The Dynamic Nelson-Siegel Approach for Yield Curve Modelling

Vivien Ziting Zhou

Supervisors: Gareth Peters
Maggie Jing Chen

A dissertation submitted in partial fulfillment for the degree of
Master of Science

Department of Statistical Science
University College London
4 September 2015

Abstract

The dissertation focuses on the yield curve modelling by using the dynamic Nelson-Siegel approach. We discuss the yields-only model and propose a general form of the extended model that can incorporate a set of macroeconomic variables based on the work of Diebold and Li (2006). The main contribution of this dissertation is to implement and compare the gradient-based and the EM-based estimation procedures for the state-space representation of the model. We also derive a recursive method to find the optimal estimate of the latent factors and associate it to the Kalman filter. According to our simulation study, we find that the gradient descent method performs better than the EM algorithm in term of stability and accuracy. We further test the gradient method on the real data example, and get good results.

Declaration

I declare that this dissertation is my own work and that all sources have been acknowledged.

Acknowledgement

I would like to sincerely thank my supervisor, Dr. Gareth Peters for his guidance, patience and encouragement. I have learnt a lot this summer while working on an interesting project under his supervision.

I would also like to thank my co-supervisor, Dr. Maggie Jing Chen, for her support and suggestions during the project.

Finally, I would express a deep sense of gratitude to my parents for their love and supports over the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The yield curve, also well-known as the term structure of interest rates, is a very useful tool in the world of fixed income. At any particular time, the yield curve illustrates the yields of various bonds across the maturity spectrum. Not only the yield curve at a given time, but also the evolution of the entire yield curve over time plays an important role in completing many financial tasks, including pricing financial assets as discussed in Campbell (1987) [3], conducting monetary policy as discussed in Campbell (1995) [2] and predicting recessions as discussed in Estrella et al. (1996) [12]. Besides being very important, the evolution of the yield curve is very complex as well, because the shape of the yield curve is not unique, it can be normal, inverted, flat and so on, and it does not remain the same over time. As a result, studying the evolution of the yield curve is equivalent to studying a large set of yield curves corresponding to different points in time and those curves usually do not follow a common pattern, which is a challenging problem. In Figure 1.1, as an example, we plots the U.S. Treasury bond yields in all three dimensions, showing the yields both across a range of bond maturities and over time. Due to the importance and complexity of the yield curve dynamics, a dynamic yield curve model is required. Furthermore, the yield curves are different across countries. For example, the yield curve of the U.K. Government bonds shown in Figure 1.2 is not the same as the one of the U.S. Treasury bonds shown in Figure 1.1. The dynamic model is therefore required to have good adaptability so that it can be used for modelling the yield curves in different countries.

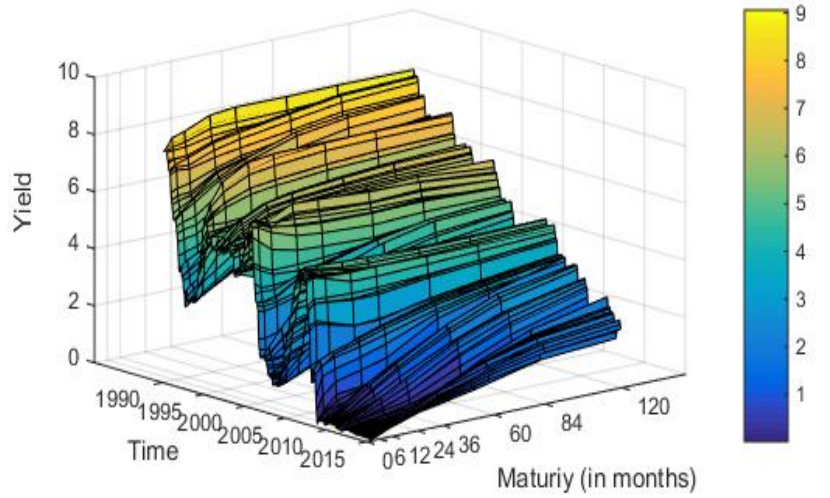

Figure 1.1: Three-dimensional plot of monthly yields of the U.S. Treasury bills and bonds. The sample maturities are 3, 6, 12, 24, 36, 84 and 12 months. The sample period is from January 1990 to August 2015. Data are from the U.S. Treasury Department.

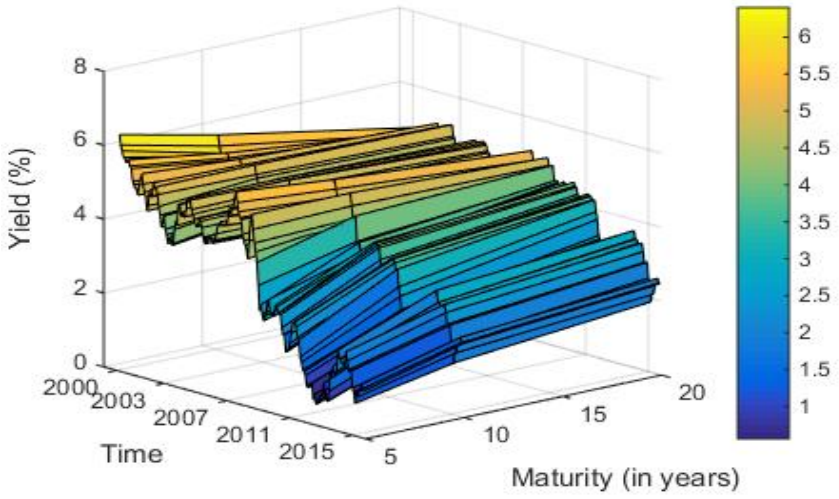

Figure 1.2: Three-dimensional plot of monthly yields of the UK government bonds. The sample maturities are 5, 10 and 20 years. The sample period is from January 2000 to August 2015. Data are from Bank of England.

A variety dynamic term structure models have been proposed and developed from a large volume of finance and econometric literature. Most of these models can be divided into two categories according to their different approaches to modelling, and these two approaches are the equilibrium approach and the arbitrage-free approach. The equilibrium models, including the Vasicek model (1977) [25] and the Cox–Ingersoll–Ross model (1985) [5], derive the yields of the bonds with longer maturities by using the assumed stochastic process of the short rates. It is worth pointing out that the equilibrium models have a significant shortcoming in addition to being not arbitrary-free, which is the inconsistency between the equilibrium and the actual term structure observed. The Ho-Lee model (1986) [16] is the first no-arbitrage model that makes up for this shortcoming, as it can be calibrated to the market data. Besides the Ho-Lee model (1986) [16], there are more examples belonging to the category of arbitrage-free models, such as the Hull–White model (1990) [17] and the Heath–Jarrow–Morton (HJM) model (1992) [15]. The arbitrage-free models are the models that specify the form of the yield curve at a given point in time to ensure that there are no arbitrage opportunities at that time. In the past thirty years, a number of researchers have made great contributions to these two approaches.

However, in this dissertation, we pay attention to neither the equilibrium approach nor the arbitrage-free approach, but another one called dynamic Nelson-Siegel approach. The models mentioned above are built by using stochastic differential equations, while the models under the dynamic Nelson-Siegel approach use time series. The Nelson-Siegel yield curve model [20] has been commonly used since it was introduced in 1987, but it did not used for modelling the dynamics of the yield curve until Diebold and Li (2006) [8] incorporated dynamics into it with appropriate interpretations. The dynamic Nelson-Siegel (DNS) model generally refers to the Diebold-Li extension (2006) [8]. This relatively new approach for dynamic yield curve modelling has been widely researched over the past ten years, and many extended, more complex models have been proposed. The aim of this dissertation is to provide a comprehensive review of the DNS model, with an emphasis on the model estimation under the state-space framework of the DNS model introduced by Diebold et al. (2006) [10]. We implement two gradient-based methods (the gradient descent algorithm and the Newton's method), and the expectation-maximization (EM) algorithm. After implementing the three algorithms, we examine the performance of the gradient descent algorithm and

the EM algorithm by using the synthetic data.

The remainder of this dissertation is organized as follows: Chapter 2 discusses the relevant literature and also briefly introduces the evolution of the dynamic Nelson-Siegel approach since Diebold and Li (2006) [8] and Diebold et al. (2006 [10] made a prominent contribution in this area. In Chapter 3, the yield curve model itself is introduced with detailed explanation. We firstly introduce the static Nelson-Siegel yield curve, and then move to its dynamic extension by Diebold and Li (2006) [8], which is followed by the discussion about the state space representation and the extended dynamic yield curve model with the incorporated macroeconomic variables introduced by Diebold et al. (2006) [10]. Subsequently, Chapter 4 implement several techniques that can be used for model estimation under the state-space framework of the DNS model. In Chapter 5, we perform a simulation study for comparing two estimation procedures. Chapter 6 presents a real data example of the DNS model. We model the U.S. Treasury bill and bond yields and discuss the results. Chapter 7 provides some final conclusions and directions for future work.

Chapter 2

Related literature

The study about the dynamic Nelson-Siegel yield curve modelling is highly related to two important papers in the area: Diebold and Li (2006) [8] and Diebold et al. (2006) [10]. Diebold and Li (2006) [8] extend the original Nelson-Siegel yield curve (1987) [20] to a dynamic model, and interpret the model as a dynamic factor structure. They also interpret the three time-varying parameters in Nelson-Siegel as latent factors corresponding to the level, slope and curvature of the entire yield curve. The approach for model estimation introduced in Diebold and Li (2006) [8] is called two-step DNS. They firstly fit the static Nelson-Siegel yield curve at each point in time so that a three-dimensional times series of estimated factors and the corresponding error terms can be obtained. After that, they model and estimate autoregressive models for the factors. Following the work of Diebold and Li(2006) [8], Diebold et al. (2006) [10] introduce a unified state-space representation of the model. Moreover, under the state-space framework, they introduce another model estimation procedure, known as the one-step approach. This approach estimates the state-space form of the DNS model by using the Kalman filter together with a numerical method for maximum likelihood estimation, which allows fitting the single yield curve and estimating the underlying dynamics of the factors at that point of time to be done simultaneously.

In this dissertation, we pay our attention to the one-step estimation approach and hence the study is related to the literature studying the likelihood-based estimation procedures for state-space models. In Engle and Watson (1981)

[11], the maximum likelihood estimation is achieved by using the the Fisher's scoring algorithm, which is based on Newton's method. Shumway and Stoffer (1982) [23] propose a recursive procedure that uses EM algorithm [7] in conjunction with the Kalman filter [18] for estimating the unknown parameters in the state-space model by maximum likelihood. Watson and Engle (1983) [27] present and compare these two methods, and suggest to use a mix of EM and Scoring in practice.

Deibold and Li (2006) [8] and Diebold et al. (2006) [10] have been shifting more attention back to the Nelson-Siegel approach for modelling the dynamics of the yield curve, and there are a number of literature further studying the extended models. Benefiting from the state-space structure of the DNS model, Diebold et al. (2008) [9] extend the model to a global context, modelling sets of yield curves corresponding different countries in a framework that allows to capture both country-specific factors and global factors. Yu and Zivot (2010) [28] extend the DNS model to a broader empirical prospective by including different ratings for corporate bonds. In order to enforce absence of arbitrage, Christensen et al. (2011) [4] introduce a new class of affine arbitrage-free models based on the dynamic Nelson-Siegel model, which is the arbitrage-free Nelson-Siegel (AFNS) models. They argue that this new class of models are not only empirically tractable but also theoretically rigorous.

Most recently, researchers have been trying to incorporate different macroeconomic variables to the DNS model. Diebold et al. (2006) [10] is the first work in this area. They include manufacturing capacity, the federal funds rate and the annual price inflation in the model, and find significant evidence of the effect of these variables on the yield curve. Pooter et al. (2010) [6] incorporate fifteen categories of macro factors to the model, and find that the models incorporated with macroeconomic factors are more accurate in and around recession periods.

Chapter 3

Nelson-Siegel Approach for Modelling

This chapter introduces the dynamic Nelson-Siegel (DNS) model, including the models with and without macroeconomic variables. In the section of the model without macroeconomic variables, we start with the original Nelson-Siegel yield curve model (1987) [20] for fitting a static yield curve at a fixed time. By taking the time-varying environment into account, we then proceed to and focus on the dynamic representation of the Nelson-Siegel and its interpretation by Diebold and Li (2006) [8]. This is followed by the construction of the state-space structure of the DNS model. After introducing the yields-only model, we discuss the extended model that can incorporate the macroeconomic variables. In this section, we give a general form of extended state-space model based on the work of Diebold et al. (2006) [10].

3.1 Model without Macroeconomic Variables

3.1.1 Static Nelson-Siegel Model

At any fixed time, a large set of yields across different maturities for a type of bond is available. Nelson and Siegel (1987) [20] propose a model for fitting a smooth curve to such set of yields. This is the original Nelson-Siegel model,

and it has the following modified functional form which is commonly used in the literature:

$$y(\tau) = \beta_1 + \beta_2\left(\frac{1-e^{-\lambda\tau}}{\lambda\tau}\right) + \beta_3\left(\frac{1-e^{-\lambda\tau}}{\lambda\tau} - e^{-\lambda\tau}\right), \tag{3.1}$$

where $y(\tau)$ is the yield to maturity, τ is the maturity, and λ, β_1, β_2, β_3 are the parameters of the model.

The Nelson-Siegel yield curve model is one of the most popular model for yield curve fitting in practice. There are several significant reasons for its popularity. We firstly discuss it from a financial economics perspective. As we know, the discount rate curve and the forward rate curve are two other curves of interest in addition to the yield curve in finance, and both of them can be produced from the corresponding yield curve. To study how well the Nelson-Siegel yield curve performs from a financial economics perspective, we analyze all these three curves under the Nelson-Siegel framework, especially properties of their asymptotes. The discount curve is a line graph showing the prices of discount bond across different time periods. The forward rate curve shows the future yields on a bond over different time periods. We denote the discount curve and forward rate curve at maturity τ by $P(\tau)$ and $r(\tau)$ respectively.

In Nelson and Siegel (1987) [20], the corresponding forward rate curve is given, and it is actually used to derive the Nelson-Siegel yield curve in conjunction with the relationship between these two curves. The yield is an average of the corresponding forward rate, and this relationship can be written as $y(\tau) = \frac{1}{\tau}\int_0^\tau r(u)du$ by definition. The forward rate curve under Nelson-Siegel framework is given by:

$$r(\tau) = \beta_1 + \beta_2 e^{-\lambda\tau} + \beta_3\lambda\tau e^{-\lambda\tau}. \tag{3.2}$$

From equation 3.2, we can find that the initial value of the forward rate curve is a constant, as $f(0) = \beta_1 + \beta_2$. This can be interpreted as the instantaneous short rate by definition, hence we can easily compute the short rate if a Nelson-Siegel model is given, which is an advantage of the model. The behaviour of the forward curve near infinity is also of our interest, because it can be interpreted as the long-term interest rate, and thus it is expected to be a constant. The limit of equation 3.2, as maturity approaches to infinity,

is evaluated with the help of L'Hopital's rule, and the result is:

$$\begin{aligned}\lim_{\tau\to\infty} f(\tau) &= \beta_1 + \beta_2 \lim_{\tau\to\infty} \frac{1}{e^{\lambda\tau}} + \beta_3\lambda \lim_{\tau\to\infty} \frac{\tau}{e^{\lambda\tau}} \\ &= \beta_1 + 0 + \beta_3\lambda \lim_{\tau\to\infty} \frac{1}{\lambda e^{\lambda\tau}} \\ &= \beta_1. \end{aligned} \tag{3.3}$$

The forward rate, as maturity becomes arbitrage large, is indeed a constant. Based on these results and the relationship between yield curve and forward rate curve, we know that the asymptotic results of the yield curve itself are also constants as required. The corresponding limits are:

$$\lim_{\tau\to 0} y(\tau) = f(0) = \beta_1 + \beta_2, \tag{3.4}$$

and

$$\lim_{\tau\to\infty} y(\tau) = \lim_{\tau\to\infty} f(\tau) = \beta_1. \tag{3.5}$$

The discount rate curve produced from the Nelson-Siegel yield curve can be obtained by using the relationship between these two curves, which is expressed as $P(\tau) = e^{-\tau y(\tau)}$ by the definition of price of the discount bond. The discount curve under Nelson-Siegel framework has the following expression:

$$P(\tau) = exp\left\{-\tau\beta_1 - \frac{\beta_2}{\lambda}\Big(1 - exp(-\lambda\tau)\Big) - \frac{\beta_3}{\lambda}\Big(1 - exp(-\lambda\tau) - \lambda\tau exp(-\lambda\tau)\Big)\right\}. \tag{3.6}$$

We can then calculate the initial value of the discount curve using equation 3.6, which is $P(0) = 1$. The limit of the discount rate, as maturity approaches to infinity, can be easily found as well, which is $\lim_{\tau\to\infty} P(\tau) = 0$. Both of these two results are the expected characteristics of a discount curve.

Hence we can see that the Nelson-Siegel yield curve itself and the related curves derived from it all have some desirable properties of curves from the view of finance and economics. This is one of the most appealing advantages of the Nelson-Siegel model.

There are at least two more reasons why Nelson-Siegel is a popular and widely-used model among financial institutions. One of the reasons is that the shape of the Nelson-Siegel yield curve is controlled by the parameters

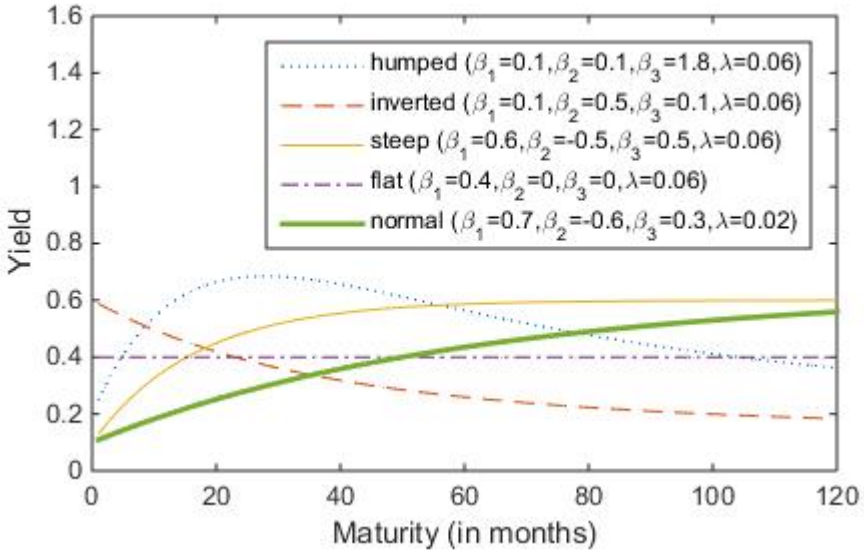

Figure 3.1: Some possible shapes of the Nelson-Siegel yield curve

λ, β_1, β_2 and β_3, and with different values of these parameters, the shape of the Nelson-Siegel curve can be any one of the typical shapes of the yield curve, including normal, inverted, steep, flat and humped shape. Figure 3.1 shows some possible shapes of the Nelson-Siegel yield curve. Furthermore, as discussed in the original paper of Nelson-Siegel (1987) [20], this yield curve model is parsimonious, promoting desirable smoothness to the curve. Providing parsimony with only four parameters is another big advantage of Nelson-Siegel model.

Up until this point, we have discussed the static model for a single yield curve across various maturities at a fixed time. In practice, there is a sequence of curves over time and these curves move in a stochastic fashion. In the next subsection, we will discuss the dynamic version of the Nelson-Siegel model, with focus on its representation by Diebold and Li (2006) [8].

3.1.2 Dynamic Nelson-Siegel and Diebold-Li Interpretation

The original dynamic Nelson-Siegel (DNS) model is constructed by simply changing the static parameters in equation 3.1 to be time-varying, which is given by:

$$y_t(\tau) = \beta_{1t} + \beta_{2t}\left(\frac{1-e^{-\lambda_t\tau}}{\lambda_t\tau}\right) + \beta_{3t}\left(\frac{1-e^{-\lambda_t\tau}}{\lambda_t\tau} - e^{-\lambda_t\tau}\right), \tag{3.7}$$

where time $t = 1, 2, \ldots, T$, maturity $\tau = \tau_1, \tau_2, \ldots, \tau_N$, and $y_t(\tau)$ is the yield for maturity τ at time t. It is important to note that β_1, β_2 and β_3 are acted as both variables and parameters in the model, because they are time-varying for any fixed maturity τ, but static across different maturities for any fixed time t. The remaining λ_t is a parameter at time t that controls the second and third components.

In Diebold and Li (2006) [8], the DNS model is interpreted as a dynamic factor model with three dynamic, latent factors β_{1t}, β_{2t}. The parameter λ_t is considered to a time-invariant parameter with a fixed value of 0.0609 in their model, and we will discuss the value of it later. After simply dropping t from λ_t, we can see that the structure of factor loading on each factor is clearly shown in equation 3.7.

Based on the structure, Diebold and Li (2006) [8] also argue that the three latent factors not only can be classified as long-term, short-term and medium-term factors according to their relative effect on the yield at different maturities, which is fairly common in the literature before Diebold and Li (2006), but also can be interpreted as the level, the slope and the curvature according to their impact on the shape of the overall yield curve.

The factor β_{1t} is viewed as the level or long-term factor, because the loading on it is 1, a constant, and therefore at time t, it loads equally on the yield regardless of the maturity. This is also coincident with our discussion result in the subsection 3.1.1, where we concludes that at a fixed time t, the horizontal line at β_{1t} is the asymptote of the forward curve near infinity and hence it is linked to the long-term interest rate and the level of overall yield curve.

The second factor β_{2t} is termed as the slope or short-term factor. The loading on factor β_{2t} is $((1-e^{-\lambda\tau})/\lambda\tau)$, which starts from infinitely near 1 and decays

monotonically to 0 as the maturity increases. The decay takes place at a fast rate due to the exponential term, which means that the factor has more impact on yields of the short-term bonds and determines the slope of the whole yield curve.

The third factor β_{3t} is interpreted as the curvature or medium-term factor. $((1 - e^{-\lambda\tau})/\lambda\tau - e^{-\lambda\tau})$ is the loading on this factor. It has a humped shape, starting at a point infinitely close to 0, increasing until it reaches the peak, and then decaying back to 0. It has little impact on either short-term or long-term yields, because it starts from 0 and finally decays to 0. While looking at the overall yield curve instead of the range of maturities, it is the factor that contributes to the determination of the curvature of the curve. Figure 3.2 shows the shape of the three factor loadings, where λ is fixed at 0.0598. The value of λ will be discussed later.

In the rest of this dissertation, we focus on the Diebold-Li interpretation of the DNS model [8], considering the latent factors as level, slope and curvature factors. The equation 3.7 is modified a bit express this representation. The Diebold-Li model [8] is displayed as follows:

$$y_t(\tau) = L_t + S_t\left(\frac{1 - e^{-\lambda\tau}}{\lambda\tau}\right) + C_t\left(\frac{1 - e^{-\lambda\tau}}{\lambda\tau} - e^{-\lambda\tau}\right), \tag{3.8}$$

where L_t, S_t, C_t are the level, slope and curvature at time t respectively, and λ is a time-invariant parameter.

As we can see, the constant parameter λ is an indispensable part of the Diebold-Li model [8], because it not only governs the rates of decay of the factor loading on the slope (the larger the value of λ, the faster the decay), but also determines the maturity at which the factor loading on the curvature is maximized. The estimated value of λ is concluded to be 0.0609 by Diebold and Li (2006) [8]. They argue that this is the value of λ that maximizes the loading on the medium-term factor (the curvature) at exactly 30 months. The medium-term maturity is generally defined as the maturity ranging from 1 year to 5 years. Hence setting the maturity τ at the averaging 2.5 years (30 months) is reasonable. However, 0.0609 is not the correct value for the maximization. As shown in Figure 3.3, we can observe that the factor loading on the curvature has a unique maximum when λ approximately equals to 0.0598 rather than 0.0609. Moreover, Figure 3.4 shows that the corresponding maturity to the λ value of 0.0609 is in fact less than 30 months,

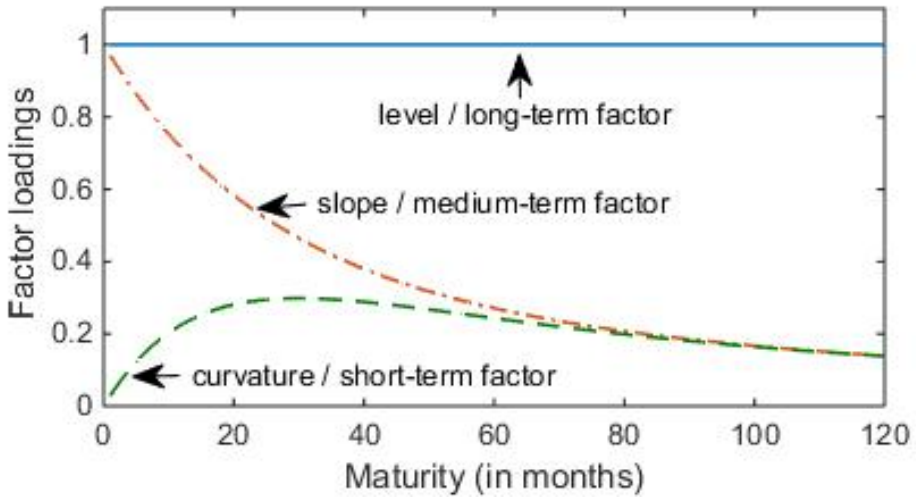

Figure 3.2: Plot of factor loadings as functions of maturity (fix $\lambda = 0.0598$)

roughly 29.4 months instead. Paul Veerhusi (2011) [26] corrects and fixes the value of λ in this case at 0.059776. In addition, it is worth mentioning that although the fixed λ is advocated in many studies, the value of it is diverse across different studies, mainly due to the differences in the maturity and the unit of measurement they used. For example, the maturity of 23.3 months is chosen by Diebold et al. (2006) [10], and they estimate the corresponding λ to be 0.077. Changing the unit of measurement in maturity from month to year, the estimates of λ are changed as well, even though the maturities used have the same length of time, for instance, the estimated λ of 0.0598 and 0.7173 imply that the factor loading on the curvature is maximized at a maturity of 30 months and a maturity of 2.5 years respectively.

To sum up, the dynamic Nelson-Siegel model retains the advantages of the static Nelson-Siegel yield curve, and is applied for modelling the yield curve in time-varying environment. By using Diebold-Li representation of the DNS model [8], we can give interpretations to the latent factors, viewing them as the level, slope and curvature of the overall yield curve, which is useful in estimating the factors and forecasting the future yield curve.

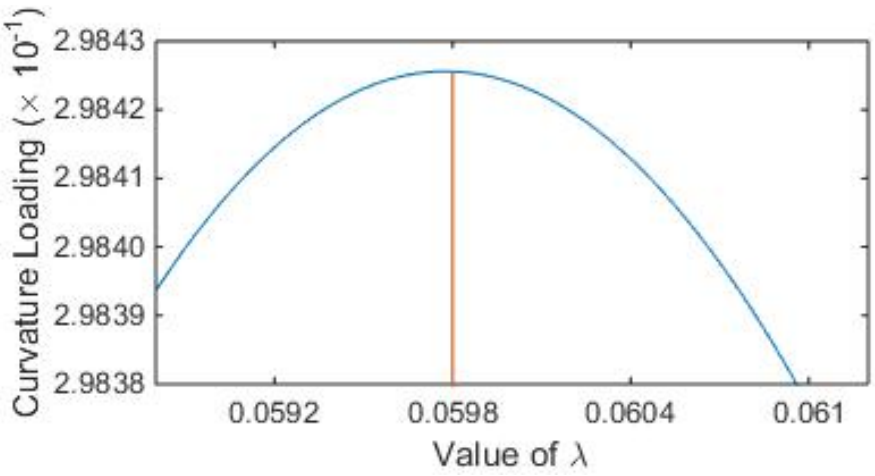

Figure 3.3: Plot of curvature loading against lambda (fix $\tau = 30$ months)

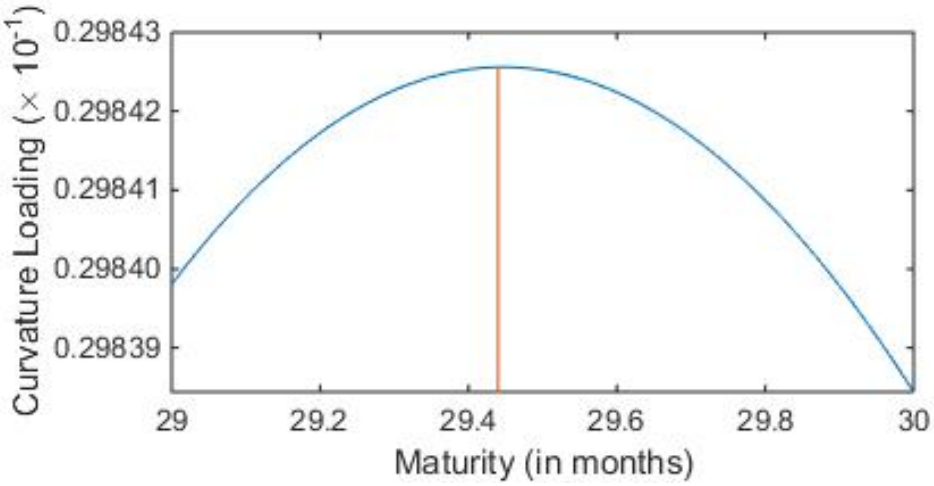

Figure 3.4: Plot of curvature loading against maturity (fix $\lambda = 0.0609$)

3.1.3 State-Space model representation

The dynamic Nelson-Siegel model can be specified and estimated by using the state-space framework as presented in Diebold et al. (2006) [10]. In generally, the state-space framework of a system is given by two equations: the measurement equation and the transition equation. In the case of representing the DNS model, the measurement equation should be an equation that expresses the relationship between the observable yields and the three unobservable latent factors. This equation can be constructed by adding the maturity-specific error terms to the deterministic DNS model shown in equation 3.8. We consider N different maturities, say $\tau_1, \tau_2, \ldots, \tau_N$, and their corresponding errors, also known as the measurement noises, are denoted by $\epsilon(\tau_1), \epsilon(\tau_2), \ldots, \epsilon(\tau_N)$. The measurement equation is written as:

$$\begin{bmatrix} y_t(\tau_1) \\ y_t(\tau_2) \\ \vdots \\ y_t(\tau_N) \end{bmatrix} = \begin{bmatrix} 1 & \frac{1-exp(-\tau_1\lambda)}{\tau_1\lambda} & \frac{1-exp(-\tau_1\lambda)}{\tau_1\lambda} - exp(-\tau_1\lambda) \\ 1 & \frac{1-exp(-\tau_2\lambda)}{\tau_2\lambda} & \frac{1-exp(-\tau_2\lambda)}{\tau_2\lambda} - exp(-\tau_2\lambda) \\ \vdots & \vdots & \vdots \\ 1 & \frac{1-exp(-\tau_N\lambda)}{\tau_N\lambda} & \frac{1-exp(-\tau_N\lambda)}{\tau_N\lambda} - exp(-\tau_N\lambda) \end{bmatrix} \begin{bmatrix} L_t \\ S_t \\ C_t \end{bmatrix} + \begin{bmatrix} \epsilon_t(\tau_1) \\ \epsilon_t(\tau_2) \\ \vdots \\ \epsilon_t(\tau_N) \end{bmatrix}. \tag{3.9}$$

The random vector of the observation errors is assumed to be homogeneous in time, correlated to the space, and independently and identically distributed in calendar time. The distribution of it is usually treated as normal, and we will discuss this later.

As shown in equation 3.9, the three latent factors L_t, S_t and C_t remain the same across maturities but are time-varying. The transition equation under the state-space framework for the DNS model should then be an equation that models the time series of these factors. In Diebold et al. (2006) [10], the time series model is assumed to be a vector autoregressive model of order 1 (VAR(1)). This is not the only choice. For example, Boswijk (2013) [1] uses the model with 2 lags (VAR(2)). Pooter et al. (2010) [6] examine AR and VAR models with multiple lags, and find that using multiple lags give nearly identical results compared to the models with one lag. In this dissertation, we select VAR(1) like Diebold et al. (2006) [10]. The reason to choose VAR(1) is that any vector autoregressive model with other lag always has a VAR(1) form as its alternative representation, and hence VAR(1) allows better adaptability. In addition, we center the factors around the corresponding time-invariant factor means, denoted by μ^L, μ^S and μ^C, so that we can drop

the constant term in the VAR(1) model. The factor dynamics is governed by a time-invariant 3×3 matrix, denoted by the matrix $\mathbf{A}$ with entries $= a_{ij}$, where $i, j = 1, 2, 3$. The error terms included in VAR(1) are denoted by η_t^L, η_t^S and η_t^C, and they are also known as process noises. The transition equation can then be expressed by a VAR(1) model as follows:

$$\begin{bmatrix} L_t - \mu^L \\ S_t - \mu^S \\ C_t - \mu^C \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} l_{t-1} - \mu^L \\ s_{t-1} - \mu^S \\ c_{t-1} - \mu^C \end{bmatrix} + \begin{bmatrix} \eta_t^L \\ \eta_t^S \\ \eta_t^C \end{bmatrix}. \tag{3.10}$$

The assumptions about the measurement noises and the process noises are essential in the state-space framework. Diebold et al. (2006) [10] make the following standard assumptions about $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\eta}_t$. Firstly, they assume that both of the two noises are white noises and they are orthogonal to each other and to the initial state $\mathbf{f}_0$ which is assumed to be known. Secondly, the covariance matrix of $\boldsymbol{\epsilon}_t$ is assumed to be a $N \times N$ diagonal matrix, denoted by $\mathbf{H}$, which implies an assumption that the deviations of observed yields at various maturities are uncorrelated. In contrast, $\boldsymbol{\eta}_t$ is assumed to be a 3×3 non-diagonal matrix, denoted by $\mathbf{Q}$, which implies the consideration of the correlated shocks of the three factors. After making these assumptions, the complete state-space model can then be summarized by writing both equations 3.9 and 3.10 in vector and matrix notation. Using the state-space form, the Diebold-Li representation of the dynamic Nelson-Siegel model is represented as the following two equations:
measurement equation:

$$\mathbf{y}_t = \boldsymbol{\Lambda}(\lambda)\mathbf{f}_t + \boldsymbol{\epsilon}_t, \tag{3.11}$$

transition equation:

$$\mathbf{f}_t - \boldsymbol{\mu} = \mathbf{A}(\mathbf{f}_{t-1} - \boldsymbol{\mu}) + \boldsymbol{\eta}_t, \tag{3.12}$$

alternatively,

$$\mathbf{f}_t = \mathbf{A}\mathbf{f}_{t-1} + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu} + \boldsymbol{\eta}_t, \tag{3.13}$$

with assumptions:

$$\begin{bmatrix} \boldsymbol{\eta}_t \\ \boldsymbol{\epsilon}_t \end{bmatrix} \sim WN\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{H} \end{bmatrix} \right),$$

$$\mathrm{E}(\mathbf{f}_0 \boldsymbol{\eta}_t^\intercal) = \mathbf{0},\ \mathrm{E}(\mathbf{f}_0 \boldsymbol{\epsilon}_t^\intercal) = \mathbf{0},$$

where the vectors are defined as:

$$\mathbf{y}_t = \begin{bmatrix} y_t(\tau_1) \\ y_t(\tau_2) \\ \vdots \\ y_t(\tau_N) \end{bmatrix}, \mathbf{f}_t = \begin{bmatrix} L_t \\ S_t \\ C_t \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \mu^L \\ \mu^S \\ \mu^C \end{bmatrix}, \boldsymbol{\epsilon}_t = \begin{bmatrix} \epsilon_t(\tau_1) \\ \epsilon_t(\tau_2) \\ \vdots \\ \epsilon_t(\tau_N) \end{bmatrix}, \boldsymbol{\eta}_t = \begin{bmatrix} \eta_t^L \\ \eta_t^S \\ \eta_t^C \end{bmatrix},$$

and the matrices are defined as:

$$\boldsymbol{\Lambda}(\lambda) = \begin{bmatrix} 1 & \frac{1-exp(-\tau_1\lambda)}{\tau_1\lambda} & \frac{1-exp(-\tau_1\lambda)}{\tau_1\lambda} - exp(-\tau_1\lambda) \\ 1 & \frac{1-exp(-\tau_2\lambda)}{\tau_2\lambda} & \frac{1-exp(-\tau_2\lambda)}{\tau_2\lambda} - exp(-\tau_2\lambda) \\ \vdots & \vdots & \vdots \\ 1 & \frac{1-exp(-\tau_N\lambda)}{\tau_N\lambda} & \frac{1-exp(-\tau_N\lambda)}{\tau_N\lambda} - exp(-\tau_N\lambda) \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

Diebold et al.(2006) [10] do not make any assumption about the distributions of the measurement noises and process noises themselves. In this dissertation, we make additional assumptions of normality so that we can have a linear Gaussian state-space framework of the DNS model. These assumptions are required on the estimation problem that will be discussed in the next chapter. We assume that each of the two noise terms identically and independently follows a multivariate normal distribution respectively. The two distributions are given by:

$$\boldsymbol{\eta}_t \overset{iid}{\sim} \mathcal{N}_3(\mathbf{0}, \mathbf{Q}),$$
$$\boldsymbol{\epsilon}_t \overset{iid}{\sim} \mathcal{N}_N(\mathbf{0}, \mathbf{H}).$$

3.2 Yield Curve Model with Macroeconomic Variables

The state-space representation of the DNS model is extended to incorporate the macroeconomic variables by Doebold et al. (2006) [10]. To be more specific, they incorporate three key measures of the economy, which are the manufacturing capacity utilization, the federal funds rate and the annual

price inflation. However, the macroeconomic factors incorporated to the model should not be limit to these three variables, as many empirical and theoretical studies in the finance literature have demonstrated that there are another macroeconomic factors that are important drivers of the yield curve. For example, Vargas (2005) [24] finds that real activity, budget deficit and exchange rate have significant impact on the level and slope of the yield curve, and budget deficit has significant impact on all the level, slope and curvature of the yield curve. Longstaff et al. (2005) [19] find that the yield spread is strongly related to measures of liquidity and credit quality. This implies that liquidity and credit quality are the determinants of the yield curve as well. Moreover, Piazzesi (2005) [22] believe that the Federal Reserve's interest rate is a key macroeconomic variable. Hence incorporating these variable to the model may also improve the accuracy of the DNS model.

To provide more flexibility to the covariates in the model, we provide the extended model that includes a set of macroeconomic variables. We denote such set of covariates observed at time t by $\mathbf{x}$, which is a $p \times 1$ vector and p is the number of macroeconomic variables included.

The incorporation is done by using link functions to relate the time-varying covariates $\mathbf{x}_t$ to the level, slope and curvature factors respectively. Hence the latent factors $\boldsymbol{\mu}$ in the yields-only model are no longer time-invariant in this macro-yield model. The time-varying latent factors are expressed as:

$$\mu_t^i = \mu^i + \boldsymbol{\beta}^i \mathbf{x}_t, \quad i = L, S, C, \tag{3.14}$$

where $\boldsymbol{\beta}^i$s are $1 \times p$ parameter vectors.

The state-space structure of the extended model are almost the same as the one of the model without macroeconomic variables. The only difference is the transition equation, which is slightly modified as follows:

$$\mathbf{f}_t - \boldsymbol{\mu}_t = \mathbf{A}(\mathbf{f}_{t-1} - \boldsymbol{\mu}_t) + \boldsymbol{\eta}_t, \tag{3.15}$$

with

$$\boldsymbol{\mu}_t = \boldsymbol{\mu} + \mathbf{B}\mathbf{x}_t \tag{3.16}$$

where $\mathbf{B}$ is a $3 \times p$ matrix of coefficients.

As we can see, benefiting from the powerful state-space structure, the dynamic Nelson-Siegel model can be easily extended to the one with macroeconomic variables.

Chapter 4

Model Estimation

This chapter introduces some procedures for estimating the dynamic Nelson-Siegel model by using one-step DNS approach [10]. The estimation of all unknowns are done simultaneously based on the state-space structure of the DNS model presented in the previous chapter. The chapter is divided into two sections. In the first section, we derive the optimal estimate of the latent factor based on the property of linear unbiased, and then associate this recursive fashion with the well-known Kalman filter [18]. In the second section, we introduce three popular numerical methods, the gradient descent algorithm, the Newton's method and the EM algorithm. We implement each of them in details for the purpose of finding the maximum likelihood estimation of the unknown parameters in the DNS model.

4.1 Latent Factor Estimation

Given the observed yields and the assumed parameters, the unobservable, latent factors L_t, S_t and C_t in the DNS model are need to be estimated. As presented in the previous chapter, by making appropriate assumptions, the dynamic Nelson-Siegel model has a linear Gaussian state-space representation. The state $\mathbf{f}_t$ is corresponding to the latent factors at time t. We therefore only need to estimate the state for the given values of the parameters and the observations (the yields data) under the state-space framework. In this section, we assume that the parameters are known.

To begin with, we summarize the dimensions of matrices and vectors that are involved in our discussion in Table 4.1. Some of them have been introduced in Chapter 3, and the rest of them will be defined in this Chapter. In addition, in this paper, we use $\intercal$ to represent the transpose in matrix operation.

Vector	Dimension	Matrix	Dimension
$\mathbf{y}_t$	$N \times 1$	$\mathbf{\Lambda}$	$N \times 3$
$\mathbf{f}_t$	3×1	$\mathbf{A}$	3×3
$\boldsymbol{\mu}$	3×1	$\mathbf{Q}$	3×3
$\boldsymbol{\epsilon}_t$	$N \times 1$	$\mathbf{H}$	$N \times N$
$\boldsymbol{\eta}_t$	3×1	$\mathbf{P}_t$	3×3
$\mathbf{v}_t$	$N \times 1$	$\mathbf{K}_t$	$3 \times N$
		$\mathbf{W}_t$	$N \times N$

Table 4.1: Dimensions of Matrices and Vectors

We do the estimation with minimum mean square error (MMSE). The optimal estimate of the state $\mathbf{f}_t$, denoted by $\hat{\mathbf{f}}_t$, is defined as the one that minimises the expectation:

$$\mathbb{E}[||\mathbf{f}_t - \hat{\mathbf{f}}_t||^2] = \mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_t)^\intercal(\mathbf{f}_t - \hat{\mathbf{f}}_t)]. \tag{4.1}$$

By differentiating equation 4.1 with respect to $\hat{\mathbf{f}}_t$, setting the result to be zero and doing second derivative test, we can obtain the expression of the MMSE estimate of $\mathbf{f}_t$:

$$\hat{\mathbf{f}}_t = \mathbb{E}[\mathbf{f}_t]. \tag{4.2}$$

The estimation of the state is based on the observations. We want to find estimate of the state at time t by using all the available observations up to time t, , $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t$, which we denote by $\mathbf{f}_{t|t}$. This estimate is named as the updated state, and it can be expressed as:

$$\hat{\mathbf{f}}_{t|t} = \mathbb{E}[\mathbf{f}_t|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t]. \tag{4.3}$$

From the state-space model shown in equation 3.11 and 3.12, we know that it is possible to estimate the state at time t by using the observations up to time $t-1$, $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}$. This estimate is the predicted state, which we denote by $\hat{\mathbf{f}}_{t|t-1}$, and based on the result in equation 4.2, the predicted state has the following expression:

$$\hat{\mathbf{f}}_{t|t-1} = \mathbb{E}[\mathbf{f}_t|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}]. \tag{4.4}$$

The predicted state can then be written in terms of the model parameters and the updated state one step behind by using the equations 3.11, 4.3 and 4.4, along with the assumption that the process noises $\boldsymbol{\eta}_t$ are Gaussian distributed with zero mean and the fact that they are independent to the observations. The expression is as follows:

$$\begin{aligned}
\hat{\mathbf{f}}_{t|t-1} &= \mathbb{E}[\mathbf{f}_t|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}] \\
&= \mathbb{E}[\mathbf{A}\mathbf{f}_{t-1} + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu} + \boldsymbol{\eta}_t|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}] \\
&= \mathbf{A}\mathbb{E}[\mathbf{f}_{t-1}|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}] + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu} + \mathbb{E}[\boldsymbol{\eta}_t|\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{t-1}] \\
&= \mathbf{A}\hat{\mathbf{f}}_{t-1|t-1} + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu} \qquad (4.5)
\end{aligned}$$

Furthermore, we need to determine the covariance matrix of errors. The covariance matrices that correspond to the predicted state $\mathbf{f}_{t|t-1}$ and the updated state $\mathbf{f}_{t|t}$ are denoted by $\mathbf{P}_{t|t-1}$ and $\mathbf{P}_{t|t}$ respectively, namely the predicted covariance matrix and the updated covariance matrix. They are given by the expressions below:

$$\mathbf{P}_{t|t-1} = \mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})^\intercal], \qquad (4.6)$$

$$\mathbf{P}_{t|t} = \mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t})^\intercal]. \qquad (4.7)$$

The expression of predicted covariance matrix $\mathbf{P}_{t|t-1}$ can be derived from equations 3.11, 4.3 4.5 and 4.6. We use the fact that the process noises $\boldsymbol{\eta}_t$ are uncorrelated to the states, and they are assumed to follow a multivariate normal distribution, $\boldsymbol{\eta}_t \overset{iid}{\sim} \mathcal{N}_N(\mathbf{0}, \mathbf{Q})$. It can written as follows:

$$\begin{aligned}
\mathbf{P}_{t|t-1} &= \mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})^\intercal] \\
&= \mathbb{E}[(\mathbf{A}\mathbf{f}_{t-1} + \boldsymbol{\eta}_t - \mathbf{A}\hat{\mathbf{f}}_{t-1|t-1})(\mathbf{A}\mathbf{f}_{t-1} + \boldsymbol{\eta}_t - \mathbf{A}\hat{\mathbf{f}}_{t-1|t-1})^\intercal] \\
&= \mathbf{A}\mathbb{E}[(\mathbf{f}_{t-1} - \hat{\mathbf{f}}_{t-1|t-1})(\mathbf{f}_{t-1} - \hat{\mathbf{f}}_{t-1|t-1})^\intercal]\mathbf{A}^\intercal + \mathbb{E}[\boldsymbol{\eta}_t\boldsymbol{\eta}_t^\intercal] \\
&= \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^\intercal + \mathbf{Q}. \qquad (4.8)
\end{aligned}$$

We are not satisfied to just obtain the predicted state $\hat{\mathbf{f}}_{t|t-1}$, because in addition to the observations up to and including at time $t-1$, the observation at time t is also observed. Including the available observations as more as possible will increase the precision in estimation. Hence, the updated state $\hat{\mathbf{f}}_{t|t}$ is more desirable and should be the one used as the estimate of the state

that we are looking for. Under the state-space framework, at time t, both the observation $\mathbf{y}_t$ and the predicted state $\hat{\mathbf{f}}_{t|t-1}$ are related to the updated state $\hat{\mathbf{f}}_{t|t}$. We then make a reasonable assumption that $\hat{\mathbf{f}}_{t|t}$ is a linear weight sum of $\mathbf{y}_t$ and $\hat{\mathbf{f}}_{t|t-1}$. This relationship is given by:

$$\hat{\mathbf{f}}_{t|t} = \mathbf{K}'\hat{\mathbf{f}}_{t|t-1} + \mathbf{K}\mathbf{y}_t, \tag{4.9}$$

where $\mathbf{K}'$ and $\mathbf{K}$ are two different weights.

The optimal values $\mathbf{K}'$ and $\mathbf{K}$ are to be determined. We achieve these by using the unbiased property of the MMSE estimator. Being an unbiased estimator, the updated state must has the expectation that $\hat{\mathbf{f}}_{t|t}$ that satisfy:

$$\mathbb{E}[\hat{\mathbf{f}}_{t|t}] = \mathbb{E}[\mathbf{f}_t]. \tag{4.10}$$

We now derive the expectation of the updated state $\hat{\mathbf{f}}_t$ by using the results shown in equations 3.11, 3.12, 4.3, 4.5, 4.9, and the assumption that the measurement noises $\boldsymbol{\epsilon}_t$ are Gaussian distributed with zero mean. The expression is derived as follows:

$$\begin{aligned}
\mathbb{E}[\hat{\mathbf{f}}_{t|t}] &= \mathbb{E}[\mathbf{K}'\hat{\mathbf{f}}_{t|t-1} + \mathbf{K}(\boldsymbol{\Lambda}\mathbf{f}_t + \boldsymbol{\epsilon}_t)] \\
&= \mathbf{K}'\mathbb{E}[\hat{\mathbf{f}}_{t|t-1}] + \mathbf{K}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{f}_t] + \mathbf{K}\mathbb{E}[\boldsymbol{\epsilon}_t] \\
&= \mathbf{K}'\mathbb{E}[\hat{\mathbf{f}}_{t|t-1}] + \mathbf{K}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{f}_t] \\
&= \mathbf{K}'\mathbb{E}[\mathbf{A}\hat{\mathbf{f}}_{t-1|t-1} + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu}] + \mathbf{K}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{f}]_t \\
&= \mathbf{K}'\{\mathbf{A}\mathbb{E}[\hat{\mathbf{f}}_{t-1|t-1}] + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu}\} + \mathbf{K}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{f}_t] \\
&= \mathbf{K}'\mathbb{E}[\mathbf{f}_t] + \mathbf{K}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{f}_t] \\
&= (\mathbf{K}' + \mathbf{K}\boldsymbol{\Lambda})\mathbb{E}[\mathbf{f}_t].
\end{aligned} \tag{4.11}$$

Combing equations 4.10 and 4.11, we can find that the two weights $\mathbf{K}'$ and $\mathbf{K}$ must have a certain relationship, and this relationship is unique, which is:

$$\mathbf{K}' = \mathbf{I} - \mathbf{K}\boldsymbol{\Lambda}. \tag{4.12}$$

Now we can modify the equation 4.9, expressing the predicted state in terms of only one weight $\mathbf{K}$ as follows:

$$\begin{aligned}
\hat{\mathbf{f}}_{t|t} &= (\mathbf{I} - \mathbf{K}\boldsymbol{\Lambda})\hat{\mathbf{f}}_{t|t-1} + \mathbf{K}\mathbf{y}_t \\
&= \hat{\mathbf{f}}_{t|t-1} + \mathbf{K}(\mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1}).
\end{aligned} \tag{4.13}$$

The optimal value of $\mathbf{K}$ will be determined later.

Before proceeding to the updated covariance matrix of $\mathbf{P}_{t|t}$, we firstly find the difference between the true state $\mathbf{f}_t$ and the updated state $\hat{\mathbf{f}}_{t|t}$ by using the results in equations 3.11 and 4.13:

$$\begin{aligned}
\mathbf{f}_t - \hat{\mathbf{f}}_{t|t} &= \mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1} - \mathbf{K}(\mathbf{y}_t - \mathbf{\Lambda}\hat{\mathbf{f}}_{t|t-1}) \\
&= \mathbf{f}_t - \mathbf{K}\mathbf{y}_t - (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\hat{\mathbf{f}}_{t|t-1} \\
&= \mathbf{f}_t - \mathbf{K}(\mathbf{\Lambda}\mathbf{f}_t + \boldsymbol{\epsilon}_t) - (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\hat{\mathbf{f}}_{t|t-1} \\
&= (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1}) - \mathbf{K}\boldsymbol{\epsilon}_t. \qquad (4.14)
\end{aligned}$$

The updated covariance matrix $\mathbf{P}_{t|t}$ can then be derived based on equations 4.7 and 4.14, along with the assumption that the Gaussian distributed measurement noises $\boldsymbol{\epsilon}_t$ is orthogonal to the state. The derivation is as follows:

$$\begin{aligned}
\mathbf{P}_{t|t} &= \mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t})^\intercal] \\
&= \mathbb{E}\{[(\mathbf{I} - \mathbf{K}\mathbf{\Lambda})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1}) - \mathbf{K}\boldsymbol{\epsilon}_t][(\mathbf{I} - \mathbf{K}\mathbf{\Lambda})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1}) - \mathbf{K}\boldsymbol{\epsilon}_t]^\intercal\} \\
&= (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\mathbb{E}[(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})(\mathbf{f}_t - \hat{\mathbf{f}}_{t|t-1})^\intercal](\mathbf{I} - \mathbf{K}\mathbf{\Lambda})^\intercal + \mathbf{K}\mathbb{E}[\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t^\intercal]\mathbf{K}^\intercal \\
&= (\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\mathbf{P}_{t|t-1}(\mathbf{I} - \mathbf{K}\mathbf{\Lambda})^\intercal + \mathbf{K}\mathbf{H}\mathbf{K}^\intercal. \qquad (4.15)
\end{aligned}$$

The weight $\mathbf{K}$ is still undetermined at this stage, and we need to find the optimal value of it. As mentioned earlier, our estimator should be a MMSE estimator, minimizing the expression in equation 4.1. Equivalently, the optimal $\mathbf{K}$ should minimize the trace of the updated covariance matrix $\mathbf{P}_{t|t}$. The trace of $\mathbf{P}_{t|t}$ can be expressed as follows:

$$\begin{aligned}
tr(\mathbf{P}_{t|t}) &= tr((\mathbf{I} - \mathbf{K}\mathbf{\Lambda})\mathbf{P}_{t|t-1}(\mathbf{I} - \mathbf{K}\mathbf{\Lambda})^\intercal + \mathbf{K}\mathbf{H}\mathbf{K}^\intercal) \\
&= tr(\mathbf{P}_{t|t-1} - \mathbf{K}\mathbf{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal\mathbf{K}^\intercal + \mathbf{K}\mathbf{\Lambda}\mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal\mathbf{K}^\intercal + \mathbf{K}\mathbf{H}\mathbf{K}^\intercal) \\
&= tr(\mathbf{P}_{t|t-1} + \mathbf{K}\mathbf{\Lambda}\mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal\mathbf{K}^\intercal + \mathbf{K}\mathbf{H}\mathbf{K}^\intercal) - 2tr(\mathbf{K}\mathbf{\Lambda}\mathbf{P}_{t|t-1}). \qquad (4.16)
\end{aligned}$$

The derivative of the trace given above with respect to $\mathbf{K}$ is:

$$2\mathbf{K}\mathbf{\Lambda}\mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal + 2\mathbf{K}\mathbf{H} - 2\mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal = 0. \qquad (4.17)$$

By equating equation 4.17 to zero and solving it, the optimal $\mathbf{K}$ is then obtained, which has the following expression:

$$\mathbf{K} = \mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal(\mathbf{\Lambda}\mathbf{P}_{t|t-1}\mathbf{\Lambda}^\intercal + \mathbf{H})^{-1}. \qquad (4.18)$$

We can determine this $\mathbf{K}$ is the optimal one that minimizes the trace instead of maximizing it, as we can easily find that the second derivative of the trace with respect to $\mathbf{K}$ is always greater than zero. Moreover, as we can see, the optimal $\mathbf{K}$ at time t depends on the predicted covariance matrix $\mathbf{P}_{t|t}$ at that time, it therefore time-varying. We denote it by $\mathbf{K}_t$ instead.

Multiplying both sides of equation 4.18 by $(\boldsymbol{\Lambda}\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal + \mathbf{H})\mathbf{K}^\intercal$, then we have the equation:

$$\mathbf{K}_t(\boldsymbol{\Lambda}\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal + \mathbf{H})\mathbf{K}_t^\intercal = \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{K}_t^\intercal. \tag{4.19}$$

By substituting the result in equation 4.19 into the equation 4.15, we can have a simplified expression of the updated covariance matrix $\mathbf{P}_{t|t}$, which is:

$$\begin{aligned}
\mathbf{P}_{t|t} &= \mathbf{P}_{t|t-1} - \mathbf{K}_t\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{K}_t^\intercal + \mathbf{K}_t\boldsymbol{\Lambda}\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{K}_t^\intercal + \mathbf{K}_t\mathbf{H}\mathbf{K}_t^\intercal \\
&= \mathbf{P}_{t|t-1} - \mathbf{K}_t\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{K}_t^\intercal + \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{K}_t^\intercal \\
&= \mathbf{P}_{t|t-1} - \mathbf{K}_t\boldsymbol{\Lambda}\mathbf{P}_{t|t-1}.
\end{aligned} \tag{4.20}$$

Now we can summarize the algorithm that we can use to estimate the state at each discrete time point. A known initial predicted estimate of the state and a known predicted covariance matrix, denoted by $\hat{\mathbf{f}}_{1|0}$ and $\mathbf{P}_{1|0}$ are required to start the algorithm and there are two stages involved. The algorithm is defined as follows:

Prediction stage:

$$\hat{\mathbf{f}}_{t|t-1} = \mathbf{A}\hat{\mathbf{f}}_{t-1|t-1} + (\mathbf{I} - \mathbf{A})\boldsymbol{\mu} \tag{4.21}$$

$$\mathbf{P}_{t|t-1} = \mathbf{A}\mathbf{P}_{t-1|t-1}\mathbf{A}^\intercal + \mathbf{Q} \tag{4.22}$$

Update stage:

$$\hat{\mathbf{f}}_{t|t} = \hat{\mathbf{f}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1}) \tag{4.23}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} \tag{4.24}$$

This is actually a form of the well-known Kalman filter algorithm [18]. Compared with the original form of the algorithm, we just drop t off the parameters, because our parameters $\boldsymbol{\mu}$, Λ, $\mathbf{A}$, $\mathbf{Q}$ and $\mathbf{H}$ are all time-invariant.

Kalman filtering is an optimal recursive data processing algorithm that is typically used for smoothing noisy data and producing estimates of parameters of interest for linear systems with Gaussian distributed error. The filter

was invented and preliminarily developed in the work of Kalman [18], and hence is named after Rudolf E. Kalman.

It has been more than 50 years since the Kalman filter is first described, and it is still one of the most important algorithms for linear quadratic estimation nowadays. The Kalman filter is computationally efficient due to its elegant recursive structure, and it is also proved to be the optimal estimator in the sense that it minimises both mean square error of the estimated parameters and estimated covariance matrix given that all noises in the system are Gaussian distributed.

4.2 Parameter Estimation

As discussed in the previous section, the latent yield factors in the state-space representation of the DNS model can be estimated by using the Kalman filter with given parameters. However, these parameters that are necessary to run the algorithm are unknown in reality and need to be estimated. Maximizing the log-likelihood function is the method that we use to find the estimation. We denote this set of parameters with $\boldsymbol{\theta}$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\Lambda}, \mathbf{A}, \mathbf{Q}, \mathbf{H}\}$.

The log-likelihood function is constructed based on the assumption that the prediction errors are multivariate Gaussian distributed. We firstly write the vector of prediction errors at time t, $\mathbf{v}_t$, by definition as follow:

$$\mathbf{v}_t = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1} = \mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1}. \tag{4.25}$$

The covariance matrix of $\mathbf{v}_t$ is denoted by $\mathbf{W}_t$, and it can be written in terms of the measurement noises defined in equation 3.11 and covariance matrix of the predicted state, which is given by:

$$\begin{aligned} \mathbf{W}_t &= \mathrm{Var}(\mathbf{v}_t) \\ &= \mathrm{Var}(\mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1}) \\ &= \mathbf{H} + \boldsymbol{\Lambda}\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal. \end{aligned} \tag{4.26}$$

Since the prediction error vector is assumed to be Gaussian distributed, the distribution of the random variable $\mathbf{y}_t$ conditional on the given value of prediction, which we denote by $\mathbf{y}_{t|t-1}$, is Gaussian as well. The distribution is

given by:

$$\mathbf{y}_t|\mathbf{y}_{t|t-1} \overset{d}{\sim} \mathcal{N}(\Lambda\hat{\mathbf{f}}_{t|t-1}, \mathbf{W}_t) \tag{4.27}$$

Based on this conditional distribution, we can now compute the log-likelihood function of $\boldsymbol{\theta}$ by computing the joint density of $\mathbf{y}_t|\mathbf{y}_{t-1|t}$, $t = 1, 2, \ldots, T$.

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= \log \prod_{t=1}^{T} \left\{ \frac{1}{(2\pi)^{N/2}|\mathbf{W}_t|^{1/2}} exp\left(\frac{-(\mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1})^\intercal \mathbf{W}_t^{-1} (\mathbf{y}_t - \boldsymbol{\Lambda}\hat{\mathbf{f}}_{t|t-1})}{2} \right) \right\} \\
&= \log \prod_{t=1}^{T} \left\{ \frac{1}{(2\pi)^{N/2}|\mathbf{W}_t|^{1/2}} exp\left(\frac{-\mathbf{v}_t{}^\intercal \mathbf{W}_t^{-1} \mathbf{v}_t}{2} \right) \right\} \\
&= \sum_{t=1}^{T} \left\{ -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{W}_t| - \frac{1}{2}\mathbf{v}_t{}^\intercal \mathbf{W}_t^{-1} \mathbf{v}_t \right\} \\
&= -\frac{NT}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{T}\log|\mathbf{W}_t| - \frac{1}{2}\sum_{t=1}^{T}\mathbf{v}_t{}^\intercal \mathbf{W}_t{}^{-1}\mathbf{v}_t.
\end{aligned} \tag{4.28}$$

Analytical solution that maximizes this log-likelihood function is not available. Hence we use numerical methods to find the maximum likelihood estimation. There are many iterative methods which have be developed to the optimization problem. In this paper, we discuss and implement two popular approaches: the gradient-based method and the expectation-maximization (EM) algorithm. Both the gradient descent algorithm and the Newton-Raphson method will be explained when we discuss the gradient-based methods, and we mainly focus on the former.

4.2.1 Estimation via the Gradient Descent Algorithm

The first method we discuss and implement is the gradient descent algorithm. Gradient descent algorithm, also known as steepest descent algorithm, is a numeric iterative method to find the nearest local minimum of a function that has a computable gradient. It is based on the observation that a function will decrease fastest if its variable moves from a given value in the direction of the negative gradient of the function at that value.

The algorithm starts with a initial value of the targeted variable $\mathbf{x}$, denoted by $\mathbf{x}_0$, and the given function $f(\mathbf{x})$ is presumed to be defined and differentiable

at this point. Then it calculates the negative gradient of the function at the point $\mathbf{x}_0$, $-\nabla f(\mathbf{x}_0)$. After that, this negative gradient is used to find a updated value of $\mathbf{x}$, $\mathbf{x}_1$, where $f(\mathbf{x}_1) \leqslant f(\mathbf{x}_0)$. The updated value of $\mathbf{x}$ is then used as the initial value, the above steps are repeated until the approximate minimum to the predefined accuracy is found. The algorithm takes the following form of iteration:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \boldsymbol{\gamma}_n \nabla f(\mathbf{x}^{(n)}), \quad n \geqslant 0, \tag{4.29}$$

where γ_n is the chosen step size at iteration n, $\mathbf{x}^{(n)}$ is the approximate at iteration n, and $f(\mathbf{x}^{(n+1)}) \leqslant f(\mathbf{x}_{(n)})$.

We want to do the maximum log-likelihood estimation, equivalently, we need to find the minimum of the negative log-likelihood function. Moreover, the gradient of the log-likelihood $\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is also called the score for $\boldsymbol{\theta}$, which is denoted by $S(\boldsymbol{\theta})$. We then modify the form in equation 4.29 so that we have a specific form of the gradient descent algorithm for our case of maximizing log-likelihood function. The approach that approximates the local maximum of a function is also known as gradient ascent. The form of each iteration for our case is:

$$\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} + \boldsymbol{\gamma} S(\boldsymbol{\theta}^{(n)}), \quad n \geqslant 0. \tag{4.30}$$

where $\boldsymbol{\gamma}$ is the preset step size for all iterations, $\boldsymbol{\theta}^{(n)}$ is the approximation of $\boldsymbol{\theta}$ at iteration n, and $\boldsymbol{\theta}^{(n+1)}$ is a better approximation obtained at iteration $n+1$ compared with $\boldsymbol{\theta}^{(n)}$.

Now we need to calculate the score function $S(\boldsymbol{\theta})$. From the expression in equation 4.28, we can find that the log-likelihood function $\ell(\boldsymbol{\theta})$ can be decomposed into T parts:

$$\ell(\boldsymbol{\theta}) = \sum_{t=1}^{T} \ell_t(\boldsymbol{\theta}), \tag{4.31}$$

where each part has the expression:

$$\ell_t(\boldsymbol{\theta}) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{W}_t| - \frac{1}{2}\mathbf{v}_t{}^{\intercal}\mathbf{W}_t{}^{-1}\mathbf{v}_t \quad t = 1, 2, \ldots, T. \tag{4.32}$$

Finding the first derivative of $\ell_t(\boldsymbol{\theta})$ is still not trivial. However, we can compute it by using the method presented in the book of Harvey (1990) [14].

The derivatives of the determinant and the inverse of the symmetric matrix will be frequently used in the later discussion. We use the results given in the Matrix Cookbook by Petersen and Pedersen [21]. For any symmetric matrix $\mathbf{M}$, the derivatives of the determinant and the inverse of it with respect to a variable $\mathbf{x}$, are given by:

$$\frac{\partial|\mathbf{M}|}{\partial\mathbf{x}} = |\mathbf{M}|\, tr\left[\mathbf{M}^{-1}\frac{\partial\mathbf{M}}{\partial\mathbf{x}}\right], \tag{4.33}$$

and

$$\frac{\partial\mathbf{M}^{-1}}{\partial\mathbf{x}} = -\mathbf{M}^{-1}\frac{\partial\mathbf{M}}{\partial\mathbf{x}}\mathbf{M}^{-1}. \tag{4.34}$$

Differentiating the equation 4.32 with respect to a component of $\boldsymbol{\theta}$, denoted by $\boldsymbol{\theta}_i$, $i = 1, 2 \ldots, 5$, we can obtain the first derivative of the log-likelihood at time t. We use the derivative results in equations 4.33 and 4.34 and then take trace of the non-trace term in order to get simpler expression:

$$\begin{aligned}
\frac{\partial\ell_t(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i} &= -\frac{1}{2}\frac{1}{|\mathbf{W}_t|}\frac{\partial|\mathbf{W}_t|}{\partial\boldsymbol{\theta}_i} - \frac{1}{2}\frac{\partial(\mathbf{v}_t{}^{\intercal}\mathbf{W}_t{}^{-1}\mathbf{v}_t)}{\partial\boldsymbol{\theta}_i} \\
&= -\frac{1}{2}tr\left[\mathbf{W}_t^{-1}\frac{\partial\mathbf{W}_t}{\partial\boldsymbol{\theta}_i}\right] - \frac{1}{2}\left[\frac{\partial\mathbf{v}_t^{\intercal}}{\partial\boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\mathbf{v}_t + \mathbf{v}_t^{\intercal}\frac{\partial\mathbf{W}_t^{-1}}{\partial\boldsymbol{\theta}_i}\mathbf{v}_t + \mathbf{v}_t^{\intercal}\mathbf{W}_t^{-1}\frac{\partial\mathbf{v}_t}{\partial\boldsymbol{\theta}_i}\right] \\
&= -\frac{1}{2}tr\left[\mathbf{W}_t^{-1}\frac{\partial\mathbf{W}_t}{\partial\boldsymbol{\theta}_i}\right] + \frac{1}{2}tr\left[\mathbf{v}_t^{\intercal}\mathbf{W}_t^{-1}\frac{\partial\mathbf{W}_t}{\partial\boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\mathbf{v}_t\right] - \left(\frac{\partial\mathbf{v}_t}{\partial\boldsymbol{\theta}_i}\right)^{\intercal}\mathbf{W}_t^{-1}\mathbf{v}_t \\
&= -\frac{1}{2}tr\left[\left[\mathbf{W}_t^{-1}\frac{\partial\mathbf{W}_t}{\partial\boldsymbol{\theta}_i}\right]\left[\mathbf{I} - \mathbf{W}_t^{-1}\mathbf{v}_t\mathbf{v}_t^{\intercal}\right]\right] - \left(\frac{\partial\mathbf{v}_t}{\partial\boldsymbol{\theta}_i}\right)^{\intercal}\mathbf{W}_t^{-1}\mathbf{v}_t.
\end{aligned} \tag{4.35}$$

As we can see, the derivatives of $\mathbf{v}_t$ and $\mathbf{W}_t$ with respect to the vector $\boldsymbol{\theta}_i$ are needed in the computation. These two derivatives can be obtained by differentiating the equations 4.26 and 4.25 respectively, which are:

$$\frac{\partial\mathbf{W}_t}{\partial\boldsymbol{\theta}_i} = \frac{\partial\boldsymbol{\Lambda}}{\partial\boldsymbol{\theta}_i}\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^{\intercal} + \boldsymbol{\Lambda}\frac{\partial\mathbf{P}_{t|t-1}}{\partial\boldsymbol{\theta}_i}\boldsymbol{\Lambda}^{\intercal} + \boldsymbol{\Lambda}\mathbf{P}_{t|t-1}\frac{\partial\boldsymbol{\Lambda}^{\intercal}}{\partial\boldsymbol{\theta}_i} + \frac{\partial\mathbf{H}}{\partial\boldsymbol{\theta}_i}, \tag{4.36}$$

and

$$\frac{\partial\mathbf{v}_t}{\partial\boldsymbol{\theta}_i} = -\boldsymbol{\Lambda}\frac{\partial\hat{\mathbf{f}}_{t|t-1}}{\partial\boldsymbol{\theta}_i} - \frac{\partial\boldsymbol{\Lambda}}{\partial\boldsymbol{\theta}_i}\hat{\mathbf{f}}_{t|t-1}. \tag{4.37}$$

From equations 4.36 and 4.37, we know that the expression for the derivatives of $\hat{\mathbf{f}}_{t|t-1}$ and $\mathbf{P}_{t|t-1}$ with respect to the vector $\boldsymbol{\theta}_i$ are also needed. These two derivatives can be obtained based on the equations involved in the Kalman filter algorithm described in the previous section, to be more specific, the equations 4.21 to 4.24. They have the following expressions:

$$\frac{\partial \hat{\mathbf{f}}_{t|t-1}}{\partial \boldsymbol{\theta}_i} = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}_i}\hat{\mathbf{f}}_{t-1|t-1} + \mathbf{A}\frac{\partial \hat{\mathbf{f}}_{t-1|t-1}}{\partial \boldsymbol{\theta}_i} + \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}_i} - \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\mu} - \mathbf{A}\frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}_i}, \tag{4.38}$$

and

$$\frac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i} = \frac{\partial \mathbf{A}}{\partial \boldsymbol{\theta}_i}\mathbf{P}_{t-1|t-1}\mathbf{A}^\intercal + \mathbf{A}\frac{\partial \mathbf{P}_{t-1|t-1}}{\partial \boldsymbol{\theta}_i}\mathbf{A}^\intercal + \mathbf{A}\mathbf{P}_{t-1|t-1}\frac{\partial \mathbf{A}^\intercal}{\partial \boldsymbol{\theta}_i} + \frac{\partial \mathbf{Q}}{\partial \boldsymbol{\theta}_i}, \tag{4.39}$$

where the updated derivatives of $\hat{\mathbf{f}}_{t|t}$ and $\mathbf{P}_{t|t}$ with respect to the vector $\boldsymbol{\theta}_i$ are given by:

$$\begin{aligned}
\frac{\partial \hat{\mathbf{f}}_{t|t}}{\partial \boldsymbol{\theta}_i} &= \tfrac{\partial \hat{\mathbf{f}}_{t|t-1}}{\partial \boldsymbol{\theta}_i} + \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\mathbf{v}_t + \mathbf{P}_{t|t-1}\tfrac{\partial \boldsymbol{\Lambda}^\intercal}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\mathbf{v}_t \\
&\quad -\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\tfrac{\partial \mathbf{W}_t^{-1}}{\partial \boldsymbol{\theta}_i}\mathbf{v}_t + \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{v}_t}{\partial \boldsymbol{\theta}_i} \\
&= \tfrac{\partial \hat{\mathbf{f}}_{t|t-1}}{\partial \boldsymbol{\theta}_i} + \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\mathbf{v}_t + \mathbf{P}_{t|t-1}\tfrac{\partial \boldsymbol{\Lambda}^\intercal}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\mathbf{v}_t \\
&\quad -\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{W}_t}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\mathbf{v}_t + \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{v}_t}{\partial \boldsymbol{\theta}_i},
\end{aligned} \tag{4.40}$$

and

$$\begin{aligned}
\frac{\partial \mathbf{P}_{t|t}}{\partial \boldsymbol{\theta}_i} &= \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i} - \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\tfrac{\partial \boldsymbol{\Lambda}^\intercal}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} \\
&\quad - \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\tfrac{\partial \mathbf{W}_t^{-1}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \boldsymbol{\Lambda}}{\partial \boldsymbol{\theta}_i}\mathbf{P}_{t|t-1} \\
&\quad -\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i} \\
&= \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i} - \tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\tfrac{\partial \boldsymbol{\Lambda}^\intercal}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} \\
&\quad + \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{W}_t}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\mathbf{P}_{t|t-1} \\
&\quad -\mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\tfrac{\partial \boldsymbol{\Lambda}}{\partial \boldsymbol{\theta}_i}\mathbf{P}_{t|t-1} - \mathbf{P}_{t|t-1}\boldsymbol{\Lambda}^\intercal\mathbf{W}_t^{-1}\boldsymbol{\Lambda}\tfrac{\partial \mathbf{P}_{t|t-1}}{\partial \boldsymbol{\theta}_i}.
\end{aligned} \tag{4.41}$$

The required initial derivatives $\partial\hat{\mathbf{f}}_{1|0}/\partial\boldsymbol{\theta}_i$ and $\partial\mathbf{P}_{1|0}/\partial\boldsymbol{\theta}_i$ can be calculated based on the given known initial predicted estimate of the state $\hat{\mathbf{f}}_{1|0}$ and the initial predicted covariance matrix $\mathbf{P}_{1|0}$. If the initial value is independent to $\boldsymbol{\theta}_i$, then the derivative is zero.

For now, most of the derivatives required for the computation of the first derivative of the log-likelihood function have been derived, except for the first-order derivative of the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Lambda}$, $\mathbf{A}$, $\mathbf{Q}$ and $\mathbf{H}$. These derivatives are straightforward, as they simply depend on the structure of the parameters.

After driving all derivatives needed, we can write the expression for the score vector $S(\boldsymbol{\theta})$ as follows:

$$S(\boldsymbol{\theta}) = \frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{4.42}$$

where

$$\frac{\partial \ell(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = \sum_{t=1}^{T} \left\{ -\frac{1}{2} tr\left[\left[\mathbf{W}_t^{-1} \frac{\partial \mathbf{W}_t}{\partial \boldsymbol{\theta}_i} \right] \left[\mathbf{I} - \mathbf{W}_t^{-1} \mathbf{v}_t \mathbf{v}_t^{\intercal} \right] \right] - \left(\frac{\partial \mathbf{v}_t}{\partial \boldsymbol{\theta}_i} \right)^{\intercal} \mathbf{W}_t^{-1} \mathbf{v}_t \right\},$$

for $i = 1, 2, \ldots, 5$. The book of Harvey (1990) [14] provides a simplified expression of the Fisher's information, and the Fisher's information is just the expectation of the observed information. The expression in equation 4.48 can then be simplified based on the result given by Harvey [14].

4.2.2 Estimation via the Newton–Raphson method

The gradient descent algorithm uses gradient information for optimization, we now introduce another gradient-based method, which uses the information about the derivative of the gradient, the curvature information. This approach is called the Newton–Raphson method, which is named after Isaac Newton and Joseph Raphson and also known as the Newton's method. It is a algorithm for finding the root of the derivative of a twice-differentiable function], and it is based on successive approximations to the root by using Taylor's theorem.

The algorithm starts with a initial guess of the variable $\mathbf{x}$, denoted by $\mathbf{x}_0$. By using Taylor's theorem, the approximation of the function $f(\mathbf{x})$ has the following expression:

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^{\intercal} H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0), \tag{4.43}$$

where $\nabla f(\mathbf{x}_0)$ is the gradient of the function $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_0$, and $H_f(\mathbf{x}_0)$ is the matrix of second-order partial derivatives of the function $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_0$,

which is also known as the Hessian matrix. Differentiating the right-hand side of the equation (4.38) with respect to $\mathbf{x}-\mathbf{x}_0$ and setting it to zero vector, the following can be obtained:

$$\nabla f(\mathbf{x}_0) + H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) = \mathbf{0}. \tag{4.44}$$

Based on equation 4.44, a better approximation of $\mathbf{x}$ can be calculated, and it is used as an initial value in the next iteration. Iterations are run until the approximation with sufficient accuracy is found. The algorithm takes the following form of iteration:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - [H_f(\mathbf{x}^{(n)})]^{-1}\nabla f(\mathbf{x}^{(n)}), \quad n \geqslant 0, \tag{4.45}$$

In our case for maximizing the log-likelihood, we need to find an approximation of $\boldsymbol{\theta}$ that can let the score function equal to zero:

$$S(\boldsymbol{\theta}) = \mathbf{0}. \tag{4.46}$$

In addition, the negative Hessian matrix of the log-likelihood is known as the observed Fisher information, denoted by $\mathcal{J}(\cdot)$. We then rewrite the iteration expression of Newton's method for our case:

$$\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} + [\mathcal{J}(\boldsymbol{\theta}^{(n)})]^{-1}S(\boldsymbol{\theta}^{(n)}). \tag{4.47}$$

where $\boldsymbol{\theta}^{(n)}$ is the approximation of $\boldsymbol{\theta}$ at iteration n, and $\boldsymbol{\theta}^{(n+1)}$ is a better approximation obtained at iteration $n+1$ compared with $\boldsymbol{\theta}^{(n)}$.

The score can be calculated by using the same method presented in the subsection of the gradient descent, and the observed information can be calculated by definition once the second derivatives of the log-likelihood are known. The component of the second derivative at time t, $\ell_t(\boldsymbol{\theta})$, is obtained by further differentiating equation 4.35 with respect to $\boldsymbol{\theta}_j$, $j = 1, 2, \ldots, 5$:

$$\begin{aligned}\frac{\partial^2 \ell_t(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j} \quad &= -\tfrac{1}{2} tr\left[\left(\partial\big[\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{W}_t}{\partial \boldsymbol{\theta}_i}\big]/\partial \boldsymbol{\theta}_j\right)\big[\mathbf{I} - \mathbf{W}_t^{-1}\mathbf{v}_t\mathbf{v}_t^\intercal\big]\right] \\ &\quad -\tfrac{1}{2} tr\left[\big[\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{W}_t}{\partial \boldsymbol{\theta}_i}\big]\left(\partial\big[\mathbf{I} - \mathbf{W}_t^{-1}\mathbf{v}_t\mathbf{v}_t^\intercal\big]/\partial \boldsymbol{\theta}_j\right)\right] \\ &\quad -\tfrac{\partial^2 \mathbf{v}_t^\intercal}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}\mathbf{W}_t^{-1}\mathbf{v}_t - \tfrac{\partial \mathbf{v}_t^\intercal}{\partial \boldsymbol{\theta}_i}\tfrac{\partial \mathbf{W}_t^{-1}}{\partial \boldsymbol{\theta}_j}\mathbf{v}_t - \tfrac{\partial \mathbf{v}_t^\intercal}{\partial \boldsymbol{\theta}_i}\mathbf{W}_t^{-1}\tfrac{\partial \mathbf{v}_t}{\partial \boldsymbol{\theta}_j}.\end{aligned} \tag{4.48}$$

for $i = 1, 2, \ldots, 5$. The book of Harvey (1990) [14] provides a simplified expression of the Fisher's information, and the Fisher's information is just the expectation of the observed information. The expression in equation 4.48 can then be simplified based on the result given by Harvey [14].

4.2.3 Estimation via the EM Algorithm

The Expectation-Maximization (EM) algorithm, introduced by Dempster, Laird, and Rubin (1977) [7], is the third approach that we will present in this section, which is different from the gradient-based methods. The EM algorithm is a recursive method for computing maximum likelihood and it performs two steps in each iteration: an expectation (E) step and a maximization (M) step. On the E step, the expectation of the log-likelihood will be calculated by using the current estimates for the unknown parameters. On the M step, the parameters that maximize the expected value of the log-likelihood on the E step will be found and extracted, and then these updated parameters will be used as the current estimates on the E step of next iteration. Both E step and M step are repeated alternately until the difference between two successive values of the log-likelihood falls within the targeted range.

The EM algorithm can be used for parameter estimation in the space-state models, and Shumway and Stoffer (1982) [23] and Watson and Engle (1983) [27] have proposed the way to use it. Hence we can apply the EM algorithm for estimating the parameters of the DNS model expressed in the state-space form. Now we will explain how this algorithm works in details.

To start the EM algorithm for our case, we need to provide some relevant inputs, which are the observations of yields $\mathbf{y}$, an initial predicted estimate of the state $\mathbf{f}_{1|0}$, a initial predicted covariance matrix $\mathbf{P}_{1|0}$, initial guess of the unknown parameters, denoted by $\boldsymbol{\theta}^{(0)}$, the maximum number of iterations allowed and the targeted percentage of the log-likelihood difference between two successively iterations. Described below are the two main stages of the algorithm:

- **Modified E step:**
  The E step for our case should not be exactly the same as the one involved in the original expression of EM algorithm. In a typical E step, the expected value of the log-likelihood is calculated under the current estimate of parameters, which can be expressed as:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) = \mathbb{E}_{\mathbf{z}|\mathbf{h},\boldsymbol{\theta}^{(n)}}[\ell(\boldsymbol{\theta};\mathbf{z},\mathbf{h})] = \int_{\boldsymbol{\Omega}} \ell(\boldsymbol{\theta};\mathbf{z},\mathbf{h}) g(\mathbf{z}|\mathbf{h},\boldsymbol{\theta}^{(n)}) d\mathbf{z}, \quad (4.49)$$

  where $\mathbf{z}$ is a sequence of yield curves, $\mathbf{z} = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$, $\mathbf{h}$ is a sequence

of states, $\mathbf{h} = [\mathbf{f}_1, \ldots, \mathbf{f}_T]$, $\boldsymbol{\Omega}$ is the support of the density function of $\mathbf{z}$, $\boldsymbol{\theta}^{(n)}$ is the current estimate of parameters, and the function $g(\cdot)$ is the conditional density of $\mathbf{z}$ given $\mathbf{h}$ and $\boldsymbol{\theta}^{(n)}$. However, the state sequence $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_T$ in our state-space model is unknown and unobservable in reality. On our E step, instead of calculating the expectation in equation 4.49, we extract the estimated state sequence optimally. For the given observations $\mathbf{y}$ and the current estimate of the unknown parameters $\boldsymbol{\theta}^{(n)}$, this extraction can be done by using the Kalman filter iteratively, starting with the provided $\mathbf{f}_{1|0}$ and $\mathbf{P}_{1|0}$. We denote the extracted state sequence as $\mathbf{f}_1^{(n)}, \mathbf{f}_2^{(n)}, \ldots, \mathbf{f}_T^{(n)}$.

- **Modified M step;**
  Like the M step in a typical EM algorithm, we want to find the updated estimate of the unknown parameters on our M step. The updated estimate $\boldsymbol{\theta}^{(n+1)}$ has the following expression:

$$\boldsymbol{\theta}^{(n+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \quad Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}). \tag{4.50}$$

  For our state-space model, this step is straightforward once we know the estimated state sequence from the previous modified E step. Firstly, based on equation 3.11, the new estimates of $\boldsymbol{\Lambda}$ and $\mathbf{H}$, denoted by $\hat{\boldsymbol{\Lambda}}^{(n+1)}$ and $\hat{\mathbf{H}}^{(n+1)}$, can be obtained by running the seemingly unrelated regression, which is defined as follows:

$$\mathbf{y}_t = \boldsymbol{\Lambda}\hat{\mathbf{f}}_t^{(n)} + \boldsymbol{\epsilon}_t, \quad t = 1, 2, \ldots, T, \tag{4.51}$$

  where $\mathbf{y}_t$ is the tth column of $\mathbf{y}$, $\mathbf{y}_t$ and $\boldsymbol{\epsilon}_t$ are $N \times 1$ vectors, $\hat{\mathbf{f}}_t^{(n)}$ is a 3×1 vector, $\boldsymbol{\Lambda}$ is a $N \times 3$ matrix and the covariance matrix of $\boldsymbol{\epsilon}_t$ is the $N \times N$ matrix $\mathbf{H}$. The expression of the updated estimate of λ is given by:

$$\hat{\lambda}^{(n+1)} = \max_{0 \le \lambda \le 1} \sum_{t=1}^{T} \sum_{n=1}^{N} \left[y_t(\tau_n) - \boldsymbol{\Lambda}_{n\cdot} \hat{\mathbf{f}}_t^{(n)} \right]^2, \tag{4.52}$$

  where $\boldsymbol{\Lambda}_{n\cdot}$ is the nth nth row of the matrix $\boldsymbol{\Lambda}$. The derivative of the

summation term in equation 4.52 with respect to λ is:

$$\sum_{t=1}^{T}\sum_{n=1}^{N}\Bigg\{\frac{exp(-\lambda\tau_n)\big(\lambda\tau_n - exp(\lambda\tau_n)+1\big)}{\lambda^2\tau_n}\hat{S}_t^{(n)} + \Bigg[\frac{exp(-\lambda\tau_n)\big(\lambda\tau_n - exp(\lambda\tau_n)+1\big)}{\lambda^2\tau_n} + \tau_n exp(-\lambda\tau_n)\Bigg]\hat{C}_t^{(n)}\Bigg\}. \tag{4.53}$$

The optimal λ can be found by letting equation 4.53 equal to 0 and then solve the equation. However, it is difficult to solve it analytically. Simplifying the problem, we use numerical method to deal with it, finding λ that maximizes the least squares by grid search. As discussed in Section 3.1.2, the suggested value of λ is small and has to be positive, and based on the data in the literature and our own data, we have reason to believe that it should be greater than 0 and smaller than 0.1 given that the maturity is measured in months. We therefore set 0.01 and 0.2 as the initial grid limits in the implementation of the algorithm. Given grid spacing of 0.001, the grid will contain 191 grid points. The updated estimate $\hat{\mathbf{\Lambda}}^{(n+1)}$ can then be obtained by substituting $\hat{\lambda}^{(n+1)}$ into the expression of the parameter matrix in measurement equation shown in equation 3.11. Furthermore, the estimate $\hat{\mathbf{H}}^{(n+1)}$ can be updated by finding the covariance

$$\hat{\mathbf{H}}^{(n+1)} = \frac{1}{T-1}\sum_{t=1}^{T}\Bigg\{\big[\mathbf{y}_t - \hat{\mathbf{\Lambda}}^{(n+1)}\hat{\mathbf{f}}_t^{(n)}\big]\big[\mathbf{y}_t - \hat{\mathbf{\Lambda}}^{(n+1)}\hat{\mathbf{f}}_t^{(n)}\big]^{\intercal}\Bigg\}. \tag{4.54}$$

Secondly, the updated estimate of $\hat{\boldsymbol{\mu}}^{(n+1)}$ can be easily found by taking an average of the current estimates of each factor over without constant term, which has the following form:

$$\hat{\boldsymbol{\mu}}^{(n+1)} = \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{f}}_t^{(n)}. \tag{4.55}$$

The remaining updated estimates of $\mathbf{A}$, and $\mathbf{Q}$, denoted by $\mathbf{A}^{(n+1)}$, $\boldsymbol{\mu}^{(n+1)}$ and $\mathbf{Q}^{(n)}$, can be gotten based on equation 3.12, by running the vector autoregression of first order, which is defined as follows:

$$\hat{\mathbf{f}}_t^{(n)} - \hat{\boldsymbol{\mu}}^{(n+1)} = \mathbf{A}\big(\hat{\mathbf{f}}_{t-1}^{(n)} - \hat{\boldsymbol{\mu}}^{(n+1)}\big) + \boldsymbol{\eta}_t, \quad t = 1, 2, \ldots, T. \tag{4.56}$$

The estimate of the autoregressive matrix $\mathbf{A}$ is the updated $\hat{\mathbf{A}}^{(n+1)}$, and the estimated covariance matrix of the vector of serially uncorrelated innovations $\boldsymbol{\eta}_t$ is the updated $\hat{\mathbf{Q}}^{(n+1)}$.

The updated parameter estimates $\hat{\boldsymbol{\mu}}^{(n+1)}$, $\hat{\mathbf{A}}^{(n+1)}$, $\hat{\boldsymbol{\Lambda}}^{(n+1)}$, $\hat{\mathbf{Q}}^{(n+1)}$ and $\hat{\mathbf{H}}^{(n+1)}$ are then used to find the state extraction $\hat{\mathbf{f}}_1^{(n+1)}, \hat{\mathbf{f}}_2^{(n+1)}, \ldots, \hat{\mathbf{f}}_T^{(n+1)}$ in the next iteration. These two stages will be repeated until the difference of change between the parameters update becomes small enough as defined beforehand.

Chapter 5

Simulation Study for Comparing Estimation Procedures

As discussed in the previous chapter, applying different numerical methods together with the Kalman filter can constitute a procedure to estimate both the latent factors and the parameters in the state-space form of the DNS model. This chapter will discuss and compare the performance of the gradient descent method and the EM algorithm. The state-space model is assumed to be linear Gaussian, and the estimation procedures we discussed all rely on this assumption. We therefore use synthetic data, which is more appropriate compared with real data for the purpose of simulation study. This chapter is divided into four sections: test case, data preparation, results and discussion.

5.1 Test Case

We test both gradient descent and EM algorithm on the diagonal case. We restrict the transition matrix $\mathbf{A}$ and the covariance matrices $\mathbf{Q}$ and $\mathbf{H}$ to be diagonal, setting their off-diagonal entries all equal to zero. These settings imply that not only the the deviations of the observed yields across maturities are uncorrelated, but also the three factors are independent to each other and

governed by their own dynamics. The transition equation shown in equation 3.10 is now replaced by:

$$\begin{bmatrix} L_t - \mu^L \\ S_t - \mu^S \\ C_t - \mu^C \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} l_{t-1} - \mu^L \\ s_{t-1} - \mu^S \\ c_{t-1} - \mu^C \end{bmatrix} + \begin{bmatrix} \eta_t^L \\ \eta_t^S \\ \eta_t^C \end{bmatrix}. \tag{5.1}$$

On this case, the set of unknowns $\boldsymbol{\theta}$ has 20 free parameters in total. The mean factor vector $\boldsymbol{\mu}$ contains 3 parameters (μ^L, μ^S, μ^C), the measurement matrix $\mathbf{\Lambda}$ contains one parameter (λ), the transition matrix $\mathbf{A}$ contains 3 free parameters (a_{11}, a_{22}, a_{33}) and the transition covariance matrix also $\mathbf{Q}$ contains 3 parameters (q_{11}, q_{22}, q_{33}). Given that there are ten maturities, the measurement covariance matrix $\mathbf{H}$ contains 10 free parameters $(h_{11}, \ldots, h_{1010})$. We use index to represent each free parameter $\boldsymbol{\theta}_i$. The indexes are list in Table 5.1.

Index	1	2	3	4	5	6	7	8	9	10
$\boldsymbol{\theta}_i$	μ^L	μ^S	μ^C	λ	a_{11}	a_{22}	a_{33}	q_{11}	q_{22}	q_{33}
Index	11	12	13	14	15	16	17	18	19	20
$\boldsymbol{\theta}_i$	h_{11}	h_{22}	h_{33}	h_{44}	h_{55}	h_{66}	h_{77}	h_{88}	h_{99}	h_{1010}

Table 5.1: Indexes of the free parameters in the set of unknowns $\boldsymbol{\theta}$

5.2 Synthetic Data Preparation

Twenty sets of synthetic data are used in the case study, and each set of data contains synthetic yields at 100 discrete points in time for ten different maturities. These sets are generated by using the same set of parameters, $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \mathbf{\Lambda}, \mathbf{A}, \mathbf{Q}, \mathbf{H}\}$. These parameters are determined by a set of fixed values, namely the seed values. Each seed value corresponds to one free parameter $\boldsymbol{\theta}_i$, and it is set randomly within a appropriated range based on the observations in the literature. For example, the seed value of the decay rate λ must be positive, and based on the observations in the literature and our observed data, it is reasonable to set it less than 1 given that the maturities are measured in months. To generate such set of synthetic yields, we firstly generate a set of synthetic factors by using the given parameters

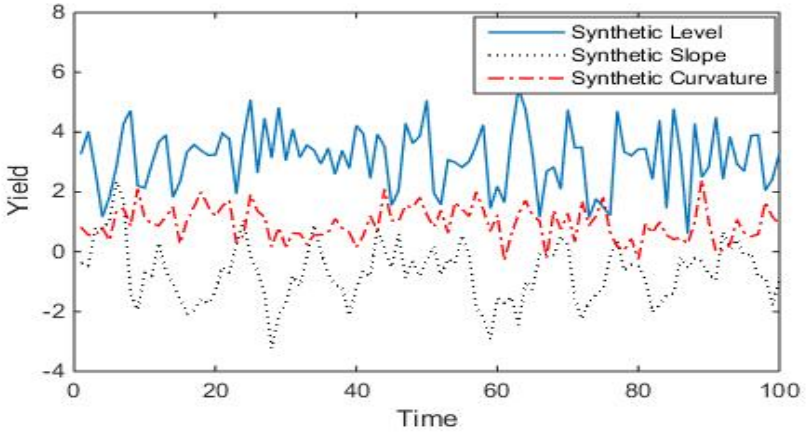

Figure 5.1: Plots of synthetic level, slope and curvature factors

based on the transition equation shown in equation 3.13. The initial state $\mathbf{f}_0$ is required, and we set it equal to the seed values for $\boldsymbol{\mu}$. After that, we use this set of synthetic factors and the given parameters to generate a set of synthetic yields based on the measurement equation shown in equation 3.11. Figure 5.1 and 5.2 shows the plots of the synthetic factors and the synthetic yields respectively. As shown in the figures, the synthetic data generated are within the appropriate ranges. Moreover, in the later discussion, we regard the test on one set of synthetic data as one run.

5.3 Results

First of all, we find that the gradient descent is extremely sensitive to the initial values of the parameters and the value of step size. Given that the initial values of the parameters are set ideally, close to the seed values, the gradient descent converges only if the value of step size is sufficiently small. After trying different values, we find that a value of 10^{-4} is appropriate one on the diagonal case. Such small value of the step size results in a very low speed of convergence, especially on the case that the initial values of the parameters are far away from the seed values.

In EM algorithm, we use the grid as defined in Section 4.2.3 to find the

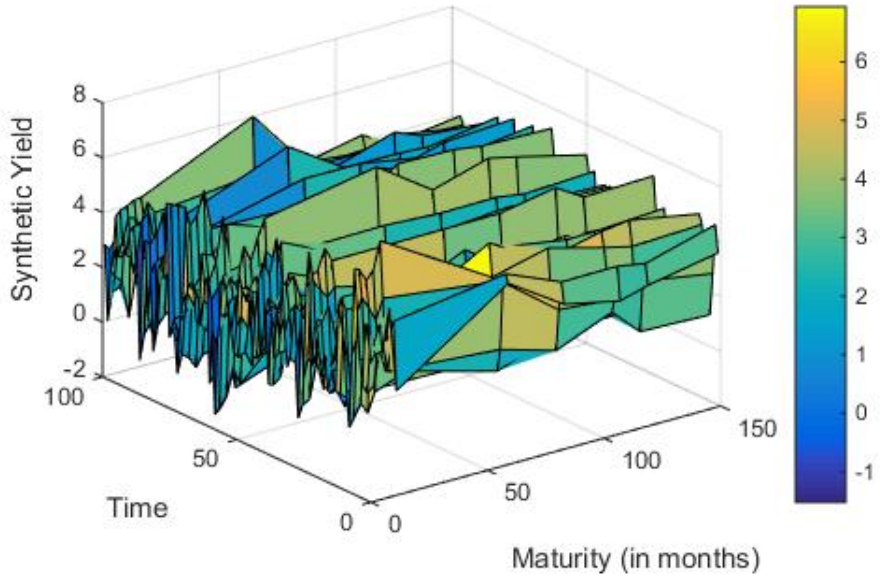

Figure 5.2: Three-dimensional plot of synthetic yields

best estimate of λ in each iteration. It is worth mentioning that the grid search within the EM algorithm does slow down the speed of convergence, but this impact is greatly reduced by using matrix operations instead of loops. As expected, the EM algorithm converges faster than the gradient descent regardless of the initial values of the parameters, although its rate of convergence is slow as well.

We now discuss their performance in term of accuracy. Both methods provide reasonable results. The sum of squared errors (SSE) for each parameter $\boldsymbol{\theta}_i$ is good measure of accuracy, which is defined as the sum of the squares of the differences between the parameter estimate and the corresponding seed value. According to the box plots of SSE shown in Figures 5.3-5.5, we can see that the gradient descent gives very small values of SSE for all parameters and these values do not change a lot when different sets of synthetic data are used, while the EM algorithm gives large values for some parameters on some sets of data, such as the parameter $\boldsymbol{\theta}_7$ (a_{33}). Hence the gradient methods is more stable in convergence and has more accurate estimates. By using the estimated parameter, the estimates of the level, slope and curvature factors can be obtained. The relevant plots are given in Figures 5.6-5.11. We can see

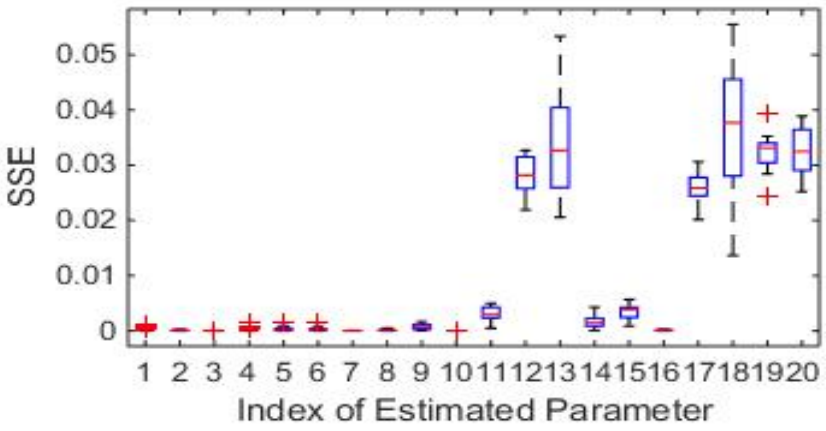

Figure 5.3: Box plots of sum of squared errors given by Gradient descent

that both gradient descent and EM algorithm provide good estimates to the level and slope with small uncertainly. However, both of perform worse on the curvature estimation. The EM algorithm fails to estimate it accurately as shown in Figure. The gradient method is better, providing good estimate although the 95% confidence interval of it is large.

The estimates and the total sum of the SSE of all the parameters provided by the two methods in the first five runs are summarized in Tables A.1 and A.2 respectively in Appendix.

5.4 Discussion

It is not surprising that using numerical methods for maximum likelihood estimation of the state-space framework of the DNS model may be a challenging task, because the log-likelihood, as shown in equation 4.28, is a highly non-linear function and may have many local optimum. However, based on our simulation study, we find that this task is achievable by either using the gradient descent method or the EM algorithm. Given appropriate initial settings, we conclude that the gradient descent method is more favourable than the EM algorithm in term of accuracy and stability.

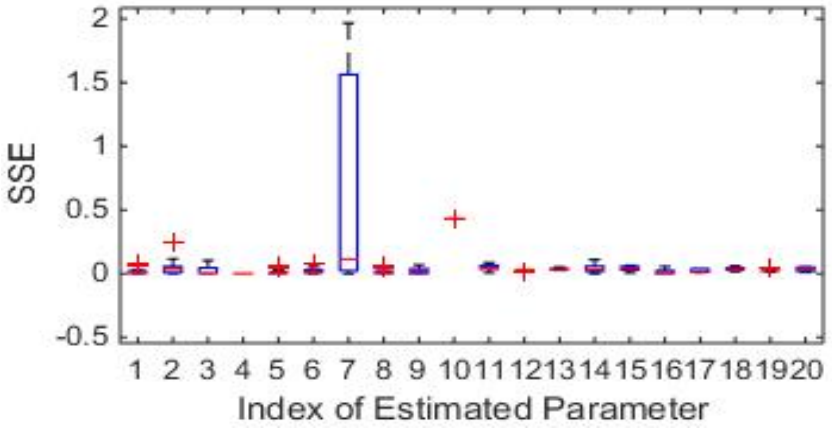

Figure 5.4: Box plots of sum of squared errors given by EM algorithm

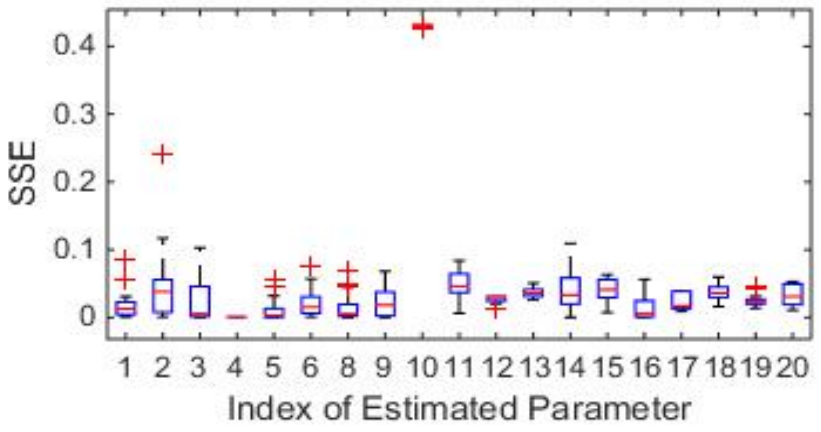

Figure 5.5: Box plots of sum of squared errors given by EM algorithm (except for the 7th parameter

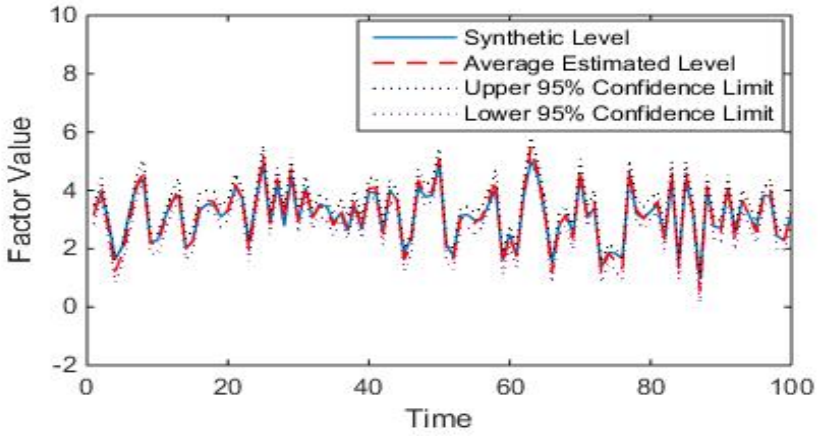

Figure 5.6: Synthetic level and estimated level by gradient descent

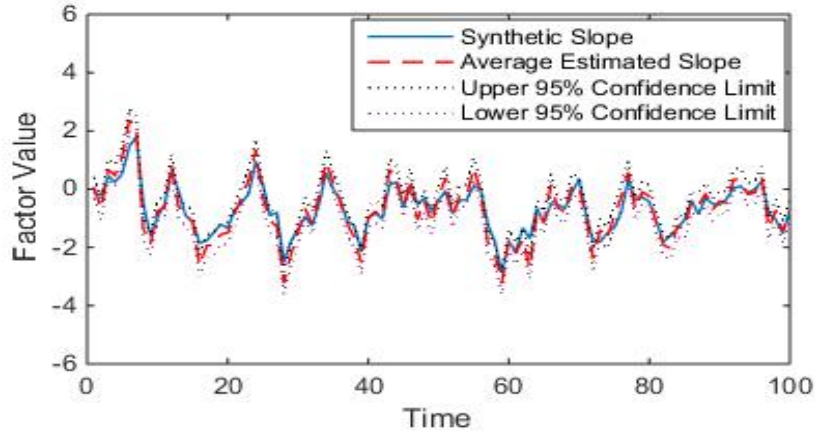

Figure 5.7: Synthetic slope and estimated slope by gradient descent

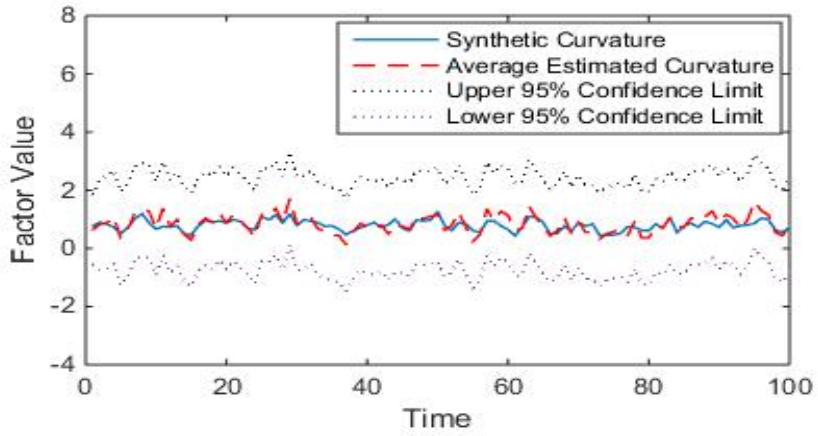

Figure 5.8: Synthetic curvature and estimated curvature by gradient descent

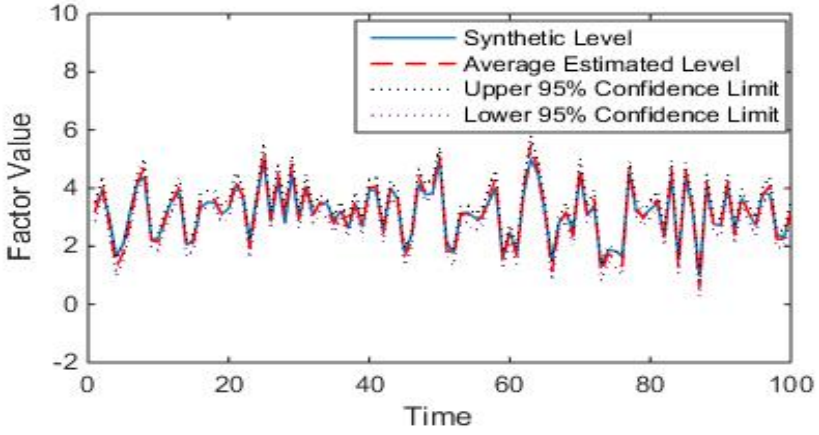

Figure 5.9: Synthetic level and estimated level by EM algorithm

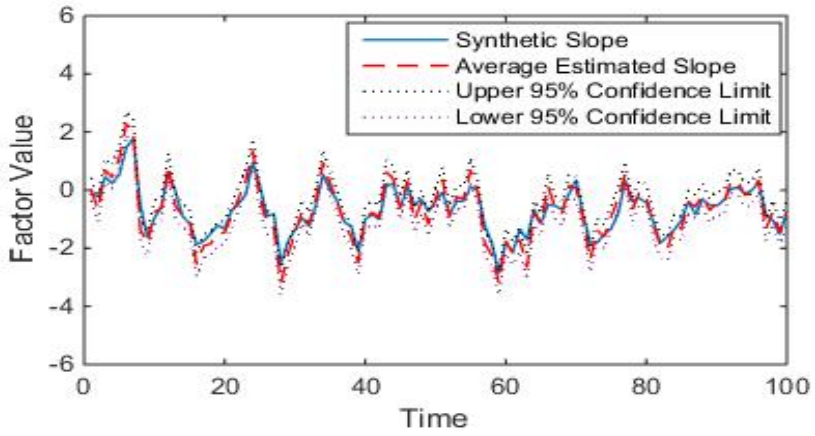

Figure 5.10: Synthetic slope and estimated slope by EM algorithm

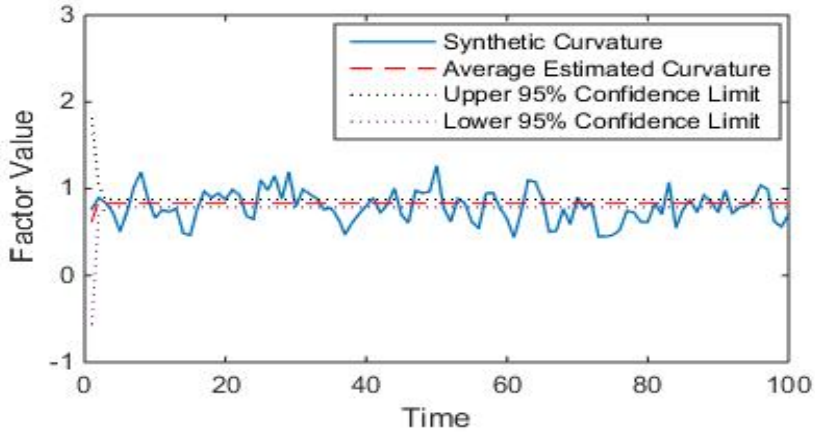

Figure 5.11: Synthetic curvature and estimated curvature by EM algorithm

Chapter 6

Application

In this chapter, we present a real world application of the dynamic Nelson-Siegel model, modelling the yields of the U.S. Treasury bills and bonds. The U.S. Treasury yield curve is one of the most important yield curve both from theoretical and practical perspectives as discussed in Gurkaynak et al. (2007) [13]. It is commonly used in interest rate risk management, design of the interest rate hedging products, and considered as a benchmark for pricing other securities and assets in the market, because the U.S. Treasury market is the largest and most liquid government securities market in the world, and the U.S. Treasury securities are almost default-free. We build a yields-only model for the data. The model estimation is achieved by using the gradient descent method together with the Kalman filter. Unlike the simulation study in Chapter 5, we assume that the transition matrix $\mathbf{A}$ and covariance matrices $\mathbf{Q}$ and $\mathbf{H}$ in the state-space form of the DNS model are all non-diagonal. The chapter will start with the data section, which is followed by the results and discussion section.

6.1 Real Data Example

The data for the U.S. Treasury yields is taken from the U.S. Treasury database on Quandl, and the data is validated by the U.S. Treasury Department. We use end-of-month yields of bills and bonds with maturities of 3, 6, 12, 24, 36, 60, 84 and 120 months. The sample period is from January 1990 to August

2015. Early in Chapter 1, we have plotted the data in three dimensions in Figure 1.1. The descriptive statistics of the data is given in Table 6.1.

Maturity (months)	Mean	Std. Dev.	$\hat{\rho}(1)$	$\hat{\rho}(12)$
3	3.0277	2.3428	0.9857	0.7176
6	3.1603	2.3786	0.9863	0.7212
12	3.2938	2.3780	0.9864	0.7366
24	3.6170	2.3804	0.9852	0.7599
36	3.8523	2.3069	0.9840	0.7695
60	4.2828	2.1303	0.9824	0.7779
84	4.6083	1.9936	0.9812	0.7734
120	4.8619	1.8351	0.9794	0.7634

Table 6.1: Descriptive Statistics of the given U.S. Treasury yields data

6.2 Results and Discussion

The estimates of the transition matrix $\mathbf{A}$ and the factor means $\boldsymbol{\mu}$ are given in Table 6.2. The values in the diagonal entries of $\mathbf{A}$ are 0.7858, 0.6278 and 0.6584, which are much larger compared with the values in the off-diagonal entries. This indicates that the value of each factor is mainly governed by its own dynamics. However, the off-diagonal values are not small enough to be ignored, which implies the possibility of correlation between factors. The estimate of the covariance matrix of the process noises $\mathbf{Q}$ is given in Table 6.3.

Furthermore, the estimates of the level, slope and curvature factor over time are plotted in Figure 6.1. It is clearly show that the estimated level factor has a deceasing trend, and the estimated slope and curvature have similar shape. In order to see how well the model performs, we plot both the estimated yield and the actual yield over time in one figure for each maturity. The plots are shown in Figure 6.2-6.9. As we can see from the plots, the model fits the data very well for the bonds with maturities of 10-year, 7-year and 3-month. For the rest of the bonds, the model is still acceptable, because it captures main movements of the yields.

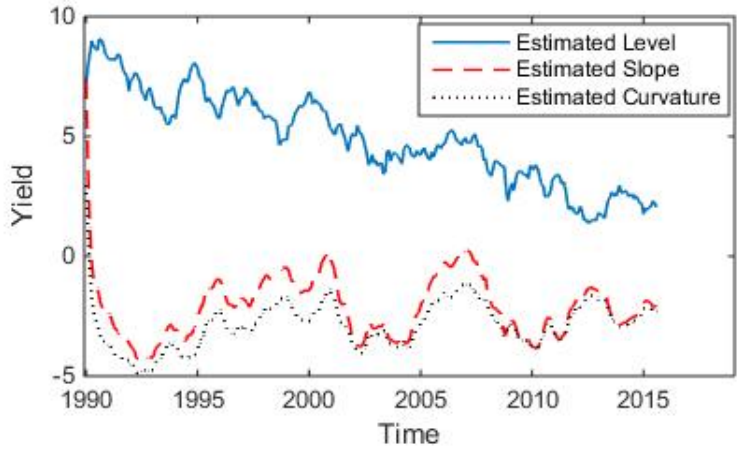

Figure 6.1: Synthetic curvature and estimated curvature for EM algorithm on non-diagonal case

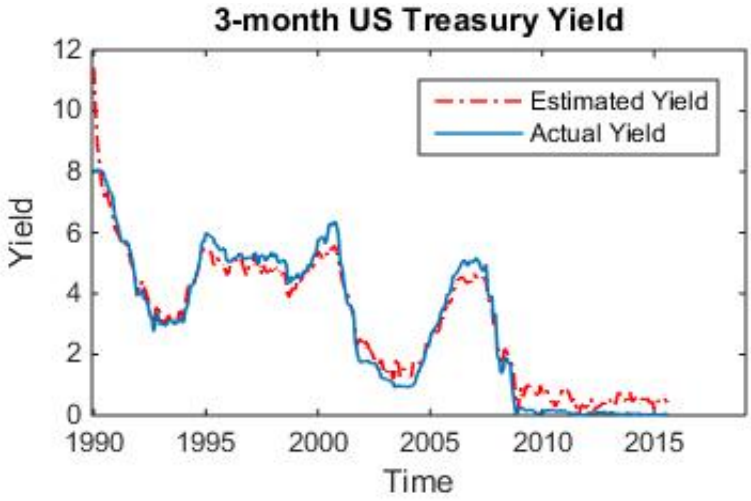

Figure 6.2: Plots of actual and estimated yield of 3-month US Treasury

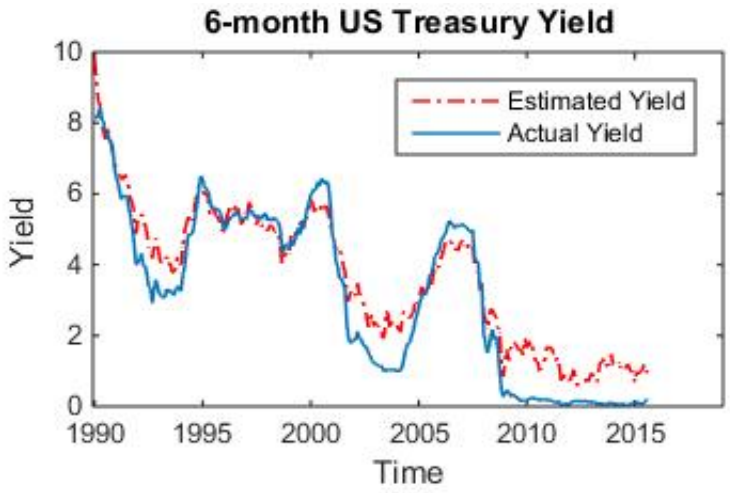

Figure 6.3: Plots of actual and estimated yield of 6-month US Treasury

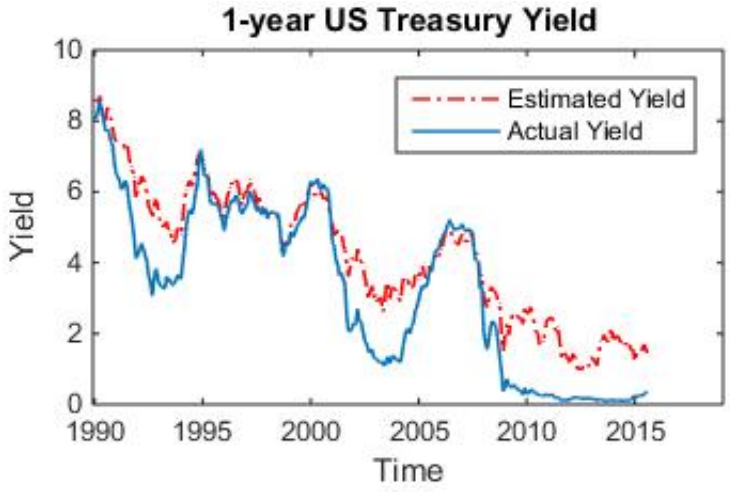

Figure 6.4: Plots of actual and estimated yield of 1-year US Treasury

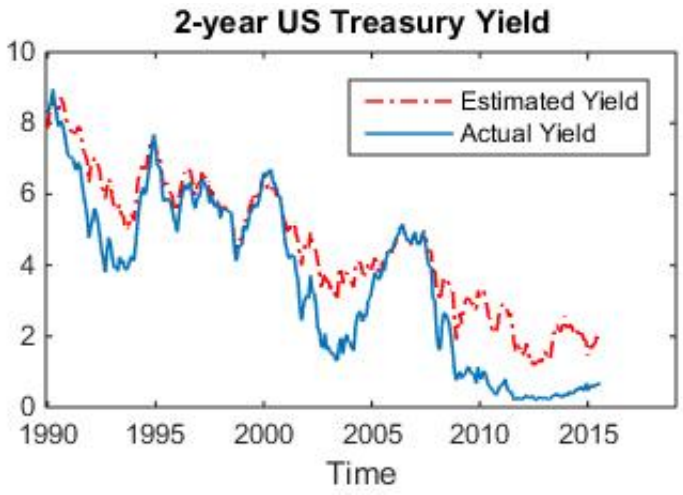

Figure 6.5: Plots of actual and estimated yield of 2-year US Treasury

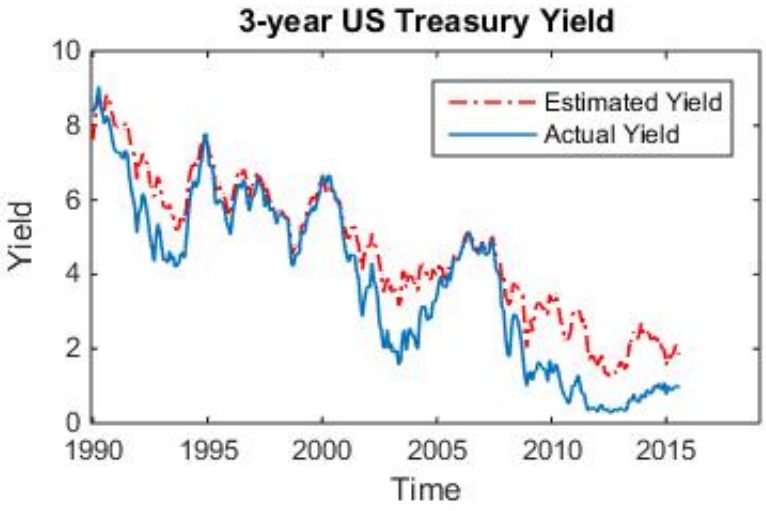

Figure 6.6: Plots of actual and estimated yield of 3-year US Treasury

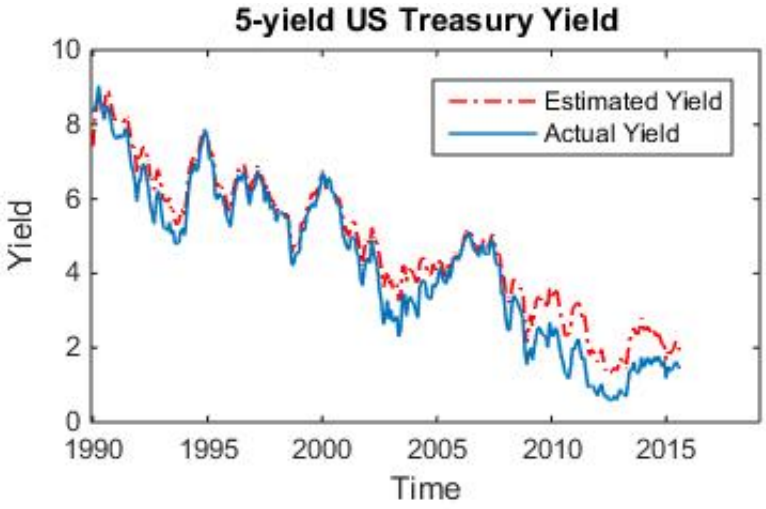

Figure 6.7: Plots of actual and estimated yield of 5-year US Treasury

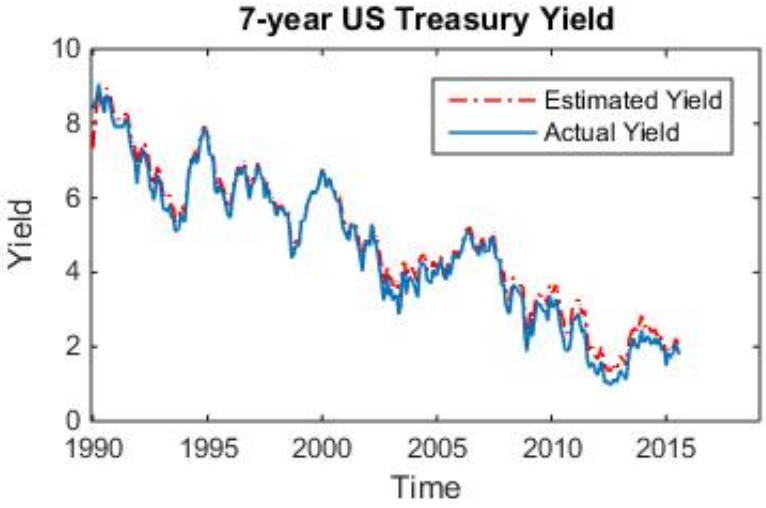

Figure 6.8: Plots of actual and estimated yield of 7-year US Treasury

$\hat{\mathbf{A}}$	L_{t-1}	S_{t-1}	C_{t-1}	$\hat{\boldsymbol{\mu}}$
L_t	0.7858	-0.1345	-0.1278	0.1817
S_t	-0.1822	0.6278	0.0714	0.9644
C_t	-0.1503	0.0983	0.6584	0.2804

Table 6.2: Estimated transition matrix of the state vector ($\hat{\mathbf{A}}$) and their means ($\hat{\boldsymbol{\mu}}$)

$\hat{\mathbf{Q}}$	L_t	S_t	C_t
L_t	0.8674	0.1977	-0.0036
S_t	-0.1977	0.5899	0.3284
C_t	-0.0036	0.3284	0.0865

Table 6.3: Estimated covariance matrix of the state vector ($\hat{\mathbf{Q}}$)

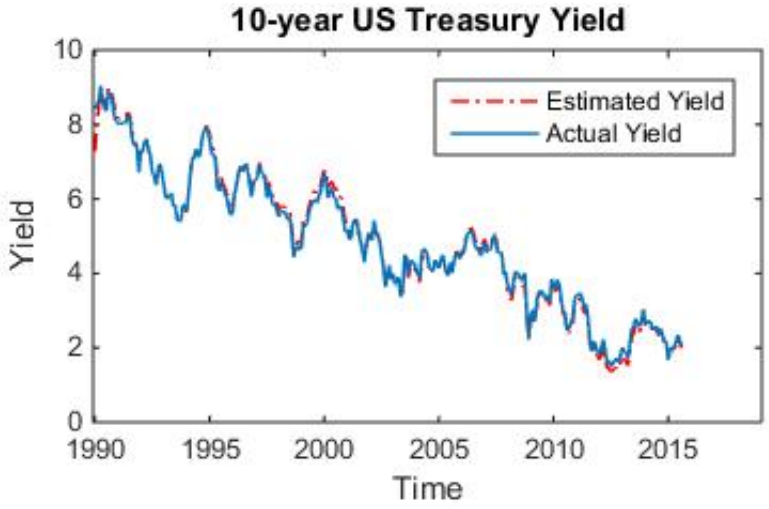

Figure 6.9: Plots of actual and estimated yield of 10-year US Treasury

Chapter 7

Conclusion

The dissertation studies the dynamic Nelson-Siegel approach for yield curve modelling and focuses on the model estimation based on the state-space structure of the DNS model. The motivation to use the state-space framework is that the DNS model can be easily extended to incorporate macroeconomic variables under this framework. Following the work of Diebold et. al (2006) [10], we have provided a general form of the extended model that can include a set of macroeconomic variables.

On model estimation, we have derived the optimal estimation of the latent factors in a recursive fashion and associated it to the well-known Kalman filter algorithm [18]. We have also completed the implementation of the gradient-based methods and the EM algorithm for maximum likelihood estimation of the unknown parameters in the state-space model. After that, we have performed a simulation study demonstrating that the gradient descent method is a better approach on the compared with the EM algorithm in term of the accuracy of the estimation and the stability of the convergence. We have also applied the gradient descent method to estimate the dynamic Nelson-Siegel model of the U.S. Treasury yield curve. The model fits the data quite well, which further demonstrates the appropriateness of the dynamic Nelson-Siegel model and also the practicability of gradient descent method.

The low speed of convergence is significant weakness of our gradient descent method. We believe that it is the constant, extremely small step size that

leads to the low rate. We can foresee that the gradient descent may not work well or take a extremely long time when there are much larger set of parameters in the model. One possibility to speed up the gradient descent would be using the time-varying step size instead of the fixed one. In addition, it would be also interesting to implement the EM algorithm without using the grid search method. Using the grid search method to estimate the rate of decay is a straightforward, however, some smarter and more accurate methods may be available, which would improve the performance of the EM algorithm.

Bibliography

[1] H Peter Boswijk. "Cointegration analysis of the dynamic Nelson-Siegel model using the wild bootstrap". In: *Aenorm* 21.81 (2013), pp. 30–34.

[2] John Y Campbell. "Some lessons from the yield curve". In: (1995).

[3] John Y Campbell. "Stock returns and the term structure". In: *Journal of financial economics* 18.2 (1987), pp. 373–399.

[4] Jens HE Christensen, Francis X Diebold, and Glenn D Rudebusch. "The affine arbitrage-free class of Nelson–Siegel term structure models". In: *Journal of Econometrics* 164.1 (2011), pp. 4–20.

[5] John C Cox, Jonathan E Ingersoll, and Stephen A Ross. "A theory of the term structure of interest rates". In: *Econometrica* 53.2 (1985), pp. 385–407.

[6] Michiel De Pooter, Francesco Ravazzolo, and Dick JC Van Dijk. "Term structure forecasting using macro factors and forecast combination". In: *FRB International Finance Discussion Paper* 993 (2010).

[7] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[8] Francis X Diebold and Canlin Li. "Forecasting the term structure of government bond yields". In: *Journal of econometrics* 130.2 (2006), pp. 337–364.

[9] Francis X Diebold, Canlin Li, and Vivian Z Yue. "Global yield curve dynamics and interactions: a dynamic Nelson–Siegel approach". In: *Journal of Econometrics* 146.2 (2008), pp. 351–363.

[10] Francis X Diebold, Glenn D Rudebusch, and S Boragan Aruoba. "The macroeconomy and the yield curve: a dynamic latent factor approach". In: *Journal of econometrics* 131.1 (2006), pp. 309–338.

[11] Robert Engle and Mark Watson. "A one-factor multivariate time series model of metropolitan wage rates". In: *Journal of the American Statistical Association* 76.376 (1981), pp. 774–781.

[12] Arturo Estrella and Frederic S Mishkin. "The yield curve as a predictor of US recessions". In: *Current Issues in Economics and Finance* 2.7 (1996).

[13] Refet S Gürkaynak, Brian Sack, and Jonathan H Wright. "The US Treasury yield curve: 1961 to the present". In: *Journal of Monetary Economics* 54.8 (2007), pp. 2291–2304.

[14] Andrew C Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[15] David Heath, Robert Jarrow, Andrew Morton, et al. "Bond pricing and the term structure of interest rates: A new methodology for contingent claims valuation". In: *Econometrica* 60.1 (1992), pp. 77–105.

[16] Thomas SY Ho and Sang-Bin Lee. "Term structure movements and pricing interest rate contingent claims". In: *Journal of Finance* (1986), pp. 1011–1029.

[17] John Hull and Alan White. "Pricing interest-rate-derivative securities". In: *Review of financial studies* 3.4 (1990), pp. 573–592.

[18] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of Fluids Engineering* 82.1 (1960), pp. 35–45.

[19] Francis A Longstaff, Sanjay Mithal, and Eric Neis. "Corporate yield spreads: Default risk or liquidity? New evidence from the credit default swap market". In: *The Journal of Finance* 60.5 (2005), pp. 2213–2253.

[20] Charles R Nelson and Andrew F Siegel. "Parsimonious modeling of yield curves". In: *Journal of business* (1987), pp. 473–489.

[21] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. "The matrix cookbook". In: *Technical University of Denmark* 7 (2008), p. 15.

[22] Monika Piazzesi. "Bond yields and the Federal Reserve". In: *Journal of Political Economy* 113.2 (2005), pp. 311–344.

[23] Robert H Shumway and David S Stoffer. "An approach to time series smoothing and forecasting using the EM algorithm". In: *Journal of time series analysis* 3.4 (1982), pp. 253–264.

[24] Gregorio A Vargas. "Macroeconomic Determinants of the Movement of the Yield Curve". In: (2005).

[25] Oldrich Vasicek. "An equilibrium characterization of the term structure". In: *Journal of financial economics* 5.2 (1977), pp. 177–188.

[26] Paul Veerhuis. “Arbitrage free Nelson Siegel yield curve modelling: An application to assess unconventional monetary policy”. In: (2011).

[27] Mark W Watson and Robert F Engle. “Alternative algorithms for the estimation of dynamic factor, mimic and varying coefficient regression models”. In: *Journal of Econometrics* 23.3 (1983), pp. 385–400.

[28] Wei-Choun Yu and Eric Zivot. “Forecasting the term structures of treasury and corporate yields: Dynamic nelson-siegel models evaluation”. In: *International Journal of Forecasting, Forthcoming* (2010).

Appendix A

Appendix

A.1 Tables

Table A.1: Estimates in the first 5 runs by gradient descent on synthetic data

Parameter	Seed	Run 1	Run 2	Run 3	Run 4	Run 5
muL	3.300500	3.287002	3.286026	3.304652	3.270966	3.307992
muS	-0.373100	-0.384877	-0.379122	-0.361052	-0.368788	-0.369788
muC	0.815500	0.812668	0.822000	0.820519	0.813916	0.826176
lambda	0.068900	0.056770	0.073876	0.064927	0.069880	0.091881
a11	0.120200	0.124882	0.150336	0.120859	0.139182	0.090376
a22	0.571200	0.551297	0.551031	0.570048	0.561581	0.610474
a33	0.412800	0.416339	0.413878	0.409435	0.411208	0.413454
q11	0.987000	0.995510	0.991587	0.971882	0.979287	0.994210
q22	0.759600	0.728859	0.731841	0.720402	0.730234	0.746043
q33	0.657200	0.654293	0.654944	0.654062	0.656400	0.655387
h11	0.603900	0.535649	0.535572	0.552871	0.534038	0.543267
h22	0.176900	0.000188	0.012597	0.015197	-0.001281	-0.002099
h33	0.307500	0.108621	0.076527	0.146431	0.146492	0.164139
h44	0.713180	0.668365	0.666790	0.703779	0.663343	0.675899
h55	0.595400	0.535137	0.526401	0.533483	0.539919	0.532873
h66	1.046800	1.059474	1.046171	1.053730	1.030323	1.043505
h77	0.198000	0.037436	0.030027	0.032693	0.028959	0.023033
h88	0.327700	0.132851	0.134231	0.122093	0.139791	0.150292
h99	0.238300	0.056349	0.052950	0.054939	0.055975	0.061013
h1010	0.229600	0.054284	0.065291	0.050900	0.039992	0.059026
SSE	n/a	0.210760	0.221349	0.195812	0.203737	0.187574

Table A.2: Estimates of the first 5 runs by EM algorithm on synthetic data

Parameter	Seed	Run 1	Run 2	Run 3	Run 4	Run 5
muL	3.300500	3.184170	3.223801	3.264512	3.062920	3.417821
muS	-0.373100	-0.560267	-0.614724	-0.031318	-0.267500	-0.346331
muC	0.815500	0.860076	0.980913	0.892600	0.857178	0.581014
lambda	0.068900	0.041800	0.100000	0.043200	0.065800	0.100000
a11	0.120200	0.159306	0.267162	0.132787	0.133428	-0.058474
a22	0.571200	0.424823	0.371992	0.505335	0.485447	0.684414
a33	0.412800	0.000012	0.149691	0.332479	0.157551	0.355739
q11	0.987000	1.060286	1.025415	0.829306	0.917839	1.072921
q22	0.759600	0.498631	0.594907	0.523702	0.615049	0.746581
q33	0.657200	0.000028	0.000266	0.000004	0.000001	0.000000
h11	0.603900	0.319575	0.394623	0.386281	0.326597	0.371895
h22	0.176900	0.024040	0.000000	0.018172	0.004351	0.000000
h33	0.307500	0.112337	0.081869	0.111809	0.132935	0.145034
h44	0.713180	0.468665	0.477821	0.700884	0.448682	0.521508
h55	0.595400	0.447656	0.344191	0.359369	0.393313	0.358640
h66	1.046800	1.244902	1.063380	1.202943	0.910638	1.020927
h77	0.198000	0.020800	0.000000	0.066552	0.004415	0.000000
h88	0.327700	0.109796	0.140428	0.112118	0.132408	0.138102
h99	0.238300	0.060842	0.092036	0.089817	0.072956	0.089026
h1010	0.229600	0.057979	0.113667	0.016366	0.069230	0.090605
SSE	n/a	1.152969	1.037147	0.970445	0.994766	0.879624

A.2 Matlab Codes

```
%%%% Generate synthetic data %%%%
clear all
clear
clc

randn('seed',3)
rand('seed',3)

filename = 'seeds.xlsx';
seeddata = xlsread(filename);

seeds = seeddata(:,1);
tau = seeddata(1:10,2);
tau = tau';

% number of maturities
N = length(tau);
% number of factors
nf = 3;
% number of parameters
np = 20;
T = 100;

% generate sythetic parameter
mu = seeds(1:3);
l = seeds(4);
L = calc_Lambda3(seeds(4),tau,N);
A = diag(seeds(5:7));
Q = diag(seeds(8:10));
H = diag(seeds(11:np));

runs = 20;
StoreSynthf = cell(runs,1);
for r=1:runs
```

```
    % generate sythetic state f
    synthf = zeros(nf,T);
    synthf(:,1) = seeds(1:nf);
    for i=2:T
        synthf(:,i) = A*synthf(:,i-1)+(eye(nf)-A)*mu
                      +normrnd(0,diag(Q));
    end
    StoreSynthf{r} = synthf;
end

StoreSynthy = cell(runs,1);
for r=1:runs
    % generate sythetic data y
    synthy = zeros(N,T);
    for j=1:T
        synthy(:,j) = L*StoreSynthf{r}(:,j)
                      + normrnd(0,diag(H));
    end
    StoreSynthy{r} = synthy;
end


pf1 = seeds(1:3);
pP1 = diag(seeds(5:7));
theta0.l = l;
theta0.L = L;
theta0.A = A;
theta0.Q = Q;
%theta0.Q = 0.4*eye(nf);
theta0.H = H;
theta0.mu = mu;



%%%%% EM and gradient method on synthetic data %%%%%
StoreGD = cell(runs,1);
StoreEM = cell(runs,1);
for r=1:runs
```

```
    display(r)
    GD_synth = GradientDescent(StoreSynthy{r},
               tau,pf1,pP1,theta0,0.00001,
               0.001,1000);
    StoreGD{r} = [GD_synth.mu;GD_synth.l;
                  diag(GD_synth.A);
                  diag(GD_synth.Q);
                  diag(GD_synth.H)];

    EM_synth = ExpectationMaximization(StoreSynthy{r},
                        tau,pf1,pP1,theta0,
                        0.0001,1000);
    StoreEM{r} = [EM_synth.mu;EM_synth.l;
                  diag(EM_synth.A);
                  diag(EM_synth.Q);
                  diag(EM_synth.H)];
end

StoreGD_matrix = reshape(cell2mat(StoreGD),20,20);
StoreEM_matrix = reshape(cell2mat(StoreEM),20,20);

seeds_matrix = repmat(seeds,1,20);
% sum of squared error
GD_sse = sum(((StoreGD_matrix - seeds_matrix).^2), 1);
EM_sse = sum(((StoreEM_matrix - seeds_matrix).^2), 1);



%%%% Implementation of gradient descent (diagonal) %%%%
function theta = GradientDescent(Y,tau,pf_init,pP_init,
                  theta_init,stepsize,ltol,maxiter,ltol2)

% define dimensions
T = size(Y,2);
N = size(Y,1);

% number of latent factors
nf = size(pf_init,1);
```

```
np = nf + 1 + nf + nf + N;

% initialize the unknown parameters
theta.l = theta_init.l;
theta.L= theta_init.L;
theta.A = theta_init.A;
theta.mu = theta_init.mu;
theta.H = theta_init.H;
theta.Q = theta_init.Q;

% initialize the unknown component
param.mu = theta.mu;
param.l = theta.l;
param.A = diag(theta.A);
param.Q = diag(theta.Q);
param.H = diag(theta.H);

% intialize the parameter components in vector
po = [param.mu',param.l,param.A',param.Q',param.H'];
storeKF = StoreKF(Y, pf_init, pP_init, theta);
Lo=LogLikelihood(Y,storeKF.W,storeKF.e);
Ln=2*Lo;

% initialize the score
s = Score(Y,tau,param.l, pf_init, pP_init, theta);
pn = po + stepsize*s;

% implementation of gradient descent
niter = 1;
tic;
while  abs(Ln-Lo) > ltol2 && norm(pn-po) > ltol
    && (niter<=maxiter),
    display(norm(pn-po));
    po = pn;
    Lo = Ln;

    theta.mu = po(1:3)';
    theta.l = po(4);
```

```
    theta.L= calc_Lambda3(po(4), tau, N);
    theta.A = diag(po(5:7));
    theta.Q = diag(po(8:10));
    theta.H = diag(po(11:np));
    %display(theta);
    storeKF = StoreKF(Y, pf_init, pP_init, theta);
    Ln = LogLikelihood(Y, storeKF.W, storeKF.e);

    s = Score(Y, tau, po(4), pf_init, pP_init, theta);

    pn = po + stepsize*s;
    display(niter);
    niter = niter + 1;
end
toc;
param = pn;
mu = param(1:3)';
display(mu);
l = param(4);
display(l);
L = calc_Lambda3(param(4), tau, N);
display(L);
A = diag(param(5:7));
display(A);
Q = diag(param(8:10));
display(Q);
H = diag(param(11:np));
display(H);
return



%%%% Implementation of the EM (diagonal) %%%%
function theta = ExpectationMaximization(Y,
    tau, pf_init, pP_init, theta_init, ltol, maxiter)

% define dimensions
T = size(Y,2);
```

```
N = size(Y,1);

% number of latent factors
nf = size(pf_init,1);

% initialize the unknown parameters
theta.l = theta_init.l;
theta.L= theta_init.L;
theta.A = theta_init.A;
theta.mu = theta_init.mu;
theta.H = theta_init.H;
theta.Q = theta_init.Q;

% initialize the log-likelihood
storeKF = StoreKF(Y, pf_init, pP_init, theta_init);
Ln = LogLikelihood(Y, storeKF.W, storeKF.e);
Lo = 2*Ln;

% EM algorithm
niter = 0;
tic;
while (abs(100*(Ln-Lo)/Lo)>ltol) && (niter<=maxiter),
%niter
%pause(0.1)

% extract state vector optimally using Kalman filter
%(rougly the E-step)
%display(theta.A);
storeKF = StoreKF(Y, pf_init, pP_init, theta);
fmatrix = zeros(nf,T);
    for t=1:T
    f = storeKF.f{t};
    fmatrix(:,t) = f;
    end
    %display(f_matrix);

    % conditional on the extract state vector, run
    % seemingly unrelated regression and vector
```

```
% autoregression to update the unknown parameters
% (roughly the M step)
grid = 0.01:0.0001:0.1;
[lambda,L,H] = GridSearchLambda(Y, tau, grid, fmatrix);

theta.l = lambda;
%display(theta.l);
theta.L = L;
theta.H = H;
theta.H = diag(diag(theta.H));
%display(theta.L);
%display(theta.H)

% run vector autoregrssion
theta.mu = mean(fmatrix,2);
% estimates parameters of VAR model by
% using maximum likelihood estimation
SpecX = vgxset('n',3,'nAR',1);
%display(SpecX);
Covartype = 'diagonal';
f = fmatrix - repmat(theta.mu,1,T);
EstSpec = vgxvarx(SpecX,f',Covartype);
%vgxdisp(EstSpec);

% update A,Q and mu
theta.A = EstSpec.AR{1};
theta.A = diag(diag(theta.A));
%display(theta.A);
theta.Q = EstSpec.Q;
theta.Q = diag(diag(theta.Q));
%display(theta.Q);
%display(theta);

% update the log-likelihood
Lo = Ln;
storeKF = StoreKF(Y, pf_init, pP_init, theta);
%display(storeKF.W{1});
Ln = LogLikelihood(Y,storeKF.W,storeKF.e);
```

```
    %pause(0.1)
    %fig = gcf;
    %plot(niter,Ln)
    display(Ln);
    display(niter)
    niter = niter + 1;
end
toc;
% return the maximum log-likelihood
l = theta.l;
display(l);
L = theta.L;
display(L);
A = theta.A;
display(A);
mu = theta.mu;
display(mu);
H = theta.H;
display(H);
Q = theta.Q;
display(Q);
display(niter-1);
display(Ln);

return



%%%% Implementation of gradient descent (full) %%%%
function theta = GradientDescentFull(Y,tau,pf_init,
        pP_init,theta_init,stepsize,ltol,maxiter)

% define dimensions
T = size(Y,2);
N = size(Y,1);

% number of latent factors
nf = size(pf_init,1);
```

```
np = nf + 1 + 9 + 6 + N + (N*N-N)/2;

% initialize the unknown parameters
theta.l = theta_init.l;
theta.L= theta_init.L;
theta.A = theta_init.A;
theta.mu = theta_init.mu;
theta.H = theta_init.H;
theta.Q = theta_init.Q;

% initialize the unknown components
param.mu = theta.mu;
param.l = theta.l;
param.A = [diag(theta.A)', diag(theta.A,1)',
           diag(theta.A,2)',diag(theta.A,-1)',
           diag(theta.A,-2)'];
param.Q = [diag(theta.Q)', diag(theta.Q,1)',
           diag(theta.Q,2)'];
param.H = diag(theta.H)';
for i=1:N-1
    param.H = [param.H, diag(theta.H,i)'];
end

% intialize the parameter components in vector
po = [param.mu',param.l,param.A,param.Q,param.H];
storeKF = StoreKF(Y, pf_init, pP_init, theta);
%Lo = LogLikelihood(Y, storeKF.W, storeKF.e);
LogLikelihood(Y,storeKF.W,storeKF.e)
% initialize the score
s = ScoreFull(Y,tau,param.l, pf_init, pP_init, theta);
pn = po + stepsize*s;
%Ln = 1.1*Lo;

% implementation of gradient descent
niter = 1;
tic;
while norm(pn-po) > ltol && (niter<=maxiter),
    display(norm(pn-po));
```

```
    po = pn;
    %Lo = Ln;

    theta.mu = po(1:3)';
    theta.l = po(4);
    theta.L= calc_Lambda3(po(4), tau, N);
    theta.A = diag(po(5:7))+diag(po(8:9),1)
              +diag(po(10),2)+diag(po(11:12),-1)
              +diag(po(13),-2);
    theta.Q = diag(po(14:16))+diag(po(17:18),1)
              +diag(po(19),2)+diag(po(17:18),-1)
              +diag(po(19),-2);
    h = po(20:np);
    c = N:-1:1;
    hcell = mat2cell(h,1,c);
    theta.H = diag(hcell{1});
    for j=1:N-1
        theta.H = theta.H + diag(hcell{j+1},j)
                  + diag(hcell{j+1},-j);
    end

    %display(theta);
    storeKF = StoreKF(Y, pf_init, pP_init, theta);
    %Ln = LogLikelihood(Y, storeKF.W, storeKF.e);
    LogLikelihood(Y, storeKF.W, storeKF.e)

    s = ScoreFull(Y,tau,po(4), pf_init, pP_init, theta);

    pn = po + stepsize*s;
    display(niter);
    niter = niter + 1;
end
toc;
param = pn;
mu = param(1:3)';
display(mu);
l = param(4);
display(l);
```

```
L = calc_Lambda3(param(4), tau, N);
display(L);
A = diag(param(5:7))+diag(param(8:9),1)
    +diag(param(10),2)+diag(param(11:12),-1)
    +diag(param(13),-2);
display(A);
Q = diag(param(14:16))+diag(param(17:18),1)
    +diag(param(19),2)+diag(param(17:18),-1)
    +diag(param(19),-2);
display(Q);
h = param(20:np);
c = N:-1:1;
hcell = mat2cell(h,1,c);
H = diag(hcell{1});
for j=1:N-1
    H = H + diag(hcell{j+1},j) + diag(hcell{j+1},-j);
end
display(H);

return



%%%% Implementation of discrete Kalman Filter %%%%
function Store = StoreKF(y,pf_init,pP_init,theta)
   % number of discrete time points
   T = size(y,2);

   % initialize the sets for storing results
   Store.pf = cell(T,1);
   Store.pf{1} = pf_init;

   Store.pP = cell(T,1);
   Store.pP{1} = pP_init;

   Store.f = cell(T,1);
   K = pP_init*theta.L'*inv(theta.L*pP_init*theta.L'+theta.H);
   f = pf_init + K*(y(:,1) - theta.L*pf_init);
```

```
    Store.f{1} = f;

    Store.P = cell(T,1);
    P = pP_init - K*theta.L*pP_init;
    Store.P{1} = P;

    Store.W = cell(T,1);
    Store.W{1} = theta.H + theta.L*pP_init*theta.L';

    Store.e = cell(T,1);
    Store.e{1} = y(:,1) - theta.L*pf_init;

    for t=2:T
        % Prediction stage
        f = theta.A*f
        + (eye(length(f))-theta.A)*theta.mu;
        P = theta.A*P*theta.A' + theta.Q;
        % Store the preditive results
        Store.pf{t} = f;
        Store.pP{t} = P;
        Store.W{t} = theta.H + theta.L*P*theta.L';
        Store.e{t} = y(:,t) - theta.L*f;

        % Compute Kalman gain factor:
        K = P*theta.L'*inv(theta.L*P*theta.L'+theta.H);

        % Measurement update stage
        f = f + K*(y(:,t) - theta.L*f);
        P = P - K*theta.L*P;
        % Store the updated results
        Store.f{t} = f;
        Store.P{t} = P;
    end
return

%%%% Computation of Log-likelihood %%%%
function L = LogLikelihood(Y,W_store,e_store)
```

```
% define dimensions
N = size(Y,1);
T = size(Y,2);

% initialize the log-likelihood
l = 0;

for t=1:T
    l = l -0.5*(N*log(2*pi)+
    log(abs(det(W_store{t})))
    +e_store{t}'*inv(W_store{t})*e_store{t});
end
L = l;
return



%%%%% Computation of Score (diagonal)%%%%%
function s= Score(Y,tau,lambda,pf_init,pP_init,theta)

N = size(Y,1);
T = size(Y,2);
nf = length(pf_init);
np = nf + 1 + nf + nf + N;

% derivatives of the unknown parameters
dmu = calc_dmu(nf,np);
dL = calc_dL(tau,lambda,nf,np);
dA = calc_dA(nf,np);
dQ = calc_dQ(nf,np);
dH = calc_dH(N,np,nf);

% results of Kalman filter
storeKF = StoreKF(Y,pf_init,pP_init,theta);
store_f = storeKF.f;
store_P = storeKF.P;
store_pf = storeKF.pf;
store_pP = storeKF.pP;
```

```
store_W = storeKF.W;
store_e = storeKF.e;

store_dpf = cell(T,1);
store_dpP = cell(T,1);
store_df = cell(T,1);
store_dP = cell(T,1);
store_dW = cell(T,1);
store_de = cell(T,1);
store_dl = cell(T,1);


% at time=1
store_dpf{1} = zeros(3,1,np);
store_dpP{1} = zeros(3,3,np);

dei = zeros(N,1,np);
for i=1:np
    dei(:,:,i) = calc_de(store_dpf{1}(:,:,i),
    dL(:,:,i),theta.L,store_pf{1});
end
store_de{1} = dei;

dWi = zeros(N,N,np);
for i=1:np
    dWi(:,:,i) = calc_dW(dL(:,:,i),
    store_dpP{1}(:,:,i),dH(:,:,i),
    theta.L,store_pP{1});
end
store_dW{1} = dWi;

dfi = zeros(nf,1,np);
for i=1:np
    dfi(:,:,i) = calc_df(store_dpf{1}(:,:,i),
    store_dpP{1}(:,:,i),dL(:,:,i),store_dW{1}(:,:,i),
    store_de{1}(:,:,i),theta.L,store_e{1},
    store_W{1},store_pP{1});
end
```

```
store_df{1} = dfi;

dPi = zeros(nf,nf,np);
for i=1:np
    dPi(:,:,i) = calc_dP(store_dpP{1}(:,:,i),
    dL(:,:,i),store_dW{1}(:,:,i),theta.L,store_W{1},
    store_pP{1});
end
store_dP{1} = dPi;

dli = zeros(1,1,np);
for i=1:np
    dli(:,:,i) = -0.5*trace(inv(store_W{1})
    *store_dW{1}(:,:,i)
    *(eye(N)-inv(store_W{1})*store_e{1}*store_e{1}'))
    -store_de{1}(:,:,i)'*inv(store_W{1})*store_e{1};
end
store_dl{1} = dli;

for t=2:T
    for i=1:np
        store_dpP{t}(:,:,i) = calc_dpP(dA(:,:,i),
        store_dP{t-1}(:,:,i),
        dQ(:,:,i),theta.A,store_P{t-1});
        store_dpf{t}(:,:,i) = calc_dpf(dA(:,:,i),
        store_df{t-1}(:,:,i),
        dmu(:,:,i),theta.A,store_f{t-1},theta.mu);
        store_dW{t}(:,:,i) = calc_dW(dL(:,:,i),
        store_dpP{t}(:,:,i),dH(:,:,i),
        theta.L,store_pP{t});
        store_de{t}(:,:,i) = calc_de(store_dpf{t}(:,:,i),
        dL(:,:,i),theta.L,store_pf{t});
        store_dl{t}(:,:,i) = -0.5*trace(inv(store_W{t})
        *store_dW{t}(:,:,i)*(eye(N)-inv(store_W{t})
        *store_e{t}*store_e{t}'))-store_de{t}(:,:,i)'
        *inv(store_W{t})*store_e{t};

        store_dP{t}(:,:,i) = calc_dP(store_dpP{t}(:,:,i),
```

```
            dL(:,:,i),store_dW{t}(:,:,i),theta.L,
            store_W{t},store_pP{t});
            store_df{t}(:,:,i) = calc_df(store_dpf{t}(:,:,i),
            store_dpP{t}(:,:,i),
            dL(:,:,i),store_dW{t}(:,:,i),
            store_de{t}(:,:,i),theta.L,store_e{t},
            store_W{t},store_pP{t});
        end
end

dli_matrix = zeros(T,np);
for j=1:T
    dli_matrix(j,:) = store_dl{j};
end

% sum up each column of dlt to obtain dl
s = sum(dli_matrix, 1);
return



%%%%% Implementation of Score (Full) %%%%%
function s= ScoreFull(Y,tau,lambda,pf_init,pP_init,theta)

N = size(Y,1);
T = size(Y,2);
nf = length(pf_init);
np = nf + 1 + 9 + 6 + N + (N*N-N)/2;

% derivatives of the unknown parameters
dmu = calc_dmu(nf,np);
dL = calc_dL(tau,lambda,nf,np);
dA = calc_dAFull(nf,np);
dQ = calc_dQFull(nf,np);
dH = calc_dHFull(N,np,nf);

% results of Kalman filter
storeKF = StoreKF(Y,pf_init,pP_init,theta);
```

```
store_f = storeKF.f;
store_P = storeKF.P;
store_pf = storeKF.pf;
store_pP = storeKF.pP;
store_W = storeKF.W;
store_e = storeKF.e;

store_dpf = cell(T,1);
store_dpP = cell(T,1);
store_df = cell(T,1);
store_dP = cell(T,1);
store_dW = cell(T,1);
store_de = cell(T,1);
store_dl = cell(T,1);


% at time=1
store_dpf{1} = zeros(3,1,np);
store_dpP{1} = zeros(3,3,np);

dei = zeros(N,1,np);
for i=1:np
    dei(:,:,i) = calc_de(store_dpf{1}(:,:,i),dL(:,:,i),
    theta.L,store_pf{1});
end
store_de{1} = dei;

dWi = zeros(N,N,np);
for i=1:np
    dWi(:,:,i) = calc_dW(dL(:,:,i),store_dpP{1}(:,:,i),d
    H(:,:,i),theta.L,store_pP{1});
end
store_dW{1} = dWi;

dfi = zeros(nf,1,np);
for i=1:np
    dfi(:,:,i) = calc_df(store_dpf{1}(:,:,i),
    store_dpP{1}(:,:,i),dL(:,:,i),store_dW{1}(:,:,i),
```

```
    store_de{1}(:,:,i),theta.L,
    store_e{1},store_W{1},store_pP{1});
end
store_df{1} = dfi;

dPi = zeros(nf,nf,np);
for i=1:np
    dPi(:,:,i) = calc_dP(store_dpP{1}(:,:,i),
    dL(:,:,i),store_dW{1}(:,:,i),theta.L,store_W{1},
    store_pP{1});
end
store_dP{1} = dPi;

dli = zeros(1,1,np);
for i=1:np
    dli(:,:,i) = -0.5*trace(inv(store_W{1})
    *store_dW{1}(:,:,i)*(eye(N)-inv(store_W{1})
    *store_e{1}*store_e{1}'))-store_de{1}(:,:,i)'
    *inv(store_W{1})*store_e{1};
end
store_dl{1} = dli;

for t=2:T
    for i=1:np
        store_dpP{t}(:,:,i) = calc_dpP(dA(:,:,i),
                              store_dP{t-1}(:,:,i),
                              dQ(:,:,i),
                              theta.A,store_P{t-1});
        store_dpf{t}(:,:,i) = calc_dpf(dA(:,:,i),
                              store_df{t-1}(:,:,i),
                              dmu(:,:,i),theta.A,
                              store_f{t-1},theta.mu);
        store_dW{t}(:,:,i) = calc_dW(dL(:,:,i),
                             store_dpP{t}(:,:,i),
                             dH(:,:,i),theta.L,store_pP{t});
        store_de{t}(:,:,i) = calc_de(store_dpf{t}(:,:,i),
                             dL(:,:,i),theta.L, store_pf{t});
        store_dl{t}(:,:,i) = -0.5*trace(inv(store_W{t})
```

```
                                    *store_dW{t}(:,:,i)
                                    *(eye(N)-inv(store_W{t})
                                    *store_e{t}*store_e{t}'))
                                    -store_de{t}(:,:,i)'
                                    *inv(store_W{t})*store_e{t};

        store_dP{t}(:,:,i) = calc_dP(store_dpP{t}(:,:,i),
        dL(:,:,i),store_dW{t}(:,:,i),
        theta.L,store_W{t},store_pP{t});
        store_df{t}(:,:,i) = calc_df(store_dpf{t}(:,:,i),
        store_dpP{t}(:,:,i),dL(:,:,i),
        store_dW{t}(:,:,i),store_de{t}(:,:,i),
        theta.L,store_e{t},store_W{t},store_pP{t});
    end
end

dli_matrix = zeros(T,np);
for j=1:T
    dli_matrix(j,:) = store_dl{j};
end

% sum up each column of dlt to obtain dl
s = sum(dli_matrix, 1);
return



%%%% grid search for lambda %%%%
% Y is a N x T matrix; grid is a row vector;
% fmatrix is a 3 x T matrix
function [lambda,L,H] =GridSearchLambda(Y,tau,grid,fmatrix)
T = size(Y,2);
N = size(Y,1);
Ydesign = reshape(Y,N*T,1);
fdesign = reshape(fmatrix,3*T,1);
gridsize = length(grid);
storeL = cell(gridsize,1);
storelq = zeros(gridsize,1);
```

```
for g=1:gridsize
    L = calc_Lambda3(grid(g), tau, N);
    storeL{g}= L;
    Ldesign = designL(L,T);
    dmatrix = (Ydesign - Ldesign*fdesign).^2;
    storelq(g) = sum(dmatrix(:));
end
[M,I] = min(storelq);
%pause(0.1)
%display(M)
%display(I)
lambda = grid(I);
L = storeL(I);
L = L{1};
e = Ydesign - Ldesign*fdesign;
e = reshape(e,N,T);
H = (1/T).*e*e';
end


%%%%% Other Functions %%%%%

% function to calculate Lambda for 3 factor DNS
function val=calc_Lambda3(lambda, tau, N)
c1 = ones(N,1);
c2 = (1-exp(-lambda.*tau)) ./ (lambda.*tau);
c3 = c2 - exp(-lambda.*tau);

val = [c1 c2' c3'];
return

% calculate the derivative of A
function dA = calc_dA(nf,np)
res = zeros(nf,nf,np);
i = nf + 1 + 1;
j = nf + 1 + nf;
for k=i:j
```

```
    res(k-i+1,k-i+1,k) = 1;
end
dA = res;
end

% calculate the derivative of Lambda
function dL = calc_dL(tau,lambda,nf,np)
N = size(tau,2);
res = zeros(N,nf,np);

dx1lT = (exp(-lambda.*tau) .*
(lambda.*tau - exp( lambda.*tau) + 1))
./ (lambda.^2 .* tau);
dx2lT = dx1lT + tau .* exp(-lambda.*tau);
i = nf + 1;
res(:,:,i) = reshape([zeros(N,1) dx1lT' dx2lT'],N,nf);
dL = res;
end

% calculate the derivative of mu
function dmu = calc_dmu(nf,np)
res = zeros(nf,1,np);
m = eye(nf);
for i=1:nf
    res(:,:,i) = m(:,i);
end
dmu = res;
end

% calculate the derivative of A (diagonal case)
function dA = calc_dA(nf,np)
res = zeros(nf,nf,np);
i = nf + 1 + 1;
j = nf + 1 + nf;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
dA = res;
```

```
end

% calculate the derivative of A (non-diagonal case)
function dA = calc_dAFull(nf,np)
res = zeros(nf,nf,np);
i = nf + 1 + 1;
j = nf + 1 + nf;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
res(1,2,j+1) = 1;
res(2,3,j+2) = 1;
res(1,3,j+3) = 1;
res(2,1,j+4) = 1;
res(3,2,j+5) = 1;
res(3,1,j+6) = 1;
dA = res;
end

% calculate the derivative of Q (diagonal case)
function dQ = calc_dQ(nf,np)
res = zeros(nf,nf,np);
i = nf + 1 + nf +1;
j = nf + 1 + nf + nf;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
dQ = res;
end

% calculate the derivative of Q (non-diagonal case)
function dQ = calc_dQFull(nf,np)
res = zeros(nf,nf,np);
i = nf + 1 + nf + 6 + 1;
j = nf + 1 + nf + 6 + nf;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
```

```
% for 3 factor case only
res(1,2,j+1) = 1;
res(2,1,j+1) = 1;
res(2,3,j+2) = 1;
res(3,2,j+2) = 1;
res(1,3,j+3) = 1;
res(3,1,j+3) = 1;
dQ = res;
end

% calculate the derivative of H
function dH = calc_dH(N,np,nf)
res = zeros(N,N,np);
i = nf + 1 + nf + nf + 1;
j = nf + 1 + nf + nf + N;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
dH = res;
end

% calculate the derivative of H
function dH = calc_dHFull(N,np,nf)
res = zeros(N,N,np);
i = nf + 1 + nf + 6 + nf + 3 + 1;
j = nf + 1 + nf + 6 + nf + 3 + 1 + N - 1;
for k=i:j
    res(k-i+1,k-i+1,k) = 1;
end
c = N-1:-1:1;
i = j+1;
for t=1:length(c)
    j = i + c(t)-1;
    for k=i:j
        res(k-i+1,k-i+1+N-c(t),k) = 1;
        res(k-i+1+N-c(t),k-i+1,k) = 1;
    end
    i = j + 1;
```

```
end
dH = res;
end

% calculate the derivative of prediction error (v)
function de = calc_de(dpf,dL,L,pf)
de = -L*dpf - dL*pf;
end

% calculate the derivative of W
function dW = calc_dW(dL,dpP,dH,L,pP)
dW = dL*pP*L' + L*dpP*L' + L*pP*dL' + dH;
end

% calculate the derivative of predicted f
function dpf = calc_dpf(dA,df,dmu,A,f,mu)
dpf = dA*f + A*df + dmu - dA*mu - A*dmu;
end

% calculate the derivative of predicted P
function dpP = calc_dpP(dA,dP,dQ,A,P)
dpP = dA*P*A' + A*dP*A' + A*P*dA' + dQ;
end

% calculate the derivative of updated f
function df = calc_df(dpf,dpP,dL,dW,de,L,e,W,pP)
df = dpf + dpP*L'*inv(W)*e + pP*dL'*inv(W)*e
     - pP*L'*inv(W)*dW*inv(W)*e + pP*L'*inv(W)*de;
end

% calculate the derivative of updated P
function dP = calc_dP(dpP,dL,dW,L,W,pP)
dP = dpP - dpP*L'*inv(W)*L*pP - pP*dL'*inv(W)*L*pP
     + pP*L'*inv(W)*dW*inv(W)*L*pP
     - pP*L'*inv(W)*dL*pP - pP*L'*inv(W)*L*dpP;
end
```