Quantitative Methods in Fixed Income - Assignment 1

Tommaso Garutti - 424192
Bas van Andel - 508077
Lemeng Li - 432203
Team 4

November 27, 2019

Abstract

In this paper we evaluate the performance of the DNS model. The research commences by evaluating the ability of the DNS model to capture stylized facts of the empirical yield data, after which out-of-sample forecasts of the DNS model are benchmarked against a random walk, VAR(1) and ECM(1) model. Finally, a simulation study is performed to evaluate whether the DNS estimation procedure produces correct parameters. We find that the DNS model can capture all stylized facts of the yield curve and the simulation study suggest that the estimation procedure is able to produce correct and robust parameter estimates. In forecasting, the DNS model cannot consistently outperform forecasts of the benchmark. Especially the random walk model often emerges as the best specification for predicting yield curves.

1 Introduction

Estimating and forecasting term structures has been an important area of research in recent decades. While substantial progress has been made over the years, forecasting yield curves remains especially challenging (Diebold and Li, 2006). Moreover, many studies focus on modeling empirical data, as opposed to using simulations as a data source. This paper aims to evaluate and compare a variety of different models, while putting emphasis on the Dynamic Nelson Siegel model using the formulation of Diebold and Li (2006).

In order to be able to provide accurate estimates and forecasts, it is important for term structure models to comply with the stylized facts of yields curves. The stylized facts, as stated by Van der Wel (2019), are given below:

i The average yield curve over time is increasing and concave.

ii The yield curve can take on a variety of shapes.

iii Yield dynamics are persistent.

iv Yields for long maturities are more persistent than yields for shorter maturities.

v The short end of the yield curve is more volatile than the long end of the curve.

vi Yields for different maturities have high cross-correlations.

This report is structured as follows. Section 2 provides a comprehensive description of the models applied in the estimation and forecasting of yield curve data, including motivation for choosing the various methods. Section 3 introduces the data set used in the empirical study and investigates whether the key stylized facts are observed in the yield curves. It also presents the setup of a simulated set of yields. Section 4 analyzes both the empirical results and simulation results, in which the former section compares yield forecasts against various benchmarks. Finally, Section 5 concludes this research and offers points for improvement.

2 Methodology

To estimate yields curves, we apply the Dynamic Nelson-Siegel (DNS) model as formulated by Diebold and Li (2006). The DNS model is estimated in two steps in order to facilitate a smooth estimation procedure. Both an empirical and a simulated set of yields is used, which are presented in Section 3.

In the empirical study, forecast yields with the DNS model using the latest 48 observations available at the time of forecasting, corresponding to a period of 4 years when using monthly observations. Allowing parameters to adjust over time takes into account changing market conditions. In order to benchmark our DNS forecasts, we consider two variations to the first estimation step of the DNS. We also compare the forecasts to five different yield curve models.

In the simulation study, we examine the performance of the DNS by comparing the model parameters estimated on the simulated data against the parameters of the data generating process.

2.1 Dynamic Nelson-Siegel model

The yield curve of the DNS model formulated by Diebold and Li (2006) is presented as

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

The yield curve above can be written in a dynamic factor model form where a VAR(1) is specified for the factor dynamics:

$$y_t = B(\lambda_t)\beta_t + \varepsilon_t \tag{2}$$

$$\beta_t = c + \Phi\beta_{t-1} + \eta_t \tag{3}$$

In order to smoothen the estimation procedure, we linearize the model following Ibanez (2015) by fixing $\lambda_t = \lambda$ such that the correlation between the slope loading $\left(\frac{1-\exp(-\lambda\tau)}{\lambda\tau}\right)$ and the curvature loading $\left(\frac{1-\exp(-\lambda\tau)}{\lambda\tau} - \exp(-\lambda\tau)\right)$ is minimized. If the loadings are correlated, Equation (9) will contain endogenous explanatory variables and OLS will not produce consistent estimates. The correlation is minimized for $\lambda = 0.0703$.

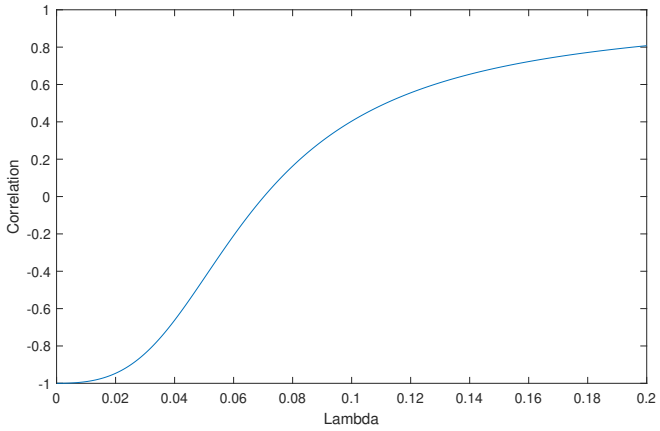

Figure 1: Correlation of the slope and curvature loadings for λ between 0.0001 and 0.2. The correlation is minimized for $\lambda = 0.0703$

2.1.1 Varying lambda in forecasts

For comparing forecasts, we consider two variations on determining the λ_t. The first variation consists setting $\lambda_t = \lambda$ such that the curvature loading $\left(\frac{1-\exp(-\lambda\tau)}{\lambda\tau} - \exp(-\lambda\tau)\right)$ is maximized for a specific τ. By maximizing for $\tau = 30$ months, Diebold and Li (2006) obtain $\lambda = 0.0609$. A plot of the lambda against the curvature loading is provided in Appendix A.

A possible criteria for optimizing λ_t is minimizing the estimation errors of the DNS model. This is be done by first obtaining parameter estimates for β in Equation (9) by fixing $\lambda = 0.0609$, then use those estimates for β to minimize the squares of error terms ε_t. The λ is re-estimated for every forecasting period.

2.2 Benchmark models

To benchmark forecasts provided by the DNS model presented in Section 2.1, we consider a random walk, VAR(1) and ECM specifications.

2.2.1 Random walk

The first model used to provide a benchmark against the DNS model is a random walk, which can capture the third stylized fact of persistency. For $y_t(\tau_i)$ the yield with maturity τ_i $(i = 1, ..., n)$ at time t $(t = 1, ..., T)$, this model is defined as

$$y_t(\tau_i) = y_{t-1}(\tau_i) + \varepsilon_{i,t} \tag{4}$$

with ε assumed to be noise, such that the forecast for period $t + k$ equals the current observation at t.

2.2.2 Vector autoregressive model

In order to model the sixth stylized fact of high cross-correlations in yields, we consider a vector autoregressive model with lag order 1:

$$\begin{pmatrix} y_t(\tau_1) \\ \vdots \\ y_t(\tau_n) \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} + \begin{pmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} \end{pmatrix} \begin{pmatrix} y_{t-1}(\tau_1) \\ \vdots \\ y_{t-1}(\tau_n) \end{pmatrix} + \begin{pmatrix} \varepsilon_{1,t} \\ \vdots \\ \varepsilon_{n,t} \end{pmatrix} \tag{5}$$

The VAR(1) can capture many relations amongst yields and can take into account all stylized facts of the yield curve due to the large amount of parameters specified. Unfortunately, the amount of parameters is a double-edged sword, making estimation difficult and forecasts inaccurate especially when the data set is small relative to the amount of maturities.

2.2.3 Error-correction model

Empirically, time series of yields often appear to be cointegrated (Diebold and Sharpe, 1990; Pagan, 1996). We may exploit this cointegration relation by modeling the following relations:

$$y_t(\tau_i) = \beta_{i,0} + \beta_{i,1} y_{t-1}(\tau_i) + \varepsilon_{i,t} \tag{6}$$

$$\varepsilon_{i,t} = \rho_i \varepsilon_{i,t-1} + v_{i,t} \tag{7}$$

The error-correction model (ECM) follows by rewriting Equations (6) and (7):

$$\Delta y_t(\tau_i) = \phi_i \Delta y_{t-1}(\tau_i) - \gamma_i [y_{t-1}(\tau_i) - \beta_{i,0} - \beta_{i,1} y_{t-2}(\tau_i)] + v_{i,t} \tag{8}$$

The estimation procedure starts with the cointegration relation of Equation (6), after which estimation of the ECM of Equation (8) follows. Using the relation $\rho_i = 1 - \gamma_i$, we can forecast yields using Equations (6) and (7).

3 Data

3.1 Empirical data

For the empirical part of this study, data from Italian government bonds was extracted from the Bloomberg terminal[1]. More specifically, we used end-of-day bid price quotes from the first (trading) day of each month, starting in November 2009 until November 2019. The term structure over time is visualized in Figure 9. Each monthly data point consists of a set of yields for fixed maturities of 3, 6, 12, 24, 36, 48, 60, 72, 84, 96, 108 and 120 months. Fixed maturities of larger than ten years have been discarded as there is very little bond price data available and since existing bonds are often "off-the-run" (Van der Wel, 2019). Hence, this would otherwise result in poor estimates and subsequent forecasts for the long end of the curve. Finally, we chose to model the Italian yield curve due to its large size and liquidity (Beber et al., 2008) as well as its volatility over the past decade, mainly driven by geopolitical events and the recent European sovereign debt crisis. A surface plot of the yield curve can be found in Appendix B.

3.1.1 The stylized facts of the yield curve

In this section, we will provide empirical evidence that the data exhibits the stylized facts, as described in Section 1. Table 1 presents descriptive statistics of the Italian yield curve data. From this information, it is clear that the yield curve is on average upward sloping and concave. Moreover, the short end of the yield curve is more volatile than long end of the curve, as shown by relatively larger standard errors compared to the means. Furthermore, both 1-month and 12-month autocorrelations imply that long term yields are indeed more persistent than short term yields. Generally speaking, though, yield dynamics are fairly persistent across the board.

In order to investigate whether the yield curve can take on a variety of shapes (i.e. stylized fact (ii) in Section 1), Figure 2 displays four different shapes that are observed in the data. Finally, the last stylized fact from Section 1 is concerned with high cross-correlations between yields of different fixed maturities. Table 4 in Appendix 4 presents these cross-correlations. The lowest cross-correlation of 0.789 is observed between the 3 month yield and the 10 year yield, which simultaneously represents the largest spread in yields in the data. Moreover, most maturity-pairs enjoy very strong cross-correlations of above 0.900, implying that the final stylized fact is also satisfied.

[1] The following functions on Bloomberg have been consulted in order to obtain data for each fixed maturity: GBOTS3MO Index, GBOTS6MO Index, GBOTS12M Index, GBTP2YR Index, GBTP3YR Index, GBTP4YR Index, GBTP5YR Index, GBTP6YR Index, GBTP7YR Index, GBTP8YR Index, GBTP9YR Index and GBTP10YR Index.

Table 1: Descriptive statistics of the Italian yield curve between November 2009 and November 2019

Maturity (Months)	Mean	Std. dev.	Minimum	Maximum	$\hat{\rho}(1)$	$\hat{\rho}(12)$
3	0.245	0.737	-0.618	4.826	0.842	0.497
6	0.482	0.884	-0.434	5.243	0.906	0.533
12	0.660	1.032	-0.441	5.220	0.937	0.571
24	0.832	1.278	-0.768	6.007	0.931	0.527
36	1.310	1.334	-0.240	6.071	0.939	0.591
48	1.457	1.426	-0.341	6.326	0.955	0.593
60	1.849	1.390	0.015	6.209	0.959	0.633
72	1.981	1.399	0.072	6.349	0.963	0.621
84	2.258	1.350	0.271	6.387	0.961	0.623
96	2.355	1.425	0.272	6.336	0.970	0.655
108	2.570	1.358	0.422	6.138	0.971	0.667
120 (level)	2.783	1.327	0.518	6.201	0.970	0.673
Slope	2.538	0.872	0.829	4.874	0.876	0.484
Curvature	-1.363	0.943	-3.350	1.271	0.827	0.124

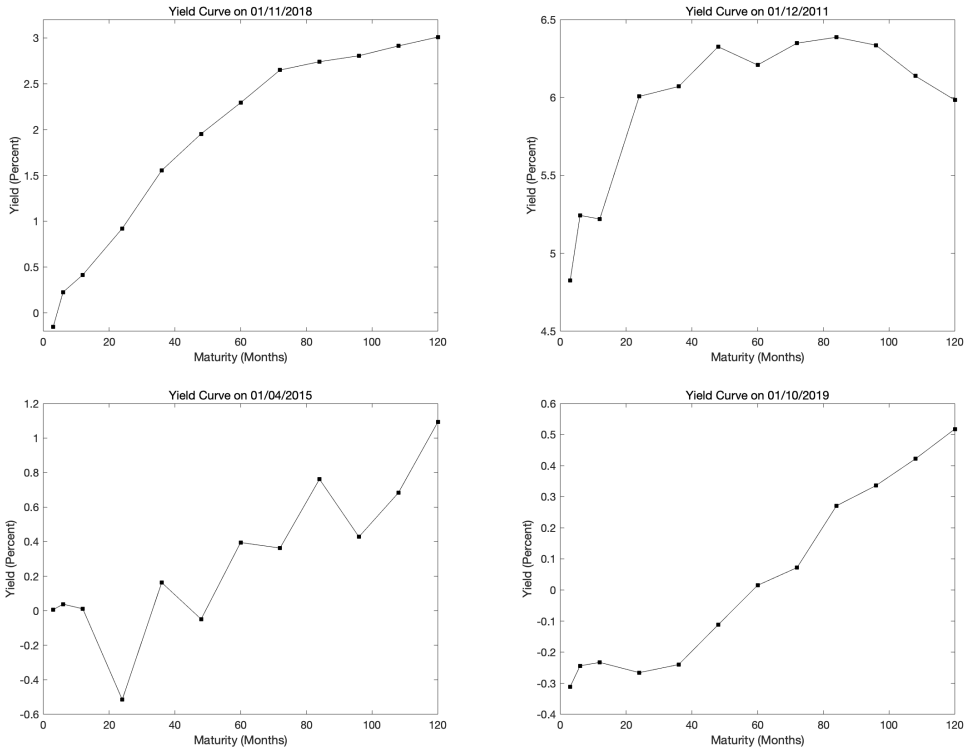

Figure 2: The Italian yield curve data satisfies stylized fact (ii): the yield curve can take on a variety of shapes

3.2 Simulated data

In order to study the estimation performance of the DNS model, we specify a data generating process that simulates yields with the same 12 maturities as in the empirical data set. To obtain sensible yield curve dynamics, we follow a DNS model as defined in Section 2.1 using estimates on the full empirical data set to obtain parameter specifications:

$$y_t = B(\lambda_t)\beta_t + \varepsilon_t \tag{9}$$

$$\beta_t = c + \Phi\beta_{t-1} + \eta_t \tag{10}$$

Using $\lambda = 0.0703$, we set the parameters $c = [0.2962, -0.1494, -0.7163]'$ and

$$\Phi = \begin{bmatrix} 1.0113 & 0.0686 & 0.0427 \\ -0.3118 & 0.5726 & 0.0077 \\ 0.0397 & 0.0390 & 0.8079 \end{bmatrix}$$

The noise terms are $\eta_t \sim N(0, \sigma_\eta I_{12})$ with $\sigma_\eta^2 = 0.1$ and $\varepsilon_t \sim N(0, \sigma_\varepsilon I_{12})$ with $\sigma_{\varepsilon,i}^2 = 0.01 * 0.9^{i+1}$ for maturity $i = 1, ..., 12$ to decrease the variance of yields for longer maturities. The simulations are initialized by setting $\beta_0 = [1, 1, 1]'$.

We simulate 100 different data sets of 121 months. For each yield data set we calculate summary statistics and take an average of these statistics across the simulations. These summary statistics are described in Appendix F. From Table 8 it is evident that the simulated yields display some of the stylized facts of the yield curve. Specifically, the first-order autocorrelations are greater than 0.74 for every maturity implying that the dynamics of the simulated yields are persistent and, since the autocorrelation is increasing for maturity, that this persistence is greater for yields with long maturities. The short end and long end yields have similar standard deviations, however, taking into consideration that the mean of the short end yields is smaller than that of the long end yields, the short end of the simulated yield curve is more volatile than the long end.

We evaluate the DNS parameters following the model explained in section 2.1 over each simulated data set. To check the robustness of the estimations, we vary the variance of the factor noise σ_η^2 in 10 steps between 0.001 and 2, such that we simulate 10x100 data sets, each data set containing 121 months. We obtain 10 different mean parameter estimates for c and Φ, whose values are tested against the true parameter values of the DGP using a t-test with 100-1 degrees of freedom, thus obtaining 10 different p-values for each of the 12 parameters.

4 Results

4.1 Empirical results

Figure 3 shows the estimated parameter values for $\hat{\beta}_{1t}$ (level factor), $\hat{\beta}_{2t}$ (slope factor) and $\hat{\beta}_{3t}$ (curvature factor) after applying the DNS model with fixed $\lambda = 0.0703$, as described in Section 2.1, for the entire sample set.

Using the summary statistics provided in Table 2, it can be investigated whether each of the aforementioned stylized facts for yield curves are satisfied.

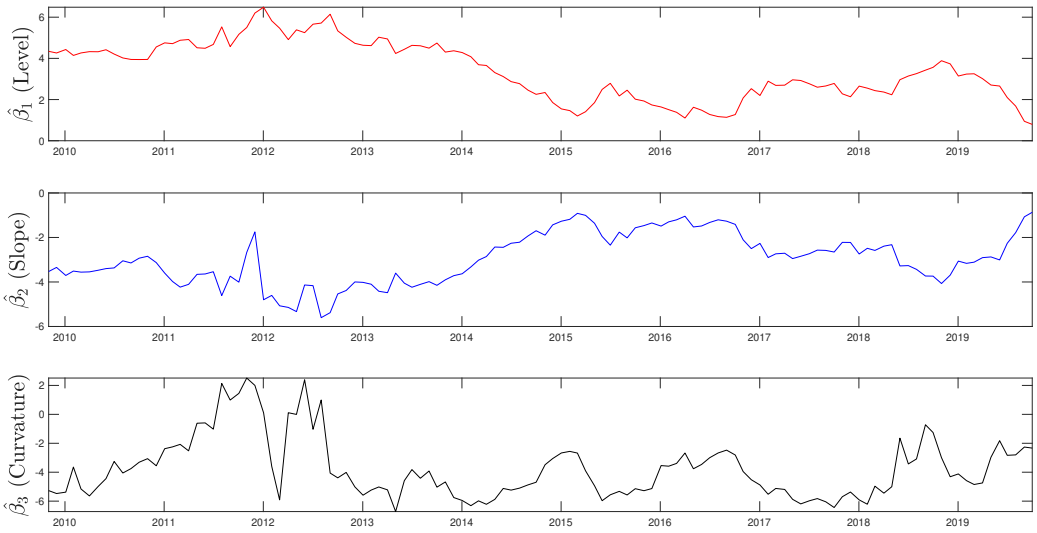

Figure 3: Beta estimations of the DNS model on the yield curve of Italian government bonds between November 2009 and November 2019.

Table 2: Descriptive statistics of estimated factors using DNS

Factor	Mean	Std. dev.	Minimum	Maximum	$\hat{\rho}(1)$	$\hat{\rho}(12)$
$\hat{\beta}_{1t}$	3.391	1.401	0.794	6.484	0.969	0.642
$\hat{\beta}_{2t}$	-2.938	1.137	-5.604	-0.869	0.907	0.476
$\hat{\beta}_{3t}$	-3.743	2.084	-6.721	2.51	0.812	0.097

First, estimations using average values of $\hat{\beta}_{1t}$, $\hat{\beta}_{2t}$ and $\hat{\beta}_{3t}$ show that the corresponding average yield curve is increasing and concave, which is visualized in Figure 4. Second, the yield curve can take a variety of shapes, which is depicted in Figure 5. Unfortunately, as not all the shapes from Section 3.1.1 are obtained using the model, thus the second stylized fact is not fully satisfied. Third, yield dynamics are persistent, which is demonstrated by high persistency of β_{1t} (Diebold and Li, 2006). Moreover, spread dynamics are less persistent, corresponding to weaker persistence of β_{2t}. Fourth, since β_{1t} is the most persistent factor, yields for long maturities will be more persistent than yields for shorter maturities. Fifth, as mentioned by Diebold and Li (2006), the short end of the yield curve is more volatile than the long end in the DNS model, as the short end depends positively on both β_{1t} and β_{2t}, while the long end solely depends on β_{1t}. Finally, Table 5 in Appendix D shows that the estimated yields for different maturities also have high cross-correlations. This satisfies the last stylized fact, as described in Section 1.

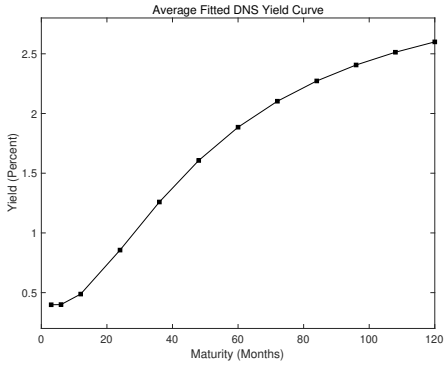

Figure 4: Fitted DNS yield curve obtained by evaluating the model at the mean values of $\hat{\beta}_{1t}$, $\hat{\beta}_{2t}$ and $\hat{\beta}_{3t}$ from Table 2.

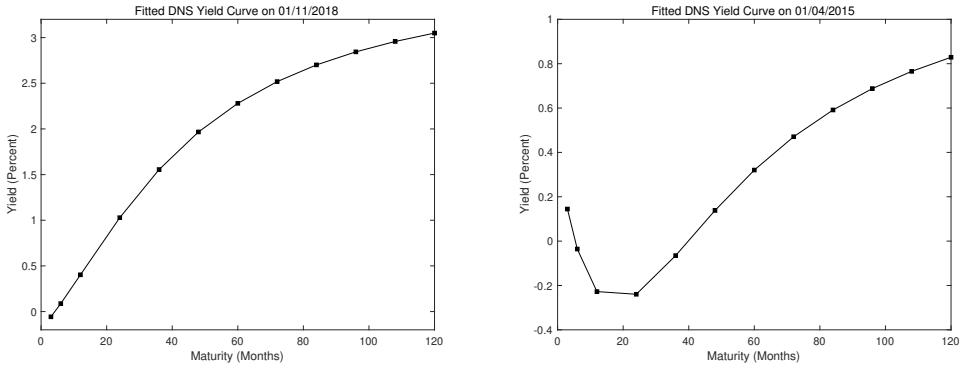

Figure 5: Fitted DNS yield for two time points, showing different shapes of the yield curve.

4.1.1 Empirical forecasting results

We perform 1-, 6- and 12-month-ahead forecasts for variations of DNS models as well as the benchmark models described in Section 2.2. Table 3 with root mean squared forecast errors for the 1-month-ahead forecasts is displayed below and RMSFEs for the 6 and 12 month forecasts can be found in Appendix E. The 1-month-ahead RMSFEs indicate that three DNS models seem to perform similarly, outperforming the VAR and ECM models for yields with 1-year maturities or greater. This is supported by 6- and 12-month-ahead forecasts in Appendix E, where the RMSFEs of the DNS models are lower than those of the VAR and ECM for longer term yields. This could be explained by the fact that the DNS models are able to capture the stylized facts of the yield curve more accurately. However, in most cases the random walk clearly outperforms all the models.

Table 3: Out-of-same 1-month-ahead forecasting results: Root Mean Square Forecasting Error

Maturity	Random Walk	VAR	ECM	DNS (1)	DNS (2)	DNS (3)
3mth	0.060	0.127	0.098	0.251	0.232	0.231
6mth	0.060	0.152	0.090	0.112	0.113	0.113
1yr	0.089	0.180	0.135	0.136	0.126	0.125
2yr	0.148	0.243	0.231	0.170	0.171	0.171
3yr	0.191	0.297	0.262	0.215	0.212	0.212
4yr	0.202	0.272	0.291	0.329	0.311	0.310
5yr	0.194	0.281	0.271	0.249	0.233	0.232
6yr	0.204	0.256	0.282	0.251	0.242	0.241
7yr	0.205	0.274	0.306	0.220	0.219	0.219
8yr	0.207	0.266	0.303	0.254	0.254	0.254
9yr	0.186	0.242	0.269	0.208	0.204	0.204
10yr	0.194	0.263	0.273	0.252	0.232	0.231

Note: The forecast period spans November 2013 to November 2019. DNS (1) refers to fixing $\lambda_t = \lambda = 0.0703$ (i.e. Diebold and Li (2006)), DNS (2) refers to fixing $\lambda_t = \lambda = 0.0609$ (i.e. minimizing the correlation between the slope and curvature loadings) and DNS (3) refers to a time-varying λ_t that minimizes squared errors.

4.2 Simulation results

Using the 100 simulated data sets defined in Section 3.2, we estimate the yield curve dynamics following the DNS model with $\lambda = 0.0703$. The resulting parameter values are illustrated in a set of histograms in Figure 6, where each histogram shows the distribution of values for each parameter. Below are the means of the DNS model parameters estimated over 100 simulated data sets with standard deviations provided in brackets,

$$c = \begin{bmatrix} 0.4603 \\ (0.1653) \\ -0.1665 \\ (0.1408) \\ -1.2437 \\ (0.2196) \end{bmatrix}, \quad \Phi = \begin{bmatrix} 0.9612 & 0.0499 & 0.0624 \\ (0.0533) & (0.0674) & (0.0397) \\ -0.3354 & 0.5397 & 0.0056 \\ (0.0525) & (0.0644) & (0.0387) \\ 0.135 & 0.2012 & 0.6237 \\ (0.0827) & (0.1043) & (0.0631) \end{bmatrix}.$$

The means of the DNS model parameters over the 100 simulated data sets are within two standard deviations of the values used for the DGP. The histograms in Figure 6 also indicate that the estimated parameters are similar to those used to generate the yield data sets in Section 3.2.

To study the robustness of our estimations, we vary the level of noise in our data generating process in 10 different magnitudes. Results of t-tests for all 12 parameters of the DNS model show that almost none of the estimations are significantly different from the real values when the noise in the data generating process is increased or decreased. Histograms of p-values can be found in Appendix G.

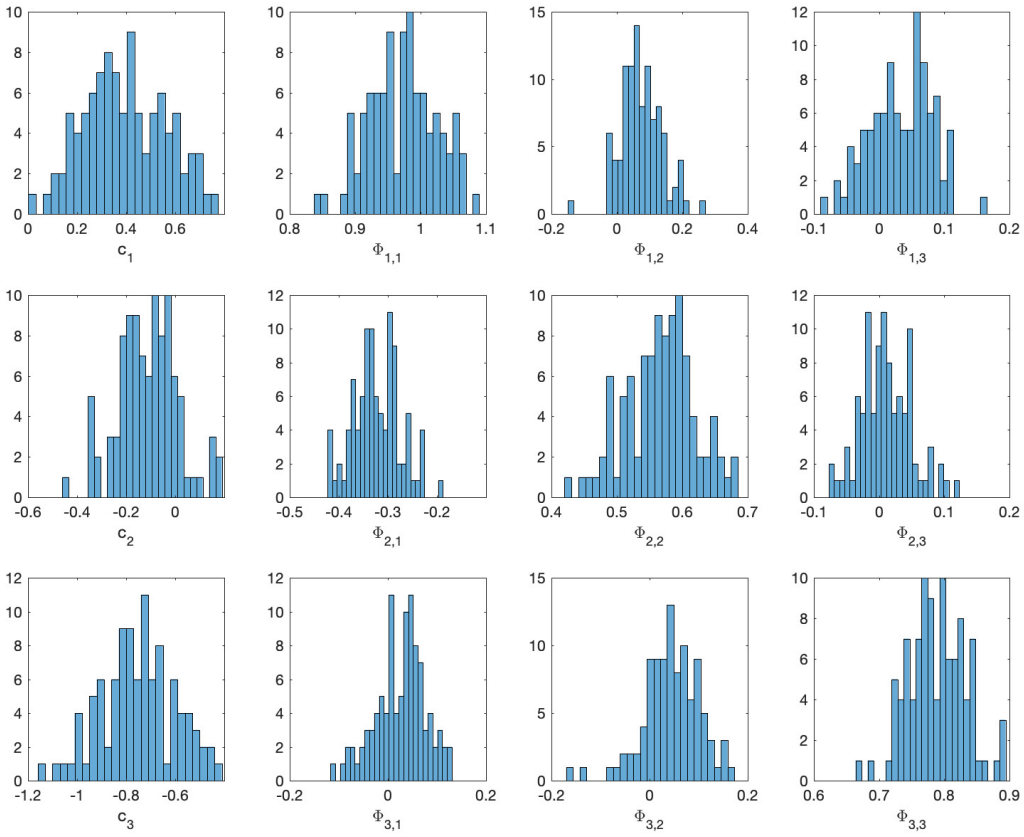

Figure 6: Histogram of DNS parameter estimates for 100 simulated data sets.

5 Conclusion

In this report the performance of the Dynamic Nelson Siegel model in estimating and forecasting a yield curve has been researched. Forecasts of the yields are compared against three benchmark models that can account for a variety of yield curve characteristics. We have found that the DNS model has been able to capture all six stylized facts of the yield curve in an in-sample setting. A simulation study confirms the ability of the DNS model to make relative accurate and robust in-sample estimations. In forecasting yields, the random walk often appears to produce the best results in terms of RMSFE, which might be caused by the larger amount of parameters that the VAR, ECM and DNS specifications contain.

A clear limitation of this research is that all models considered can be described as statistical. One would, however, expect yields to be closely related to the state of the economy. Therefore, an interesting extension would be to evaluate models that relate yields to the macroeconomy and financial markets.

References

Beber, A., Brandt, M. W., and Kavajecz, K. A. (2008). Flight-to-quality or flight-to-liquidity? evidence from the euro-area bond market. *The Review of Financial Studies*, 22(3):925–957.

Diebold, F. X. and Li, C. (2006). Forecasting the term structure of government bond yields. *Journal of econometrics*, 130(2):337–364.

Diebold, F. X. and Sharpe, S. A. (1990). Post-deregulation bank-deposit-rate pricing: The multivariate dynamics. *Journal of Business & Economic Statistics*, 8(3):281–291.

Ibanez, F. (2015). Calibrating the dynamic nelson-siegel model: A practitioner approach.

Pagan, A. (1996). The econometrics of financial markets. *Journal of empirical finance*, 3(1):15–102.

Van der Wel, M. (2019). Lecture 2: Yield curve construction, stylized facts, pca, and nelson-siegel.

Appendix

A DNS λ plot

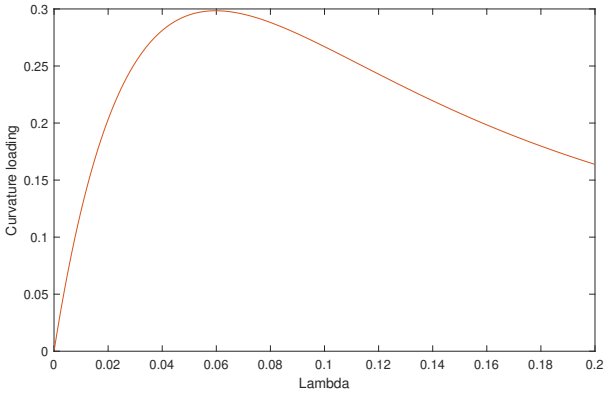

Figure 7: Loading of the curvature factor for λ between 0.0001 and 0.2.

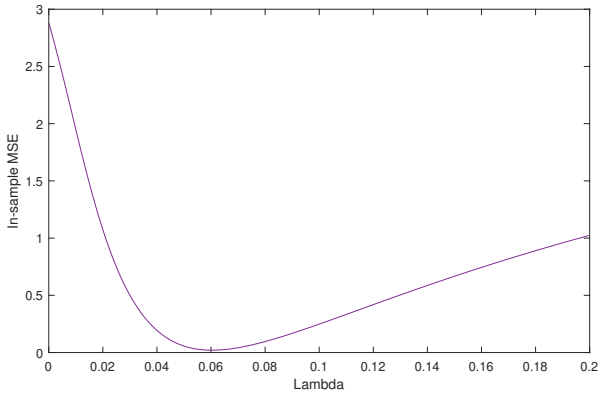

Figure 8: MSE of the DNS estimations for λ between 0.0001 and 0.2.

B Surface plot of empirical yields

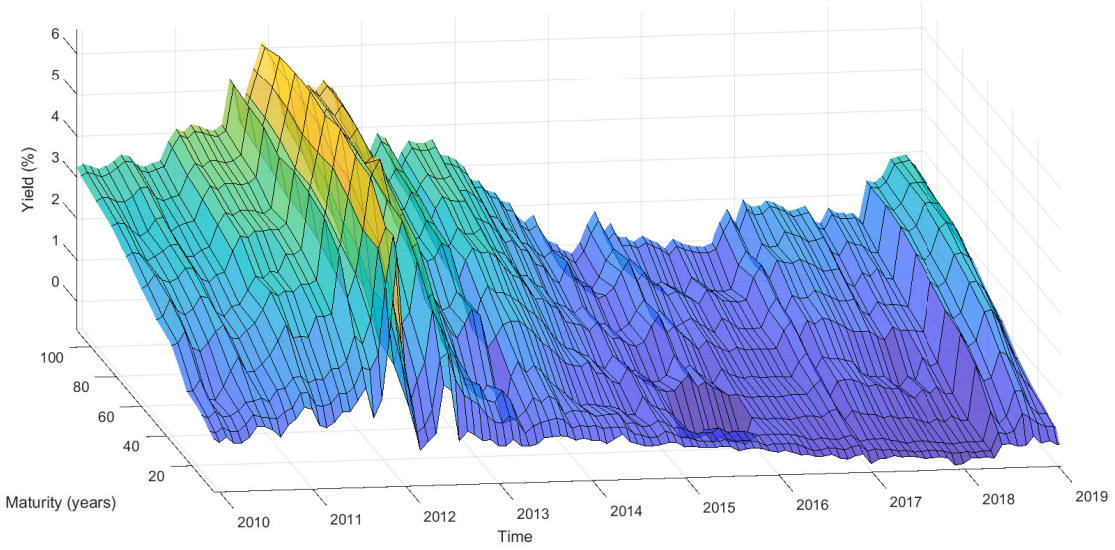

Figure 9: Italian yield curve data between November 2009 and November 2019 across fixed maturities

C Cross-correlation of empirical yields

Table 4: Cross-correlations of yields for different fixed maturities

height	3mth	6mth	1yr	2yr	3yr	4yr	5yr	6yr	7yr	8yr	9yr	10yr
3mth	1											
6mth	0.972	1										
1yr	0.941	0.987	1									
2yr	0.896	0.94	0.964	1								
3yr	0.88	0.935	0.97	0.974	1							
4yr	0.861	0.921	0.957	0.969	0.99	1						
5yr	0.843	0.909	0.948	0.949	0.987	0.991	1					
6yr	0.821	0.888	0.93	0.943	0.977	0.987	0.994	1				
7yr	0.826	0.893	0.933	0.938	0.977	0.985	0.995	0.997	1			
8yr	0.797	0.864	0.907	0.919	0.959	0.972	0.986	0.994	0.994	1		
9yr	0.794	0.86	0.905	0.916	0.957	0.97	0.984	0.992	0.993	0.997	1	
10yr	0.789	0.854	0.897	0.902	0.95	0.963	0.98	0.985	0.99	0.992	0.996	1

D Cross-correlation of in-sample DNS estimated yields

Table 5: Cross-correlations of yields estimated by DNS for different fixed maturities

	3mth	6mth	1yr	2yr	3yr	4yr	5yr	6yr	7yr	8yr	9yr	10y
3mth	1											
6mth	0.986	1										
1yr	0.945	0.987	1									
2yr	0.901	0.959	0.991	1								
3yr	0.879	0.940	0.978	0.996	1							
4yr	0.863	0.924	0.964	0.988	0.997	1						
5yr	0.848	0.908	0.948	0.977	0.991	0.998	1					
6yr	0.836	0.893	0.934	0.965	0.984	0.994	0.999	1				
7yr	0.825	0.881	0.921	0.954	0.976	0.989	0.996	0.999	1			
8yr	0.815	0.869	0.909	0.945	0.968	0.984	0.993	0.998	1.000	1		
9yr	0.807	0.860	0.899	0.936	0.962	0.979	0.99	0.995	0.998	1.000	1	
10y	0.800	0.852	0.891	0.929	0.956	0.975	0.986	0.993	0.997	0.999	1.000	1

E RMSFE of empirical yield curves

Table 6: Out-of-same 6-month-ahead forecasting results: Root Mean Square Forecasting Error

Maturity	Random Walk	VAR	ECM	DNS (1)	DNS (2)	DNS (3)
3mth	0.135	0.212	0.361	0.458	0.439	0.438
6mth	0.204	0.294	0.370	0.321	0.327	0.327
1yr	0.263	0.391	0.378	0.312	0.329	0.330
2yr	0.382	0.484	0.429	0.404	0.408	0.409
3yr	0.506	0.567	0.531	0.480	0.470	0.470
4yr	0.623	0.617	0.564	0.632	0.615	0.615
5yr	0.580	0.631	0.538	0.546	0.535	0.534
6yr	0.645	0.663	0.578	0.581	0.576	0.576
7yr	0.658	0.685	0.594	0.575	0.576	0.576
8yr	0.661	0.708	0.606	0.579	0.580	0.580
9yr	0.605	0.652	0.551	0.578	0.575	0.575
10yr	0.630	0.649	0.599	0.616	0.610	0.610

Note: The forecast period spans November 2013 to November 2019. DNS (1) refers to fixing $\lambda_t = \lambda = 0.0703$ (i.e. Diebold and Li (2006)), DNS (2) refers to fixing $\lambda_t = \lambda = 0.0609$ (i.e. minimizing the correlation between the slope and curvature loadings) and DNS (3) refers to a time-varying λ_t that minimizes squared errors.

Table 7: Out-of-same 12-month-ahead forecasting results: Root Mean Square Forecasting Error

Maturity	Random Walk	VAR	ECM	DNS (1)	DNS (2)	DNS (3)
3mth	0.195	0.301	0.497	0.526	0.510	0.508
6mth	0.310	0.404	0.493	0.417	0.421	0.421
1yr	0.404	0.500	0.513	0.464	0.472	0.473
2yr	0.557	0.588	0.591	0.597	0.599	0.599
3yr	0.715	0.648	0.687	0.676	0.673	0.673
4yr	0.867	0.825	0.773	0.830	0.826	0.826
5yr	0.823	0.676	0.664	0.692	0.688	0.688
6yr	0.922	0.797	0.806	0.743	0.741	0.741
7yr	0.922	0.749	0.791	0.673	0.676	0.676
8yr	0.940	0.868	0.924	0.689	0.691	0.691
9yr	0.835	0.779	0.813	0.688	0.685	0.685
10yr	0.872	0.737	0.847	0.714	0.709	0.708

Note: The forecast period spans November 2013 to November 2019. DNS (1) refers to fixing $\lambda_t = \lambda = 0.0703$ (i.e. Diebold and Li (2006)), DNS (2) refers to fixing $\lambda_t = \lambda = 0.0609$ (i.e. minimizing the correlation between the slope and curvature loadings) and DNS (3) refers to a time-varying λ_t that minimizes squared errors.

F Summary statistics of simulated data sets

Table 8: Mean of the summary statistics of the 100 simulated data sets.

Maturity	Mean	Std. Dev.	Minimum	Maximum	$\rho(1)$	$\rho(12)$
3	0.32	0.31	-0.22	1.99	0.74	0.09
6	0.32	0.32	-0.24	1.98	0.76	0.12
12	0.38	0.32	-0.20	1.94	0.79	0.15
24	0.65	0.31	0.04	1.82	0.84	0.20
36	0.96	0.29	0.34	1.74	0.87	0.24
48	1.25	0.28	0.63	1.88	0.89	0.26
60	1.49	0.28	0.87	2.11	0.90	0.26
72	1.69	0.29	1.06	2.32	0.90	0.25
84	1.85	0.30	1.18	2.49	0.90	0.24
96	1.98	0.31	1.23	2.63	0.90	0.23
108	2.08	0.32	1.24	2.74	0.89	0.23
120	2.17	0.32	1.24	2.83	0.89	0.22

G Robustness of simulation study

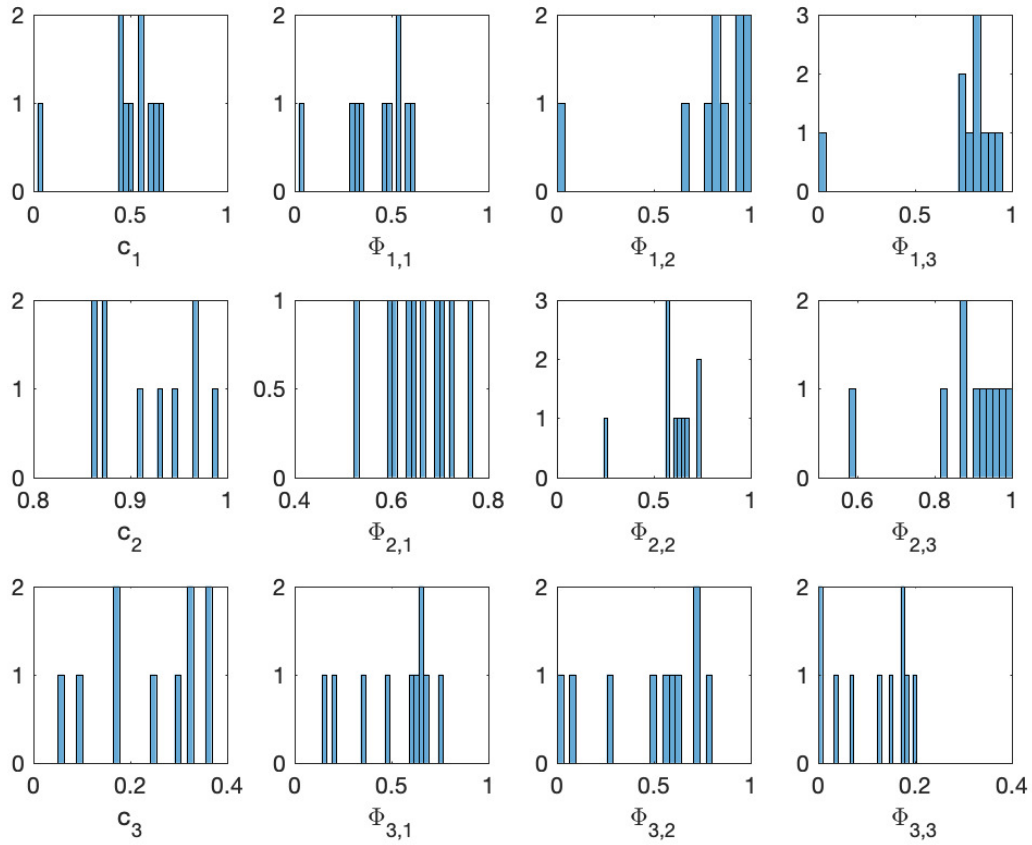

Figure 10: P-values of DNS estimations against true values after varying the noise the data generating process

H Code

```
%% Assignment 1 - Main

%% Get data
clc
clear
DATA = readtable('FinalDataMonthly10Y.csv', 'Format',...
    '%{dd/MM/yyyy}D %f %f %f %f %f %f %f %f %f %f %f %f');
tau = 12*[3/12, 6/12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]';
lambda = 0.0609; %Diebold-Li (2006)
yields = DATA{:,2:end}';
```

```
dates = DATA{:,1};
[m,T] = size(yields);
mWindow = 48; %In-sample of four years / 40% of the empirical data
R = 100;

%% Plot yield curve
surf(dates,tau,yields,'FaceAlpha',0.75)
ylabel('Maturity (years)');
xlabel('Time');
zlabel('Yield (%)');

%% Stationarity and Cointegration tests
preTests = preTestYields(yields, 5); % 5 lags

%% Simulation study lambda
lambda_in = 0.0001:0.0001:0.2;
[minCorr, maxB3, minMSE, ~] = lambda_simStudy(yields, tau, ...
    lambda_in, false); %Set last variable to true for plots

%% DNS estimation (2-step), FULL SAMPLE
DNS = DNS_2step(yields, lambda, tau, 1);

DNSCorr = DNS_2step(yields, minCorr(1), tau, 1);
plotDNS(DNSCorr, dates);

%% Dynamic Nelson-Siegel data generating process
simulatedYields = dgpDNS(tau, lambda, [1;1;1], 0.01, DNS{3}, 0.1, T, R);

%% DNS estimation (2-step), MOVING WINDOW

%Forecast: Dynamic Nelson-Siegel (2-step) using lambda = 0.0609
mwDNS_DiLi = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @
    DNS_2step);

%Forecast: Dynamic Nelson-Siegel (2-step) with time-varying lambda
mwDNS_timeVarying = movingWindow(mWindow, yields, lambda, tau, [1,6,12],...
```

```
    @timeVaryingDNS);

%Forecast: Dynamic Nelson-Siegel (2-step) using lambda as min-correlation
mwDNS_Corr = movingWindow(mWindow, yields, minCorr(1), tau, [1,6,12], @
    DNS_2step);

%Forecast: Random walk
mwRW = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @random_walk);

%Forecast: AR(1)
mwAR1 = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @AR1);

%Forecast: VAR(1)
mwVAR1 = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @VAR1);

%Forecast: ECM(1)
mwECM = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @ECM);

%Forecast: VECM(1)
mwVECM = movingWindow(mWindow, yields, lambda, tau, [1,6,12], @VECM);

%% Estimation results

% RMSFE: 1 month forecast length
RMSFE_1m = [mwRW{end}(:,1), mwAR1{end}(:,1), mwVAR1{end}(:,1), ...
    mwECM{end}(:,1), mwVECM{end}(:,1), ...
    mwDNS_DiLi{end}(:,1), mwDNS_timeVarying{end}(:,1), mwDNS_Corr{end}(:,1)
        ];

% RMSFE: 6 month forecast length
RMSFE_6m = [mwRW{end}(:,2), mwAR1{end}(:,2), mwVAR1{end}(:,2), ...
    mwECM{end}(:,2), mwVECM{end}(:,2), ...
    mwDNS_DiLi{end}(:,2), mwDNS_timeVarying{end}(:,2), mwDNS_Corr{end}(:,2)
        ];

% RMSFE: 12 month forecast length
```

```
RMSFE_12m = [mwRW{end}(:,3), mwAR1{end}(:,3), mwVAR1{end}(:,3), ...
    mwECM{end}(:,3), mwVECM{end}(:,3), ...
    mwDNS_DiLi{end}(:,3), mwDNS_timeVarying{end}(:,3), mwDNS_Corr{end}(:,3)
        ];

%% Model evaluation for simulated yields

DNS_sim = evaluateSimulations(simulatedYields, minCorr(1), tau, [1,6,12],...
    @DNS_2step);
meanStats = mean(DNS_sim{1},3);
[statsBeta, statsPhi] = statsPlots(DNS_sim, 0);

%% Robustness check for simulation parameters
t_stats = robustnessSim([tau, minCorr(1), {[1;1;1]}, 0.01, DNS{3}, 0.1,...
    T, R], @dgpDNS, @DNS_2step, 10, true);
```

```
function [output] = preTestYields(yields, lags)
%ADFTESTYIELDS performs the ADF stationarity test and Engle-Granger
%cointegration test

[m,T] = size(yields);
L = length(lags);
IsStationary = nan(m,L);
IsCointegrated = nan(m,1);
pValue_ADF = nan(m,L);
pValue_EG = nan(m,1);

% perform tests
for s = 1:m
    % ADF: 1 indicates stationarity | last input is most recent observation
    [IsStationary(s,:), pValue_ADF(s,:)] = adftest(yields(s,:),'lags',lags)
        ;

    % EGCI: 1 indicates cointegration | last input is most recent
        observation
    [IsCointegrated(s,:), pValue_EG(s,:)] = egcitest(([yields(s,2:end);
        yields(s,1:end-1)])', 'lags', max(lags));
```

```
end

output = cell(2,2);
output{1,1} = IsStationary;
output{1,2} = pValue_ADF;
output{2,1} = IsCointegrated;
output{2,2} = pValue_EG;

end
```

```
function [ minCorr, maxB3, minMSE, simulations ] = lambda_simStudy( yields,
    tau, lambda_in, plotBool)
%LAMBDA_SIMSTUDY
[m,T] = size(yields);
[DNS] = DNS_2step(yields, 0.0609, tau, []);
betasFull = DNS{2};
%Optimize lambda for min correlation, min MSE and max curvature loading
funCorr = @(lambda_in)(corr((1-exp(-tau*lambda_in))./(tau*lambda_in),...
    (1-exp(-tau*lambda_in))./(tau*lambda_in) - exp(-tau*lambda_in))).^2;
funB3 = @(lambda_in)-1*((1-exp(-30*lambda_in))/(30*lambda_in)...
    - exp(-30*lambda_in));
funErr = @(lambda_in)sum(sum((yields - [ones(size(tau)),...
    (1-exp(-lambda_in*tau))./(lambda_in*tau),(1-exp(-lambda_in*tau))./...
    (lambda_in*tau) - exp(-lambda_in*tau)] * betasFull).^2, 1), 2)/m/T;

%Set initial parameters
lambda0 = 0.1;
A = [];
b = [];
Aeq = [];
beq = [];
lb = 0.0001;
ub = 2;
nonlcon = [];
options = optimoptions('fmincon', 'TolFun', 1e-10);

[lambda_Corr, fvalCorr] = fmincon(funCorr, lambda0, A, b, Aeq, beq, lb,...
```

```
    ub, nonlcon, options);
minCorr = [lambda_Corr, fvalCorr];
[lambda_B3, fvalB3] = fmincon(funB3, lambda0, A, b, Aeq, beq, lb,...
    ub, nonlcon, options);
maxB3 = [lambda_B3, -fvalB3];
[lambda_Err, fvalMSE] = fmincon(funErr, lambda0, A, b, Aeq, beq, lb,...
    ub, nonlcon, options);
minMSE = [lambda_Err, fvalMSE];

%Simulation: correlation between slope and curvature loadings (and MSE)
n = size(lambda_in,2);
B2 = (1-exp(-tau*lambda_in))./(tau*lambda_in);
B3 = (1-exp(-tau*lambda_in))./(tau*lambda_in) - exp(-tau*lambda_in);
B3_30 = (1-exp(-30*lambda_in))./(30*lambda_in) - exp(-30*lambda_in);

corrB2B3 = zeros(size(lambda_in));
MSE = zeros(size(lambda_in));
for i=1:n
    corrB2B3(i) = corr(B2(:,i),B3(:,i));
    MSE(i) = sum(sum((yields - [ones(size(tau)),(1-exp(-lambda_in(i)...
        *tau))./(lambda_in(i)*tau),(1-exp(-lambda_in(i)*tau))./...
        (lambda_in(i)*tau) - exp(-lambda_in(i)*tau)]...
        * betasFull).^2, 1), 2)/m/T;
end

simulations = [corrB2B3', B3_30', MSE'];

%Plot simulation study
if (plotBool == true)
    figure(1)
    plot(lambda_in, corrB2B3); %Correlation between slope and curvature
       loading
    ylabel('Correlation');
    xlabel('Lambda');
    figure(2)
    plot(lambda_in, B3_30, 'color',[0.8500, 0.3250, 0.0980] );
```

```
    ylabel('Curvature loading');
    xlabel('Lambda');
    figure(3)
    plot(lambda_in, MSE, 'color', [0.4940, 0.1840, 0.5560]);
    ylabel('In-sample MSE');
    xlabel('Lambda');
end
end
```

```
function [output] = random_walk( yields, lambda, tau, steps_ahead )
% RANDOM_WALK returns a random walk forecasted yield of k steps ahead
F = length(steps_ahead);

yield_forecasts = repmat(yields(:,end),[1,F]);
output = cell(1,1);
output{1,1} = yield_forecasts;
end
```

```
function [output] = AR1( yields, lambda, tau, steps_ahead )
% AR1 returns a forecasted yield of k steps ahead
[m, T] = size(yields);
F = length(steps_ahead);
yield_forecasts = nan(m,F);
X = yields(:,1:end-1);
Y = yields(:,2:end);
phi = nan(m,2);

% make estimations for every maturity and forecast length
for tau = 1:m
    phi(tau,:) = [ones(1,T-1);X(tau,:)]'\Y(tau,:)';
    for s = 1:F
        step = steps_ahead(s);
        yield_forecasts(tau, s) = phi(tau,1)*(1+sum(phi(tau,2).^[1:step-1])
            )+X(tau,end)*phi(tau,1)^step;
    end
end

```

```
output = cell(1,1);
output{1} = yield_forecasts;

end
```

```
function [output] = VAR1( yields, lambda, tau, steps_ahead )
% VAR1 forecasts yields using a VAR(1) model

[m,T] = size(yields);
F = length(steps_ahead);
yield_forecasts = nan(m,F);

% estimate VAR(1) parameters
Y = yields(:,2:end);
X = [ones(1,T-1);yields(:,1:end-1)];
PHI = Y*X'*inv(X*X');

% evaluating forecasts for different lengths
for f=1:F
    y = yields(:,end);
    step = steps_ahead(f);
    % recursive forecasting until forecast length is reached
    for s=1:f
        y = PHI*[1;y];
    end
    yield_forecasts(:,f) = y;
end

output = cell(1);
output{1} = yield_forecasts;
end
```

```
function [ output ] = ECM(yields, lambda, tau, steps_ahead)
% ECM forecasts yields using the error-correction model (1)

[m,T] = size(yields);
F = length(steps_ahead);
```

```
yield_forecasts = nan(m,F);

for tau=1:m
    % Estimate cointegration relation of yield_t and yield_(t-1)
    X = [ones(T-1,1),yields(tau,1:end-1)'];
    Y = yields(tau,2:end)';
    b = X\Y;

    % Estimate error-correction model
    error = Y - X*b;
    dX = [yields(tau,2:end-1) - yields(tau, 1:end-2)]';
    Z = [dX, -error(1:end-1)];
    dY = [yields(tau,3:end) - yields(tau, 2:end-1)]';
    PHI = Z\dY; %PHI = [phi; gamma]

    %     % forecast yield / dY specification
    %     for f=1:F
    %         error_f = error(end);
    %         Z_f = [yields(tau,end)-yields(tau,end-1), -error_f];
    %         dyield_sum = 0;
    %         for s = 1:steps_ahead(f)
    %             dyield = Z_f*PHI;
    %             dyield_sum = dyield_sum + dyield;
    %             error_f = (1-PHI(2))*error_f; %estimate next error using
       rho = 1-gamma
    %             Z_f = [dyield, -error_f];
    %         end
    %         yield_forecasts(tau,f) = yields(tau,end) + dyield_sum;
    %     end

    % forecast yield / Y specification
    for f=1:F
        error_f = error(end);
        X_f = [1,yields(tau,end)];
        for s = 1:steps_ahead(f)
            Y_f = X_f*b + error_f; %predict next yield, correcting for
```

```
                error
            error_f = (1-PHI(2))*error_f; %forecast next error
            X_f = [1,Y_f]; %lagged yield as explanatory variable
        end
        yield_forecasts(tau,f) = Y_f;
    end

    output = cell(1);
    output{1} = yield_forecasts;

end

end
```

```
function [ output ] = VECM( yields, lambda, tau, steps_ahead )
% VECM returns forecasted yields using the vector error-correction model
% (1)

[m,T] = size(yields);
F = length(steps_ahead);
yield_forecasts = nan(m,F);

dY = yields(:,2:end) - yields(:,1:end-1);
beta = [-ones(1,m-1); eye(m-1)]; %cointegration relation
S = beta'*yields(:,1:end-1);
Z = [ones(1,T-1); S];
params = dY*Z'*inv(Z*Z'); %[mu; alpha]

for f=1:F
    Zf = [1; beta'*yields(:,end)];
    yields_f=yields(:,end);
    for s=1:steps_ahead(f)
        dyield = params*Zf;
        yields_f = yields_f + dyield;
        Sf = beta'*yields_f;
        Zf = [1;Sf];
    end
```

```
    yield_forecasts(:,f) = yields_f;
end

output = cell(1);
output{1} = yield_forecasts;

end
```

```
function [ output ] = DNS_2step( yields, lambda, tau, steps_ahead )
%DNS_2STEP
[m, T] = size(yields);

%Initialize Beta and B loadings
B = [ones(size(tau)), (1-exp(-lambda*tau))./(lambda*tau),...
    (1-exp(-lambda*tau))./(lambda*tau) - exp(-lambda*tau)];
beta = zeros(3,T);

%Estimate betas for each t by cross-sectional OLS
for t=1:T
    beta(:,t) = B\yields(:,t);
end

%Estimate VAR(1) model for betas
Y = beta(:,2:end);
X = [ones(1, T-1) ;beta(:,1:end-1)];
phi = Y*X' * inv(X*X');

%Forecast using DNS model
F = size(steps_ahead,2);
yield_forecasts = zeros(m,F);
beta_forecasts = zeros(3,F);
for f=1:F
    b = beta(:,end);
    step = steps_ahead(f);
    for s=1:step
        b = phi*[1;b];
    end
```

```
    yield_forecasts(:,f) = B*b;
    beta_forecasts(:,f) = b;
end

output = cell(3,1);
output{1} = yield_forecasts;
output{2} = beta;
output{3} = phi;

end
```

```
function [ output ] = timeVaryingDNS( yields, lambda, tau, steps_ahead )
%TIMEVARYINGDNS
[m,T] = size(yields);

%Estimate betas given initial lambda
outputDNS = DNS_2step(yields, lambda, tau, steps_ahead);
betas = outputDNS{2};

%Optimize lambda based on the estimated betas
funErr = @(lambda_in)sum(sum((yields - [ones(size(tau)),...
    (1-exp(-lambda_in*tau))./(lambda_in*tau),(1-exp(-lambda_in*tau))./...
    (lambda_in*tau) - exp(-lambda_in*tau)] * betas).^2, 1), 2)/m/T;

lambda0 = 0.1;
A = [];
b = [];
Aeq = [];
beq = [];
lb = 0.0001;
ub = 2;
nonlcon = [];
options = optimoptions('fmincon', 'TolFun', 1e-10);

[optLambda, ~] = fmincon(funErr, lambda0, A, b, Aeq, beq, lb,...
    ub, nonlcon, options);

```

```
output = DNS_2step(yields, optLambda, tau, steps_ahead);
end
```

```
function [ output ] = movingWindow(w, yields, lambda, tau, steps_ahead, FUN
    )
%MOVINGWINDOW
%Initialize variables
[M,T] = size(yields);
S = numel(steps_ahead);
forecasts = cell(S,1);
for s=1:S
    forecasts{s} = zeros(M,T-w);
end

%Forecast yields over moving window
for i=1:(T-w+1)
    window = i:(i+w-1);
    yields_window = yields(:, window);

    model = FUN(yields_window, lambda, tau, steps_ahead);
    for s=1:S
        forecasts{s}(:,i) = model{1}(:,s);
    end
end

%Compute RMSFE
RMSFE = zeros(M,S);
output = cell(S+1,1);

for s=1:S
    step = steps_ahead(s);
    actuals = yields(:,(w+step):end);
    forecast_s = forecasts{s}(:,1:(end-step));
    RMSFE(:,s) = mean(sqrt((actuals-forecast_s).^2),2);
    %     MSFE(:,s) = mean((actuals-forecast_s).^2,2);
    output{s} = forecasts{s};
end
```

```
34
35  output{S+1} = RMSFE;
36  % output{S+2} = MSFE
37
38  end
```

```
1   function [ ] = plotDNS( DNS, dates )
2   %PLOTDNS
3   beta = DNS{2};
4   subplot(3,1,1);
5   plot(dates(1:120), beta(1,1:(end-1)), 'color', 'r');
6   ylabel('Level factor');
7   subplot(3,1,2);
8   plot(dates(1:120), beta(2,1:(end-1)), 'color', 'b');
9   ylabel('Slope factor');
10  subplot(3,1,3);
11  plot(dates(1:120), beta(3,1:(end-1)), 'color', 'black');
12  ylabel('Curvature factor');
13  end
```

```
1   function [ yieldsEst ] = dgpDNS( tau, lambda, beta0, var1, phi, var2, T, R
        )
2   %DGPDNS
3   %Initialize variables
4   [m,n] = size(phi);
5   var2 = var2*eye(m);
6   B = [ones(size(tau)), (1-exp(-lambda*tau))./(lambda*tau),...
7       (1-exp(-lambda*tau))./(lambda*tau) - exp(-lambda*tau)];
8   y = zeros(size(tau))';
9   yieldsEst = cell(R,1);
10
11  % Run R simulations of the DNS dgp
12  for r = 1:R
13      %Set beta to initial value beta0
14      beta = beta0;
15      yieldsEst{r} = zeros(size(B,1), T);
16      for t = 1:T
```

```
        % Simulate error terms
         nu = mvnrnd(zeros(m, 1), var2)';   % correlated nu

        % Evaluate DNS dynamics
        var_y = var1;
        for i = 1:size(tau,1)
            eps = normrnd(0, sqrt(var_y));
            y(i) = B(i,:)*beta + eps;
            var_y = 0.9*var_y;
        end
        beta = phi*[1; beta] + nu;
        yieldsEst{r}(:, t) = y;
    end
end

end
```

```
function [ output ] = evaluateSimulations( simYields, lambda, tau,
    steps_ahead, FUN)
%EVALUATESIMULATIONS
[R,~] = size(simYields);
[m,T] = size(simYields{1});
output = cell(3,1);
summaryStats = zeros(m,6,R);
beta = zeros(3,T,R);
phi = zeros(3,4,R);
for r=1:R
    yields = simYields{r};
    model = FUN(yields, lambda, tau, steps_ahead);
    beta(:,:,r) = model{2};
    phi(:,:,r) = model{3};
    summaryStats(:,:,r) = summaryStatistics(yields);
end
output{1} = summaryStats;
output{2} = beta;
output{3} = phi;

```

```
end
```

```
function [ beta, phi ] = statsPlots( sims, plot )
%STATSPLOTS
R = 100;
[I,J] = size(sims{3}(:,:,1));
stdBetas = zeros(I,1);
stdPhi = zeros(I,J);
skewBetas = zeros(I,1);
skewPhi = zeros(I,J);
kurtBetas = zeros(I,1);
kurtPhi = zeros(I,J);

minBetas = zeros(I,1);
maxBetas = zeros(I,1);
ac1Betas = zeros(I,1);
ac12Betas = zeros(I,1);

beta = cell(8,1);
phi = cell(4,1);
beta{1} = mean(mean(sims{2},2),3);
phi{1} = mean(sims{3},3);

for i=1:I
    b = sims{2}(i,:,:);
    stdBetas(i) = std(b(:));
    skewBetas(i) = skewness(b(:));
    kurtBetas(i) = kurtosis(b(:));

    minBetas(i) = mean(min(b,[],2),3);
    maxBetas(i) = mean(max(b,[],2),3);
    ac1

    for j=1:J
        stdPhi(i,j) = std(sims{3}(i,j,:));
        skewPhi(i,j) = skewness(sims{3}(i,j,:));
        kurtPhi(i,j) = kurtosis(sims{3}(i,j,:));
```

```
36
37      end
38  end
39
40
41  beta{2} = stdBetas;
42  beta{3} = skewBetas;
43  beta{4} = kurtBetas;
44
45  phi{2} = stdPhi;
46  phi{3} = skewPhi;
47  phi{4} = kurtPhi;
48
49  if plot == 1
50      subplot(3,1,1);
51      histogram(mean(sims{2}(1,:,:),3),R/2);
52      subplot(3,1,2);
53      histogram(mean(sims{2}(2,:,:),3),R/2);
54      subplot(3,1,3);
55      histogram(mean(sims{2}(3,:,:),3),R/2);
56  elseif plot == 2
57      subplot(3,4,1);
58      histogram(sims{3}(1,1,:),R/4);
59      xlabel('c_1');
60      subplot(3,4,2);
61      histogram(sims{3}(1,2,:),R/4);
62      xlabel('\Phi_{1,1}');
63      subplot(3,4,3);
64      histogram(sims{3}(1,3,:),R/4);
65      xlabel('\Phi_{1,2}');
66      subplot(3,4,4);
67      histogram(sims{3}(1,4,:),R/4);
68      xlabel('\Phi_{1,3}');
69      subplot(3,4,5);
70      histogram(sims{3}(2,1,:),R/4);
71      xlabel('c_2');
```

```
    subplot(3,4,6);
    histogram(sims{3}(2,2,:),R/4);
    xlabel('\Phi_{2,1}');
    subplot(3,4,7);
    histogram(sims{3}(2,3,:),R/4);
    xlabel('\Phi_{2,2}');
    subplot(3,4,8);
    histogram(sims{3}(2,4,:),R/4);
    xlabel('\Phi_{2,3}');
    subplot(3,4,9);
    histogram(sims{3}(3,1,:),R/4);
    xlabel('c_3');
    subplot(3,4,10);
    histogram(sims{3}(3,2,:),R/4);
    xlabel('\Phi_{3,1}');
    subplot(3,4,11);
    histogram(sims{3}(3,3,:),R/4);
    xlabel('\Phi_{3,2}');
    subplot(3,4,12);
    histogram(sims{3}(3,4,:),R/4);
    xlabel('\Phi_{3,3}');
elseif plot == 0

end


end
```

```
function [ output ] = summaryStatistics( yields )
%SUMMARYSTATISTICS
[m,T] = size(yields);

% Simple summary statistics
mu = mean(yields,2);
std_dev = std(yields')';
mi = min(yields')';
ma = max(yields')';
```

```
10
11  % Autocorrelations
12  lags = [1,12];
13  ac = zeros(m,lags(end)+1);
14  for i=1:m
15      ac(i,:) = autocorr(yields(i,:), lags(end))';
16  end
17  ac = ac(:,1+lags);
18
19  output = [mu, std_dev, mi, ma, ac];
20  end
```

```
1   function [ t_test ] = robustnessSim( params, DGP, MDL, N, plot )
2   %ROBUSTNESSSIM
3   %Select parameters for dgp
4   tau = params{1};
5   lambda = params{2};
6   beta0 = params{3};
7   sigma1 = params{4};
8   phi = params{5};
9   sigma2 = params{6};
10  T = params{7};
11  R  = params{8};
12
13  t_test = cell(2,1);
14  tStat = zeros(3,4,N);
15  pVal = zeros(3,4,N);
16  sigma2 = [0.001:(1/N):2];
17  for i=1:N
18      yields = DGP(tau, lambda, beta0, sigma1, phi, sigma2(i), T, R);
19      mdlEst = evaluateSimulations(yields, lambda, tau, 1, MDL);
20      [~, simPhi] = statsPlots(mdlEst, 0);
21      tStat(:,:,i) = (simPhi{1} - phi)./(simPhi{2});
22      pVal(:,:,i) = 2*(1-tcdf(abs(tStat(:,:,i)), R-1));
23  end
24  t_test{1} = tStat;
25  t_test{2} = pVal;
```

```
26
27 if plot == true
28          subplot(3,4,1);
29      histogram(pVal(1,1,:),R/4);
30      xlabel('c_1');
31      subplot(3,4,2);
32      histogram(pVal(1,2,:),R/4);
33      xlabel('\Phi_{1,1}');
34      subplot(3,4,3);
35      histogram(pVal(1,3,:),R/4);
36      xlabel('\Phi_{1,2}');
37      subplot(3,4,4);
38      histogram(pVal(1,4,:),R/4);
39      xlabel('\Phi_{1,3}');
40      subplot(3,4,5);
41      histogram(pVal(2,1,:),R/4);
42      xlabel('c_2');
43      subplot(3,4,6);
44      histogram(pVal(2,2,:),R/4);
45      xlabel('\Phi_{2,1}');
46      subplot(3,4,7);
47      histogram(pVal(2,3,:),R/4);
48      xlabel('\Phi_{2,2}');
49      subplot(3,4,8);
50      histogram(pVal(2,4,:),R/4);
51      xlabel('\Phi_{2,3}');
52      subplot(3,4,9);
53      histogram(pVal(3,1,:),R/4);
54      xlabel('c_3');
55      subplot(3,4,10);
56      histogram(pVal(3,2,:),R/4);
57      xlabel('\Phi_{3,1}');
58      subplot(3,4,11);
59      histogram(pVal(3,3,:),R/4);
60      xlabel('\Phi_{3,2}');
61      subplot(3,4,12);
```

```
    histogram(pVal(3,4,:),R/4);
    xlabel('\Phi_{3,3}');
end

end
```

```
import pandas as pd
import numpy as np

# read in data
df = pd.read_csv('data.csv')

# remove date column and maturities >10yr
df = df[df.columns[1:13]]
df.columns = ['3', '6', '12', '24', '36', '48', '60', '72', '84', '96',
              '108', '120 (level)']
# add slope and curvature (according to Diebold and Li, 2006)
df['Slope'] = df['120 (level)'] - df['3']
df['Curvature'] = 2*df['24'] - (df['3'] + df['120 (level)'])

# Autocorrelation
def autocorr(x, t=1):
    return np.corrcoef(np.array([x[:-t], x[t:]]))

# Calculate autocorrelations
autocorrel = []
for term in df.columns: # Loop through all terms
    autocorrelTerm = []
    for t in [1, 12]:
        autocorrelTerm.append(autocorr(df[term], t)[0,1])
    autocorrel.append(autocorrelTerm)

# create dataframe with summary statistics
summaryDf = pd.DataFrame(autocorrel, columns=['AC 1m', 'AC 12m'],
                         index=[df.columns])
summaryDf['Mean'] = df.mean().to_list()
summaryDf['Std. dev.'] = df.std().to_list()
```

```
summaryDf['Minimum'] = df.min().to_list()
summaryDf['Maximum'] = df.max().to_list()
summaryDf = summaryDf[['Mean', 'Std. dev.', 'Minimum', 'Maximum', 'AC 1m',
                       'AC 12m']].round(3)
summaryDf.to_csv('SummaryStatisticsYieldCurve.csv')
print(summaryDf)

# create dataframe with cross-correlations
corrDf = df[['3', '6', '12', '24', '36', '48', '60', '72', '84', '96',
             '108', '120 (level)']].corr().round(3)
corrDf.to_csv('CorrelationYieldCurve.csv')
print(corrDf)
```