1553;1583;1614;1644;1675;1706;1736;1767;
    1797;1828;1859;1887;1918;1948;1979;2009;2040;2071;
    2101;2132;2162;2193;2224;2252;2283;2313;2344;2374;
    2405;2436;2466;2497;2527;2558;2589;2617;2648;2678;
    2709;2739;2770;2801;2831;2862;2892;2923;2954;2983;
    3014;3044;3075;3105;3136;3167;3197;3228;3258;3289;
    3320;3348;3379;3409;3440;3470;3501;3532;3562;3593;
    3623;3654;3685;3713;3744;3774;3805;3835;3866;3897;
    3927;3958;3988;4019;4050;4078;4109;4139;4170;4200;
    4231;4262;4292;4323;4353;4384;4415;4444;4475;4505;
    4536;4566;4597;4628;4658;4689;4719;4750;4781;4809;
    4840;4870;4901;4931;4962;4993;5023;5054;5084;5115;
    5146;5174;5205;5235;5266;5296;5327;5358;5388;5419;
    5449;5480;5511;5539;5570];

%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
```

```
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;



% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector
```

```
%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);   % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
%Interpolated Apple stock price
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price and inearly interpolated from
   there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
vnz3 = find(SandP500 ~= 0);
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
```

```
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
ir_nz = ir1(ir1 ~= 0);
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
rd = interp1(vnz8, rd_nz, iv, 'linear')';

%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
```

```
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
vnz12 = find(iphone1 ~= 0);
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting on December 1,2007, so now we
   account for that and start all vectors on January 1,
   2008
```

```
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-----------------------------------------------------------
%Complete the same process with the older data files
%-----------------------------------------------------------
%Apple's Net Income
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
```

```
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
 %Apple didn't have anybuybacks prior to 2008 so this
    variable is 0 pple's EPS
buyback_old=zeros(mo1(end),1);

eps_old=csvread('old_eps.csv',1,1);

%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);              %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);  %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0);%Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
```

```
%Interpolated APPL function is above


%The same process is repeated for each variable

%Apple's Net Income
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
cpi_onz = cpi_old(cpi_old ~= 0);
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
```

```
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';

%GLD
gld_onz = gld_old(gld_old ~= 0);
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);

%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';

%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';

%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';

%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
```

```
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);

%Confidence Index
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';

%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special cases) instead of January 1,2018,
   which is where the new data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
```

```
%caculate P/E ratio using EPS and stock price
pe_old=AAPL_old./eps(1:mo1(end)-1);

%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];

mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Set up the time steps                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t= (mo(a):mo(b));
```

```
t_size = size(t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Setting up x_1(t)~x_13(t)                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_1(t)=cpi(mo(a):mo(b),:);
x_2(t)=inflation(mo(a):mo(b),:);
x_3(t)=pe(mo(a):mo(b),:);
x_4(t)=ir(mo(a):mo(b),:);
x_4d(t)=ir(mo(a-20):mo(a-20)+t_size(2)-1,:); % 20 month
   delay
x_5(t)=NASDAQ100(mo(a):mo(b),:);
x_6(t)=rev(mo(a):mo(b),:);
x_7(t)=rd(mo(a):mo(b),:);
x_7d(t)=rd(mo(a-17):mo(a-17)+t_size(2)-1,:); % 17 month
   delay
x_8(t)=AAPL(mo(a):mo(b),:);
x_9(t)=gld(mo(a):mo(b),:);
x_10(t)=sp500(mo(a):mo(b),:);
x_11(t)=iphone(mo(a):mo(b),:);
x_12(t)=conf(mo(a):mo(b),:);
x_13(t)=buyback(mo(a):mo(b),:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Setting up g                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g = [x_1(t)-x_1(1);x_2(t)-x_2(1);...
```

```
    x_3(t)-x_3(1);x_4(t)-x_4(1);...
    x_5(t)-x_5(1);x_6(t)-x_6(1);...
    x_7(t)-x_7(1);x_8(t)-x_8(1);...
    x_9(t)-x_9(1);x_10(t)-x_10(1);...
    x_11(t)-x_11(1);x_12(t)-x_12(1);...
    x_13(t)-x_13(1)];
g_size=size(g);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Define the hisotry vector for dde23                       %
% See Chapter 2 Section 2 for details.                                %
% We use constant functions as our history vector.                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0(1)=cpi(mo(v));
x0(2)=inflation(mo(v));
x0(3)=pe(mo(v));
x0(4)=ir(mo(v));
x0(5)=NASDAQ100(mo(v));
x0(6)=rev(mo(v));
x0(7)=rd(mo(v));
x0(8)=AAPL(mo(v));
x0(9)=gld(mo(v));
x0(10)=sp500(mo(v));
x0(11)=iphone(mo(v));
x0(12)=conf(mo(v));
x0(13)=buyback(mo(v));
```

```
% psi_1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Gradient Descent using Psi_1                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up psi_0, u, u', and u_psi.                 %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(good_psis(:,1),[1,t_size(2)]);
ux=u(t,g,s,mo(a),mo(b));
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the L2 Gradient                          %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
in=ux-u_psi_0+g-T_psi_0;
L2Gpsi_0_n=-2*T_star_data(in,x_1(t),x_2(t),x_3(t),x_4d(t),
   x_4(t),...
x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t)
   ,...
x_12(t),x_13(t),s,psi_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the H1 Gradient using Neuman   %
%       boundary condition.             %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H1 = [];
L2size = size(L2Gpsi_0_n);
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_0_n(i,:),mo(a),mo(b),s); %See the
       function called "boundary"
    H1 = [H1;S];
end
H1Gpsi_0 = H1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up alpha.                         %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_h=max(abs(H1),[],"all");
low = max_h*1e3;
```

```
high = max_h*1e4;
alpha_options=[1e10,1e11,1e12,1e13,1e14,1e15,1e16,1e17,1
   e18,1e19,1e20,1e21,1e22,1e23,1e24,1e25,1e26,1e27,1e28,1
   e29,1e30,1e31,1e32];
for i = alpha_options
   if (i<high) && (i>low)
       alpha=1/i;
   else
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up u'_psi and Calculate G(psi)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
g_psi_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
   s);
fprintf('Gpsi: %d',g_psi_0);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we keep the values      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_s=[psi_0];      % All the psi_m
fm_s=[1.0e+80];             % All the G(psi_m)
u_psi_s=[u_psi_0];  % All the u_psi_m
```

```
L2G=[L2Gpsi_0_n];   % All the L2 gradient with respect to
   psi_m
H1G=[H1Gpsi_0];     % All the H1 gradient with respect to
   psi_m
T_psi_s = [T_psi_0];% All the T(psi_m)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     For loop to update psi_m            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if g_psi_0<0.0000001
    fprintf('Minimizing Psi is \n');
    disp(psi_0);
    fprintf('G_psi is %d \n',sum(trapz((T_psi_0-g).^2+(
       u_psi_prime-up).^2,2)*s));
else
T_psi_m = T_psi_0;
k=1;
while k<100
    % we need below i1 and i2 to get the correct columns
    i1 = t_size(2)*(k-1)+1;
    i2 = t_size(2)*k;

    % Step 1) Update psi_m
    psi_m=psi_s(:,[i1:i2])-(alpha)*H1G(:,[i1:i2]);
```

```
    psi_s=[psi_s,psi_m];              % Append psi_m to the
       right

    % Step 2) Get T(psi_m)
    T_psi_m=T_reg_data(psi_m,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

    % Step 3) Get u_psi_m
    u_psi_m=u_psi(T_psi_m,mo(a),mo(b),s,t);

    % Step 4) Get G(psi_m)=fm
    fm=sum(trapz((T_psi_m-g).^2,2)*s);
    fm_s=[fm_s,fm];                    % Appending fm to the
       right
    if fm_s(end-1)-fm<0
        fprintf('STOP%d\n',k)
        break
    end

    % Step 5) Get L2 gradient with respect to psi_m
    in_m = ux-u_psi_m+g-T_psi_m;
    L2Gpsi_m=-2*T_star_data(in_m,x_1(t),x_2(t),x_3(t),x_4d
       (t),x_4(t)...
```

```
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s,psi_m);

    % Step 6) Get H1 gradient with respect to psi_m
    H1G_psi_m = [];
    parfor i = 1:L2size(1)
        S = boundary(L2Gpsi_m(i,:),mo(a),mo(b),s);
        H1G_psi_m = [H1G_psi_m;S];
    end
    H1G=[H1G,H1G_psi_m];            % Append H1Gpsi_m to
       the right

    k = k+1;

    if rem(k,10)==0
        c1 = psi_m(:,end);
        c = KJ_to_J(c1);
        writematrix(c,'forward_kj.txt');

        %-----------------------------------------------
        %Solving the DDE system
        A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare
           DDE resuts to
        time=linspace(v,v+l,length(A));
        tspan=[v,v+l]; %time for which the DDE is solved--
           in months
```

```
        sol=dde23(@delay_buy_kj,lags, x0,tspan);      %
           solve DDE
        y=interp1(sol.x,sol.y(8,:),time,'linear'); %
           linearly interpolate sol of

        if min(y)<x0(8)-bound
           break
        elseif min(y)<0
            break
        elseif max(y)>x0(8)+bound
            break
        else
            fprintf('Iteration %d\n',k);
            fprintf('G(psi_m) first term: %d\n',fm);
            writematrix(psi_m,'round1_min_1.txt');
        end
    end

end  % end of while loop
end  % end of if loop in the big if loop
end

% psi_2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Gradient Descent using Psi_2                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up psi_0, u, u', and u_psi.                 %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(good_psis(:,2),[1,t_size(2)]);

ux=u(t,g,s,mo(a),mo(b));

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the L2 Gradient                          %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
in=ux-u_psi_0+g-T_psi_0;
L2Gpsi_0_n=-2*T_star_data(in,x_1(t),x_2(t),x_3(t),x_4d(t),
   x_4(t),...
```

```
x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t)
   ,...
x_12(t),x_13(t),s,psi_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the H1 Gradient using Neuman   %
%       boundary condition.             %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H1 = [];
L2size = size(L2Gpsi_0_n);
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_0_n(i,:),mo(a),mo(b),s); %See the
       function called "boundary"
    H1 = [H1;S];
end
H1Gpsi_0 = H1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up alpha.                         %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_h=max(abs(H1),[],"all");
low = max_h*1e3;
high = max_h*1e4;
```

```
alpha_options=[1e10,1e11,1e12,1e13,1e14,1e15,1e16,1e17,1
   e18,1e19,1e20,1e21,1e22,1e23,1e24,1e25,1e26,1e27,1e28,1
   e29,1e30,1e31,1e32];
for i = alpha_options
   if (i<high) && (i>low)
       alpha=1/i;
   else
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up u'_psi and Calculate G(psi)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
g_psi_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
   s);
fprintf('Gpsi: %d',g_psi_0);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we keep the values      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_s=[psi_0];      % All the psi_m
fm_s=[1.0e+80];             % All the G(psi_m)
u_psi_s=[u_psi_0];  % All the u_psi_m
```

```
L2G=[L2Gpsi_0_n];   % All the L2 gradient with respect to
   psi_m
H1G=[H1Gpsi_0];     % All the H1 gradient with respect to
   psi_m
T_psi_s = [T_psi_0];% All the T(psi_m)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             For loop to update psi_m                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if g_psi_0<0.0000001
    fprintf('Minimizing Psi is \n');
    disp(psi_0);
    fprintf('G_psi is %d \n',sum(trapz((T_psi_0-g).^2+(
       u_psi_prime-up).^2,2)*s));
else
T_psi_m = T_psi_0;
k=1;
while k<100
    % we need below i1 and i2 to get the correct columns
    i1 = t_size(2)*(k-1)+1;
    i2 = t_size(2)*k;

    % Step 1) Update psi_m
    psi_m=psi_s(:,[i1:i2])-(alpha)*H1G(:,[i1:i2]);
    psi_s=[psi_s,psi_m];            % Append psi_m to the
       right
```

```
    % Step 2) Get T(psi_m)
    T_psi_m=T_reg_data(psi_m,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

    % Step 3) Get u_psi_m
    u_psi_m=u_psi(T_psi_m,mo(a),mo(b),s,t);

    % Step 4) Get G(psi_m)=fm
    fm=sum(trapz((T_psi_m-g).^2,2)*s);
    fm_s=[fm_s,fm];                    % Appending fm to the
        right
    if fm_s(end-1)-fm<0
        fprintf('STOP%d\n',k)
        break
    end

    % Step 5) Get L2 gradient with respect to psi_m
    in_m = ux-u_psi_m+g-T_psi_m;
    L2Gpsi_m=-2*T_star_data(in_m,x_1(t),x_2(t),x_3(t),x_4d
       (t),x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s,psi_m);
```

```
% Step 6) Get H1 gradient with respect to psi_m
H1G_psi_m = [];
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_m(i,:),mo(a),mo(b),s);
    H1G_psi_m = [H1G_psi_m;S];
end
H1G=[H1G,H1G_psi_m];            % Append H1Gpsi_m to
   the right

k = k+1;

if rem(k,10)==0
    c1 = psi_m(:,end);
    c = KJ_to_J(c1);
    writematrix(c,'forward_kj.txt');

    %-----------------------------------------------
    %Solving the DDE system
    A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare
       DDE resuts to
    time=linspace(v,v+l,length(A));
    tspan=[v,v+l]; %time for which the DDE is solved--
       in months
    sol=dde23(@delay_buy_kj,lags, x0,tspan);      %
       solve DDE
```

```
        y=interp1(sol.x,sol.y(8,:),time,'linear'); %
           linearly interpolate sol of

        if min(y)<x0(8)-bound
           break
        elseif min(y)<0
            break
        elseif max(y)>x0(8)+bound
            break
        else
            fprintf('Iteration %d\n',k);
            fprintf('G(psi_m) first term: %d\n',fm);
            writematrix(psi_m,'round1_min_2.txt');
        end
    end
end  % end of while loop
end  % end of if loop in the big if loop
end
% psi_3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Gradient Descent using Psi_3                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up psi_0, u, u', and u_psi.                 %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
psi_0 = repmat(good_psis(:,3),[1,t_size(2)]);
ux=u(t,g,s,mo(a),mo(b));
common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);
T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);
u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the L2 Gradient                    %
% See Chapter 5 Section 5.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
in=ux-u_psi_0+g-T_psi_0;
L2Gpsi_0_n=-2*T_star_data(in,x_1(t),x_2(t),x_3(t),x_4d(t),
   x_4(t),...
x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t)
   ,...
x_12(t),x_13(t),s,psi_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the H1 Gradient using Neuman       %
%       boundary condition.                 %
% See Chapter 5 Section 5.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
H1 = [];
L2size = size(L2Gpsi_0_n);
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_0_n(i,:),mo(a),mo(b),s); %See the
       function called "boundary"
    H1 = [H1;S];
end
H1Gpsi_0 = H1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up alpha.                                  %
% See Chapter 5 Section 5.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_h=max(abs(H1),[],"all");
low = max_h*1e3;
high = max_h*1e4;
alpha_options=[1e10,1e11,1e12,1e13,1e14,1e15,1e16,1e17,1
   e18,1e19,1e20,1e21,1e22,1e23,1e24,1e25,1e26,1e27,1e28,1
   e29,1e30,1e31,1e32];
for i = alpha_options
   if (i<high) && (i>low)
       alpha=1/i;
   else
   end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up u'_psi and Calculate G(psi)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
g_psi_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
   s);
fprintf('Gpsi: %d',g_psi_0);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we keep the values      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_s=[psi_0];      % All the psi_m
fm_s=[1.0e+80];             % All the G(psi_m)
u_psi_s=[u_psi_0];  % All the u_psi_m
L2G=[L2Gpsi_0_n];   % All the L2 gradient with respect to
   psi_m
H1G=[H1Gpsi_0];     % All the H1 gradient with respect to
   psi_m
T_psi_s = [T_psi_0];% All the T(psi_m)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      For loop to update psi_m          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if g_psi_0<0.0000001
```

```
    fprintf('Minimizing Psi is \n');
    disp(psi_0);
    fprintf('G_psi is %d \n',sum(trapz((T_psi_0-g).^2+(
       u_psi_prime-up).^2,2)*s));
else
T_psi_m = T_psi_0;
k=1;
while k<100
    % we need below i1 and i2 to get the correct columns
    i1 = t_size(2)*(k-1)+1;
    i2 = t_size(2)*k;

    % Step 1) Update psi_m
    psi_m=psi_s(:,[i1:i2])-(alpha)*H1G(:,[i1:i2]);
    psi_s=[psi_s,psi_m];            % Append psi_m to the
       right

    % Step 2) Get T(psi_m)
    T_psi_m=T_reg_data(psi_m,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

    % Step 3) Get u_psi_m
    u_psi_m=u_psi(T_psi_m,mo(a),mo(b),s,t);
```

```
    % Step 4) Get G(psi_m)=fm
    fm=sum(trapz((T_psi_m-g).^2,2)*s);
    fm_s=[fm_s,fm];                    % Appending fm to the
        right
    if fm_s(end-1)-fm<0
        fprintf('STOP%d\n',k)
        break
    end

    % Step 5) Get L2 gradient with respect to psi_m
    in_m = ux-u_psi_m+g-T_psi_m;
    L2Gpsi_m=-2*T_star_data(in_m,x_1(t),x_2(t),x_3(t),x_4d
       (t),x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s,psi_m);

    % Step 6) Get H1 gradient with respect to psi_m
    H1G_psi_m = [];
    parfor i = 1:L2size(1)
        S = boundary(L2Gpsi_m(i,:),mo(a),mo(b),s);
        H1G_psi_m = [H1G_psi_m;S];
    end
    H1G=[H1G,H1G_psi_m];              % Append H1Gpsi_m to
       the right

    k = k+1;
```

```
if rem(k,10)==0
    c1 = psi_m(:,end);
    c = KJ_to_J(c1);
    writematrix(c,'forward_kj.txt');

    %-----------------------------------------------
    %Solving the DDE system
    A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare
       DDE resuts to
    time=linspace(v,v+l,length(A));
    tspan=[v,v+l]; %time for which the DDE is solved--
       in months
    sol=dde23(@delay_buy_kj,lags, x0,tspan);      %
       solve DDE
    y=interp1(sol.x,sol.y(8,:),time,'linear'); %
       linearly interpolate sol of

    if min(y)<x0(8)-bound
       break
    elseif min(y)<0
        break
    elseif max(y)>x0(8)+bound
        break
    else
        fprintf('Iteration %d\n',k);
        fprintf('G(psi_m) first term: %d\n',fm);
```

```
            writematrix(psi_m,'round1_min_3.txt');
        end
    end

end  % end of while loop
end  % end of if loop in the big if loop
end


% psi_4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Gradient Descent using Psi_4                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up psi_0, u, u', and u_psi.                 %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(good_psis(:,4),[1,t_size(2)]);

ux=u(t,g,s,mo(a),mo(b));

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
```

```
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the L2 Gradient                    %
% See Chapter 5 Section 5.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
in=ux-u_psi_0+g-T_psi_0;
L2Gpsi_0_n=-2*T_star_data(in,x_1(t),x_2(t),x_3(t),x_4d(t),
   x_4(t),...
x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t)
   ,...
x_12(t),x_13(t),s,psi_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the H1 Gradient using Neuman       %
%       boundary condition.                 %
% See Chapter 5 Section 5.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H1 = [];
L2size = size(L2Gpsi_0_n);
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_0_n(i,:),mo(a),mo(b),s); %See the
       function called "boundary"
```

```
    H1 = [H1;S];
end
H1Gpsi_0 = H1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up alpha.                                  %
% See Chapter 5 Section 5.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_h=max(abs(H1),[],"all");
low = max_h*1e3;
high = max_h*1e4;
alpha_options=[1e10,1e11,1e12,1e13,1e14,1e15,1e16,1e17,1
   e18,1e19,1e20,1e21,1e22,1e23,1e24,1e25,1e26,1e27,1e28,1
   e29,1e30,1e31,1e32];
for i = alpha_options
   if (i<high) && (i>low)
       alpha=1/i;
   else
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up u'_psi and Calculate G(psi)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
g_psi_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
   s);
```

```
fprintf('Gpsi: %d',g_psi_0);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is where we keep the values      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_s=[psi_0];      % All the psi_m
fm_s=[1.0e+80];             % All the G(psi_m)
u_psi_s=[u_psi_0];  % All the u_psi_m
L2G=[L2Gpsi_0_n];   % All the L2 gradient with respect to
   psi_m
H1G=[H1Gpsi_0];     % All the H1 gradient with respect to
   psi_m
T_psi_s = [T_psi_0];% All the T(psi_m)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       For loop to update psi_m          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if g_psi_0<0.0000001
    fprintf('Minimizing Psi is \n');
    disp(psi_0);
    fprintf('G_psi is %d \n',sum(trapz((T_psi_0-g).^2+(
       u_psi_prime-up).^2,2)*s));
else
```

```
T_psi_m = T_psi_0;
k=1;
while k<100
    % we need below i1 and i2 to get the correct columns
    i1 = t_size(2)*(k-1)+1;
    i2 = t_size(2)*k;

    % Step 1) Update psi_m
    psi_m=psi_s(:,[i1:i2])-(alpha)*H1G(:,[i1:i2]);
    psi_s=[psi_s,psi_m];            % Append psi_m to the
       right

    % Step 2) Get T(psi_m)
    T_psi_m=T_reg_data(psi_m,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

    % Step 3) Get u_psi_m
    u_psi_m=u_psi(T_psi_m,mo(a),mo(b),s,t);

    % Step 4) Get G(psi_m)=fm
    fm=sum(trapz((T_psi_m-g).^2,2)*s);
    fm_s=[fm_s,fm];                 % Appending fm to the
        right
    if fm_s(end-1)-fm<0
```

```
        fprintf('STOP%d\n',k)
        break
    end

    % Step 5) Get L2 gradient with respect to psi_m
    in_m = ux-u_psi_m+g-T_psi_m;
    L2Gpsi_m=-2*T_star_data(in_m,x_1(t),x_2(t),x_3(t),x_4d
       (t),x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s,psi_m);

    % Step 6) Get H1 gradient with respect to psi_m
    H1G_psi_m = [];
    parfor i = 1:L2size(1)
        S = boundary(L2Gpsi_m(i,:),mo(a),mo(b),s);
        H1G_psi_m = [H1G_psi_m;S];
    end
    H1G=[H1G,H1G_psi_m];            % Append H1Gpsi_m to
       the right

    k = k+1;

    if rem(k,10)==0
        c1 = psi_m(:,end);
        c = KJ_to_J(c1);
        writematrix(c,'forward_kj.txt');
```

```
        %-----------------------------------------------
        %Solving the DDE system
        A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare
           DDE resuts to
        time=linspace(v,v+l,length(A));
        tspan=[v,v+l]; %time for which the DDE is solved--
           in months
        sol=dde23(@delay_buy_kj,lags, x0,tspan);      %
           solve DDE
        y=interp1(sol.x,sol.y(8,:),time,'linear'); %
           linearly interpolate sol of

        if min(y)<x0(8)-bound
           break
        elseif min(y)<0
            break
        elseif max(y)>x0(8)+bound
            break
        else
            fprintf('Iteration %d\n',k);
            fprintf('G(psi_m) first term: %d\n',fm);
            writematrix(psi_m,'round1_min_4.txt');
        end
    end

end  % end of while loop
```

```
end  % end of if loop in the big if loop
end


% psi_5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Gradient Descent using Psi_5                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up psi_0, u, u', and u_psi.                 %
% See Chapter 5 Section 5.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_0 = repmat(good_psis(:,5),[1,t_size(2)]);

ux=u(t,g,s,mo(a),mo(b));

common = s*cumtrapz(g,2);
up=-1*common+(1/(mo(b)-mo(a)))*s*trapz(common, 2);

T_psi_0=T_reg_data(psi_0,x_1(t),x_2(t),x_3(t),x_4d(t),x_4(
   t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

u_psi_0 = u_psi(T_psi_0,mo(a),mo(b),s,t);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the L2 Gradient                %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
in=ux-u_psi_0+g-T_psi_0;
L2Gpsi_0_n=-2*T_star_data(in,x_1(t),x_2(t),x_3(t),x_4d(t),
   x_4(t),...
x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t)
   ,...
x_12(t),x_13(t),s,psi_0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up the H1 Gradient using Neuman   %
%       boundary condition.             %
% See Chapter 5 Section 5.              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H1 = [];
L2size = size(L2Gpsi_0_n);
parfor i = 1:L2size(1)
    S = boundary(L2Gpsi_0_n(i,:),mo(a),mo(b),s); %See the
       function called "boundary"
    H1 = [H1;S];
end
H1Gpsi_0 = H1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Set up alpha.                          %
% See Chapter 5 Section 5.               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
max_h=max(abs(H1),[],"all");
low = max_h*1e3;
high = max_h*1e4;
alpha_options=[1e10,1e11,1e12,1e13,1e14,1e15,1e16,1e17,1
   e18,1e19,1e20,1e21,1e22,1e23,1e24,1e25,1e26,1e27,1e28,1
   e29,1e30,1e31,1e32];
for i = alpha_options
   if (i<high) && (i>low)
       alpha=1/i;
   else
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up u'_psi and Calculate G(psi)    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
u_psi_prime = u_psi_p(T_psi_0,mo(a),mo(b),s);
g_psi_0 = sum(trapz((T_psi_0-g).^2+(u_psi_prime-up).^2,2)*
   s);
fprintf('Gpsi: %d',g_psi_0);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% This is where we keep the values      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
psi_s=[psi_0];      % All the psi_m
fm_s=[1.0e+80];             % All the G(psi_m)
u_psi_s=[u_psi_0];  % All the u_psi_m
L2G=[L2Gpsi_0_n];   % All the L2 gradient with respect to
   psi_m
H1G=[H1Gpsi_0];     % All the H1 gradient with respect to
   psi_m
T_psi_s = [T_psi_0];% All the T(psi_m)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               For loop to update psi_m                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if g_psi_0<0.0000001
    fprintf('Minimizing Psi is \n');
    disp(psi_0);
    fprintf('G_psi is %d \n',sum(trapz((T_psi_0-g).^2+(
       u_psi_prime-up).^2,2)*s));
else
T_psi_m = T_psi_0;
k=1;
while k<100
    % we need below i1 and i2 to get the correct columns
    i1 = t_size(2)*(k-1)+1;
```

```
    i2 = t_size(2)*k;

    % Step 1) Update psi_m
    psi_m=psi_s(:,[i1:i2])-(alpha)*H1G(:,[i1:i2]);
    psi_s=[psi_s,psi_m];              % Append psi_m to the
       right

    % Step 2) Get T(psi_m)
    T_psi_m=T_reg_data(psi_m,x_1(t),x_2(t),x_3(t),x_4d(t),
       x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s);

    % Step 3) Get u_psi_m
    u_psi_m=u_psi(T_psi_m,mo(a),mo(b),s,t);

    % Step 4) Get G(psi_m)=fm
    fm=sum(trapz((T_psi_m-g).^2,2)*s);
    fm_s=[fm_s,fm];                    % Appending fm to the
       right
    if fm_s(end-1)-fm<0
        fprintf('STOP%d\n',k)
        break
    end

    % Step 5) Get L2 gradient with respect to psi_m
```

```
    in_m = ux-u_psi_m+g-T_psi_m;
    L2Gpsi_m=-2*T_star_data(in_m,x_1(t),x_2(t),x_3(t),x_4d
       (t),x_4(t)...
,x_5(t),x_6(t),x_7d(t),x_7(t),x_8(t),x_9(t),x_10(t),x_11(t
   ),...
x_12(t),x_13(t),s,psi_m);

    % Step 6) Get H1 gradient with respect to psi_m
    H1G_psi_m = [];
    parfor i = 1:L2size(1)
        S = boundary(L2Gpsi_m(i,:),mo(a),mo(b),s);
        H1G_psi_m = [H1G_psi_m;S];
    end
    H1G=[H1G,H1G_psi_m];            % Append H1Gpsi_m to
       the right

    k = k+1;

    if rem(k,10)==0
        c1 = psi_m(:,end);
        c = KJ_to_J(c1);
        writematrix(c,'forward_kj.txt');

        %-----------------------------------------------
        %Solving the DDE system
        A=AAPL(mo(v):mo(v+1)); %Defining 'A' to compare
           DDE resuts to
```

```
        time=linspace(v,v+l,length(A));
        tspan=[v,v+l]; %time for which the DDE is solved--
           in months
        sol=dde23(@delay_buy_kj,lags, x0,tspan);      %
           solve DDE
        y=interp1(sol.x,sol.y(8,:),time,'linear'); %
           linearly interpolate sol of

        if min(y)<x0(8)-bound
           break
        elseif min(y)<0
            break
        elseif max(y)>x0(8)+bound
            break
        else
            fprintf('Iteration %d\n',k);
            fprintf('G(psi_m) first term: %d\n',fm);
            writematrix(psi_m,'round1_min_5.txt');
        end
    end

end  % end of while loop
end  % end of if loop in the big if loop
end
```

Appendix D: Apply_KJ_C_to_predict_good_psi.m

Apply_KJ_C_to_predict_good_psi.m

```
clc
clear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code uses the minimized psis to calculate the                    %
% solution to DDE. It averages the solutions and find                   %
% the psi that gives the solution closest to the                        %
% average psi.                                                          %
% Input: v, l, lags, minimized psis.                                    %
% Output: will provide us the forecast of stock trend.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        Set v and lags                                 %
% Each one of these should be equal to the values from                  %
%   "Get_Constants_Final.m".                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=339;
lags=[12,1,1,20,1,12,17,1,1,1,1,1,1];
l=2;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Importing the minimized psis                       %
% Comment or uncomment depending on how many psis are                    %
% minimized.                                                             %
% For example, if one minimized three psis, uncomment                    %
% m1, m2, and m3 and leav m4 and m5 commented.                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m1 = readmatrix('round1_min_1.txt');
% m2 = readmatrix('round1_min_2.txt');
% m3 = readmatrix('round1_min_3.txt');
% m4 = readmatrix('round1_min_4.txt');
% m5 = readmatrix('round1_min_5.txt');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Getting the value the the last time step                     %
% Comment or uncomment depending on how many psis are                    %
% minimized.                                                             %
% For example, if one minimized three psis, uncomment                    %
% the one with m1, m2, and m3 and leave the others                       %
%commented.                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c0=[m1(:,end)];
% c0=[m1(:,end),m2(:,end)];
% c0=[m1(:,end),m2(:,end),m3(:,end)];
% c0=[m1(:,end),m2(:,end),m3(:,end),m4(:,end)];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Preprocess the data                              %
% This portion is a modified portion of Barnett's data                  %
% preprocessing process. If a user decides to extend the%
% data, 'mo' must be changed accordingly.                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];
```

```
%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;1156;
    1187;1217;1248;1278;1309;1340;1370;1401;1431;1462;
    1493;1522;1553;1583;1614;1644;1675;1706;1736;1767;
    1797;1828;1859;1887;1918;1948;1979;2009;2040;2071;
    2101;2132;2162;2193;2224;2252;2283;2313;2344;2374;
    2405;2436;2466;2497;2527;2558;2589;2617;2648;2678;
    2709;2739;2770;2801;2831;2862;2892;2923;2954;2983;
    3014;3044;3075;3105;3136;3167;3197;3228;3258;3289;
    3320;3348;3379;3409;3440;3470;3501;3532;3562;3593;
    3623;3654;3685;3713;3744;3774;3805;3835;3866;3897;
    3927;3958;3988;4019;4050;4078;4109;4139;4170;4200;
    4231;4262;4292;4323;4353;4384;4415;4444;4475;4505;
    4536;4566;4597;4628;4658;4689;4719;4750;4781;4809;
    4840;4870;4901;4931;4962;4993;5023;5054;5084;5115;
    5146;5174;5205;5235;5266;5296;5327;5358;5388;5419;
    5449;5480;5511;5539;5570];

%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
```

```
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;



% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector
```

```
%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);   % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
%Interpolated Apple stock price
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price and inearly interpolated from
   there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
vnz3 = find(SandP500 ~= 0);
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
```

```
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
ir_nz = ir1(ir1 ~= 0);
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
rd = interp1(vnz8, rd_nz, iv, 'linear')';

%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
```

```
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
vnz12 = find(iphone1 ~= 0);
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting on December 1,2007, so now we
   account for that and start all vectors on January 1,
   2008
```

```
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-----------------------------------------------------------
%Complete the same process with the older data files
%-----------------------------------------------------------
%Apple's Net Income
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
```

```
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
 %Apple didn't have anybuybacks prior to 2008 so this
    variable is 0 pple's EPS
buyback_old=zeros(mo1(end),1);

eps_old=csvread('old_eps.csv',1,1);

%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);               %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);  %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0);%Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
```

```
%Interpolated APPL function is above


%The same process is repeated for each variable

%Apple's Net Income
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
cpi_onz = cpi_old(cpi_old ~= 0);
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
```

```
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';

%GLD
gld_onz = gld_old(gld_old ~= 0);
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);

%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';

%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';

%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';

%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
```

```
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);

%Confidence Index
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';

%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special cases) instead of January 1,2018,
   which is where the new data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
```

```
%caculate P/E ratio using EPS and stock price
pe_old=AAPL_old./eps(1:mo1(end)-1);

%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];

mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Define the hisotry vector for dde23          %
% See Chapter 2 Section 2 for details.                    %
% We use constant functions as our history vector.        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
x0(1)=cpi(mo(v));
x0(2)=inflation(mo(v));
x0(3)=pe(mo(v));
x0(4)=ir(mo(v));
x0(5)=NASDAQ100(mo(v));
x0(6)=rev(mo(v));
x0(7)=rd(mo(v));
x0(8)=AAPL(mo(v));
x0(9)=gld(mo(v));
x0(10)=sp500(mo(v));
x0(11)=iphone(mo(v));
x0(12)=conf(mo(v));
x0(13)=buyback(mo(v));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Calculate the solutions and save                %
% ys: This will hold all the solutions.                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s_c = size(c0);
ys = [];
for i = 1:s_c(2)
    c1 = c0(:,i);
    c = KJ_to_J(c1);
    writematrix(c,'forward_kj.txt');

    %-----------------------------------------------------
    %Solving the DDE system
```

```
    A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare DDE
       resuts to
    time=linspace(v,v+l,length(A));
    tspan=[v,v+l]; %time for which the DDE is solved--in
       months
    sol=dde23(@delay_buy_kj,lags, x0,tspan);      %solve
       DDE
    y=interp1(sol.x,sol.y(8,:),time,'linear'); %linearly
       interpolate sol of stock price down to days
    ys = [ys,y']; %vertical
    der=interp1(sol.x,sol.yp(8,:),time, 'linear'); %
       lineaerly interpolate derivative down to days
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Calculate average of the solutions                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
avg = mean(ys,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Calculate the mape between average and the solutions %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close = [];
for i = 1:s_c(2)
    close = [close,mape(avg,ys(:,i))];
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find the psi which gives the solution closest to the  %
% average.                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[number, index] = mink(close,1);

close_avg = c0(:,index);

fprintf('Psi closest to average = %d\n',index);

writematrix(ys(:,index),'cloesest_to_avg_psi.txt');


% Compare DDE solution to actual stock price
figure
hold on
t1=datetime(1994,7,1); %Start Date of Data
t2=datetime(2023,3,15); %End Date of Data
t3=t1:t2;               %Create a vector of days between t1
    and t2
% plot(t3(mo(v):mo(v+1)),ys(:,index),'r');          %plot
   DDE solution
plot(t3(mo(v):mo(v+1)),ys(:,index),'r');          %plot DDE
    solution
```

```
plot(t3(mo(v):mo(v+l)),AAPL(mo(v):mo(v+l)),'b')%plots
   actual stock price

lgd=legend('Predicted Apple Trend','Actual Apple Stock
   Price');
lgd.FontSize = 12;          %legend size
lgd.Location='northwest';  %legend location
%set(gca,'Xtick',291:295,'XTickLabel
   ',{'9/1/18','10/1/18','11/1/18','12/1/18','1/1/19'})

hold off

c1 = c0(:,index);
c = KJ_to_J(c1);
writematrix(c,'forward_kj.txt');

A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare DDE resuts
    to
time=linspace(v,v+l,length(A));
tspan=[v,v+l]; %time for which the DDE is solved--in
   months
sol=dde23(@delay_buy_kj,lags, x0,tspan);      %solve DDE
y=interp1(sol.x,sol.y(8,:),time,'linear');

mapes=mape(y',A); %mean absolute percentage error
fprintf('MAPE %d', mapes);
```

Appendix E: Apple_Stock_Pred_Guess.m

Apple_Stock_Pred_Guess.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This code uses the minimized psis to calculate the                    %
% solution to DDE. It averages the solutions and find                   %
% the psi that gives the solution closest to the                        %
% average psi.                                                          %
% Input: v, l, lags, max_diff, bound, and                               %
%the chosen psi from "Apply_KJ_C_to_predict_good_psi.m"                 %
% Output: will provide us the forecast of stock price.                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v=243;
l=2;
lags=[12,1,1,20,1,12,17,1,1,1,1,1,1];
max_diff = get_max_diff(v);
bound = max_diff*2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Preprocess the data                                %
% This portion is a modified portion of Barnett's data                  %
% preprocessing process. If a user decides to extend the%
% data, 'mo' must be changed accordingly.                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for h=1
```

```
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];

%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;1156;
    1187;1217;1248;1278;1309;1340;1370;1401;1431;1462;
    1493;1522;1553;1583;1614;1644;1675;1706;1736;1767;
    1797;1828;1859;1887;1918;1948;1979;2009;2040;2071;
```

```
    2101;2132;2162;2193;2224;2252;2283;2313;2344;2374;
    2405;2436;2466;2497;2527;2558;2589;2617;2648;2678;
    2709;2739;2770;2801;2831;2862;2892;2923;2954;2983;
    3014;3044;3075;3105;3136;3167;3197;3228;3258;3289;
    3320;3348;3379;3409;3440;3470;3501;3532;3562;3593;
    3623;3654;3685;3713;3744;3774;3805;3835;3866;3897;
    3927;3958;3988;4019;4050;4078;4109;4139;4170;4200;
    4231;4262;4292;4323;4353;4384;4415;4444;4475;4505;
    4536;4566;4597;4628;4658;4689;4719;4750;4781;4809;
    4840;4870;4901;4931;4962;4993;5023;5054;5084;5115;
    5146;5174;5205;5235;5266;5296;5327;5358;5388;5419;
    5449;5480;5511;5539;5570];

%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
```

```
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;



% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector

%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);   % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
%Interpolated Apple stock price
```

```
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price and inearly interpolated from
   there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
vnz3 = find(SandP500 ~= 0);
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
ir_nz = ir1(ir1 ~= 0);
```

```
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
rd = interp1(vnz8, rd_nz, iv, 'linear')';

%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
vnz12 = find(iphone1 ~= 0);
```

```
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting on December 1,2007, so now we
   account for that and start all vectors on January 1,
   2008
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
```

```
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-------------------------------------------------------
%Complete the same process with the older data files
%-------------------------------------------------------
%Apple's Net Income
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
```

```
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
 %Apple didn't have anybuybacks prior to 2008 so this
    variable is 0 pple's EPS
buyback_old=zeros(mo1(end),1);

eps_old=csvread('old_eps.csv',1,1);

%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);               %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);  %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0);%Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
%Interpolated APPL function is above


%The same process is repeated for each variable

%Apple's Net Income
```

```
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
cpi_onz = cpi_old(cpi_old ~= 0);
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';

%GLD
gld_onz = gld_old(gld_old ~= 0);
```

```
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);

%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';

%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';

%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';

%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);

%Confidence Index
```

```
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';

%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special cases) instead of January 1,2018,
   which is where the new data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
%caculate P/E ratio using EPS and stock price
pe_old=AAPL_old./eps(1:mo1(end)-1);

%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
```

```
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];

mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Input the chosen psi here                              %
% For example, if "Psi closest to average = 3", then              %
%        c0 = readmatrix('round1_min_3.txt');                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c0 = readmatrix('round1_min_1.txt');
c1 = c0(:,end);
c = KJ_to_J(c1);
writematrix(c,'forward_kj.txt');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Define the hisotry vector for dde23                          %
% See Chapter 2 Section 2 for details.                                   %
% We use constant functions as our history vector.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0(1)=cpi(mo(v));
x0(2)=inflation(mo(v));
x0(3)=pe(mo(v));
x0(4)=ir(mo(v));
x0(5)=NASDAQ100(mo(v));
x0(6)=rev(mo(v));
x0(7)=rd(mo(v));
x0(8)=AAPL(mo(v));
x0(9)=gld(mo(v));
x0(10)=sp500(mo(v));
x0(11)=iphone(mo(v));
x0(12)=conf(mo(v));
x0(13)=buyback(mo(v));

%-----------------------------------------------------------
%Solving the DDE system

A=AAPL(mo(v):mo(v+l)); %Defining 'A' to compare DDE resuts
    to
time=linspace(v,v+l,length(A));
tspan=[v,v+l]; %time for which the DDE is solved--in
   months
```

```
sol=dde23(@delay_buy_kj,lags, x0,tspan);      %solve DDE
y=interp1(sol.x,sol.y(8,:),time,'linear'); %linearly
   interpolate sol of stock price down to days
der=interp1(sol.x,sol.yp(8,:),time, 'linear'); %lineaerly
   interpolate derivative down to days


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Get the stock price forecast                %
% We repeat this process 1000 times to take away user   %
% control of choosing the best outcome. Please read     %
% Chapter 5 and 6.                                      %
% num: This sets the number of times we want to solve   %
% the GBM.                                              %
% Browns: This keeps the price forecasts.               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Browns = [];
count = 0;
num = 1000;
while count<num
    %-------------------------solving for GBM------------
    %Local Volatility results imported here
    % sigma=csvread('sigma6_29-7_6.csv',0,1);
    sigma=csvread('sigma8_3_9_21.csv',0,1);
%     sigma=csvread('sigma_sept_oct.csv',0,1);
    % sigma1=csvread('sigma9_28-11_2.csv',0,1);
    % sigma=csvread('sigma11_9_12_28.csv',0,1);
```

```
% sigma=[sigma1;sigma];
% sigma2 = [sigma;sigma1];
% sigma = csvread('01-13-01-20_vol.csv',0,1);

mu=ts2func(der./y,'times',time); %drift term as
   function of time;
GM=gbm(mu,sigma,'StartState',AAPL(mo(v)),'StartTime',v
   );
dt=1/(length(y)-1);     % time increment
nperiods=length(y)-1;  % # of simulated observations
trials=100;            % # of Simulated trials

[X,T]=GM.simBySolution(nperiods,'DeltaTime',dt,'
   ntrials',trials);
for i=1:trials
    Brown(i,:)=X(:,i)'; %set random vector solutions
       as rows of Brown matrix
end

Brown1=1/trials.*sum(Brown); %calculate average; sum
   the rows of Brown

if min(Brown1)<x0(8)-bound
elseif min(Brown1)<0
elseif max(Brown1)>x0(8)+bound
else
    Browns = [Browns;Brown1];
```

```
        count = count+1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Get the average stock price forecast                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
avg_brown = mean(Browns);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate the mape between average and the predictions%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close = [];
for i = 1:num
    close = [close,mape(avg_brown,Browns(i,:))];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Find the prediction closest to the average                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[C,D] = mink(close,1);
fprintf('Actual Stock vs. Trend: %d\n',mape(y',A));
fprintf('Actual Stock vs. Predicted Price: %d\n', mape(
   Browns(D,:)',A));

figure
hold on
plot(T(1:end-1),y(1:end-1),'r');          %plot DDE
   solution
```

```
plot(T(1:end-1),Browns(D,1:end-1),'b'); % T(1:61),Brown1
   (1:61)
plot(T(1:end-1),AAPL(mo(v):mo(v+l)-1),'black')%plots
   actual stock price
lgd=legend('Predicted Apple Stock Price','Predicted Apple
   Trend','Real Stock');
lgd.FontSize = 12;        %makes legend larger
lgd.Location='northwest'; %location of legend
set(gca,'Xtick',290:295,'XTickLabel',{'8/1/18','9/1/18','
   10/1/18','11/1/18','12/1/18','1/1/19'})
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Saves the outputs                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
writematrix(c0,'round1_psi.txt');
writematrix(sol.x,'round1_sol_x.txt');
writematrix(sol.y,'round1_sol_y.txt');
writematrix(sol.yp,'round1_sol_yp.txt');
writematrix(X,'round1_X.txt');
writematrix(T,'round1_T.txt');
writematrix(Browns(D,:),'round1_Brown.txt');
writematrix(AAPL(mo(v):mo(v+l)-1),'round1_plot_2_2.txt');
writematrix(time,'round1_time.txt');
writematrix(A,'round1_A.txt');
writematrix(tspan,'round1_tspan.txt');
writematrix(y,'round1_y.txt');
```

Appendix F: Other Functions

Other Functions

boundary.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Boundary Value Problem                              %
% This function solves the boundary value problem                        %
% assuming the Neumann boundary conditions.                              %
%            y1' = y2                                                    %
%            y2' = y1-L                                                  %
% It is solving the system of equations above.                           %
% L2G : L2 gradient of G with respect to psi                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function now = boundary(L2G,a,b,s)
Tamb = transpose(L2G);
c = round(1+(b-a)/s);
% time dependent window
tspan = linspace(a,b,c);              % calculate intervals
solinit = bvpinit(tspan,[0 0]);      % initial conditions
% define your ode-function
odefunc = @bvpfcn;
sol = bvp4c(odefunc, @bcfcn, solinit);
ysize = size(sol.y);
xsize = size(sol.x);
```

```
now2 = zeros(1,c);
k=sol.y(1,:);
% Notice that sometimes, bvp4c creates more solutions to
   better estimate
% Here, we are only taking the estimates close to the time
    steps we gave.
if xsize(2)== c
    now = sol.y(1,:);
else
    parfor i=1:c
        [M,I]=min(abs(sol.x-tspan(i)));
        now2(1,i) = k(I);
    end
    now = now2;
end

function dTempdt = bvpfcn(ts,y)
    Tamb_t = interp1(tspan,Tamb,ts);
    y1 = y(1);
    y2 = y(2);
    dTempdt = zeros(2,1);
    % define the set of equations
    dTempdt(1) = y2;                % y1'=y2
    dTempdt(2) = y1-Tamb_t;         % y2'=y1-L
end
    function res = bcfcn(ya,yb)
```

```
        res = [ya(2);yb(2)];  % Neumann Boundary condition
            = 0
    end
end
```

****

delay_buy_kj.m

****

```
function dxdt = delay_buy_kj(t,x,Z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         This function is used in dde23                  %
% The funciton below is the funciton dde23 will solve.  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Import coefficients found in main code
filename = 'forward_kj.txt';
[c]=importdata(filename);

%Delays
cpi_delay=Z(:,1);
inf_delay=Z(:,2);
pe_delay=Z(:,3);
ir_delay=Z(:,4);   %
NASDAQ_delay=Z(:,5);
rev_delay=Z(:,6);
rd_delay=Z(:,7);   %
apple_delay=Z(:,8);
gld_delay=Z(:,9);
```

```
djia_delay=Z(:,10);
iphone_delay=Z(:,11);
conf_delay=Z(:,12);
buyback_delay=Z(:,13);

dxdt = zeros(13,1);

dxdt(1)  = c(1)*x(2)*x(1); %CPI
dxdt(2)  = c(2)*x(1)+c(3)*ir_delay(4)*x(2)+c(4)*x(2);  %
   inflation
dxdt(3)  = c(5)*x(2)+c(6)*x(6)+c(7)*x(8)+c(49)*x(13); %P/E
    ratio
dxdt(4)  = c(8)*dxdt(1)+c(9)*x(2)*ir_delay(4)+c(10)*x(4)+c
   (11)*x(4)*x(8); %Fed funds rate
dxdt(5)  = c(12)*x(8)+c(13)*x(10)+c(14)*x(4)+c(15)*x(12);%
   Nasdaq-100
dxdt(6)  = c(16)*x(6)+c(17)*rd_delay(7)+c(18)*x(12)+c(19)*
   x(11); %Apple's Net Income
dxdt(7)  = c(20)*x(7)+c(21)*x(6)+c(22)*x(11);  %R&D
dxdt(8)  = c(23)*x(6)+c(24)*dxdt(5)+c(25)*x(3)+c(26)*x(4)*
   x(8)+c(27)*x(8)+c(47)*dxdt(10)+c(48)*dxdt(13); %Apple
dxdt(9)  = c(28)*x(10)+c(29)*x(12)+c(30)*x(2)+c(31)*x(9);
   %GLD
dxdt(10) = c(32)*x(4)+c(33)*x(1)+c(34)*x(5)+c(35)*x(12)+c
   (36)*x(10); %S&P 500
dxdt(11) = c(37)*x(11)+c(38)*rd_delay(7)+c(39)*ir_delay(4)
   *x(11); %iPhone Revenue
```

```
dxdt(12) = c(40)*dxdt(10)+c(41)*x(4)+c(42)*x(1)+c(43)*x(9)
   ;  %CCI
dxdt(13) = c(44)*x(10)+c(45)*x(6)+c(46)*x(7); %Buybacks

end
```

****

get_max_diff.m

****

```
function get_max_diff = get_max_diff(last_v)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Preprocess the data                   %
% This portion is a modified portion of Barnett's data  %
% preprocessing process. If a user decides to extend the%
% data, 'mo' must be changed accordingly.               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%location of the first day of every month in old data
   value file
mo1=[1;32;63;93;124;154;185;216;244;275;305;337;366;397;
    428;458;489;519;550;581;610;641;671;702;732;763;794;
    824;855;885;916;947;975;1006;1036;1067;1097;1128;
    1159;1189;1220;1250;1281;1312;1340;1371;1401;1432;
    1462;1493;1524;1554;1585;1615;1646;1677;1705;1736;
    1766;1797;1827;1858;1889;1919;1950;1980;2011;2043;
    2071;2102;2132;2163;2193;2224;2255;2285;2316;2346;
    2377;2408;2436;2467;2497;2528;2558;2589;2620;2650;
    2681;2711;2742;2773;2801;2832;2862;2893;2923;2954;
```

```
    2985;3015;3046;3076;3107;3138;3166;3197;3227;3258;
    3288;3319;3350;3380;3411;3441;3472;3503;3532;3563;
    3593;3624;3654;3685;3716;3746;3777;3807;3838;3869;
    3897;3928;3958;3989;4019;4050;4081;4111;4142;4172;
    4203;4234;4262;4293;4323;4354;4384;4415;4446;4476;
    4507;4537;4568;4599;4627;4658;4688;4719;4749;4780;
    4811;4841;4872;4902;4933];

%location of the first day of every month in new data
   value file
mo=[1;32;61;92;122;153;183;214;245;275;306;336;367;398;
    426;457;487;518;548;579;610;640;671;701;732;763;791;
    822;852;883;913;944;975;1005;1036;1066;1097;1128;1156;
    1187;1217;1248;1278;1309;1340;1370;1401;1431;1462;
    1493;1522;1553;1583;1614;1644;1675;1706;1736;1767;
    1797;1828;1859;1887;1918;1948;1979;2009;2040;2071;
    2101;2132;2162;2193;2224;2252;2283;2313;2344;2374;
    2405;2436;2466;2497;2527;2558;2589;2617;2648;2678;
    2709;2739;2770;2801;2831;2862;2892;2923;2954;2983;
    3014;3044;3075;3105;3136;3167;3197;3228;3258;3289;
    3320;3348;3379;3409;3440;3470;3501;3532;3562;3593;
    3623;3654;3685;3713;3744;3774;3805;3835;3866;3897;
    3927;3958;3988;4019;4050;4078;4109;4139;4170;4200;
    4231;4262;4292;4323;4353;4384;4415;4444;4475;4505;
    4536;4566;4597;4628;4658;4689;4719;4750;4781;4809;
    4840;4870;4901;4931;4962;4993;5023;5054;5084;5115;
    5146;5174;5205;5235;5266;5296;5327;5358;5388;5419;
```

```
    5449;5480;5511;5539;5570];

%-------------------------------------------------------
%Importing data csv files
%Apple's net income
rev1=csvread('net_income.csv',1,1).*10^9;
%S&P 500
SandP500=csvread('GSPC.csv',1,1);
%Consumer Price Index
cpi1=csvread('CPI2.csv',1,1);
%inflation rate as decimal
inflation1=csvread('inf.csv',1,1).*0.01;
%Fed funds rate as as decimal
ir1=csvread('EFFR.csv',1,1).*0.01;
%SPDR Gold Trust
gld1=csvread('GLD.csv',1,1);
%Apple's R&D Spending
rd1=csvread('RandD.csv',1,1).*10^9;
%P/E ratio
pe1=csvread('pe_apple.csv',366,1);
%NASDAQ 100 index
NASDAQ=csvread('NASDAQ100.csv',1,1);
%Apple's Stock Price
AAPL1=csvread('AAPL3.csv',1,1);
%iphone revenue
iphone1=csvread('iphone_rev.csv',1,1).*10^9 ;
%Consumer Confidence Index
```

```
conf1=csvread('cci_oecd.csv',1,1);
%buybacks
buy1=csvread('buyback.csv',1,1).*10^9;


% Linearly Interpolate the Data Down to the Day
%-------------------------------------------------------
iv = 1:mo(end);                 %Interpolation Vector

%Apple Stock Price
AAPL_nz = AAPL1(AAPL1 ~= 0);     % Find Non-Zero Elements
vnz1 = find(AAPL1 ~= 0);   % Indices Of Non-Zero Elements
AAPL = interp1(vnz1, AAPL_nz, [iv,mo(end)+1], 'linear')';
%Interpolated Apple stock price
%stock prices weren't reported on Jan 1,2008, so we
   included 12/31/07 price and inearly interpolated from
   there.

%Apple's net income
rev_nz = rev1(rev1 ~= 0);
vnz2 = find(rev1 ~= 0);
rev = interp1(vnz2, rev_nz, iv, 'linear')';

%S&P 500
sp500_nz = SandP500(SandP500 ~= 0);
vnz3 = find(SandP500 ~= 0);
```

```
sp500 = interp1(vnz3, sp500_nz, [iv,mo(end)+1], 'linear')
   ';

%Consumer Price Index
cpi_nz = cpi1(cpi1 ~= 0);
vnz4 = find(cpi1 ~= 0);
cpi = interp1(vnz4, cpi_nz, iv, 'linear')';

%Inflation rate
inf_nz = inflation1(inflation1 ~= 0);
vnz5 = find(inflation1 ~= 0);
inflation = interp1(vnz5, inf_nz, iv, 'linear')';

%Fed funds rate
ir_nz = ir1(ir1 ~= 0);
vnz6 = find(ir1 ~= 0);
ir = interp1(vnz6, ir_nz, iv, 'linear')';

%GLD
gld_nz = gld1(gld1 ~= 0);
vnz7 = find(gld1 ~= 0);
gld = interp1(vnz7, gld_nz, [iv, mo(end)+1], 'linear')';

%Apple's R&D Spending
rd_nz = rd1(rd1 ~= 0);
vnz8 = find(rd1 ~= 0);
rd = interp1(vnz8, rd_nz, iv, 'linear')';
```

```
%P/E Ratio
pe_nz = pe1(pe1 ~= 0);
vnz9 = find(pe1 ~= 0);
pe = interp1(vnz9, pe_nz, iv, 'linear')';

%NASDAQ 100
NASDAQ100_nz = NASDAQ(NASDAQ ~= 0);
vnz10 = find(NASDAQ ~= 0);
NASDAQ100 = interp1(vnz10, NASDAQ100_nz, [iv,mo(end)+1], '
   linear')';

%iPhone Revenue
iphone_nz = iphone1(iphone1 ~= 0);
vnz12 = find(iphone1 ~= 0);
iphone = interp1(vnz12, iphone_nz, iv, 'linear')';

%Confidence Index
conf_nz = conf1(conf1 ~= 0);
vnz13 = find(conf1 ~= 0);
conf = interp1(vnz13, conf_nz, iv, 'linear')';

%Buybacks
buy1(1:1736)=zeros(1,1736);
buy_nz = buy1(buy1 ~= 0);
vnz17 = find(buy1 ~= 0);
buyback = interp1(vnz17, buy_nz, iv, 'linear')';
```

```
buyback(1736:1829)=linspace(0,buyback(1828),94);
buyback(1:1736)=zeros(1736,1); %Set buybacks before
   October 2012 equal to 0 without this MATLAB uses NaN


%Make all data vectors the same length--some data was
   interpolated starting on December 1,2007, so now we
   account for that and start all vectors on January 1,
   2008
rev=rev(1:mo(end));
rd=rd(1:mo(end));
iphone=iphone(1:mo(end));
conf=conf(1:mo(end));
NASDAQ100=NASDAQ100(2:mo(end)+1);
AAPL=AAPL(2:mo(end)+1);
pe=pe(1:mo(end));
gld=gld(1:mo(end)+1);
ir=ir(1:mo(end));
inflation=inflation(1:mo(end));
cpi=cpi(1:mo(end));
sp500=sp500(2:mo(end)+1);
buyback=buyback(1:mo(end));

%-------------------------------------------------------
%Complete the same process with the older data files
%-------------------------------------------------------
%Apple's Net Income
```

```
rev_old=csvread('old_netincome.csv',1,1).*10^6;
%S&P 500
sp500_old=csvread('old_SP500.csv',1,1);
%Consumer Price Index
cpi_old=csvread('old_cpi.csv',1,1);
%inflation as decimal
inf_old=csvread('old_inf.csv',1,1).*0.01;
%Fed funds as decimal
ir_old=csvread('old_ir.csv',1,1).*0.01;
%GLD
gld_old=csvread('old_gld.csv',1,1);
%Apple's R&D Spending
rd_old=csvread('old_rd.csv',1,1).*10^6;
%NASDAQ 100 Index
nas_old=csvread('old_nasdaq.csv',1,1);
%Apple's Stock Price
AAPL_old=csvread('old_apple.csv',1,1);
%iPhone Revenue
iphone_old=csvread('old_iphone.csv',1,1).*10^6 ;
%CCI
conf_old=csvread('old_cci.csv',1,1);
 %Apple didn't have anybuybacks prior to 2008 so this
    variable is 0 pple's EPS
buyback_old=zeros(mo1(end),1);

eps_old=csvread('old_eps.csv',1,1);
```

```
%------- Linearly Interpolate the Old Data---------------

%Apple's Stock Price
iv1 = 1:mo1(end);                 %Interpolation Vector
AAPL_onz = AAPL_old(AAPL_old ~= 0);  %Non-Zero Elements
vnz1_old = find(AAPL_old ~= 0);%Indices Of Non-Zero
   Elements
AAPL_old = interp1(vnz1_old, AAPL_onz, 1:mo1(end)-1, '
   linear')';
%Interpolated APPL function is above


%The same process is repeated for each variable

%Apple's Net Income
rev_onz = rev_old(rev_old ~= 0);
vnz2_old = find(rev_old ~= 0);
rev_old = interp1(vnz2_old, rev_onz, iv1, 'linear')';

%S&P 500
sp500_onz = sp500_old(sp500_old ~= 0);
vnz3_old = find(sp500_old ~= 0);
sp500_old = interp1(vnz3_old, sp500_onz, 1:mo1(end)-1, '
   linear')';

%Consumer Price Index
cpi_onz = cpi_old(cpi_old ~= 0);
```

```
vnz4_old = find(cpi_old ~= 0);
cpi_old = interp1(vnz4_old, cpi_onz, iv1, 'linear')';

%inflation
inf_onz = inf_old(inf_old ~= 0);
vnz5_old = find(inf_old ~= 0);
inf_old = interp1(vnz5_old, inf_onz, iv1, 'linear')';

%interate rate
ir_onz = ir_old(ir_old ~= 0);
vnz6_old = find(ir_old ~= 0);
ir_old = interp1(vnz6_old, ir_onz, iv1, 'linear')';
%GLD
gld_onz = gld_old(gld_old ~= 0);
vnz7_old = find(gld_old ~= 0);
gld_old = interp1(vnz7_old, gld_onz, iv1, 'linear')';
gld_old(1:3794)=zeros(1,3794);
%Apple's R&D Spending
rd_onz = rd_old(rd_old ~= 0);
vnz8_old = find(rd_old ~= 0);
rd_old = interp1(vnz8_old, rd_onz, iv1, 'linear')';
%P/E Ratio
eps_nz = eps_old(eps_old ~= 0);
vnz9_old = find(eps_old ~= 0);
eps = interp1(vnz9_old, eps_nz, iv1, 'linear')';
%NASDAQ 100
nas_onz = nas_old(nas_old ~= 0);
```

```
vnz10_old = find(nas_old ~= 0);
nas_old = interp1(vnz10_old, nas_onz, iv1, 'linear')';
%iPhone Revenue
iphone_onz = iphone_old(iphone_old ~= 0);
vnz12_old = find(iphone_old ~= 0);
iphone_old = interp1(vnz12_old, iphone_onz, iv1, 'linear')
   ';
iphone_old(1:4931)=zeros(1,4931);
%Confidence Index
conf_onz = conf_old(conf_old ~= 0);
vnz13_old = find(conf_old ~= 0);
conf_old = interp1(vnz13_old, conf_onz, iv1, 'linear')';


%make all data same length
%subtract 1 so that the data files end on 12/31/07 (or
   12/30/07 in special cases) instead of January 1,2018,
   which is where the new data files start
rev_old=rev_old(1:mo1(end)-1);
rd_old=rd_old(1:mo1(end)-1);
iphone_old=iphone_old(1:mo1(end)-1);
conf_old=conf_old(1:mo1(end)-1);
nas_old=nas_old(1:mo1(end)-1);
AAPL_old=AAPL_old(1:mo1(end)-1);
eps=eps(1:mo1(end)-1);
gld_old=gld_old(1:mo1(end)-1);
ir_old=ir_old(1:mo1(end)-1);
```

```
inf_old=inf_old(1:mo1(end)-1);
cpi_old=cpi_old(1:mo1(end)-1);
sp500_old=sp500_old(1:mo1(end)-1);
buyback_old=buyback_old(1:mo1(end)-1);
%caculate P/E ratio using EPS and stock price
pe_old=AAPL_old./eps(1:mo1(end)-1);


%Combine old data values and new data values
rev=[rev_old;rev];
rd=[rd_old;rd];
iphone=[iphone_old;iphone];
NASDAQ100=[nas_old;NASDAQ100];
AAPL=[AAPL_old;AAPL];
gld=[gld_old;gld];
ir=[ir_old;ir];
inflation=[inf_old;inflation];
cpi=[cpi_old;cpi];
sp500=[sp500_old;sp500];
buyback=[buyback_old;buyback];
pe=[pe_old;pe];
conf=[conf_old;conf];

mo=[mo1;mo1(end)+mo(2:end)-1];%create new vector of
   locations of the start of every month

min_max=[];
```

```
PD = [];
for v=28:last_v
A = AAPL(mo(v-2):mo(v));
A2 = AAPL(mo(v-1):mo(v));
price_diff = max(A)-min(A);
price_diff2 = max(A2)-min(A2);
PD = [PD,price_diff,price_diff2];
min_max = [min_max,[min(A);max(A)]];
end
get_max_diff = maxk(PD,1);
```

****

J_to_KJ.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function reorders Barnett's coefficient into the %
%             order seen in equation (1.1).               %
% Input: Barnett's coefficients.                         %
% Output: Coefficients reorderd in the form of (1.1)     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function KJ = J_to_KJ(c)
    KJ = [c(1:7);c(49);c(8);c(10);c(9);c(11:17);c(19);c
       (18);c(20:24);...
        c(27);c(26);c(25);c(47);c(48);c(28:31);c(36);c
           (32:35);c(38);c(37);c(39:40);...
        c(43);c(41:42);c(44:46)];
end
```

****

KJ_to_J.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function reorders coefficients in the form of                    %
% (1.1) into what is in Barnett's code.                                  %
% Input: Our coefficients.                                               %
% Output: Coefficients reorderd in the form of Barnett's%
% code.                                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function KJ=KJ_to_J(c)
    KJ = [c(1:7);c(9);c(11);c(10);c(12:18);c(20);c(19);c
       (21:25);c(28);c(27);c(26);...
        c(31:34);c(36:39);c(35);c(41);c(40);c(42);c(43);c
           (45);c(46);c(44);c(47:49);c(29);c(30);c(8)];
end
```

****

T_reg_data.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Integral Operator T                                           %
% This function applies the operator T to psi                           %
% See the disseration.                                                   %
% psi: psi_ns (currently, the size of 4x6)                               %
% x_1: x_1(t)                                                            %
% x_2: x_2(t)                                                            %
```

```
% s: Time Step (in days)                                          %
% Input:                                                          %
% psi,x_1,x_2,x_3,x_4d,x_4,x_5,x_6,x_7d,x_7,x_8,x_9,x_10%
%,x_11,x_12,x_13,s                                                %
% Output: T                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T=T_reg_data(psi,x_1,x_2,x_3,x_4d,x_4,x_5,x_6,
   x_7d,x_7,x_8,x_9,x_10,x_11,x_12,x_13,s)
    psi_size = size(psi); %49x10
    r1 = psi(1,:).*x_1.*x_2; % Multiplies entrywise
    r2 = psi(2,:).*x_1+psi(3,:).*x_4d.*x_2+psi(4,:).*x_2;
    r3 = psi(5,:).*x_2+psi(6,:).*x_6+psi(7,:).*x_8+psi
       (8,:).*x_13;
    r4 = psi(9,:).*gradient(x_1)+psi(10,:).*x_4+psi(11,:)
       .*x_2.*x_4d+psi(12,:).*x_8.*x_4;
    r5 = psi(13,:).*x_8+psi(14,:).*x_10+psi(15,:).*x_4+psi
       (16,:).*x_12;
    r6 = psi(17,:).*x_6+psi(18,:).*x_7d+psi(19,:).*x_11+
       psi(20,:).*x_12;
    r7 = psi(21,:).*x_7+psi(22,:).*x_6+psi(23,:).*x_11;
    r8 = psi(24,:).*x_6+psi(25,:).*gradient(x_5)+psi(26,:)
       .*x_8+...
        psi(27,:).*x_8.*x_4+psi(28,:).*x_3+psi(29,:).*
           gradient(x_10)+...
        psi(30,:).*gradient(x_13);
    r9 = psi(31,:).*x_10+psi(32,:).*x_12+psi(33,:).*x_2+
       psi(34,:).*x_9;
```

```
    r10 = psi(35,:).*x_10+psi(36,:).*x_4+psi(37,:).*x_1+
       psi(38,:).*x_5+psi(39,:).*x_12;
    r11 = psi(40,:).*x_7d+psi(41,:).*x_11+psi(42,:).*x_4d
       .*x_11;
    r12 = psi(43,:).*gradient(x_1)+psi(44,:).*x_9+psi
       (45,:).*x_4+...
        psi(46,:).*x_1;
    r13 = psi(47,:).*x_10+psi(48,:).*x_6+psi(49,:).*x_7;
    matrix = [r1;r2;r3;r4;r5;r6;r7;r8;r9;r10;r11;r12;r13];
    T=s*cumtrapz(matrix,2);
end
```

****

T_reg_x1.m

****

Though there are thirteen different T_is, we only show T_1 here. $T_2, \cdots T_{13}$ functions are written in similar way.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Integral Operator T_1                    %
% This function applies the operator T to psi             %
% See PDF for the actual T                                %
% psi: psi_ns (currently, the size of 4x6)                %
% x_1: x_1(t)                                             %
% x_2: x_2(t)                                             %
% s: Time Step                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T=T_reg_x1(psi,x_1,x_2,s)
```

```
    r1 = psi(1,:).*x_1.*x_2;
    matrix = [r1];
     T=cumtrapz(matrix,2)*s;
end
```

****

T_star_data.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    T_Star                                              %
% This function caclulates T_star(psi) for us.                          %
% psi: This is 2x6 data ux-u_psi_m+g-T_psi_m                            %
% x_1: x_1(t)                                                           %
% x_2: x_2(t)                                                           %
% s: time step                                                          %
% See the dissertation for the exact function.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ts=T_star_data(psi,x_1,x_2,x_3,x_4d,x_4,x_5,x_6,
   x_7d,x_7,x_8,x_9,x_10,x_11,x_12,x_13,s,psi_0)
    back_int = flip(cumtrapz(flip(psi,2),2),2)*s;
    gx1 = gradient(x_1);
    gx5 = gradient(x_5);
    gx10 = gradient(x_10);
    gx13 = gradient(x_13);

    p1 = x_1.*x_2.*back_int(1,:);
    p2 = [x_1;x_4d.*x_2;x_2].*back_int(2,:);
```

```
    p3 = [x_2;x_6;x_8;x_13].*back_int(3,:);
    p4 = [gx1;x_4;x_2.*x_4d;x_8.*x_4].*back_int(4,:);
    p5 = [x_8;x_10;x_4;x_12].*back_int(5,:);
    p6 = [x_6;x_7d;x_11;x_12].*back_int(6,:);
    p7 = [x_7;x_6;x_11].*back_int(7,:);
    p8 = [x_6;gx5;x_8;x_8.*x_4;x_3;gx10;gx13].*back_int
       (8,:);
    p9 = [x_10;x_12;x_2;x_9].*back_int(9,:);
    p10 = [x_10;x_4;x_1;x_5;x_12].*back_int(10,:);
    p11 = [x_7d;x_11;x_4.*x_11].*back_int(11,:);
    p12 = [gx10;x_9;x_4;x_1].*back_int(12,:);
    p13 = [x_10;x_6;x_7].*back_int(13,:);

    Ts = [p1;p2;p3;p4;p5;p6;p7;p8;p9;p10;p11;p12;p13];
end
```

****

u.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          u(x)                                          %
% This function calculates u(x)                                         %
% t : time in 1xn matrix.                                                %
% g : This is the g we have declared.                                    %
% s : Time Step                                                          %
% a, b: From our domain [a,b]                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function u = u(t,g,s,a,b)
    int_g = cumtrapz(g,2)*s;
    int_a_b = trapz(int_g,2)*s;
    ux_1 = flip(cumtrapz(flip(int_g,2),2),2)*s;
    u = ux_1 - (1/(b-a))*(b-t).*int_a_b;
end
```

****

u_psi.m

****

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    u_psi                                               %
% This function evaluates u_psi as shown in PDF                          %
% T_psi: T(psi)                                                          %
% b: from our domain [a,b]                                               %
% x_1: x_1(t)   only here to get the size                                %
% s: Step size                                                           %
% t: times                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function u_psi=u_psi(T_psi,a,b,s,t)
    common = cumtrapz(T_psi,2)*s;
    middle = (s/(b-a))*(t-a).*trapz(common,2);
    u_psi_0_2 = cumtrapz(common,2)*s;
    u_psi = middle - u_psi_0_2;
end
```

****

u_psi_p.m

```
**********************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  u'_psi                                                %
% This function evaluates u_psi as shown in PDF                          %
% T_psi: T(psi)                                                          %
% a,b: from our domain [a,b]                                             %
% s: Step size                                                           %
% Please see the dissertation for exact function.                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function u_psi_p=u_psi_p(T_psi,a,b,s)
    common = cumtrapz(T_psi,2)*s;
    middle = (s/(b-a))*trapz(common,2);
    u_psi_p = middle - common;
end
```

Appendix G: The LSTM Model

PYTHON CODE

The Python code below is the LSTM model we built to forecast stock prices.

Predicting Apple's stock price using Long Short-Term Memory (LSTM) networks

This process was motivated by the program built on https://python.plainenglish.io/predicting-apple-stock-price-36f329cda530 (https://python.plainenglish.io/predicting-apple-stock-price-36f329cda530)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

Step 1) Load your data

Let υ represent the first day of the month where $\upsilon \in [28, 340]$.

$\upsilon = 28$ represents October 1, 1996, $\upsilon = 29$ represents November 1, 1996 and so on.

We will first load our data and initialize 'a', 'b' and 'diff'.

- a: The row before υ
- diff : Number of rows needed to give two full months starting at υ
- b: Number of rows starting at υ and ending at the very end of the data

```
data = pd.read_csv("AAPL_all.csv")
a = 6440
diff = 42
data[a:a+diff]
```

```
b = 7326-a+1
a,b,a+b
```

Step 2) Prep the data to train our Model

```
data_to_train = data[:a]
data_to_test = data[a:]
```

```
# Here, we select the 'closed' column from the imported data
aapl_data = data.iloc[:,4:5]
aapl_data.head()
```

```
training_set = aapl_data.iloc[:a,:].values
test_set = aapl_data.iloc[a:,:].values
```

```
# Here we normalize the data
from sklearn.preprocessing import MinMaxScaler
sc= MinMaxScaler(feature_range=(0,1))
training_set_scaled= sc.fit_transform(training_set)
```

```
# Create a data structure with 30 timesteps and 1 output
# We also re-shape the data into (# of values, # of time-steps, and # of one dimensional output)
X_train=[] #Independent variables
y_train= [] # Dependent variables
# I am going to append prior 30 data points not including current
for i in range(a-30,a):
    X_train.append(training_set_scaled[i-30:i,0]) # Appending prior 30 data points
    y_train.append(training_set_scaled[i,0])      # Value of the next current point

X_train, y_train= np.array(X_train), np.array(y_train)
```

Step 3) Build our model using tensorflow.

Our model will have total of 6 layers including the output layer and each layer will consist of LSTM layer and a dropout layer.

```
# Importing the Keras libraries and packages
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM

#  Initialising the RNN
model= Sequential()

# Adding first LSTM layer and some dropout Dropout regularisation
model.add(LSTM(units=100,return_sequences=True, input_shape=(X_train.shape[1],1)))
model.add(Dropout(rate=0.2))

# Adding second LSTM layer and some dropout Dropout regularisation
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))

# Adding third LSTM layer and some dropout Dropout regularisation
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))

# Adding fourth LSTM layer and some dropout Dropout regularisation
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))

# Adding fifth LSTM layer and some dropout Dropout regularisation
model.add(LSTM(units=100))
model.add(Dropout(rate=0.2))

# Adding the Output Layer
model.add(Dense(units=1))

# Compiling the Model
# Because we're doing regression hence mean_squared_error
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
model.summary()
```

```
# Fitting the model to the Training set
history=model.fit(X_train,y_train,epochs=100,batch_size=32)
```

Step 4) Use the model built to make predictions.

```
# Here we get the actual dates of each data point
appl_date = data.iloc[:,0:1]
date_set = appl_date.iloc[a:,:]
```

```
# Setting the amount of days to forecast
c = 30
```

```
# Check the length of the month you want to predict
one = data['Date'].tolist()
two = one[a:a+diff]
p = a+diff
two
```

```
# Concatenate the dataset needed and then scale them
data_total= pd.concat([data_to_train['Close'], data_to_test['Close']],  axis=0)
# inputs= data_total[len(data_total)-len(data_to_test)-c:].values
inputs = data_total.values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)

X_test = []
for i in range(a,p):
    X_test.append(inputs[i-c:i, 0])

X_test = np.array(X_test)
# 3D format
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

```
#preict the model
predicted_stock_price = model.predict(X_test)
```

```
# Inverse the scaling
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
k=len(predicted_stock_price)
```

```
real_stock_price=data_to_test['Close'].tolist()
len(real_stock_price[:k])
```

```
pred_stock = []
for i in range(len(predicted_stock_price)):
    pred_stock.append(predicted_stock_price[i,0])
len(pred_stock)
```

```
df = pd.DataFrame(list(zip(pred_stock,real_stock_price[:k])),index = two,
                  columns=['Predicted Apple Stock Price','Actual Apple Stock Price'])
df.plot.line(style={'Predicted Apple Stock Price': 'r', 'Actual Apple Stock Price': 'b'},figsize=(10,5))
#df.plot.line(style={'Predicted Apple Stock Price': 'r', 'Actual Apple Stock Price': 'b'})
plt.legend(loc='upper left')
plt.show()
```

```
from sklearn.metrics import mean_absolute_percentage_error as mape
error = mape(real_stock_price[:k],pred_stock)*100
error
```