

Self-driving neural networks for yield curve modelling*

Sicco Kooiker^(a) *Janneke van Brummelen*^(a,b)

Julia Schaumburg^(a,b) *Marcin Zamojski*^(a)

^(a) Vrije Universiteit Amsterdam ^(b) Tinbergen Institute

December 2025

Abstract

In this paper, we introduce a new factor model with time-varying loading matrix, that is particularly useful for modeling and forecasting term structure data. While maintaining the interpretability of the factors as level, slope and curvature via suitable identification restrictions, we allow for flexible shapes of the factor loadings by specifying them as neural networks with time-varying weights. We apply this model to U.S. government bond yields and document its superiority in predicting yields out-of-sample, relative to well-known benchmarks such as the dynamic Nelson Siegel model and the random walk.

Keywords: time-varying neural networks, score-driven model, yield curve.

*Email addresses: marcin@zamojski.net (Marcin Zamojski), j.schaumburg@vu.nl (Julia Schaumburg), j.n.van.brummelen@vu.nl (Janneke van Brummelen), s.h.kooiker@vu.nl (Sicco Kooiker). Schaumburg and Kooiker thank the Dutch Science Foundation (NWO, grant VI.VIDI.191.169) for financial support.

1 Introduction

Accurate modeling and forecasting of the yield curve is crucial in finance and macroeconomics. Bond portfolio management, risk management, and pricing of derivatives all rely on accurate predictions of interest rates both in the time dimension and in the cross-section. The yield curve also sheds light on how monetary policy affects inflationary expectations of agents, depending on the state of the economy.

In practice, the most widely used bond yield forecasting models are linear models. They include, on the one hand, dynamic factor models in the spirit of [Nelson and Siegel \(1987\)](#), see, for instance, [Diebold et al. \(2005\)](#), [Diebold and Li \(2006\)](#), [Christensen et al. \(2011\)](#), and [Koopman et al. \(2010\)](#). On the other hand, predictive regressions using principal components as proposed in [Stock and Watson \(2002\)](#) provide a simple yet powerful dimension reduction method to forecast bond yields. Even though linear models are often computationally inexpensive, while delivering interpretable outcomes and good forecasting performance, more complex nonlinear approaches have recently gained popularity in the asset pricing literature.¹ In particular, [Bianchi et al. \(2021\)](#) show in a comparative study that even shallow neural networks are able to predict bond excess returns significantly more accurately than their linear benchmarks.

In this paper, we introduce a new factor model with time-varying loading matrix, that is particularly useful for modeling and forecasting term structure data. While maintaining the interpretability of the factors as level, slope and curvature via suitable identification restrictions, we allow for flexible shapes of the factor loadings by specifying them as neural networks with time-varying weights. We apply this model to U.S. government bond yields and document its superiority in predicting yields out-of-sample, relative to well-known benchmarks such as the dynamic Nelson Siegel model and the random walk.

The dynamics of both loading coefficients and latent factors are specified as functions of past observations. In particular, we follow [Creal et al. \(2024\)](#) in using the influence function of a conditional moments-based objective function as the driving variable of our time-varying parameters. Our approach is computationally efficient despite the nonlinear functional form of the neural network components. The derivatives are calculated using the self-driving score filter methodology [Zamojski \(2019\)](#), which only requires a criterion

¹There is a growing literature focusing on modeling stock returns using neural networks, for example, [Gu et al. \(2020\)](#), [Chen et al. \(2024\)](#), [Avramov et al. \(2023\)](#), [Gu et al. \(2021\)](#)

function as input, and is fast and easy to implement.

In addition to contributing to the literature on yield curve modeling and forecasting, our work also relates to the literature on factor models with time-varying loadings. In the context of predicting macroeconomic variables, factor models with constant loadings are very common, including static factors based on principal components (e.g., [Stock and Watson, 2002](#); [Bai and Ng, 2013](#), and many others) and dynamic versions using state space methods and either Bayesian ([Kose et al., 2003](#); [Bernanke et al., 2003](#); [Bai and Wang, 2015](#)) or frequentist estimation procedures ([Doz et al., 2012](#); [Jungbacker and Koopman, 2015](#); [Barigozzi and Luciani, 2024](#)). Taking into account the potential instability of models over time due to, for instance, changes in market environments and monetary policy, several papers have proposed extensions to the classic factor model, allowing for dynamics in the coefficients. For instance, [Negro and Otrok \(2008\)](#) study changes in international business cycles. They specify loading coefficients and variances as random walks and propose a Gibbs sampling technique for filtering and estimation. [Mikkelsen et al. \(2019\)](#), on the other hand, cast the factor model in a state space form, first extract principal components based on constant loadings, and then estimate separate vector autoregressions for each row of the loading matrix. [Su and Wang \(2017\)](#) model factor loadings as a smooth function of time, giving rise to local principal components, and [Cheung \(2024\)](#) formally studies identification conditions for factor models based on such local principal components and time-varying loadings.

Our factor model differs from these settings because it is specifically designed to model bond yield data at different maturities, which gives rise to meaningful identification restrictions at each point in time. Therefore, we avoid the problem of only achieving identification up to a rotation matrix at each time point. Crucially, while we maintain the interpretation of the first three yield curve factors as level, slope, and curvature, our dynamic neural network-based estimation approach allows for substantially more flexibility than the static structure imposed by the (dynamic) Nelson Siegel-type models of [Nelson and Siegel \(1987\)](#), [Diebold and Li \(2006\)](#) and [Diebold et al. \(2005\)](#).

The remainder of the paper is organized as follows. Section 2 presents the neural network factor model and the dynamic updating equations for both factors and loading matrices. Furthermore, we discuss the restrictions we impose to achieve unique identification. Section 3 presents an empirical application to U.S. yield data on 24 maturities.

Both in-sample and out-of-sample predictions from our model are compared to a variety of benchmarks. Section 4 concludes.

2 Methodology

2.1 Neural Factor Model

Let $\mathbf{y}_t = (y_{1t}, \dots, y_{Nt})'$ denote the vector of zero-coupon yields observed at the ordered maturities $\boldsymbol{\tau} = (\tau_1, \dots, \tau_N)'$ with $0 < \tau_1 < \dots < \tau_N$. We assume that yields are represented by a K -factor model

$$\mathbf{y}_t = \Gamma(\boldsymbol{\tau}; \boldsymbol{\gamma}_t) \boldsymbol{\beta}_t + \boldsymbol{\varepsilon}_t, \quad (1)$$

where $K \leq N$, $\boldsymbol{\beta}_t \in \mathbb{R}^K$ collects the factors, and $\Gamma(\boldsymbol{\tau}; \boldsymbol{\gamma}_t) \in \mathbb{R}^{N \times K}$ is a loading matrix parametrised by $\boldsymbol{\gamma}_t$. The disturbance vector $\boldsymbol{\varepsilon}_t \in \mathbb{R}^N$ is serially independent with $\mathbb{E}[\boldsymbol{\varepsilon}_t] = \mathbf{0}$ and $\text{Var}(\boldsymbol{\varepsilon}_t) = \sigma_t^2 \mathbf{I}_N$.

Each column of the loading matrix is generated by an independent feed-forward neural network:

$$\Gamma_{ij}(\tau_i; \boldsymbol{\gamma}_t) = \text{NN}(\tau_i; \boldsymbol{\gamma}_{jt}), \quad i = 1, \dots, N, j = 1, \dots, K. \quad (2)$$

The specification of the single-layered neural network is

$$\text{NN}(\tau; \boldsymbol{\gamma}) = \sum_{l=1}^L w_{2l} \phi(w_{1l} \tau + b_l), \quad (3)$$

where $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is an activation function (e.g., sigmoid, tanh, or ReLU) and $\boldsymbol{\gamma} = (w_{11}, \dots, w_{1L}, b_1, \dots, b_L, w_{21}, \dots, w_{2L})' \in \mathbb{R}^{3L}$ collects the weights and biases. Stacking the parameters of the K networks gives the stacked vector $\boldsymbol{\gamma}_t = (\boldsymbol{\gamma}'_{1t}, \dots, \boldsymbol{\gamma}'_{Kt})'$. The neural networks are of the one-input, one-output type, $\text{NN}(\cdot; \boldsymbol{\gamma}) : \mathbb{R}_+ \rightarrow \mathbb{R}$. The scalar maturity is the input and the corresponding factor loading is the output. We focus on small single hidden-layer neural network architectures. Although deeper networks may be more parsimonious in specific applications, a shallow single-layer neural network is sufficiently parsimonious while flexible enough in representing the relation between the maturity and the loading.

REMARK 2.1: While we specify the loading matrix using neural networks, alternative

methods exist. For instance, in yield curve modeling, the Nelson-Siegel specification Diebold and Li (2006) is widely used. It defines the loading matrix as

$$\Gamma_{ij}^{NS}(\tau_i; \lambda) = \begin{cases} 1, & j = 1, \\ \frac{1-e^{-\lambda\tau_i}}{\lambda\tau_i}, & j = 2, \\ \frac{1-e^{-\lambda\tau_i}}{\lambda\tau_i} - e^{-\lambda\tau_i}, & j = 3, \end{cases}$$

where λ governs the decay rate. Given λ , the model estimates the factors β_t using cross-sectional OLS at each time t , and then fits a VAR(p) model to forecast these factors. The yield forecasts are obtained by plugging the forecasted factors back into the model.

2.2 Unique Identification and Interpretability

In their current form, the factor loadings Γ_t and the factors β_t are not uniquely identified, because for any invertible matrix $Q \in \mathbb{R}^{K \times K}$ we have

$$\Gamma_t \beta_t = (\Gamma_t Q^{-1})(Q \beta_t).$$

Furthermore, the filtered factors β_t are not directly interpretable. Identification and interpretability are common issues in factor model estimation. Stock and Watson (2016) thoroughly discuss this for a variety of factor models. We combine some of their techniques to achieve identification and interpretability.

By appropriately transforming the output of the neural networks, we can simultaneously satisfy the identification requirements and obtain interpretable factors. To illustrate this, we specify the transformation so that the factors correspond to the three ($K = 3$) Nelson Siegel factors, namely, the *level*, *slope* and *curvature*. The requirements for this are

$$\begin{aligned} \hat{y}_t(\tau_i) &:= \mathbf{y}_{it} \\ \hat{y}_t(\tau_N) &= \beta_{1t}, && \textit{level} \\ \hat{y}_t(\tau_N) - \hat{y}_t(\tau_1) &= -\beta_{2t}, && \textit{slope} \\ [\hat{y}_t(\tau^*) - \hat{y}_t(\tau_1)] - [\hat{y}_t(\tau_N) - \hat{y}_t(\tau^*)] &\approx \beta_{3t}, && \textit{curvature} \end{aligned}$$

for some intermediate value τ^* .

We begin by fixing the first column of Γ_t to be a vector of ones, which ensures that this column represents a uniform loading across all maturities. In addition, we impose the restrictions $\Gamma_{t,N2} = 0$ and $\Gamma_{t,N3} = 0$, so that the predicted yield at the longest maturity is determined solely by β_{1t} , which can therefore be interpreted as the level factor. Likewise, to capture the slope of the yield curve, we set $\Gamma_{t,12} = 1$ and $\Gamma_{t,13} = 0$, thereby tying $-\beta_{2t}$ to the difference between the shortest and longest maturities. To identify the curvature factor, we fix the Euclidean norm of the third column and require all elements of Γ_t to be nonnegative, which allows β_{3t} to be associated with the curvature in the yield curve. Finally, to complete the set of identification conditions, we add one further restriction by setting the second-to-last entry of the second column to zero, $\Gamma_{t,(N-1)2} = 0$, a constraint that has negligible impact on the interpretation of the factors but provides full identifiability of the model. In total, these conditions yield nine necessary restrictions, summarized in Table 1.

TABLE 1: Conditions for Unique Identification of Factor Loadings

Label	Condition
L-C1	$\Gamma_{t,i1} = 1$ for all $i = 1, \dots, N$
S-C1	$\Gamma_{t,12} = 1$
S-C2	$\Gamma_{t,(N-1)2} = 0$
S-C3	$\Gamma_{t,N2} = 0$
C-C1	$\Gamma_{t,13} = 0$
C-C2	$\Gamma_{t,N3} = 0$
C-C3	$\ \Gamma_{t,3}\ _2 = 1$
C-C4	$\Gamma_{t,i3} > 0$ for all i

To establish the identifiability of the factor loadings, we must show that for Γ_t , which satisfies the conditions in Table 1, any rotated loading matrix $R = \Gamma_t Q^{-1}$ also satisfies all identification restrictions only when the rotation matrix Q^{-1} equals the identity. In other words, we need to verify that the imposed conditions cannot hold if they are multiplied with a non-identity rotation. To show this result, we examine the structure of Q^{-1} by verifying how each entry is determined by the identifying restrictions. The argument proceeds in three intermediate steps, where each step builds on the results of the previ-

ous ones. In this way, we show that no remaining degrees of freedom are available for alternative rotations and we establishing uniqueness.

PROPOSITION 2.1: *Suppose conditions L-C1, S-C2, and C-C2 hold. Then, it follows that $q_{11} = 1$, $q_{12} = 0$, and $q_{13} = 0$.*

Proof. We begin by considering the last row of the factor loading matrix, $\Gamma_{t,N}$, which corresponds to the longest maturity. Multiplying this row by each column of the rotation matrix Q^{-1} yields three equations:

$$1 = 1 \cdot q_{11} + 0 \cdot q_{21} + 0 \cdot q_{31}, \quad 0 = 1 \cdot q_{12} + 0 \cdot q_{22} + 0 \cdot q_{32}, \quad 0 = 1 \cdot q_{13} + 0 \cdot q_{23} + 0 \cdot q_{33}.$$

These equations directly imply $q_{11} = 1$, $q_{12} = 0$, and $q_{13} = 0$, fixing the first row of Q^{-1} . \square

PROPOSITION 2.2: *Given conditions L-C1, S-C1, and C-C1, and using Proposition 2.1, it follows that $q_{21} = 0$, $q_{22} = 1$, and $q_{23} = 0$.*

Proof. Next, we move to the first row of the loading matrix, $\Gamma_{t,1}$, which represents the shortest maturity. Multiplying this row by Q^{-1} leads to:

$$1 = 1 \cdot q_{11} + 1 \cdot q_{21} + 0 \cdot q_{31}, \quad 1 = 1 \cdot q_{12} + 1 \cdot q_{22} + 0 \cdot q_{32}, \quad 0 = 1 \cdot q_{13} + 1 \cdot q_{23} + 0 \cdot q_{33}.$$

Using Proposition 2.1 to substitute $q_{11} = 1$, $q_{12} = 0$, and $q_{13} = 0$, these equations reduce immediately to $q_{21} = 0$, $q_{22} = 1$, and $q_{23} = 0$. Thus, the second row of Q^{-1} is also uniquely determined. \square

PROPOSITION 2.3: *Assuming conditions L-C1, S-C2, S-C3, C-C3, and C-C4 hold, and using Propositions 2.1 and 2.2, we have $q_{31} = 0$, $q_{32} = 0$, and $q_{33} = 1$.*

Proof. Finally, consider the remaining rows 2 through $(N - 1)$ of the loading matrix, $\Gamma_{t,2:(N-1)}$. Multiplying these rows by Q^{-1} gives, for each $i \in \{2, \dots, N - 1\}$,

$$1 = 1 \cdot q_{11} + \Gamma_{t,i2} \cdot q_{21} + \Gamma_{t,i3} \cdot q_{31}, \quad R_{i2} = 1 \cdot q_{12} + \Gamma_{t,i2} \cdot q_{22} + \Gamma_{t,i3} \cdot q_{32},$$

$$R_{i3} = 1 \cdot q_{13} + \Gamma_{t,i2} \cdot q_{23} + \Gamma_{t,i3} \cdot q_{33}, \quad \|q_{33}\Gamma_{t,3}\|_2 = 1.$$

By Propositions 2.1 and 2.2, $q_{11} = 1$, $q_{21} = 0$, $q_{12} = 0$, $q_{22} = 1$, and $q_{23} = 0$. Hence, the first equation immediately implies $q_{31} = 0$. The second equation simplifies to $R_{i2} = \Gamma_{t,i2} + \Gamma_{t,i3} \cdot q_{32}$. In particular, for $i = N-1$ we know from condition S-C2 that $\Gamma_{t,(N-1)2} = 0$ and $R_{(N-1)2} = 0$, which gives $0 = \Gamma_{t,(N-1)3} \cdot q_{32}$. Since $\Gamma_{t,(N-1)3} > 0$ by C-C4, we conclude $q_{32} = 0$. For the third equation, we have $R_{i3} = \Gamma_{t,i3} \cdot q_{33}$. By the normalization condition C-C3 and absolute homogeneity of norms,

$$\|\Gamma_{t,3} \cdot q_{33}\|_2 = |q_{33}| \|\Gamma_{t,3}\|_2 = 1.$$

Thus, $|q_{33}| = 1$. By the positivity condition C-C4, we must have $q_{33} = 1$. \square

THEOREM 2.1: *Under Propositions 2.1, 2.2, and 2.3, the factor loadings are uniquely identifiable, as the only rotation matrix satisfying all imposed conditions is the identity matrix $Q^{-1} = I$.*

Proof. The theorem follows directly from the sequence of constraints derived in Propositions 2.1, 2.2, and 2.3, which together determine all rows of Q^{-1} uniquely and force it to equal the identity. \square

2.3 Neural Network Transformations

Under the conditions in Table 1, we establish Theorem 2.1. However, the question remains how to enforce that the neural networks specifying the factor loading matrix satisfy these conditions. There are multiple ways to do this. We present one approach that satisfies all conditions and has proven to be computationally efficient. We refer to this set of transformations as the Neural Nelson–Siegel (NNS) transformations. We define the transformations of the neural networks as follows.

For the first neural network, we trivially set

$$\text{NN}^*(\tau_i; \gamma_1) = 1, \tag{4}$$

which ensures that L-C1 holds.

For the second neural network, we apply a linear rescaling and impose a fixed point

to satisfy S-C1, S-C2, and S-C3:

$$\begin{aligned} \text{NN}^*(\tau_i; \gamma_2) &= \frac{\text{NN}(\tau_i; \gamma_2) - \text{NN}(\tau_N; \gamma_2)}{\text{NN}(\tau_1; \gamma_2) - \text{NN}(\tau_N; \gamma_2)} \quad \forall i = 1 \dots N - 1, \\ \text{NN}^*(\tau_N; \gamma_2) &= 0. \end{aligned} \quad (5)$$

For the third neural network, we subtract the straight line connecting the outputs at the shortest and longest maturities from every output, anchoring those two maturities at zero to satisfy C-C1 and C-C2. We then square the results to ensure positivity and normalize the resulting vector by its Euclidean norm, which satisfies C-C3 and C-C4. These transformations restrict

$$\begin{aligned} \text{NN}^\dagger(\tau_i; \gamma_3) &= \left(\text{NN}(\tau_i; \gamma_3) - \left[\text{NN}(\tau_1; \gamma_3) + \frac{\text{NN}(\tau_N; \gamma_3) - \text{NN}(\tau_1; \gamma_3)}{\tau_N - \tau_1} (\tau_i - \tau_1) \right] \right)^2, \\ \text{NN}^*(\tau_i; \gamma_3) &= \text{NN}^\dagger(\tau_i; \gamma_3) / \sqrt{\sum_{i=1}^N (\text{NN}^\dagger(\tau_i; \gamma_3))^2}. \end{aligned} \quad (6)$$

2.4 Filter

To estimate the model in Equation (1), we use a filter from the class of Generalized Autoregressive Method-of-Moments (GaMM) filters (Creal et al., 2024). As in Zamojski (2019), we refer to our specified filter as the Self-Driving (SD) filter. For our proposed factor model, we call this the Self-Driving Neural Factor Model (SDNFM). We specify the filter in the following way

$$\hat{\mathbf{y}}_t = \Gamma(\boldsymbol{\tau}; \hat{\boldsymbol{\gamma}}_t) \hat{\boldsymbol{\beta}}_t = \hat{\Gamma}_t \hat{\boldsymbol{\beta}}_t. \quad (7)$$

These predictions are built from the predicted factors and loadings, where $\hat{\Gamma}_t$ and $\hat{\boldsymbol{\beta}}_t$ denote the one-step ahead predictions of $\Gamma(\boldsymbol{\tau}; \boldsymbol{\gamma}_t)$ and $\boldsymbol{\beta}_t$. For each time t we predict both. Because of the model's nonlinearity, we adopt a three-step procedure at each time point. First, we obtain the filtered state of the factors by a cross-sectional least-squares estimate,

$$\tilde{\boldsymbol{\beta}}_{t|t} = (\hat{\Gamma}_t' \hat{\Gamma}_t)^{-1} \hat{\Gamma}_t' \mathbf{y}_t. \quad (8)$$

This step requires the loading matrix estimate to be invertible. Second, we update the factor loadings using the scaled gradient of the loss function evaluated at time t ,

conditional on $\beta_t = \tilde{\beta}_{t|t}$,

$$\hat{\gamma}_{t|t} = A\mathbf{s}_t + \hat{\gamma}_t, \quad (9)$$

$$\mathbf{s}_t = S_t \nabla_t \quad (10)$$

$$\nabla_t = \left. \frac{\partial}{\partial \gamma_t} (\mathbf{y}_t - \Gamma_t \beta_t)' (\mathbf{y}_t - \Gamma_t \beta_t) \right|_{\gamma_t = \hat{\gamma}_t, \beta_t = \tilde{\beta}_{t|t}}, \quad (11)$$

where $A \in \mathbb{R}^{3LK \times 3LK}$ is the learning-rate matrix; larger entries lead to faster updating. \mathbf{s}_t represents the scaled update direction, where S_t scales the gradient ∇_t . Third, to ensure that the factors minimize the cross-sectional squared error given the updated loadings, we recompute a least-squares estimate,

$$\hat{\beta}_{t|t} = (\hat{\Gamma}'_{t|t} \hat{\Gamma}_{t|t})^{-1} \hat{\Gamma}'_{t|t} \mathbf{y}_t. \quad (12)$$

We complete the recursion with the state predictions

$$\hat{\beta}_{t+1} = \omega_1 + B_\beta \hat{\beta}_{t|t}, \quad \hat{\gamma}_{t+1} = \omega_2 + B_\gamma \hat{\gamma}_{t|t}, \quad (13)$$

where $\omega_1 \in \mathbb{R}^K$, $B_\beta \in \mathbb{R}^{K \times K}$, $\omega_2 \in \mathbb{R}^{3LK}$, and $B_\gamma \in \mathbb{R}^{3LK \times 3LK}$. The vectors ω_1 and ω_2 capture the unconditional means, while B_β and B_γ govern the speed of mean reversion.

Given the filtering procedure described above, the static parameters for a given sample of observations $\{\mathbf{y}_t\}_{t=1}^T$ are estimated by minimizing the sum of squared residuals:

$$\hat{\theta} = \arg \min_{\theta} \sum_{t=1}^T (\mathbf{y}_t - \hat{\Gamma}_t \hat{\beta}_t)' (\mathbf{y}_t - \hat{\Gamma}_t \hat{\beta}_t),$$

where θ collects all the static parameters, $\theta = \{\omega'_1, \text{vec}(A), \omega'_1, \text{vec}(B_\beta), \omega'_2, \text{vec}(B_\gamma)\}'$.

The filter naturally supports one-step-ahead out-of-sample forecasting. For a longer forecast horizon H , we skip Equations (8)–(12) by treating the last predictions as filtered estimates,

$$\hat{\beta}_{T+h|T+h} = \hat{\beta}_{T+h}, \quad \hat{\gamma}_{T+h|T+h} = \hat{\gamma}_{T+h}, \quad h = 1, \dots, H-1.$$

In combination with Equation (7) and (13), we recursively get the predictions for \mathbf{y}_{t+H} .

2.5 Scaling

In [Creal et al. \(2024\)](#), the authors refer to the scaling methods commonly used in the score-driven literature [Creal et al. \(2013\)](#). Traditional score-driven models update parameters using the gradient of the log-likelihood, and the gradient is typically scaled by the inverse Fisher information.

Since we do not assume a log-likelihood, we cannot analytically derive the Fisher information. [Creal et al. \(2024\)](#) propose replacing the analytical Fisher information with an empirically informed quantity. This is similar to what is common in the stochastic gradient descent literature for neural network training. Some popular gradient scaling techniques are AdaGrad ([Duchi et al., 2011](#)), RMSprop ([Tieleman and Hinton, 2012](#)), and ADAM ([Kingma and Ba, 2017](#)). We adopt scaling similar to RMSprop since it aligns with the idea of scaling with the Fisher information. The empirical scaling is

$$S_t^2 = \rho S_{t-1}^2 + (1 - \rho) \mathbf{I}_{3LK, 3LK} \nabla_t^2, \quad (14)$$

where $\rho \in (0, 1)$ governs the update speed and \mathbf{I}_{\cdot} is the identity matrix. One could optimize ρ , but it requires substantial computational effort. We use $\rho = 0.98$ throughout this paper. This choice of ρ balances the adaptability to heteroskedasticity and robustness to noisy updates. We refer to this scaling as EWMA scaling. As a benchmark, we also use scaling with the identity matrix.

3 Models

The filter introduced in [Section 2.4](#) is governed by the matrices A , ω_1 , B_β , ω_2 , B_γ and the scaling S_t . We set the number of latent factors to $K = 3$ and fix the size of each neural network at $L = 3$. Each of the models is uniquely identified by the NNS transformation. Without additional structure this specification involves 1,497 free parameters. To contain estimation variance and reduce computing time we impose diagonal restrictions on A and B_γ . [Table 2](#) summarizes four nested variants: NNS, 1SD-NNS, 2SD-NNS, and 3SD-NNS that differ in the number of parameters they share. We use two scaling methods: identity scaling and EWMA scaling. In the model names, these are denoted by SD (identity) and SSD (EWMA). Thus, 1SD-NNS refers to the first method and 1SSD-NNS to the second.

Since we transform the first neural network to always output 1, we set its filter parameters to zero and do not estimate them. Although not strictly necessary, we impose the same restrictions on A as on B_γ for each model. In the static model, γ_t is constant and equals ω_2 , with A as on B_γ set to zero. In 1SD-NNS, we assume a single learning rate per neural network and identical mean reversion speeds for weights and biases. In 2SD-NNS, we allow individual learning rates and mean reversion speeds for the groups (w_{11}, w_{12}, w_{13}) , (w_{21}, w_{22}, w_{23}) , and (b_1, b_2, b_3) of each neural network. In 3SD-NNS, all weights and biases of both neural networks are assigned individual learning rates and mean reversion parameters.

TABLE 2: Diagonal specification of A and B_γ

NNS	A and B_γ
Static	$\mathbf{0}_{27 \times 27}$
1SD-NNS	$\text{diag}(\mathbf{0}'_9, s_1 \mathbf{1}'_9, s_2 \mathbf{1}'_9)$
2SD-NNS	$\text{diag}(\mathbf{0}'_9, (s_1, s_2, s_3)' \otimes \mathbf{1}'_3, (s_4, s_5, s_6)' \otimes \mathbf{1}'_3)$
3SD-NNS	$\text{diag}(\mathbf{0}'_9, (s_1, s_2, \dots, s_{18}))$

3.1 Benchmark models

We benchmark our models against the random walk forecast and several yield curve models that are widely used in the literature. The standard Nelson–Siegel (NS) model follows Diebold and Li (2006). The dynamic Nelson–Siegel (DNS) variant of Diebold et al. (2005) keeps the loading matrix with exponential basis functions but lets the latent factors be governed by a Gaussian VAR(p) process, leading to a linear state-space model estimated using the Kalman filter. Koopman et al. (2010) extend DNS by treating the decay parameter λ as an additional state. We refer to this nonlinear state space model, estimated with the extended Kalman filter, as TV- λ DNS.

Finally, we propose a new model specification that naturally arises when combining the Nelson–Siegel model with our proposed filtering method from Section 2.4. We keep the Nelson–Siegel basis functions and allow its loading parameter λ to change over time. The recursion steps are the same as in the neural network version, but the loading matrix is replaced by the Nelson–Siegel basis functions. We refer to this model as λ SD.

4 Simulation study

To guide our understanding of the capabilities and limitations of the introduced algorithm, we conduct a simulation study. The purpose is twofold. First, the simulations illustrate the differences between models with static factor loadings and models with time-varying loadings. Second, they allow us to assess to what extent neural networks can substitute for the exponential loading functions of Nelson-Siegel. The design of the data generating process combines an empirically calibrated model with deterministically imposed time-variation, which lets us study these differences while remaining in a realistic environment.

We generate yields from a dynamic three-factor Nelson-Siegel model,

$$\mathbf{y}_t = \Gamma^{NS}(\boldsymbol{\tau}; \lambda_t) \boldsymbol{\beta}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_\varepsilon),$$

where the loading functions $\Gamma^{NS}(\boldsymbol{\tau}; \cdot)$ are specified as in Remark 2.1. The factors follow a VAR(1),

$$\boldsymbol{\beta}_t = \boldsymbol{\omega} + B \boldsymbol{\beta}_{t-1} + \mathbf{u}_t, \quad \mathbf{u}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_u),$$

with

$$\boldsymbol{\omega} = \begin{bmatrix} 5.64 \\ -2.42 \\ -1.27 \end{bmatrix}, \quad B = \begin{bmatrix} 0.99 & 0.00 & 0.00 \\ 0.00 & 0.95 & 0.00 \\ 0.00 & 0.00 & 0.93 \end{bmatrix}, \quad \Sigma_u = \begin{bmatrix} 0.15 & -0.11 & -0.01 \\ -0.11 & 0.14 & 0.02 \\ -0.01 & 0.02 & 0.66 \end{bmatrix}, \quad \Sigma_\varepsilon = 0.07I_N.$$

All parameters are empirically calibrated by estimating a Dynamic Nelson-Siegel model with static loadings on U.S. Treasury yield data from January 1985 to December 2024. The values shown are rounded for clarity, but the simulations use the unrounded estimates. For numerical stability, we assume a diagonal B matrix in the simulations. The yields and factors are generated through the random components $\boldsymbol{\varepsilon}_t$ and \mathbf{u}_t , while the sequence λ_t is imposed ex ante. We consider three specifications: an AR(1) process, structural breaks, and an empirically estimated path.

AR(1): For the first specification, we generate an auxiliary variable,

$$a_t = 0.99a_{t-1} + \delta_t, \quad \delta_t \sim \mathcal{N}(0, 0.01),$$

and then transform it so that it is positive, has mean 0.07, and has the same standard deviation as the SSD-NS estimate of λ_t . The resulting paths display high persistence and gradual time-variation. Moreover, the paths are random, so each simulation repetition features a unique path, allowing the models to be evaluated across different circumstances.

Structural breaks: To introduce abrupt time-variation, we let λ_t cycle deterministically between 0.03, 0.07, 0.2, and 0.07 in intervals of length 24. The values 0.03 and 0.2 approximate the lower and upper bounds of the estimated λ_t from our empirical data, while 0.07 represents a moderate level. In contrast to the autoregressive case, this process is fully deterministic.

Empirically estimated: Lastly, we use the empirical λ_t estimates from the SSD-NS model as a deterministic path in our simulations. We have 480 monthly estimates in our empirical sample. To obtain the length required for our simulation design, we prepend the estimated series with a copy of itself. To smooth out any large jumps at the concatenation point, we linearly interpolate between the start and the tail of the path, where the slope of the interpolation matches the average slope of two consecutive empirical estimates. This specification is the most realistic of the three, since the λ_t paths are closest to those observed in the data.

We simulate yields for 24 maturities, 1-12, 18, 24, 30, 36, 48, 60, 72, 84, 96, 108, 120, 180 months, for sample sizes of 250, 500, and 2000 in-sample observations and 500 out-of-sample observations. We perform 500 simulation repetitions. For each repetition, we estimate the NS, NNS, SD-NS, SSD-NS, and 1SSD-NNS models, and generate one-month and twelve-month-ahead forecasts for all maturities. To evaluate forecast accuracy, we compare against the random walk (RW) and report the relative root mean squared error (RRMSE), defined relative to RW. Tables 3 show the RRMSEs averaged over repetitions and over all maturities except the 180-month maturity, which is included as anchoring point, for all sample sizes and all time-variation specifications.

For the one-step-ahead forecasts and the smallest sample size, the simplest model, NS, performs best across all three simulation designs. Its parsimony avoids it from overfitting, although the neural network, which has 17 more parameters, performs almost as well. Under the strongest time-variation, the structural breaks, the SSD-NS model already matches the performance of NS, while SD-NS fails to generalize out-of-sample.

TABLE 3: Average RRMSE w.r.t. RW for forecast horizons 1 and 12 (500 simulations)

Model	Horizon 1			Horizon 12		
	250	500	2000	250	500	2000
AR(1)						
RW	1.000	1.000	1.000	1.000	1.000	1.000
NS	1.003	0.981	0.974	1.102	1.005	0.968
NNS	1.008	0.982	0.973	1.098	0.988	0.944
SD-NS	1.013	0.978	0.970	1.140	1.006	0.966
SSD-NS	1.009	0.977	0.970	1.129	1.005	0.966
1SSD-NNS	1.017	0.980	0.968	1.141	1.002	0.946
Structural Breaks						
RW	1.000	1.000	1.000	1.000	1.000	1.000
NS	1.007	0.990	0.984	1.079	0.995	0.965
NNS	1.008	0.989	0.982	1.057	0.975	0.939
SD-NS	658.592	0.988	0.975	1972.743	1.010	0.957
SSD-NS	1.007	0.980	0.974	1.107	0.993	0.955
1SSD-NNS	1.012	0.982	0.972	1.095	0.986	0.937
Empirically Estimated						
RW	1.000	1.000	1.000	1.000	1.000	1.000
NS	0.996	0.976	0.972	1.090	1.003	0.971
NNS	0.998	0.975	0.969	1.067	0.972	0.940
SD-NS	1.446	0.975	0.969	2.044	1.000	0.968
SSD-NS	1.001	0.974	0.969	1.110	1.000	0.968
1SSD-NNS	1.006	0.975	0.967	1.097	0.983	0.942

Its gradient updates are often too large, leading to unrealistically large forecast errors.

For the medium sample size, the self-driving models begin to dominate. All three time-varying alternatives outperform their static counterparts. SSD-NS yields the lowest forecast errors across all data generating processes. Interestingly, NNS starts to outperform NS by a small margin, whereas 1SSD-NNS does not yet improve on SSD-NS. The 1SSD-NNS specification appears to require either more data or a stronger time-variation signal, whereas the parsimony of SSD-NS helps in shorter samples.

For the largest sample size, NNS fits the data better than NS in all experiments, even though the true factor loadings have the Nelson-Siegel shape. This improvement extends to 1SSD-NNS, which outperforms all NS-based competitors. The gains from allowing time-variation remain largest for the structural-break design and smallest for the empirically estimated case.

For the twelve-month-ahead forecasts, the results differ from those at the one-month-ahead horizon. For the smallest sample size, the NNS models have the lowest relative forecast errors among the estimated models, although all models perform worse than RW. This reflects the accumulation of estimation error in the recursive forecasting scheme. With only a short sample, the parameter estimates are less stable, and the resulting estimation error propagates and amplifies as the forecast horizon increases. For the medium and largest sample sizes, NNS remains the best-performing model in almost all settings and consistently outperforms RW. Only under structural breaks does 1SSD-NNS achieve slightly lower forecast errors. The benefits of time-varying factor loadings weaken or completely disappear at this longer horizon. The naturally larger forecast errors and longer horizons and the mean-reverting nature of the imposed time-variation in the factor loadings explain this.

We also investigate the role of scaling. SSD-NS consistently outperforms SD-NS for smaller samples, but the difference largely disappears for the largest sample. Without scaling, large errors can occasionally produce unrealistically large gradients; these exploding gradients make the subsequent forecast even worse, after which the model typically returns to normal. Figure 1 shows the λ_t forecasts of SSD-NS and SD-NS. The wider confidence bands for SD-NS reflect these episodes. Spikes appear in both the upward and downward directions, although they are more visible in the upward direction due to the positivity restriction on λ_t . Note how the estimates tend to not reach 0.2 value in

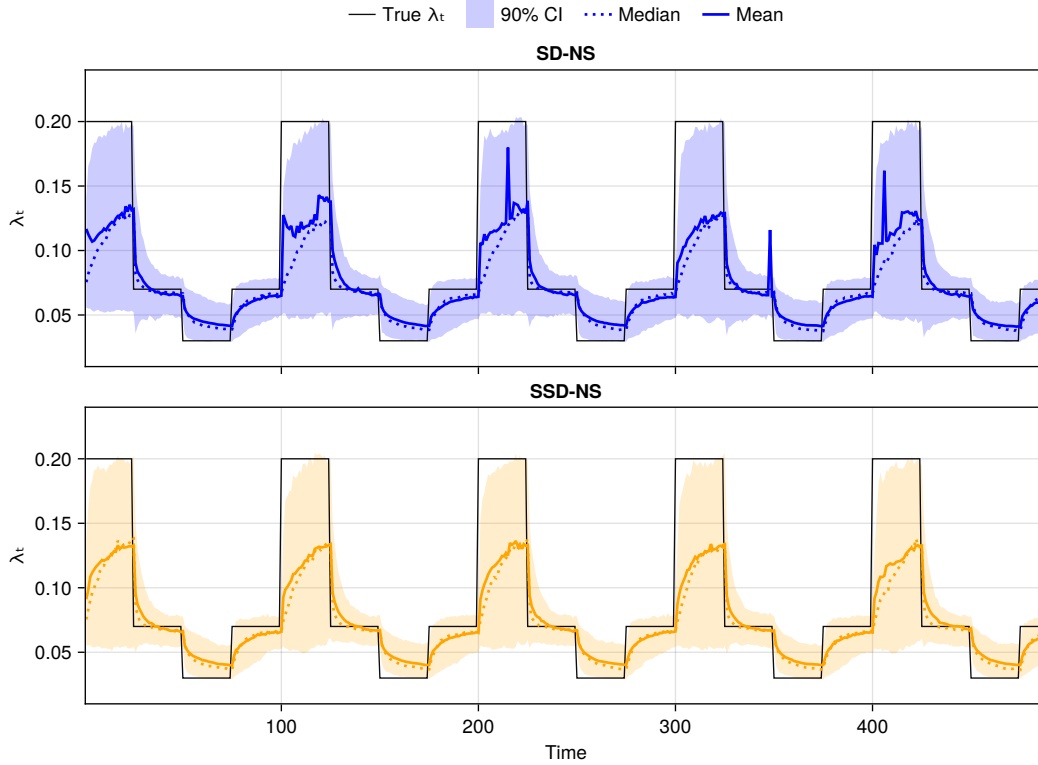


FIGURE 1: Average, median, 10% quantile, and 90% quantile of estimated λ_t for step-function λ_t paths.

the high regime and on average get stuck around 0.13. The mean reversion keeps it from reaching larger value. The models favor underestimating λ_t such that they can reach the moderate regime quicker leading to lower mean squared error.

We also investigate the role of scaling. SSD-NS consistently outperforms SD-NS for smaller samples, but the difference largely disappears for the largest sample. Without scaling, large errors can occasionally produce unrealistically large gradients; these exploding gradients make the subsequent forecast even worse, after which the model typically returns to normal. Figure 1 shows the λ_t forecasts of SSD-NS and SD-NS. The wider confidence bands for SD-NS reflect these episodes. Spikes appear in both the upward and downward directions, although they are more visible upward due to the positivity restriction on λ_t . Note that the estimates tend not to reach the value 0.2 in the high regime and instead on average only reach approximately 0.13. The mean reversion keeps the estimates from reaching larger values. The models favor underestimating λ_t in the high regime, which allows them to return to the moderate regime more quickly and leads to lower mean squared error.

For larger sample sizes, the λ_t forecasts become smoother and the two models behave more similarly. However, even with large samples, scaling remains advisable in empirical applications. In our simulations, the time-variation is repetitive and predictable, whereas real data may generate previously unseen patterns or large unprecedented shocks. Scaling makes the model more robust to such anomalies.

Finally, it is important to emphasize that the empirical λ_t paths used to inform our simulation data generating process represent a delayed and dampened projection of the true time-variation. Even when the empirical estimates show substantial movements, they can only react with a lag to the underlying variations, and they inevitably smooth out part of the true dynamics. This dampening visible is also visible in Figure 1 and illustrate how these models gradually adapt and lag behind when the underlying changes move more quickly than the model can accommodate. The actual data generating process for the yield curve is unknown and may involve richer and more abrupt forms of time-variation than those captured by our estimation. In empirical settings, the time-variation may therefore be faster and stronger. This implies that the potential benefits of allowing time-varying factor loadings could be even larger in practice than what is suggested by the simulation designs.

5 Forecast results

5.1 Data and empirical setup

We use monthly zero-coupon yields on U.S. Treasury securities from the dataset of [Liu and Wu \(2021\)](#). The yields are constructed with a non-parametric kernel-smoothing method that retains more information from the raw yield data and delivers smaller pricing errors than competing methods. The full panel spans maturities from one month to thirty years and begins in 1961.

Because changes in monetary policy have influenced yield curve dynamics, we begin the sample at the start of the Great Moderation, when the Federal Reserve committed to low and stable inflation. The data span January 1985 to December 2024 and cover the following maturities (in months): 1-12, 18, 24, 30, 36, 48, 60, 72, 84, 96, 108, 120, and 180. Over this period, data are available for all monthly maturities up to 180 months. We place greater weight on the shorter maturities since they contain most of the information

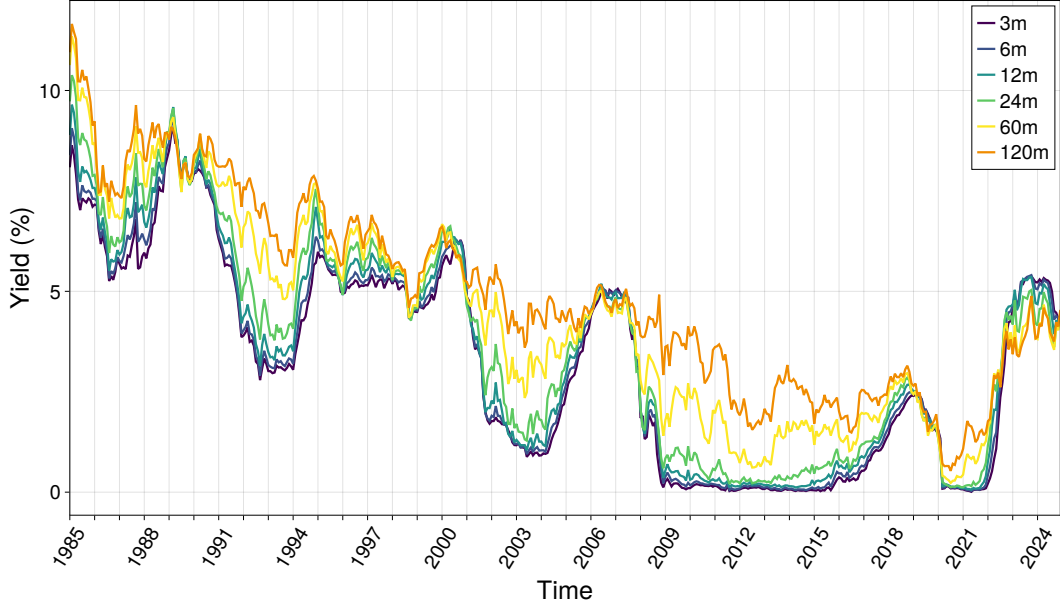


FIGURE 2: Yields over time for the 3, 6, 12, 24, 60 and 120 months

about the curve’s shape, and we include the 180-month maturity as an anchor for the factor loadings. For forecasting, we assess model performance at maturities 3, 6, 12, 24, 60, and 120 to provide a clear overview and to ensure the results are not driven solely by the shorter maturities. The results for all 24 maturities are reported in the appendix. This selection is not intended to forecast the 180-month maturity. If that maturity is of interest, we recommend adding more data on longer maturities and reducing the emphasis on shorter ones. Observations from January 1985 to February 2004 serve as the training sample, and those from March 2004 to December 2024 form the test set. A subset of the maturities is displayed in Figure 2.

Our goal is to forecast these yields at horizons one to twelve months ahead. We compare several variants of the SD-NNS model with the benchmarks. To find the best forecasting model, we also consider three ensemble specifications: the Static Ensemble, combining NS and NNS; the SD Ensemble, combining SD-NS and 1SD-NNS; and the SSD Ensemble, combining SSD-NS and 3SSD-NNS. All models are re-estimated recursively with a moving window, and out-of-sample performance is evaluated using the root mean squared error (RMSE).

Practitioners may be interested in different parts of the yield curve at different forecast horizons. To compare forecasts for a specific maturity and horizon, we assess statistical significance using the Diebold-Mariano test adjusted for multi-step horizons, as described

in [Harvey et al. \(1997\)](#). Alternatively, one may wish to compare models at a given maturity irrespective of the forecast horizon. The multi-horizon forecast comparison test for superior predictive ability (SPA) proposed by [Quaedvlieg \(2021\)](#) allows comparison across all horizons of interest simultaneously. The test has two variants: uniform SPA (uSPA), which evaluates superior performance at each individual horizon, and average SPA (aSPA), which allows inferior performance at some horizons to be compensated by superior performance at others.

Lastly, one may also be interested in comparing models across maturities at a given horizon. Noting the similarity between the multi-horizon and multi-maturity data, the multi-horizon test can be directly applied in the maturity dimension instead of the horizon dimension. For all three levels of comparison, we evaluate multiple models by their squared errors. For the Diebold-Mariano test we compare relative to the RW. For the SPA tests, we apply the model confidence set procedure of [Hansen et al. \(2011\)](#). The MCS is the set of models for which there is no significant evidence to differentiate them in terms of the chosen test statistic. [Quaedvlieg \(2021\)](#) describe how both versions of the SPA test statistic can be used to construct model confidence sets.

5.2 Out-of-sample forecast comparison

Table 6 shows the 1-step-ahead out-of-sample relative RMSE for the random walk benchmark, two static factor models, six self-driving models of which two of the Nelson-Siegel type and four specifications of our neural factor models, two Kalman filter benchmarks, and three ensembles. The RMSE for each model is divided by that of the random walk. The first six columns present results for maturities of 3, 6, 12, 24, 60, and 120 months, and the column labeled "Average" shows the root of the averaged squared errors across these maturities relative to the random walk. Stars denote the rejection levels of the null hypothesis of equal MSE.

The static models perform worse on average than the random walk, with particularly poor performance at the 60-month maturity. All self-driving models improve on their static counterparts across all maturities, and the scaled versions further enhance performance. Among the self-driving specifications, the 3SSD-NNS delivers the lowest forecast errors, highlighting the value of flexible factor structures. For the Kalman filter models, the DNS model behaves much like the static models, as expected. The TV λ -DNS model

TABLE 4: US Yield Curve Forecasting, Out-of-Sample Relative RMSE (Model RMSE / RW RMSE) for 1-Month-Ahead Forecast Horizon - Rolling Window

Model	m-3	m-6	m-12	m-24	m-60	m-120	Average
Benchmark							
RW	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Static							
NS	0.9**	0.966	1.06	1.035**	1.065**	0.989	1.01
NNS	0.94	0.946	1.044	1.032	1.063**	1.029	1.018
Self-driving							
SD-NS	0.924**	0.928*	0.959	1.023	1.055*	0.992	0.991
SSD-NS	0.918***	0.927*	0.96	1.022	1.054*	0.984	0.988*
1SD-NNS	0.932*	0.961	0.982	1.024	1.054	0.98	0.996
1SSD-NNS	0.914**	0.949*	0.983	1.027	0.99	1.004	0.984**
2SSD-NNS	0.922**	0.964	0.962*	1.014	1.011	0.995	0.984**
3SSD-NNS	0.911**	0.933**	0.954**	1.018	1.018	0.993	0.98**
Kalman filter							
DNS	0.933	0.945	1.049	1.037**	1.076**	0.988	1.013
TV λ -DNS	0.962	0.914**	1.024	1.073***	1.042	1.001	1.011
Ensembles							
Static Ensemble	0.913**	0.948	1.049	1.029*	1.061*	0.999	1.008
SD Ensemble	0.907***	0.915***	0.955*	1.018	1.042	0.975	0.979**
SSD Ensemble	0.897***	0.903***	0.943**	1.017	1.023	0.979	0.971***

Notes: Asterisks denote Diebold–Mariano rejection of equal RMSE with the RW benchmark at *10%, **5%, and ***1% significance levels.

shows mixed performance, with notably low forecast error at the 6-month maturity but high error at 24 months. Turning to the ensembles, the self-driving ensembles in particular improve on the individual models they combine and achieve the largest gains at the shorter maturities.

Table 7 reports the 12-step-ahead out-of-sample relative RMSE. All models show improved relative performance compared to the random walk at the shorter maturities (3, 6, and 12 months). For the longer maturities (24, 60, and 120 months), the results are mixed. Only the neural factor models produce lower forecast errors than the random walk across all six maturities. Consistent with the 1-month-ahead forecasts, the 3SSD-NNS delivers the lowest average forecast errors among the self-driving models. In contrast to the 1-month-ahead results, the SSD Ensemble now has higher average forecast error than the 3SSD-NNS, as the benefits of ensembling do not offset the weak performance of the NS models at the 60-month maturity. Lastly, a further difference between the 1- and 12-month-ahead horizons is that the benefit of adaptive factor loadings diminishes at longer horizons. The performance gap between the NNS and self-driving NNS models

TABLE 5: US Yield Curve Forecasting, Out-of-Sample Relative RMSE (Model RMSE / RW RMSE) for 12-Month-Ahead Forecast Horizon - Rolling Window

Model	m-3	m-6	m-12	m-24	m-60	m-120	Average
Benchmark							
RW	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Static							
NS	0.866	0.893	0.935	1.027	1.092	1.032	0.952
NNS	0.872	0.883	0.902	0.949	0.98	0.982	0.914**
Self-driving							
SD-NS	0.865	0.891	0.936	1.037	1.145	1.058	0.962
SSD-NS	0.863	0.888	0.932	1.03	1.136	1.057	0.958
1SD-NNS	0.878	0.898	0.921	0.956	0.966	0.961	0.92*
1SSD-NNS	0.868	0.89	0.917	0.95	0.956	0.963	0.913**
2SSD-NNS	0.869	0.89	0.913	0.949	0.961	0.959	0.913**
3SSD-NNS	0.858	0.88	0.905	0.943	0.956	0.96	0.905**
Kalman filter							
DNS	0.85	0.871	0.909	0.983	1.072	1.043	0.93*
TV λ -DNS	0.85**	0.886*	0.938	1.013	1.128	1.128	0.958*
Ensembles							
Static Ensemble	0.862	0.881	0.909	0.969	1.005	0.965	0.917*
SD Ensemble	0.856	0.881	0.916	0.979	1.015	0.953	0.92*
SSD Ensemble	0.844	0.87	0.907	0.97	1.008	0.953	0.91**

Notes: Asterisks denote Diebold–Mariano rejection of equal RMSE with the RW benchmark at *10%, **5%, and ***1% significance levels.

narrows, and for the NS models, the time-varying specifications often perform worse.

5.3 Multi-horizon forecast comparison

In Figure 12, we show the model confidence sets of the aSPA, Figure 3a, and the uSPA, Figure 3b, multi-horizon forecast statistics and their associated p -values. We use the forecast horizons 1 up until 12 and model confidence significance level, α_{mcs} , equal to 10%. A p -value lower than α_{MCS} indicates exclusion of the MCS. For each maturity, we indicate the models included in the MCS with a green (darker) color. Each maturity column corresponds to a new model confidence set. For six maturities there are six MCSs.

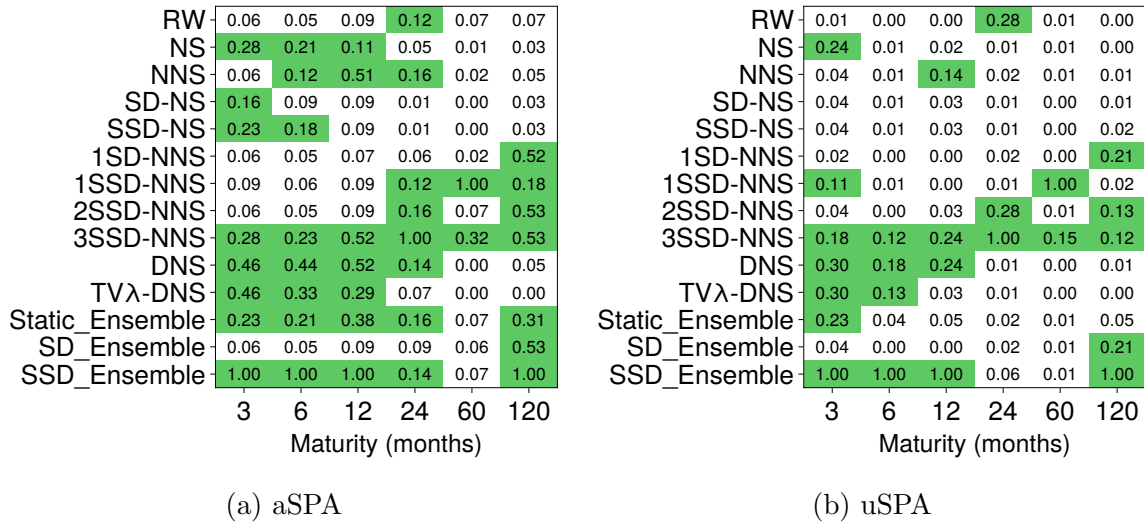


FIGURE 3: MCS and p -values per maturity at significance level $\alpha_{MCS} = 0.1$ for $B = 999$ bootstrap samples

The 3SSD-NNS is not excluded from the MCS at any maturity under either the aSPA or uSPA versions of the test. Its performance is differentiates itself the most at the 60-month maturity, where the MCS for both aSPA and uSPA consists only of the 1SSD-NNS and the 3SSD-NNS, with all other models excluded. While the 1SSD-NNS is excluded from the MCS at the lower maturities, the 3SSD-NNS is not. These findings confirm that the 3SSD-NNS is a consistently strong model across all maturities with most of its comparative advantage at the medium to medium-long maturities, the maturities shaping the curvature of the yield curve. Although many competing models provide comparable forecasts for the level and slope of the yield curve, they do not capture the curvature as well as the 3SSD-NNS. We further discuss how the curvature factor loadings of the

3SSD-NNS contribute to these results in Section 6.

5.4 Multi-maturity forecast comparison

Figure 13 presents the model confidence sets of the aSPA, Figure 4a, and the uSPA, Figure 4b, multi-maturity forecast statistics. We follow the same setup as of the multi-horizon forecast comparison. For each horizon, we form MCSs based on the forecasts of the maturities: 3, 6, 12, 24, 60, 120.

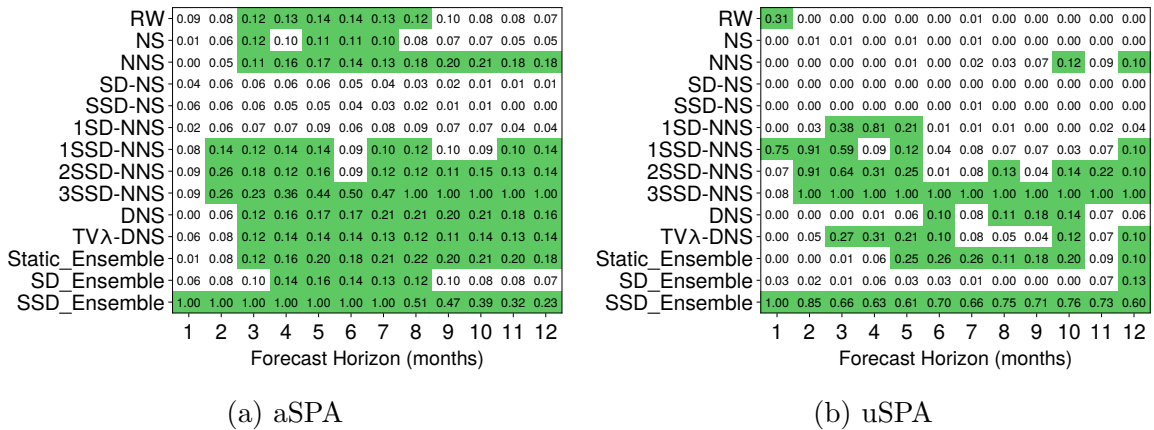


FIGURE 4: MCS and p -values per forecast horizon at significance level $\alpha_{MCS} = 0.1$ for $B = 999$ bootstrap samples

For both test statistics, the SSD Ensemble is always included in the model confidence set. The 3SSD-NNS is only excluded for the first forecast horizon for both tests. The 1-step-ahead forecast horizon Table 6 explains this. On average the SSD Ensemble has lower forecast errors than the 3SSD-NNS. The exclusion for the uniform test is the result of the SSD Ensemble having much lower forecast error at the 6-month maturity, while 3SSD-NNS does not have significantly lower forecast error than SSD Ensemble at any maturity.

Summarizing these results, the 3SSD-NNS is consistently among the best performing models, whereas the other models are not as consistent. Some modest improvement can be made at the 1 month forecast horizon when taking the ensemble of the SSD-NS and 3SSD-NNS. Beyond that, certain models incidentally produce lower forecast errors for some maturities at some horizons, but none exhibits significantly superior predictive ability for a given maturity across all horizons or for a given horizon across all maturities. This provides strong evidence in favor of using the 3SSD-NNS to forecast the US Yield

Curve.

5.5 Error accumulation over time

To better understand model performance and the evolution of RMSE over time, we analyze cumulative squared errors. Specifically, we compare the cumulative squared errors of the random walk, NS, NNS, SSD-NS and 3SSD-NNS across short (3 months), medium (5 years), and long (10 years) maturities, for both 1-month and 12-month forecast horizons. Figure 5 displays these plots. Errors accumulate more quickly for the longer 12-month forecast horizon, as expected. Notably, significant increases in errors coincide with events like the Great Recession (late 2007 to mid-2009). During this period 3SSD-NNS accumulates less error than the other models for the short maturity. At the medium maturity, the random walk accumulates the least error, yet a gap remains between the 3SSD-NNS and the other models at the end of this period. At the long maturity, the NNS accumulates most of its error. The other models are close to each other. For the 12-month forecast horizon, especially the differences at the medium maturity are visible. The NS models perform much worse during this period. Other volatile periods are during early 2020 due to the COVID-19 epidemic and starting from 2022 due to the economic disruptions that emerged after the Russian invasion of Ukraine. These periods are also clearly visible in the forecasts of the yield curve in Figure 2.

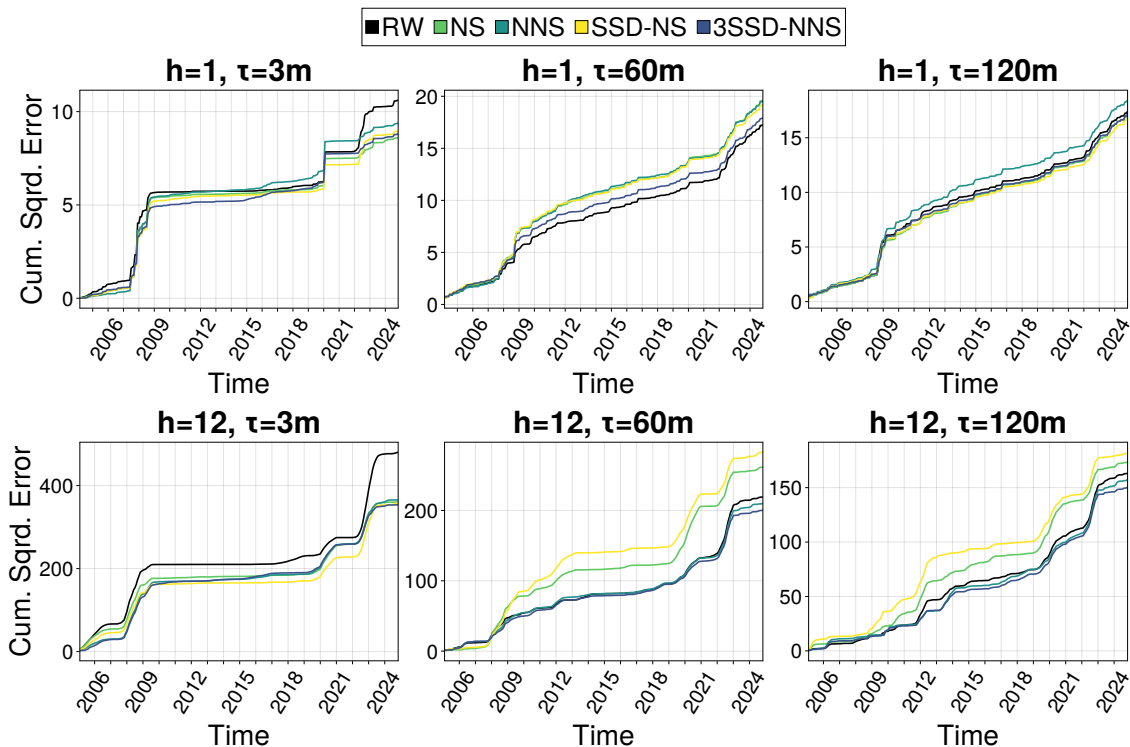


FIGURE 5: Comparison of cumulative error across forecast horizons (H) and maturities (τ): (a) $H=1$, $\tau=3$; (b) $H=12$, $\tau=3$; (c) $H=1$, $\tau=60$; (d) $H=12$, $\tau=60$; (e) $H=1$, $\tau=120$; (f) $H=12$, $\tau=120$.

6 Insample analysis

Time-varying neural factor models consistently outperform Nelson-Siegel type models. The forecast comparison results show that, in particular at the medium and medium-long maturities, these time-varying neural factor models perform better than the Nelson-Siegel models. The differences between the models arise from their factor loadings and, consequently, the factors they produce. The two classes of models differ in two main ways: the factor loading function and the time-varying structure. The neural factor models use a neural network to generate the factor loadings instead of the Nelson-Siegel specification. The time-varying component also differs fundamentally: for the neural network, time variation is governed by 18 time-varying parameters, whereas for Nelson-Siegel it is governed by a single time-varying parameter. At the 1-step-ahead forecast horizon, the time-varying structure contributes most to the improved performance, while at the 12-step-ahead forecast horizon, the neural factor loading function becomes the dominant contributor. With this in mind, we want a clearer view of how the neural factor loadings and their time-varying parameters differ from those of Nelson-Siegel to

explain the forecast improvements. We do so by estimating the models on the full sample from January 1985 to December 2024 and examining the predicted factor loadings and factors.

6.1 Estimated Factor Loadings

We begin by comparing the static neural factor loadings to the static Nelson-Siegel loadings. We are primarily interested in the slope and curvature factor loadings since the level loadings are identical for both models. The estimated loadings are illustrated in Figure 6. The shapes of both loadings show similar patterns. The loading on the second factor decreases monotonically, and approaching zero as maturity increases. The loading on the third factor initially increases sharply, reaching its maximum at the 30-month maturity for NNS model and 48-month maturity for NS model. After, reaching their maximum both decrease monotonically.

Partially, these shapes are directly imposed by each model's specific approximation method. Although neural networks offer greater flexibility in capturing shapes, their flexibility is constrained by their transformations. The Nelson-Siegel approximations are less flexible but do not anchor the loading for the shortest and longest maturity. The Nelson-Siegel model can assign more weight to the longest maturity where the transformed neural network cannot.

However, Figure 6 also highlights a limitation in interpreting the factors within the Nelson-Siegel framework. The longest-maturity yield loads substantially on both the second and third factors, implying that part of the yield's level is absorbed by these factors. This overlap reduces the interpretability of the Nelson-Siegel first factor and, consequently, also of the second and third factors. By contrast, the explicit constraints imposed by the NNS transformations on the boundary neural factor loadings provide a clearer interpretation, as they ensure that the longest-maturity yield cannot load on the second and third factors and the shortest maturity cannot load on the third factor.

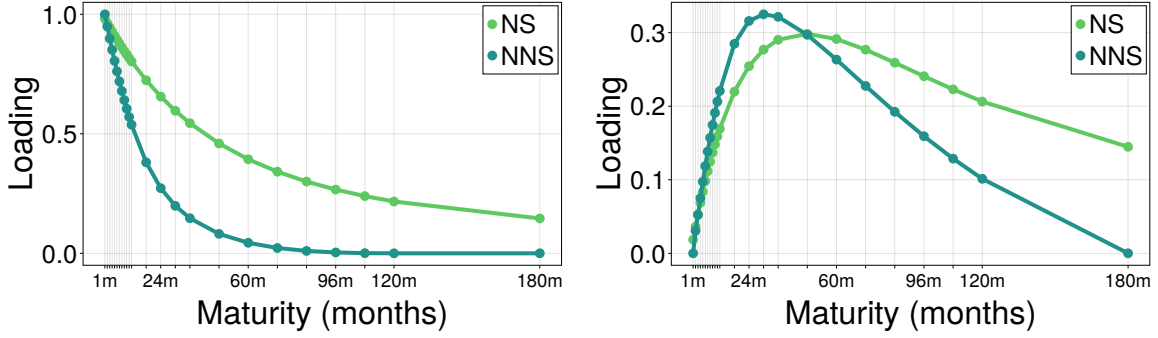


FIGURE 6: Estimated factor loadings curves for static models NNS and NS

In Figure 7, we show the time-varying paths of the one-step-ahead predicted factor loadings. Each line represents one entry in the factor loading matrix for a corresponding maturity. For the curvature factor loadings, we display only a subset of the maturities to improve readability. This comparison focuses on the 3SSD-NNS and SSD-NS models.

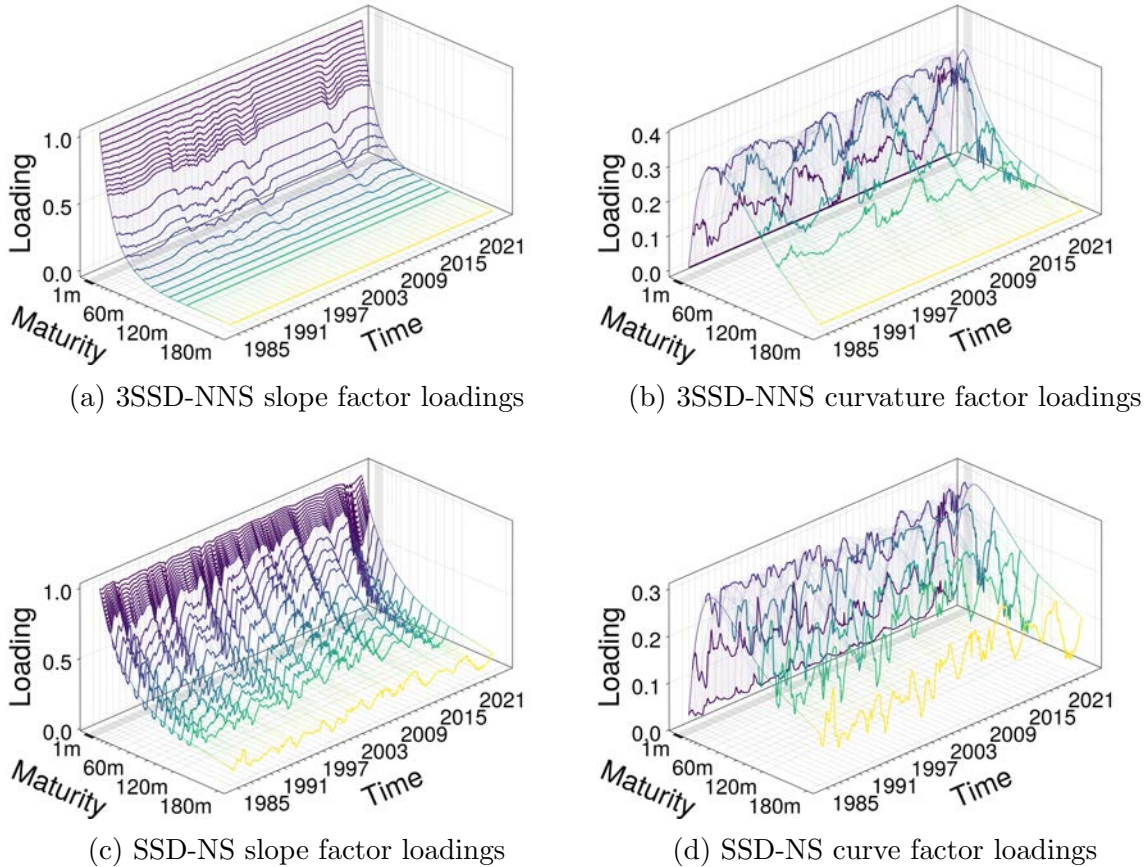


FIGURE 7: Comparison of time varying factor loadings

The SSD-NS loadings exhibit stronger time-varying behavior compared to the neural

loadings. The SSD-NS loadings are unrealistically noisy, while the neural loadings capture more persistent trends. The nine time-varying parameters of the neural network are more robust at capturing time-variation than the single time-varying parameter of the Nelson-Siegel model. Additionally, the slope loadings of the 3SSD-NNS are more static than the curvature loadings. For the SSD-NS, a single parameter produces the time-variation in both factor loadings, so more time-variation in the curvature loadings also results in more time-variation in the slope loadings. The flexibility of allowing different speeds of adjustment to new information for the slope and curvature factor loadings contributes to better performance. When examining the slope factor loadings of the 3SSD-NNS, we observe that around the Great Recession and the COVID-19 epidemic the slope loadings place less weight on higher maturities and assign relatively more weight to the lower maturities, while during the zero-lower-bound period higher maturities receive more of the slope loading.

To further understand the dynamics of the curvature loadings, Figure 8 shows the maturity with largest one-month-ahead curvature loading for every month and the shortest range between two maturities that covers at least 50% of the total curvature factor load. The loading mass moves around the 36-month into the direction where the curvature factor can explain most of the yield curve. March 1998 and December 1999 and February 2005 and October 2007 are both periods with relatively high loading on longer maturities which are followed by rapid movement toward loading on short maturities. These contribute to better capturing the inverted yield curve present at those times. Interestingly, between April 2022 and December 2024 there is also a period of an extended inverted yield curve, but here the curvature factor loads an unprecedented amount at the shorter maturities.

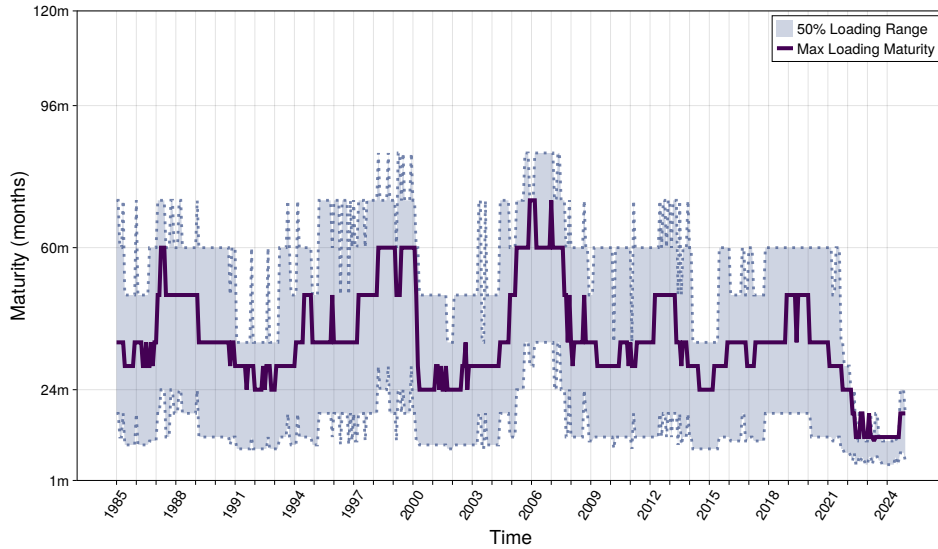


FIGURE 8: Curvature factor loading maximizing maturity and smallest 50% curvature factor loading coverage range.

6.2 Estimated Factors

In Figure 9, we compare the three predicted factors obtained with 3SSD-NNS and Nelson–Siegel (NS), and we plot their data-based proxies similar to [Koopman et al. \(2010\)](#). We proxy the level factor with the 180-month yield, the slope with the spread between the 1-month and 180-month yields, and the curvature with twice the 36-month yield minus the 3-month yield minus the 180-month yield. We choose the 36-month yield to represent curvature since it is where the 3SSD-NNS places the most curvature factor loading most of the time.

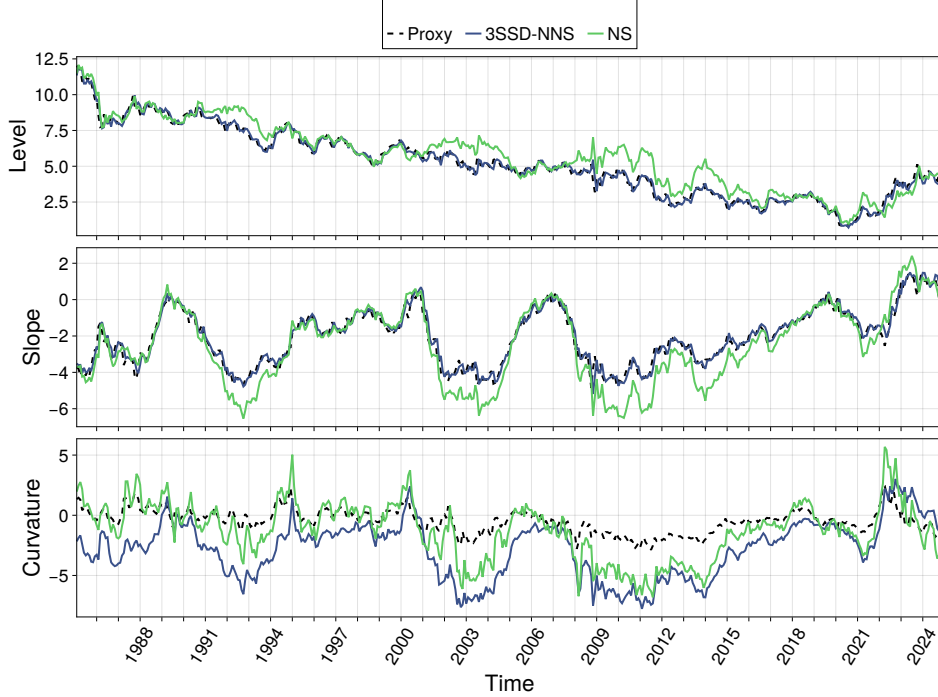


FIGURE 9: **Level, slope and curvature.** From top to bottom, the panels show the level, slope and curvature. The blue (dark) colored lines are the 1-step-ahead predicted factors of the 3SSD-NNS model; the green (light) colored lines are the NS benchmark predictions. The dashed lines represent level, slope and curvature proxies.

The neural factor model tracks the level and slope data proxies well throughout the sample, while Nelson-Siegel model does not. In particular, during 1991–1994, 2001–2005 and 2008–2016, the NS does not follow the slope and level data proxies while the neural model does. Interestingly, the NS model aligns more closely with the curvature proxy than the neural model. The curvature proxy does not align well with the neural model because the curvature data proxy relies on the curvature peak to be at the 36-month maturity while the 3SSD-NNS let’s the curvature peak vary over time. Letting the curvature peak change adaptively produces a more meaningful curvature factor than static versions.

6.3 Case study

Now that we know how the factor loadings and factors vary over time, we can go one step further by analyzing the resulting yield curve. We examine two dates where the forecast error is large and therefore expect a relatively large update in the factor loadings. We are interested in how the 3SSD-NNS adapts to these large errors. Figure 10 and Figure 11 show a snapshot of the yield curve, the one-step-ahead predictions from one month

earlier, the current one-step-ahead predictions, and the corresponding factor loadings.

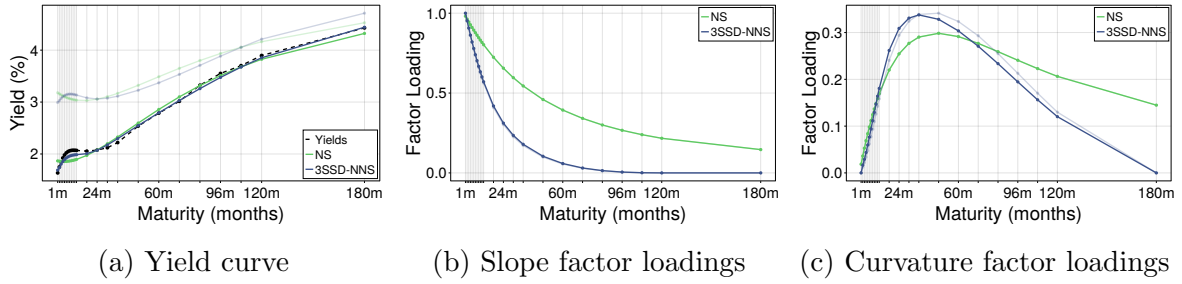


FIGURE 10: Snapshot yield curve January 2008, one-step-ahead predictions December 2007 (shaded), one-step-ahead predictions January 2008 (nonshaded)

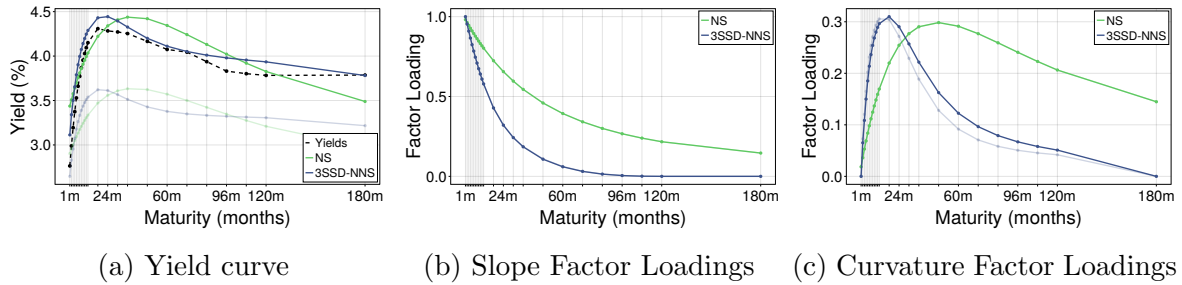


FIGURE 11: Snapshot yield curve September 2022, one-step-ahead predictions August 2022 (shaded), one-step-ahead predictions September 2022 (nonshaded)

We are interested in how the yield curve’s information is incorporated into the next predictions. In January 2008, we see that for both models there is a large shift in the slope factor and a small shift in the level factor. The curvature moves in the direction of the shorter maturities, which is visible in the corresponding change in the curvature factor loadings. Interestingly, the observed yield curve starts with a concave segment, then becomes convex, and later becomes concave again. The NS model misses the first concave episode, while the 3SSD-NNS is able to capture it.

In September 2022, the yield curve has a very different shape with a large inverted part of the yield curve. Again, there is a clear level shift, but no clear slope shift. The curvature moves slightly toward the longer maturities. The shape of the curvature factor loadings is clearly reflected in the predictions. Although the model does not fit better at some maturities, such as the 120-month maturity, it clearly provides a better fit at the medium to medium-long maturities in this month.

7 Conclusion

This paper introduces a novel framework for factor modeling with time-varying loadings, named the Self-Driving Neural Factor Model (SDNFM). The loading matrix and factors are recursively updated in the least squares direction at each time point to track variations in both. Filtered neural network parameters allow for adaptation to changing economic conditions, where static neural networks cannot.

We conduct a yield curve forecasting exercise to evaluate the practical validity of our method. We demonstrate its advantages over static models. Applied to U.S. government bond yields, the proposed SDNFM delivers significantly better out-of-sample predictive accuracy compared to the random walk, where well-known benchmarks do not.

A key contribution of this work is that despite using a neural network, the factors remain interpretable. By construction, we impose Neural Nelson–Siegel (NNS) identification restrictions so that the factors can be interpreted as level, slope, and curvature factors of the yield curve. This NNS transformation ensures unique identification of the model and have a clear interpretation than factors of the Nelson–Siegel factors (Nelson and Siegel, 1987).

We demonstrate how SDNFM is effective in the well-established field of yield curve forecasting, where the factor structure is well understood and provides a natural benchmark. Beyond this specific application, SDNFM is a flexible and generalizable framework that can be extended to other multivariate forecasting problems. In particular, when there are variables or an observed structure that describe the relationship of the entries of the loading matrix, this can be flexibly tracked by the self-driving neural networks.

Acknowledgments

The estimation of the models was carried out using the the Dutch National supercomputer Snellius at SURF and the Ada cluster (formerly Bazis) at the Vrije Universiteit Amsterdam. We thank them for providing the computational resources.

References

- Avramov, D., Cheng, S., and Metzker, L. (2023). Machine Learning vs. Economic Restrictions: Evidence from Stock Return Predictability. *Management Science*, 69(5):2587–2619.
- Bai, J. and Ng, S. (2013). Principal components estimation and identification of static factors. *Journal of Econometrics*, 176(1):18–29.
- Bai, J. and Wang, P. (2015). Identification and Bayesian Estimation of Dynamic Factor Models. *Journal of Business and Economic Statistics*, 33(2):221–240.
- Barigozzi, M. and Luciani, M. (2024). Quasi Maximum Likelihood Estimation and Inference of Large Approximate Dynamic Factor Models via the EM algorithm. *Finance and Economics Discussion Series*, (2024-086):1–135.
- Bernanke, B., Boivin, J., and Eliasziw, P. (2003). Measuring the Effects of Monetary Policy: A Factor-Augmented Vector Autoregressive (FAVAR) Approach. *Finance and Economics Discussion Series*, 2003.0(3):1–47.
- Bianchi, D., Büchner, M., and Tamoni, A. (2021). Bond Risk Premiums with Machine Learning. *Review of Financial Studies*, 34(2):1046–1089.
- Chen, L., Pelger, M., and Zhu, J. (2024). Deep Learning in Asset Pricing. *Management Science*, 70(2):714–750.
- Cheung, Y. L. (2024). Identification of Time-Varying Factor Models. *Journal of Business and Economic Statistics*, 42(1):76–94.
- Christensen, J. H., Diebold, F. X., and Rudebusch, G. D. (2011). The affine arbitrage-free class of Nelson-Siegel term structure models. *Journal of Econometrics*, 164(1):4–20.
- Creal, D., Koopman, S. J., and Lucas, A. (2013). Generalized autoregressive score models with applications. *Journal of Applied Econometrics*, 28(5):777–795.
- Creal, D., Koopman, S. J., Lucas, A., and Zamojski, M. (2024). Observation-driven filtering of time-varying parameters using moment conditions. *Journal of Econometrics*, 238(2).

- 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., Piazzesi, M., and Rudebusch, G. D. (2005). Modeling bond yields in finance and macroeconomics. *American Economic Review*, 95(2):415–420.
- Doz, C., Giannone, D., and Reichlin, L. (2012). A quasi-maximum likelihood approach for large, approximate dynamic factor models. *Review of Economics and Statistics*, 94(4):1014–1024.
- Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).
- Gu, S., Kelly, B., and Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *Review of Financial Studies*, 33(5):2223–2273.
- Gu, S., Kelly, B., and Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics*, 222(1):429–450.
- Hansen, P. R., Lunde, A., and Nason, J. M. (2011). The Model Confidence Set. *Econometrica*, 79(2):453–497.
- Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2):281–291.
- Jungbacker, B. and Koopman, S. J. (2015). Likelihood-based dynamic factor analysis for measurement and forecasting. *Econometrics Journal*, 18(2):C1–C21.
- Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.
- Koopman, S. J., Mallee, M. I., and Van Der Wel, M. (2010). Analyzing the term structure of interest rates using the dynamic Nelson-Siegel model with time-varying parameters. *Journal of Business and Economic Statistics*, 28(3):329–343.
- Kose, M. A., Otrok, C., and Whiteman, C. H. (2003). International business cycles: World, region, and country-specific factors. *american economic review*, 93(4):1216–1239.

- Liu, Y. and Wu, J. C. (2021). Reconstructing the yield curve. *Journal of Financial Economics*, 142(3):1395–1425.
- Mikkelsen, J. G., Hillebrand, E., and Urga, G. (2019). Consistent estimation of time-varying loadings in high-dimensional factor models. *Journal of Econometrics*, 208(2):535–562.
- Negro, M. D. and Otrok, C. (2008). Dynamic factor models with time-varying parameters: measuring changes in international business cycles. Technical Report 326, Federal Reserve Bank of New York.
- Nelson, C. R. and Siegel, A. F. (1987). Parsimonious Modeling of Yield Curves. *The Journal of Business*, 60(4):473.
- Quaedvlieg, R. (2021). Multi-Horizon Forecast Comparison. *Journal of Business and Economic Statistics*, 39(1):40–53.
- Stock, J. H. and Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, 97(460):1167–1179.
- Stock, J. H. and Watson, M. W. (2016). *Dynamic Factor Models, Factor-Augmented Vector Autoregressions, and Structural Vector Autoregressions in Macroeconomics*, volume 2. Elsevier B.V., 1 edition.
- Su, L. and Wang, X. (2017). On time-varying factor models: Estimation and testing. *Journal of Econometrics*, 198(1):84–101.
- Tieleman, T. and Hinton, G. (2012). Lecture 6.5 — “rmsprop: Divide the gradient by a running average of its recent magnitude”. Coursera: Neural Networks for Machine Learning. Video lecture; section 6.5 of the course.
- Zamojski, M. (2019). Self-driving score filters. Technical report.

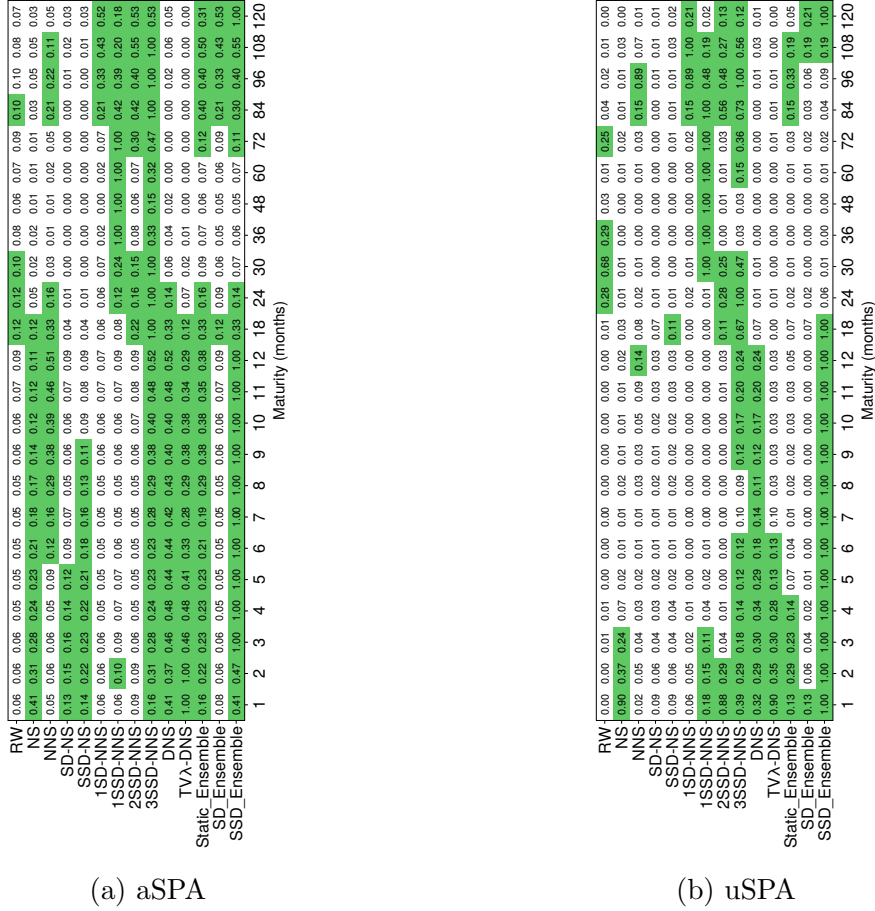


FIGURE 12: MCS and p-values per maturity at significance level $\alpha_{MCS} = 0.1$ for $B = 999$ bootstrap samples

A 24 maturities moving window

For completeness, we report the forecast evaluation results of the 23 maturities: 1 to 12, 18, 24, 30, 36, 48, 60, 72, 84, 96, 108, and 120 months. The models are the same as in Section 5, so they are not reestimated. The goal is to document that the conclusions in Section 5 are not a consequence of the smaller maturity set. Table 6 reports the one-step-ahead relative RMSE for all 24 maturities.

Table 7 shows the twelve-step-ahead results.

Figure 12 presents the aSPA and uSPA model confidence sets across maturities for forecast horizons 1 to 12. The results remain very similar to those in the main text. The 3SSD-NNS continues to perform well across the entire curve and is rarely excluded from the model confidence set. Most competing models are excluded for large parts of the maturity range.

TABLE 6: US Yield Curve Forecasting, Out-of-Sample Relative RMSE (Model RMSE / RW RMSE) for 1-Month-Ahead Forecast Horizon - Moving Window

Model	m-1	m-2	m-3	m-4	m-5	m-6	m-7	m-8	m-9	m-10	m-11	m-12	m-18	m-24	m-30	m-36	m-48	m-60	m-72	m-84	m-96	m-108	m-120	Average
Benchmark																								
RW	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Static																								
NS	0.935	0.907*	0.9**	0.913**	0.938*	0.966	0.993	1.015	1.034	1.048	1.056	1.06	1.036*	1.035**	1.052**	1.064**	1.074**	1.065**	1.052*	1.014	0.999	0.979	0.989	1.011
NNS	1.066	0.976	0.94	0.929**	0.934**	0.946	0.962	0.981	1.0	1.018	1.033	1.044	1.038	1.032	1.056*	1.078**	1.09**	1.063**	1.044*	1.001	0.987	0.977	1.029	1.017**
Self-driving																								
SD-NS	0.951	0.934*	0.924**	0.92***	0.923**	0.928*	0.933*	0.938	0.944	0.95	0.955	0.959	0.984	1.023	1.052*	1.063**	1.067*	1.055*	1.051*	1.021	1.015	0.996	0.992	0.993
SSD-NS	0.942	0.927**	0.918***	0.917***	0.921**	0.927*	0.933*	0.939	0.945	0.951	0.956	0.96	0.983	1.022	1.052*	1.063**	1.067*	1.054*	1.048	1.016	1.01	0.991	0.984	0.991
1SD-NNS	0.966	0.931	0.932*	0.943	0.954	0.961	0.963	0.964	0.967	0.971	0.976	0.982	1.002	1.024	1.041	1.051	1.069	1.054	1.04	1.005	0.993	0.967	0.98	0.996
1SSD-NNS	0.923**	0.914**	0.914**	0.927**	0.941*	0.949*	0.952*	0.954*	0.958	0.964	0.973	0.983	1.023	1.027	1.015	1.004	0.993	0.99	1.004	0.992	0.997	0.988	1.004	0.979***
2SSD-NNS	0.91**	0.911**	0.922**	0.941*	0.957	0.964	0.963	0.96*	0.957*	0.957*	0.958*	0.962*	0.989	1.014	1.023	1.026	1.021	1.011	1.018	1.0	0.997	0.981	0.995	0.982***
3SSD-NNS	0.919**	0.909**	0.911**	0.922**	0.931**	0.933**	0.933**	0.932**	0.933**	0.938**	0.946**	0.954**	0.997	1.018	1.022	1.023	1.023	1.018	1.025	1.004	0.999	0.98	0.993	0.977***
Kalman filter																								
DNS	1.023	0.968	0.933	0.92*	0.927**	0.945	0.967	0.99	1.011	1.028	1.041	1.049	1.04*	1.037**	1.049**	1.062**	1.08**	1.076**	1.067**	1.026	1.007	0.982	0.988	1.017**
TVλ-DNS	1.079	1.015	0.962	0.927	0.913**	0.914**	0.926**	0.945*	0.966	0.987	1.007	1.024	1.063***	1.073***	1.078***	1.073**	1.061*	1.042	1.046*	1.02	1.013	1.001	1.001	1.015**
Ensembles																								
Static Ensemble	0.99	0.932	0.913**	0.912***	0.927**	0.948	0.971	0.993	1.013	1.03	1.042	1.049	1.031	1.029*	1.05*	1.068**	1.079**	1.061*	1.045	1.005	0.989	0.972	0.999	1.009
SD Ensemble	0.926*	0.907**	0.907***	0.908***	0.912***	0.915***	0.919***	0.924**	0.931**	0.939**	0.947**	0.955*	0.986	1.018	1.04	1.048*	1.056	1.042	1.035	1.006	0.998	0.974	0.975	0.979***
SSD Ensemble	0.91**	0.897***	0.897***	0.899***	0.902***	0.903***	0.905***	0.909***	0.915***	0.924***	0.933**	0.943**	0.985	1.017	1.032	1.036	1.034	1.023	1.024	1.0	0.996	0.978	0.979	0.97***

Notes: Asterisks denote Diebold-Mariano rejection of equal RMSE with the RW benchmark at *10%, **5%, and ***1% significance levels.

TABLE 7: US Yield Curve Forecasting, Out-of-Sample Relative RMSE (Model RMSE / RW RMSE) for 12-Month-Ahead Forecast Horizon - Moving Window

Model	m-1	m-2	m-3	m-4	m-5	m-6	m-7	m-8	m-9	m-10	m-11	m-12	m-18	m-24	m-30	m-36	m-48	m-60	m-72	m-84	m-96	m-108	m-120	Average
Benchmark																								
RW	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Static																								
NS	0.843	0.855	0.866	0.876	0.885	0.893	0.9	0.907	0.913	0.92	0.928	0.935	0.983	1.027	1.06	1.081	1.095	1.092	1.077	1.063	1.051	1.041	1.032	0.947**
NNS	0.872	0.867	0.872	0.877	0.88	0.883	0.886	0.888	0.891	0.894	0.898	0.902	0.928	0.949	0.966	0.976	0.984	0.98	0.979	0.977	0.974	0.974	0.982	0.911***
Self-driving																								
SD-NS	0.843	0.854	0.865	0.875	0.883	0.891	0.898	0.905	0.913	0.92	0.928	0.936	0.988	1.037	1.079	1.108	1.137	1.145	1.134	1.122	1.105	1.085	1.058	0.958*
SSD-NS	0.841	0.852	0.863	0.872	0.881	0.888	0.895	0.902	0.909	0.916	0.924	0.932	0.982	1.03	1.07	1.099	1.128	1.136	1.126	1.116	1.1	1.082	1.057	0.954**
ISD-NNS	0.863	0.868	0.878	0.886	0.893	0.898	0.903	0.906	0.91	0.913	0.917	0.921	0.943	0.956	0.964	0.967	0.968	0.966	0.968	0.968	0.965	0.962	0.961	0.918***
ISSD-NNS	0.852	0.858	0.868	0.876	0.884	0.89	0.895	0.9	0.904	0.908	0.913	0.917	0.939	0.95	0.955	0.956	0.957	0.956	0.96	0.965	0.964	0.963	0.963	0.91***
2SSD-NNS	0.85	0.858	0.869	0.878	0.885	0.89	0.895	0.899	0.902	0.906	0.909	0.913	0.935	0.949	0.958	0.962	0.964	0.961	0.962	0.963	0.96	0.958	0.959	0.91***
3SSD-NNS	0.844	0.847	0.858	0.867	0.874	0.88	0.885	0.889	0.893	0.897	0.901	0.905	0.929	0.943	0.953	0.957	0.958	0.956	0.959	0.961	0.959	0.958	0.96	0.902***
Kalman filter																								
DNS	0.833	0.841	0.85	0.857	0.865	0.871	0.877	0.883	0.889	0.896	0.902	0.909	0.949	0.983	1.01	1.031	1.058	1.072	1.076	1.073	1.065	1.056	1.043	0.926***
TVA-DNS	0.819*	0.835**	0.85**	0.864**	0.876**	0.886*	0.896*	0.905*	0.914*	0.922*	0.93	0.938	0.98	1.013	1.041	1.066	1.106	1.128	1.142	1.145	1.143	1.141	1.128	0.953***
Ensembles																								
Static Ensemble	0.848	0.854	0.862	0.869	0.875	0.881	0.886	0.89	0.895	0.899	0.904	0.909	0.942	0.969	0.99	1.003	1.011	1.005	0.995	0.985	0.975	0.969	0.965	0.915***
SD Ensemble	0.836	0.845	0.856	0.866	0.874	0.881	0.887	0.893	0.899	0.904	0.91	0.916	0.952	0.979	1.0	1.012	1.019	1.015	1.006	0.997	0.983	0.97	0.953	0.917***
SSD Ensemble	0.823	0.833	0.844	0.854	0.863	0.87	0.877	0.883	0.889	0.895	0.901	0.907	0.942	0.97	0.991	1.003	1.011	1.008	1.0	0.992	0.98	0.968	0.953	0.907***

Notes: Asterisks denote Diebold-Mariano rejection of equal RMSE with the RW benchmark at *10%, **5%, and ***1% significance levels.

RW	0.12	0.10	0.12	0.13	0.15	0.14	0.13	0.12	0.10	0.09	0.08	0.08
NS	0.01	0.07	0.12	0.10	0.11	0.12	0.10	0.09	0.07	0.07	0.06	0.05
NNS	0.01	0.06	0.12	0.16	0.18	0.14	0.13	0.18	0.21	0.22	0.19	0.19
SD-NS	0.04	0.06	0.06	0.06	0.06	0.05	0.04	0.03	0.02	0.01	0.01	0.01
SSD-NS	0.04	0.05	0.06	0.06	0.05	0.04	0.03	0.02	0.01	0.01	0.00	0.00
1SD-NNS	0.03	0.05	0.07	0.07	0.10	0.06	0.09	0.09	0.08	0.07	0.04	0.04
1SSD-NNS	0.12	0.23	0.12	0.16	0.18	0.11	0.12	0.12	0.10	0.09	0.11	0.15
2SSD-NNS	0.12	0.28	0.16	0.12	0.18	0.10	0.13	0.12	0.13	0.17	0.13	0.15
3SSD-NNS	0.12	0.28	0.29	0.40	0.49	0.54	0.49	1.00	1.00	1.00	1.00	1.00
DNS	0.01	0.07	0.12	0.16	0.18	0.18	0.23	0.22	0.21	0.22	0.19	0.19
TVλ-DNS	0.06	0.09	0.12	0.15	0.18	0.14	0.13	0.12	0.13	0.17	0.14	0.15
Static_Ensemble	0.01	0.10	0.12	0.16	0.24	0.21	0.23	0.24	0.21	0.22	0.20	0.19
SD_Ensemble	0.06	0.08	0.10	0.14	0.18	0.14	0.13	0.12	0.10	0.09	0.08	0.07
SSD_Ensemble	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.50	0.47	0.39	0.32	0.23
	1	2	3	4	5	6	7	8	9	10	11	12

(a) aSPA

RW	0.14	0.15	0.05	0.03	0.03	0.02	0.02	0.01	0.01	0.01	0.02	0.01
NS	0.07	0.62	0.75	0.53	0.59	0.36	0.35	0.24	0.12	0.14	0.10	0.07
NNS	0.12	0.60	0.52	0.39	0.24	0.12	0.14	0.24	0.51	0.54	0.59	0.53
SD-NS	0.07	0.02	0.02	0.02	0.18	0.12	0.13	0.12	0.12	0.07	0.05	0.03
SSD-NS	0.12	0.04	0.05	0.02	0.18	0.02	0.02	0.12	0.10	0.07	0.05	0.03
1SD-NNS	0.07	0.41	0.68	0.97	0.47	0.19	0.19	0.22	0.20	0.12	0.50	0.53
1SSD-NNS	0.95	1.00	0.68	0.43	0.45	0.28	0.64	0.65	0.70	0.57	0.59	0.53
2SSD-NNS	0.07	0.99	0.75	0.43	0.59	0.19	0.64	0.65	0.48	0.57	0.59	0.53
3SSD-NNS	0.73	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
DNS	0.01	0.04	0.19	0.43	0.47	0.58	0.60	0.65	0.70	0.66	0.59	0.53
TVλ-DNS	0.07	0.44	0.68	0.97	1.00	0.98	0.87	0.81	0.71	0.74	0.60	0.64
Static_Ensemble	0.52	0.85	0.75	0.97	0.98	0.98	0.90	0.80	0.71	0.66	0.60	0.53
SD_Ensemble	0.07	0.05	0.04	0.37	0.40	0.24	0.22	0.14	0.12	0.37	0.45	0.53
SSD_Ensemble	1.00	0.99	0.96	0.97	0.98	0.98	0.95	0.97	0.98	0.96	0.90	0.82
	1	2	3	4	5	6	7	8	9	10	11	12

(b) uSPA

FIGURE 13: MCS and p-values per forecast horizon at significance level $\alpha_{MCS} = 0.1$ for $B = 999$ bootstrap samples

Figure 13 shows the corresponding model confidence sets across forecast horizons using all 24 maturities. The patterns again match those reported in the main text.