Arbitrage-free affine autoencoder term structure models[*]

Rastin Matin[†] Wenjie Wan[‡] Rolf Poulsen[§]

May 23, 2025

1 Introduction

An encoder-decoder model is a neural network that goes shallow in the middle. In other words, it gives a compressed representation of its input via a narrow hidden layer of factors (encoding) from which the original input – and possibly more – can be satisfactorily reproduced (decoding). Such non-linear or non-parametric dimensionality reduction fits hand-in-glove for modeling curve dynamics in finance, as pointed out by for instance Kondratyev (2018) and Sokol (2022). These models are not arbitrage-free by default though, so a condition must be imposed if they are to be used for risk-neutral term structure modeling.

The mother of all financial curves is the discount function, which tells us the value today (t) of receiving 1 unit at the future date $T = t + \tau$, where we call τ (time to) maturity. This may sound trivial, but things move stochastically with time t, and there are many τs, so it very much isn't. Term structure theory and models tell us how to handle this. It gives us the necessary no-arbitrage relations between the current yield curve, its dynamics and possible future yield curves.

In this work we follow the insight of Andreasen (2023) and impose no-arbitrage conditions at the decoding stage of the model. But whereas Andreasen implements it as a soft constraint via a regularization term, we impose it as a hard constraint in a way that

[*]The authors thank Morten Barslund, Mads Gydesen and David Skovmand for helpful comments. The views expressed in this paper are those of the authors and do not necessarily reflect the positions of their employers or affiliates.

[†]PFA Asset Management. rtm@pfa.dk
[‡]wnw19940819@gmail.com
[§]Department of Mathematical Sciences, University of Copenhagen. rolf@math.ku.dk

still allows us to specify and train/estimate models that are both flexible and exactly arbitrage-free.

2 Autoencoding Swap Curves

Encoder-decoder models often take as input a set of swap rates with discrete maturities and output a low-dimensional representation from which a continuous representation of the instantaneous forward rate is extracted via some curve constructor. The work of Sokol (2022) and Andreasen (2023) are examples of this. The novelty of our approach lies in the fact that this continuous representation is established by the encoder-decoder itself. In the following, we present the underlying theory and how to implement the model in practice.

2.1 Linear Encoding

The architecture of our model is illustrated in Figure 1. Denoting the input and output layer by n_0 and n_3, respectively, the model contains two hidden layers n_1 and n_2. The input $x_0(t)$ at n_0 is the set of observed par swap rates S with discrete maturities

$$\{1\text{Y},\ 2\text{Y},\ 3\text{Y},\ 5\text{Y},\ 10\text{Y},\ 15\text{Y},\ 20\text{Y},\ 30\text{Y}\} \tag{1}$$

Yield curves such as these are known for having low dimensionality, meaning they can be represented with few latent variables as noted by Sokol (2022). Our model subsequently employs only two latent factors $z_t \equiv (z_1(t), z_2(t))$ for the compressed representation of the input, which our encoder generates using a linear transformation

$$z_t = A_1 x_0(t) \tag{2}$$

Here A_1 is a matrix that maps $x_0(t)$ from $\mathbb{R}^8$ to z_t in $\mathbb{R}^2$. The latent factors z_t can be decoded back to par swap rates S by using zero-coupon bond prices. Ignoring the basis between forward rates and discounting, we compute the estimated swap rates $\tilde{S}$ as (see, e.g., Henry-Labordere (2008), Björk (2009) or Andreasen (2023))

$$\tilde{S}(z_t, \tau) = \frac{1 - P(z_t, \tau)}{\sum_{\tau' \leq \tau} P(z_t, \tau')} \tag{3}$$

where τ is a maturity on the annually-spaced grid

$$\tau \in \{1\text{Y}, 2\text{Y}, 3\text{Y}, \ldots, 29\text{Y}, 30\text{Y}\} \tag{4}$$

and $P(z_t, \tau)$ denotes the time-t price of zero-coupon bonds on this grid. For clarity, we note that the sum in (3) runs over maturities in (4) smaller than or equal to τ. In the following we characterize this architecture as *autoencoder*. An autoencoder is a special case of an encoder-decoder model, where the input and output domains are the same (Minaee et al., 2022). Although our input and output domains differ slightly due to the transformation in (3), we use this nomenclature to comply with recent literature in the field.

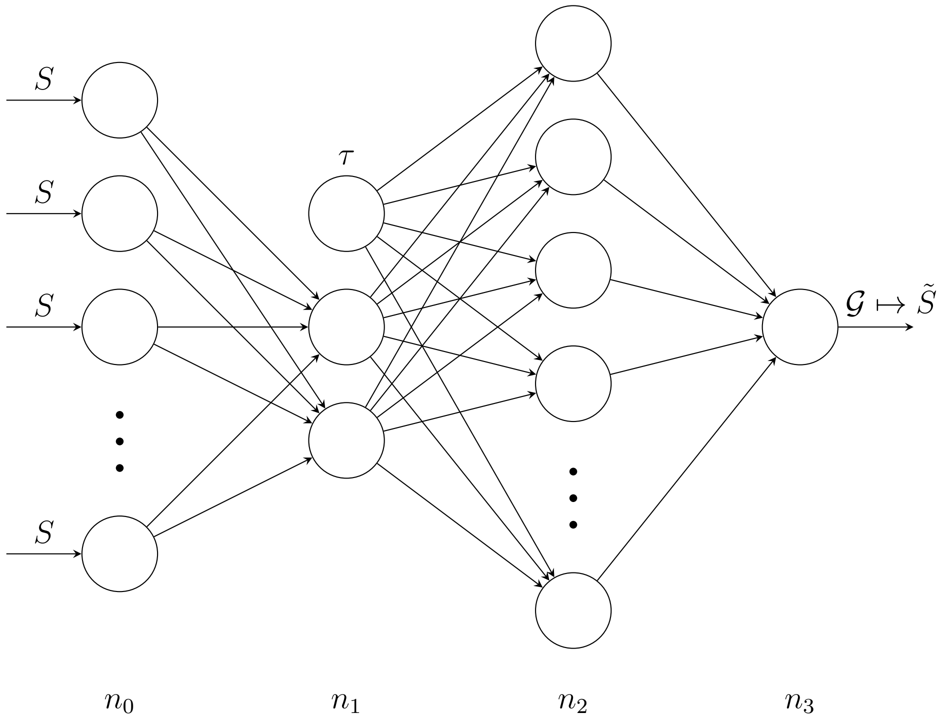


Figure 1: Architecture of the affine encoder-decoder model for par swap rates. The encoder has dimensions $(n_0, n_1) = (8, 2)$ while the decoder has dimensions $(n_1, n_2, n_3) = (3, 10, 1)$. The input S is the set of observed par swap rates with discrete maturities $\{1\text{Y}, 2\text{Y}, 3\text{Y}, 5\text{Y}, 10\text{Y}, 15\text{Y}, 20\text{Y}, 30\text{Y}\}$ while the estimated swap rates $\tilde{S}$ are computed from zero-coupon bond prices using a decoding neural network $\mathcal{G}$. The value of τ is a maturity on the annually-spaced grid $\{1\text{Y}, 2\text{Y}, 3\text{Y}, \ldots, 29\text{Y}, 30\text{Y}\}$.

2.2 Arbitrage-Free Decoding

Now that the encoder has been established, we turn our attention to the decoder and the model governing zero-coupon bond prices at n_3. We draw inspiration from the classical affine model of Duffie and Kan (1996), where a zero-coupon bond with maturity τ is expressed in terms of a vector of latent variables x

$$P(x,\tau) = \exp\left(A(\tau) + \sum_i B_i(\tau) x_i \right) \tag{5}$$

The scalar functions $A(\tau)$ and $B_i(\tau)$ can be obtained by solving a system of coupled ordinary differential equations (ODEs). We take a neural leap and replace $\sum_i B_i(\tau)x_i$ with a scalar function $B(\tau)$ multiplied by a decoding neural network $\mathcal{G}(z_t,\tau)$ which maps the tuple $(z_t,\, \tau)$ to a scalar, i.e.

$$P(z_t,\tau) = \exp(A(\tau) - B(\tau)\mathcal{G}(z_t,\tau)) \tag{6}$$

The dependency on τ we account for by including it in the first layer n_1 of the decoding network as illustrated in Figure 1. It has dimensions $(n_1, n_2, n_3) = (3, 10, 1)$ and, as is often the case when designing neural networks, we optimized these and all subsequent network dimensions on an empirical basis. We use a centered soft step function defined by

$$\psi(x) = \frac{1}{1+e^{-x}} - \frac{1}{2} \tag{7}$$

as the activation function.

We now look at the short rate and how to model it consistently. The instantaneous forward rate $f(t,\tau)$ in our framework (6) is given by

$$\begin{align} f(t,\tau) &= -\frac{\partial \log P(t,\tau)}{\partial \tau} \tag{8} \\ &= -\frac{\partial A(\tau)}{\partial \tau} + \frac{\partial B(\tau)}{\partial \tau}\mathcal{G}(z_t,\tau) + \frac{\partial \mathcal{G}(z_t,\tau)}{\partial \tau} B(\tau) \tag{9} \end{align}$$

As the short rate $r(t)$ is defined as

$$r(t) = \lim_{\tau \to 0} f(t,\tau) \tag{10}$$

we conclude that $r(t)$ must be a function of the latent factors z_t. Next, we know that absence of arbitrage implies that bond prices can be written as a risk-neutral expectation of the bank account, i.e. (with a slight abuse of notation)

$$P(t, t+\tau) = \mathbb{E}_t^{\mathbb{Q}}\left[e^{-\int_t^{t+\tau} r(u)du}\right] \tag{11}$$

Comparing (11) to (6) we furthermore conclude that z_t is a Markov process under $\mathbb{Q}$. It is therefore natural to impose a dynamics on z_t and assume that $r(t)$ is a continuous transformation $\tilde{r}$ of z_t

$$dz_t = \mu(z_t)dt + \sigma(z_t)dW_t^{\mathbb{Q}} \tag{12}$$

$$r(t) = \tilde{r}(z_t) \tag{13}$$

where $\mu(z_t)$ and $\sigma(z_t)$ denote the drift and volatility of z_t, respectively, and $W_t^{\mathbb{Q}}$ represents standard Brownian motions under the risk-neutral measure $\mathbb{Q}$. At first sight (13) might seem redundant as the short rate $r(t)$ can already be obtained through (10) and, more importantly, how to guarantee (10) and (13) yield the same value? We show at the end of this section that (13) is indeed necessary and serves as a "hard" constraint for the arbitrage-free decoding neural network $\mathcal{G}$ such that

$$r(t) = \lim_{\tau \to 0} f(t, \tau) = \tilde{r}(z_t) \tag{14}$$

is satisfied by construction.

With this knowledge at hand, we can derive a differential equation for P by using the no-arbitrage condition that the process $P(t, t+\tau)$ divided by the bank account

$$B(t) = \exp\left(\int_0^t \tilde{r}(u)du\right) \tag{15}$$

must be a martingale. This condition yields the following partial differential equation (PDE)

$$-\tilde{r}(z_t)P(z_t, \tau) - \partial_\tau P(z_t, \tau) + (\nabla_z P(z_t, \tau))^T \mu(z_t) + \frac{1}{2}\operatorname{Tr}\left[\sigma^T(z_t)H_z P(z_t, \tau)\sigma(z_t)\right] = 0 \tag{16}$$

Furthermore, it is easy to verify that (6) satisfies the following relations

$$\partial_\tau P = (\partial_\tau A(\tau) - \mathcal{G}(z_t,\tau)\partial_\tau B(\tau) - B(\tau)\partial_\tau \mathcal{G}(z_t,\tau))P \tag{17}$$

$$\nabla_z P = -B(\tau)\nabla_z \mathcal{G}(z_t,\tau)P \tag{18}$$

$$H_z P = \big(B^2(\tau)\nabla_z \mathcal{G}(z_t,\tau)\nabla_z^T \mathcal{G}(z_t,\tau) - B(\tau)H_z\mathcal{G}(z_t,\tau)\big)P \tag{19}$$

By substituting (17)-(19) into (16) we obtain

$$\begin{aligned}
&-\tilde{r}(z_t)P - (\partial_\tau A(\tau) - \mathcal{G}(z_t,\tau)\partial_\tau B(\tau) - B(\tau)\partial_\tau \mathcal{G}(z_t,\tau))\,P \\
&+ \big(-B(\tau)\nabla_z^T \mathcal{G}(z_t,\tau)\mu(z_t)\big)\,P \\
&+ \frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)\,\big(B^2(\tau)\nabla_z \mathcal{G}(z_t,\tau)\nabla_z^T \mathcal{G}(z_t,\tau) - B(\tau)H_z\mathcal{G}(z_t,\tau)\big)\,\sigma(z_t)\big]\,P = 0
\end{aligned} \tag{20}$$

which we rewrite as

$$\begin{aligned}
&-\tilde{r}(z_t) - \partial_\tau A(\tau) + \mathcal{G}(z_t,\tau)\partial_\tau B(\tau) + B(\tau)\partial_\tau \mathcal{G}(z_t,\tau) - B(\tau)\nabla_z^T \mathcal{G}(z_t,\tau)\mu(z_t) \\
&+ B^2(\tau)\frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)\nabla_z \mathcal{G}(z_t,\tau)\nabla_z^T \mathcal{G}(z_t,\tau)\sigma(z_t)\big] \\
&- B(\tau)\frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)H_z\mathcal{G}(z_t,\tau)\sigma(z_t)\big] = 0
\end{aligned} \tag{21}$$

We then group terms according to their order of $B(\tau)$

$$\begin{aligned}
&-\tilde{r}(z_t) - \partial_\tau A(\tau) + \mathcal{G}(z_t,\tau)\partial_\tau B(\tau) \\
&+ B(\tau)\left(\partial_\tau \mathcal{G}(z_t,\tau) - \nabla_z^T \mathcal{G}(z_t,\tau)\mu(z_t) - \frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)H_z\mathcal{G}(z_t,\tau)\sigma(z_t)\big]\right) \\
&+ B^2(\tau)\frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)\nabla_z \mathcal{G}(z_t,\tau)\nabla_z^T \mathcal{G}(z_t,\tau)\sigma(z_t)\big] = 0
\end{aligned} \tag{22}$$

and separate the expression in two coupled ODEs

$$\partial_\tau B(\tau) = B(\tau)\underbrace{\frac{-\partial_\tau \mathcal{G}(z_t,\tau) + \nabla_z^T \mathcal{G}(z_t,\tau)\mu(z_t) + \frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)H_z\mathcal{G}(z_t,\tau)\sigma(z_t)\big]}{\mathcal{G}(z_t,\tau)}}_{\alpha(z_t,\tau)} + \underbrace{\frac{\tilde{r}(z_t)}{\mathcal{G}(z_t,\tau)}}_{\beta(z_t,\tau)} \tag{23}$$

$$\partial_\tau A(\tau) = B^2(\tau)\underbrace{\frac{1}{2}\operatorname{Tr}\big[\sigma^T(z_t)\nabla_z \mathcal{G}(z_t,\tau)\nabla_z^T \mathcal{G}(z_t,\tau)\sigma(z_t)\big]}_{\gamma(z_t,\tau)} \tag{24}$$

with initial conditions $A(\tau = 0) = B(\tau = 0) = 0$. This system of coupled ODEs can be absorbed to a single ODE by introducing the vector $X(\tau) = (A(\tau), B(\tau))^T$ such that the no-arbitrage PDE in (16) can finally be expressed as the following ODE for each value of z_t coming from the encoding network

$$\partial_\tau X(\tau) = \begin{pmatrix} 0 & \gamma(z_t, \tau) \\ 0 & 0 \end{pmatrix} (X(\tau) \odot X(\tau)) + \begin{pmatrix} 0 & 0 \\ 0 & \alpha(z_t, \tau) \end{pmatrix} X(\tau) + \begin{pmatrix} 0 \\ \beta(z_t, \tau) \end{pmatrix} \tag{25}$$

where $X(\tau = 0) = (0, 0)^T$ and $\odot$ represents element-wise multiplication. By solving the ODE in (25) using fourth-order Runge-Kutta with 3/8 rule and applying (6) to the solved $A(\tau)$ and $B(\tau)$, we obtain the zero-coupon bond prices $P(z_t, \tau)$ used to compute the estimated swap rates $\tilde{S}$ in (3).

All that remains now is to explicitly verify that the limit (14) holds. From (9) we have

$$\lim_{\tau \to 0} f(t, \tau) = \lim_{\tau \to 0} \left(-\frac{\partial A(\tau)}{\partial \tau} + \frac{\partial B(\tau)}{\partial \tau} \mathcal{G}(z_t, \tau) + \frac{\partial \mathcal{G}(z_t, \tau)}{\partial \tau} B(\tau) \right) \tag{26}$$

We furthermore note that (23) and (24) given $\tau = 0$ yield boundary conditions for the first-order derivatives of $B(\tau)$ and $A(\tau)$

$$\partial_\tau B(\tau = 0) = \frac{\tilde{r}(z_t)}{\mathcal{G}(z_t, \tau = 0)} \tag{27}$$

$$\partial_\tau A(\tau = 0) = 0 \tag{28}$$

and since $B(\tau = 0) = 0$ we arrive at

$$\lim_{\tau \to 0} f(t, \tau) = \tilde{r}(z_t) \tag{29}$$

as advertised.

2.3 Parameter modeling

The last step remaining before a practical implementation is to parametrize (12) and (13). We start by writing (12) explicitly for a two factor model

$$\begin{pmatrix} dz_1(t) \\ dz_2(t) \end{pmatrix} = \underbrace{\begin{pmatrix} \mu_1(z_t) \\ \mu_2(z_t) \end{pmatrix}}_{\mu} dt + \underbrace{\begin{pmatrix} \sigma_1(z_t) & 0 \\ \rho(z_t)\sigma_2(z_t) & \sqrt{1 - \rho^2(z_t)}\sigma_2(z_t) \end{pmatrix}}_{\sigma} \begin{pmatrix} dW_1^{\mathbb{Q}}(t) \\ dW_2^{\mathbb{Q}}(t) \end{pmatrix} \tag{30}$$

The parameters governing this stochastic differential equation can be modelled with neural networks. For the drift μ we introduce a neural network $\mathcal{K}$ that models it as a mean-reverting process by performing the linear mapping

$$\mathcal{K}:\begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} \rightarrow \begin{pmatrix} \mu_1(z_t) \\ \mu_2(z_t) \end{pmatrix} = \mathbf{M}\begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} + \mathbf{N} \tag{31}$$

where $\mathbf{M}$ and $\mathbf{N}$ are 2×2 and 2×1 real matrices, respectively. Looking at the volatility matrix σ, we construct a neural network for determining σ_1, σ_2 and ρ. Dubbed $\mathcal{H}$, it has dimensions $(2, 4, 3)$ and performs the mapping

$$\mathcal{H}:\begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} \rightarrow \begin{pmatrix} \log(\sigma_1) \\ \log(\sigma_2) \\ \operatorname{atanh}(\rho) \end{pmatrix} \tag{32}$$

where the transformations ensure the positivity of the volatilities and the boundedness of the correlation. We transform them back to volatilities and correlation afterwards.

The risk-free rate $\tilde{r}$ in (13) is also modelled by constructing a neural network $\mathcal{R}$ with dimensions $(2, 4, 1)$ that performs the mapping

$$\mathcal{R}:\begin{pmatrix} z_1(t) \\ z_2(t) \end{pmatrix} \rightarrow \tilde{r} \tag{33}$$

The activation function in the latent layer is the centered soft step function defined in (7) for both $\mathcal{H}$ and $\mathcal{R}$ and all bias terms are set to zero.

2.4 Model Estimation

The autoencoder in Figure 1 is estimated by adjusting the weights A_1 of the encoder and the networks $\mathcal{G}$, $\mathcal{K}$, $\mathcal{H}$ and $\mathcal{R}$ of the decoder such that the ℓ_2 loss f between the input $S_t(\tau)$ and output $\tilde{S}_t(\tau)$ is minimized across all observation dates t. Denoting the total set of training parameters by Ω, the steady state solution therefore satisfies

$$\hat{\Omega} = \min_{\Omega} f(\Omega) \tag{34}$$

$$= \min_{\Omega} \frac{1}{N} \sum_{t=1}^{N} \sum_{\tau} \left(S_t(\tau) - \tilde{S}_t(\tau; \Omega) \right)^2 \tag{35}$$

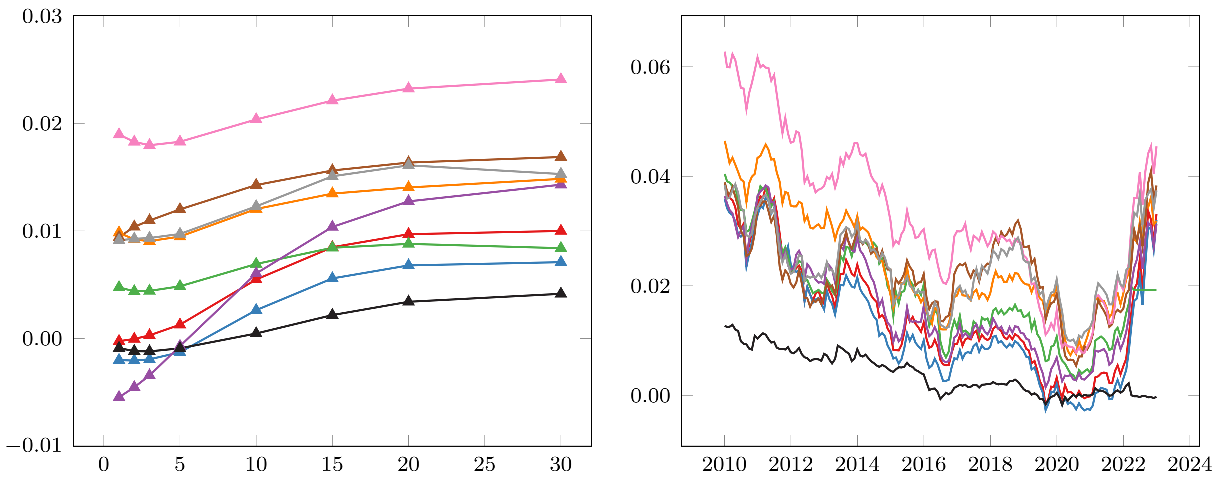
where N is the total number of observations. The numerical solution to this minimization problem is found by stochastic gradient descent using the ADAM optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3} and training batch size of 32 for 1 000 epochs. The backpropagation through the ODE solver follows the methodology by Chen et al. (2018).

2.5 Results

We employ a data set similar to that used by Andreasen (2023) consisting of liquid par swap rates for the maturities in (1) for the 9 currencies

$$\text{DKK, EUR, GBP, SEK, NOK, USD, AUD, CAD, JPY} \tag{36}$$

The rates are observed monthly over a 13-year period from January 2010 to December 2022, yielding a total of 1412 curves. Examples of the curves are illustrated in Figure 2.



(a) (b)

Figure 2: Swap rates for the currencies DKK (—▲—), EUR (—▲—), GBP (—▲—), SEK (—▲—), NOK (—▲—), USD (—▲—), AUD (—▲—), CAD (—▲—), JPY (—▲—) for (a) the maturities in (1) on the 30th August 2016 and (b) a 10 year maturity over the period 4th January 2010 – 30th December 2022.

We follow Sokol (2022) and Andreasen (2023) and assume that the currencies can be represented by the same model, meaning that we weigh all swap curves equally during training of the autoencoder. Combining curves from multiple currencies into a single data set is a clever way to obtain sufficient training data and, as remarked by Sokol (2022), enhances the autoencoder's ability to accurately represent currency-specific curves.

2.6 Performance

The resulting in-sample root mean square errors (RMSE) in basis points (bps) are presented in Table 1.

Currency	RMSE
DKK	12.76 bps
EUR	18.37 bps
GBP	10.71 bps
SEK	15.45 bps
NOK	9.59 bps
USD	11.75 bps
AUD	11.94 bps
CAD	10.44 bps
JPY	12.93 bps
Average	12.93 bps

Table 1: Steady-state in-sample RMSE of the autoencoder in Figure 1.

The RMSEs range between 9.59 bps and 18.37 bps across currencies with an average just below 13 bps. For a fairly simple model (linear encoder, two factors, constant parameters across both time and currencies), this is very decent accuracy. Andreasen (2023) considers the same time period and yield curves (except that he also includes CZK and SGD but has excluded SEK) on which he trains a two-factor autoencoder that is similar (but not completely identical) to ours. He reports an average RMSE of 10 bps. The main difference between the two approaches lies in the treatment of the no-arbitrage condition. Our model is completely arbitrage-free by construction (albeit with added computational cost stemming from having to solve ODEs), while Andreasen's isn't (enabling him to achieve a better fit).[1] Rather, he applies a no-arbitrage check by calculating the zero-coupon bond Sharpe ratios implied by the trained network. For the model to be arbitrage-free, they must all be zero – which they aren't quite; values range between -0.15 and 0.1.[2] Notice how the results from the two approaches corroborate each other; there is information not only in the similarities but also in the differences. Looking at Andreasen's results on a stand-alone basis, it would not (at least to us) be clear whether the magnitude of the Sharpe ratios should concern us. The results from our model tells us that they shouldn't. On the other hand, looking at our results on a stand-alone basis, one might worry that the hard enforcement of the no-arbitrage condition could significantly diminish the fit to data, potentially causing us to miss arbitrages or good deals. The results from Andreasen's model tell that it doesn't.

[1]In subsequent unpublished work, Andreasen includes the Sharpe ratio term as a penalty or regularization term in the loss function when training the model.

[2]Similar calculations in our model give Sharpe ratios less 10^{-10} in absolute value.

2.6.1 Relation to Principal Component Analysis

As our encoder is merely a linear transformation, we can directly compare our latent factors to the principal components simply by projecting the weight matrix of the encoder on to the space spanned by the principal components. We choose the eight components with the largest eigenvalue illustrated in Figure 3.

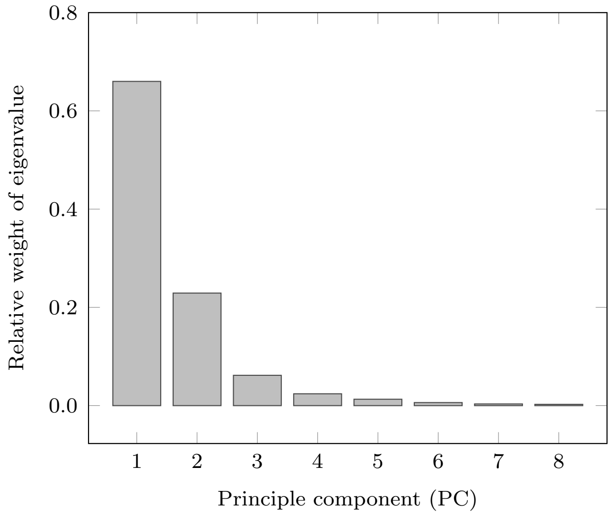


Figure 3: The relative weights of eigenvalues are sorted in a decreasing order. The weights are negligible for the fourth principal component and beyond.

To facilitate the comparison, we start by writing a given encoding weight matrix A_1 as

$$A_1 = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \tag{37}$$

where $W_1 \in \mathbf{R}^{1\times 8}$ and $W_2 \in \mathbf{R}^{1\times 8}$ are the first and the second rows of A_1, respectively. The eigenvectors $\{V_i\}_{i=1}^{8}$ obtained from PCA are grouped into a matrix following a decreasing order of the corresponding eigenvalues,

$$V = \begin{pmatrix} V_1 & V_2 & \dots & V_8 \end{pmatrix} \tag{38}$$

where $V_i \in \mathbf{R}^{8\times 1}$ for $i = 1, 2, \dots, 8$. The projection coefficient c_{ij} of weight W_i on eigenvector V_j is then merely given by the dot product $c_{ij} = W_i \cdot V_j$ and its square is the corresponding projection weight $w_{i,j} \equiv c_{i,j}^2 \in [0, 1]$. In Table 2 we present the projection weights, where it is immediately clear that the majority of the weights are concentrated on the two principal components with the largest singular values. In Figure 4 we show the two latent factors z_1 and z_2 from the encoder.

	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6	PC_7	PC_8
Latent Factor 1	12.32 %	77.13 %	2.65 %	0.56 %	5.39 %	1.73 %	0.20 %	0.01 %
Latent Factor 2	41.75 %	44.47 %	2.20 %	1.33 %	3.92 %	5.59 %	0.64 %	0.09 %

Table 2: The projection weights of the encoded two factors on all eight eigenvectors.

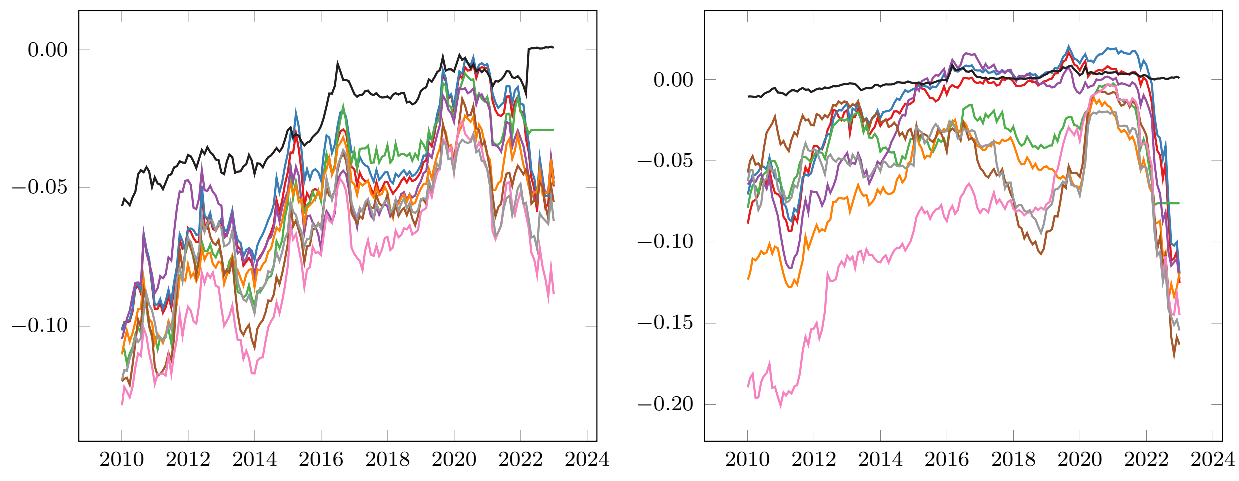
2.6.2 Risk-Neutral Parameters

In Figure 5 we present the extracted short rate r, volatilities σ_1 and σ_2, correlation ρ and drifts μ_1 and μ_2 from the autoencoder for all currencies. Interestingly, the behavior of the short rate r is in line with our intuitive expectations: The EUR and DKK short rates are almost identical as the latter is pegged to the former. Furthermore, from 2010 to 2020 most of the economies were in a low-rate environment, meaning that rates either went down or remained unchanged. The COVID-breakout in 2020 was detrimental to global economies as businesses were forced to shut down and consumer spendings were cut off almost instantly. To boost the economy and encourage spendings, central banks had to lower rates, yielding an extremely low-rate environment globally. Ever since 2022, the inflation pressure has forced most central banks to hike rates with JPY being the exception due to a fight against deflation. The volatility of JPY is also much more stable over the time period relative to other currencies, indicating the uniqueness of the monetary policy of Japan. Another interesting thing to notice is that the volatilities of the two latent factors had a significant drop when central banks started hiking rates, and their correlation is approaching one across multiple currencies.

3 Conclusion

Yield curve modeling has historically been dominated by parametric representations from the Nelson-Siegel family (Nelson and Siegel, 1987) or classical linear principal component analysis. The extension to nonlinear representations with autoencoders was pioneered by Kondratyev (2018), signifying the close relationship between autoencoders and principal component analysis. In this work we explicitly corroborate this relationship, which stems from their mutual dimension reduction ability.

Andreasen (2023) showed that the no-arbitrage condition could be checked at the decoding stage of the autoencoder, and possibly imposed as a soft constraint given a



(a) First latent factor z_1 **(b)** Second latent factor z_2

Figure 4: Encoded latent factors for the currencies DKK (—), EUR (—), GBP (—), SEK (—), NOK (—), USD (—), AUD (—), CAD (—) and JPY (—).

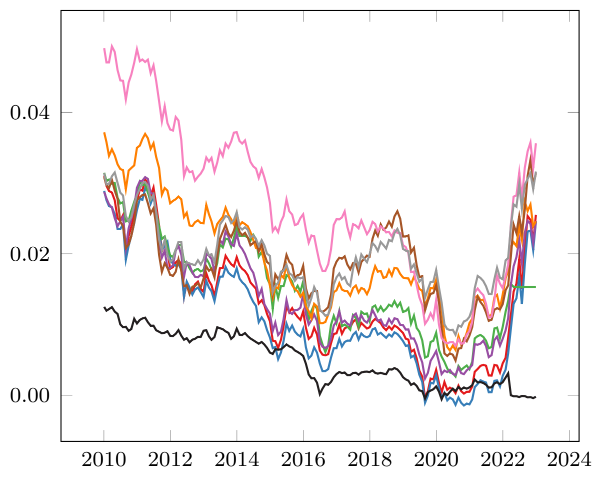
finite set of maturities. As the subsequent yield curve constructor is a spline interpolator, the generated yield curves are only (almost) arbitrage-free on this finite set of points. In contrast to Andreasen (2023) though, we impose it as a hard constraint: Inspired by the exponential affine yield curve model, we show that it is possible to design a fairly simple autoencoder that is inherently arbitrage-free and readily solved numerically, while displaying an accuracy comparable to that of Andreasen (2023).

Finally, we would be amiss if we did not try to explain how our work is related to that of Lyashenko et al. (2025). They work in a double-continuous framework (calendar time and time to maturity) and study settings where we can write the Musiela-parameterized instantaneous forward rate curves as $f(t,\tau) = \hat{f}_Z(Z(t),\tau)$ where Z is some latent low-dimensional (say d) factor process and the function $\hat{f}_Z$ is what they call the autoencoder. They then search for decoders with functional forms that ensure the absence of arbitrage or, in other words, impose the no-arbitrage condition at the decoding stage. They first look at decoders with the so-called time-shift invariance property.[3] Lyashenko et al. (2025) give a complete theoretical characterization of arbitrage-free decoders. They then develop/suggest different parametric forms that allow for efficient model training. For practical applications this is very important. However, as the authors point out, we

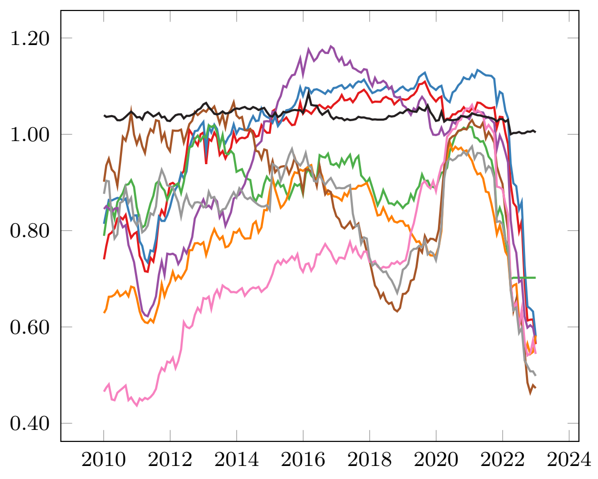
[3]Time-shift invariance is a theoretically convenient, but (at least to us) somewhat subtle concept. To illustrate: Z being a time-homogeneous Markov process does not imply time-shift invariance. Z solving a linear SDE implies (non-obviously) time-shift invariance, but not the other way round – even though time-shift invariance does imply some restrictions of the dynamics of Z.

should not be blinded by mathematical beauty and computational ease. Real-world yield curves may not be well-described by arbitrage-free time-shift invariant autoencoders.[4] Lyashenko et al. (2025) suggest treating this by starting with a base-case arbitrage-free time-shift invariance autoencoder and then add a convexity adjustment (that can be calculated explicitly given the base model) to it. This is different from our approach: Decouple encoding and decoding and solve a bunch of ODEs at decoder level to prevent arbitrage. The objectives are the same, the methods are different. There are not yet data available for a nuanced quantitative analysis of the vices and virtues of the two different methods.

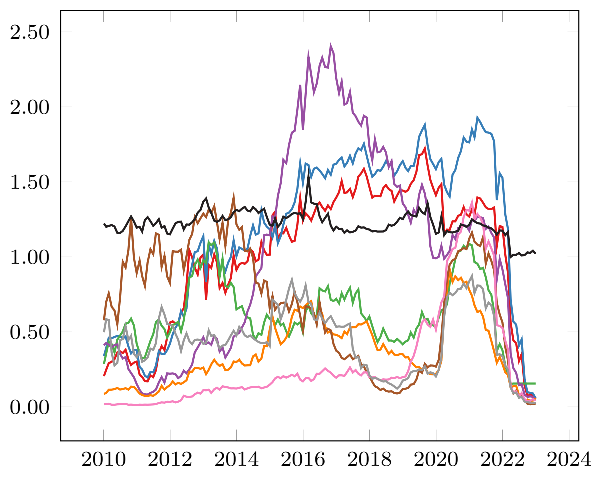
[4] Our two-factor model from Section 2 gives an indication of this: In the trained model, the dynamics of the latent Z process is of a form that is inconsistent with time-shift invariance.



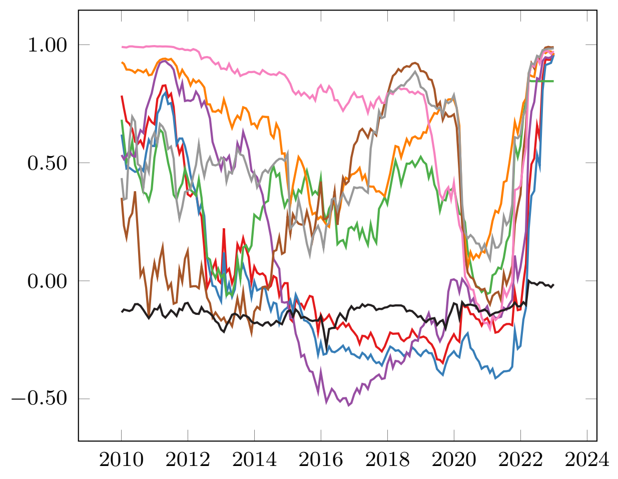
(a) r



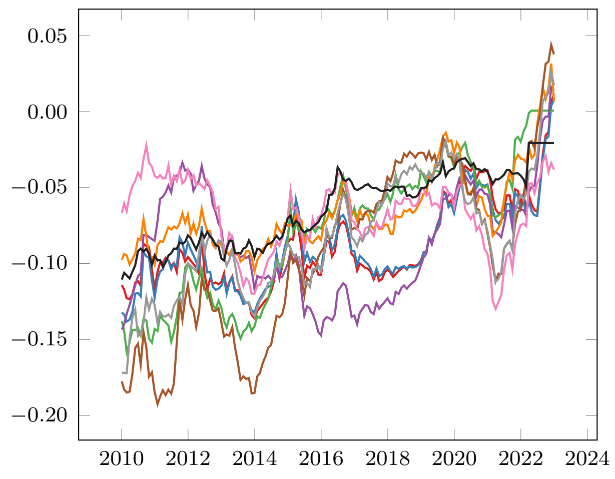
(b) σ_1



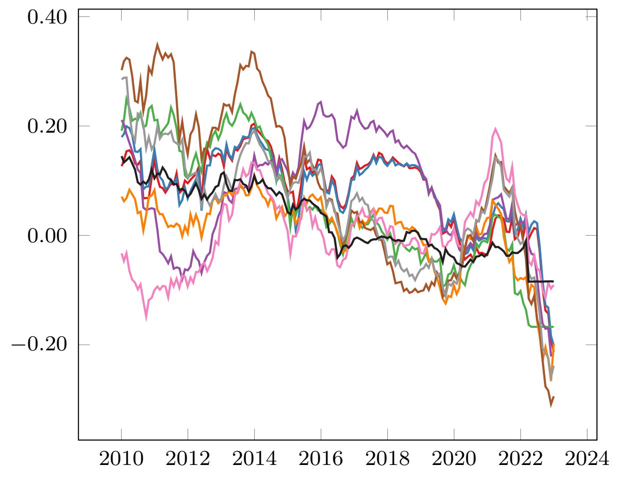
(c) σ_2



(d) ρ



(e) μ_1



(f) μ_2

Figure 5: Risk-neutral parameters of par swap rates for the currencies DKK (—), EUR (—), GBP (—), SEK (—), NOK (—), USD (—), AUD (—), CAD (—) and JPY (—).

References

Andreasen, J. (2023): "Decoding the Autoencoder," *Wilmott*, 2023.

Björk, T. (2009): *Arbitrage Theory in Continuous Time*, Oxford University Press, third ed.

Chen, R. T. Q., Y. Rubanova, J. Bettencourt, and D. K. Duvenaud (2018): "Neural Ordinary Differential Equations," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 31.

Duffie, D. and R. Kan (1996): "A Yield-Factor Model of Interest Rates," *Mathematical Finance*, 6, 379–406.

Henry-Labordere, P. (2008): *Analysis, Geometry, and Modeling in Finance: Advanced Methods in Option Pricing*, Chapman & Hall/CRC, first ed.

Kingma, D. and J. Ba (2015): "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Kondratyev, A. (2018): "Curve Dynamics with Artificial Neural Networks," *Risk*.

Lyashenko, A., F. Mercurio, and A. Sokol (2025): "Auto-Encoding Term-Structure Models," *Risk*.

Minaee, S., Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos (2022): "Image Segmentation Using Deep Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44, 3523–3542.

Nelson, C. R. and A. F. Siegel (1987): "Parsimonious Modeling of Yield Curves," *The Journal of Business*, 60, 473–489.

Sokol, A. (2022): "Autoencoder Market Models for Interest Rates," Working paper on SSRN.