Interest Rate Curve Bootstrapping & Risk Testing

Pape Kane

2026-02-17

This project constructs and analyses the U.S. Treasury zero-coupon yield curve using bootstrapping techniques applied to market data obtained from FRED. We implement spline interpolation and Nelson–Siegel parametric fitting, derive forward and swap rates, and assess bond portfolio sensitivity through duration, convexity, and key rate measures. Stress testing and Monte Carlo VaR simulations are conducted to evaluate interest rate risk under various shock scenarios. Principal Component Analysis (PCA) is applied to historical yield curve movements to identify level, slope, and curvature factors. Finally, we extend the framework with a machine learning forecasting module. The project provides a fully reproducible quantitative fixed-income framework linking term structure modelling to practical risk management.

Table of contents

1 Interest Rate Curve Bootstrapping & Risk Testing

1.1 Introduction

> Why does the yield curve matter?
>
> When you lend money for 1 year vs. 10 years, you expect a different return. The **yield curve** captures exactly that: how interest rates vary across maturities. Central banks, banks, insurers, and asset managers watch it daily, because it drives the price of virtually every financial instrument, from mortgages to derivatives.

Interest rate modelling sits at the very heart of modern finance. Every bond, every swap, every mortgage-backed security, their values all depend fundamentally on one thing: the term structure of interest rates. This project tackles that problem head-on by building a complete framework for constructing, analysing, and stress-testing the U.S. Treasury yield curve.

The motivation is straightforward. If you're managing a pension fund with billions in fixed-income assets, you need to know what happens when the Federal Reserve raises rates by 75 basis points. If you're an insurer with long-dated liabilities stretching 30 years into the future, you need to discount those cash flows at the right rate today, and understand how sensitive your balance sheet is to curve reshapings. If you're a derivatives desk pricing interest rate swaps, you need forward rates that are arbitrage-free and dynamically consistent with the spot curve.

We start from the raw data published daily by the Federal Reserve Economic Data (FRED) platform: Treasury yields at four key maturities (1Y, 2Y, 5Y, 10Y). From these four observable points, we reconstruct the full zero-coupon yield curve using bootstrapping, a recursive procedure that ensures no riskless arbitrage opportunities exist. We then fit smooth interpolation functions, both non-parametric (cubic splines) and parametric (Nelson–Siegel), allowing us to extract rates at any intermediate maturity needed for pricing.

Once the curve is built, the real work begins: risk analysis. We compute duration and convexity to quantify first- and second-order sensitivities to parallel shifts. We stress-test the curve under extreme scenarios, parallel shocks, steepeners, flatteners, and measure portfolio value-at-risk using Monte Carlo simulation with 2 000 simulated interest rate paths. We decompose historical yield curve movements using Principal Component Analysis (PCA), isolating the three fundamental drivers of curve dynamics: level shifts, slope changes, and curvature adjustments.

Finally, we push the framework one step further. Using the time series of daily yield changes we've already prepared for PCA, we build a machine learning forecasting module based on a Random Forest regressor. The model learns patterns in how short-term rates predict longer-term movements, giving us a simple but powerful tool for projecting the curve forward in time.

The objective is clear: build a reproducible, extensible quantitative framework that bridges textbook theory with the tools used daily by practitioners in asset management, banking, and actuarial science.

1.2 Literature Review

The term structure of interest rates has been extensively studied in financial economics, with foundational work establishing both the theoretical basis for arbitrage-free pricing and empirical regularities in yield curve dynamics.

Bootstrapping and arbitrage-free pricing. The construction of zero-coupon curves from coupon-bearing instruments was formalized by McCulloch (1971) and Nelson and Siegel (1987), who introduced the parametric curve that bears their names. The bootstrapping procedure ensures consistency with the no-arbitrage condition: discount factors must be such that no combination of bonds can generate riskless profit (Jarrow 2009).

Yield curve factor structure. Empirical research consistently finds that yield curve movements are driven by a small number of common factors. Litterman and Scheinkman (1991) applied Principal Component Analysis to U.S. Treasury yields and identified three dominant factors, level, slope, and curvature, which collectively explain over 95% of variation. This finding has been replicated across markets and time periods (Ang and Piazzesi 2003; Diebold and Li 2006).

Duration, convexity, and risk measures. Macaulay (1938) introduced duration as a measure of interest rate sensitivity, later refined by Hicks (1939) into modified duration. Convexity, the second-order correction for non-linearity in the price-yield relationship, was developed to improve hedging strategies in volatile markets (Fabozzi 2007).

Machine learning in yield curve forecasting. While traditional econometric models have dominated term structure forecasting (Diebold and Li 2006), recent work has explored machine learning approaches. Bianchi, Büchner, and Tamoni (2021) demonstrate that Random Forests can capture non-linear dependencies in rate dynamics, though they emphasize the challenge of non-stationarity and structural breaks.

This literature provides the foundation for our analysis. Our contribution lies in integrating these components into a single, reproducible framework with modern computational tools (Quarto, Python, Plotly) suitable for both academic instruction and professional application.

1.3 Research Questions

This project investigates how to construct, analyze, and manage interest rate risk using U.S. Treasury data. The analysis is guided by the following questions:

- How can we construct an arbitrage-free zero-coupon yield curve from observed market rates using bootstrapping?
- What are the differences between non-parametric (cubic spline) and parametric (Nelson–Siegel) interpolation methods?
- How do duration and convexity quantify bond portfolio sensitivity to interest rate shocks?
- How much of yield curve variation is explained by level, slope, and curvature factors?
- Can machine learning models successfully forecast short-term interest rate movements, or do structural breaks limit their usefulness?

- What are the practical implications for portfolio management and stress testing?

1.4 Resources & Implementation

All computational routines are centralized in the module `rates.py`, which provides:

- **Data loading and cleaning** to standardize FRED CSV files
- **Bootstrapping and interpolation** for zero-coupon curve construction
- **Risk measure computation** (duration, convexity, forward rates, swap rates)
- **Stress testing and simulation** (parallel shifts, scenario grids, Monte Carlo VaR)
- **PCA decomposition** to extract level/slope/curvature factors
- **Interactive visualization** using Plotly

By consolidating these components in a single module, the project ensures reproducibility and a clean separation between computation and presentation.

AI assistance disclosure. AI tools (Claude, GPT-4) were used to debug Python errors, improve code readability, and clarify technical writing. All modeling choices and statistical interpretations were made manually by the author.

---

2 Data Acquisition

2.1 U.S. Treasury Yields (FRED)

> What is FRED?
>
> FRED (Federal Reserve Economic Data) is the Federal Reserve's open data platform. It provides daily U.S. Treasury yields for all standard maturities, freely accessible and widely used as the reference source in quantitative finance.

We work with U.S. Treasury yields downloaded from FRED, covering four key maturities: **1-year, 2-year, 5-year, and 10-year**. These aren't randomly chosen, they span the full term structure in a way that captures short-term monetary policy expectations (1Y), medium-term growth and inflation views (2Y, 5Y), and long-run equilibrium rates (10Y). Together, they give us enough anchor points to reconstruct the entire curve through interpolation.

The data runs from February 2021 through February 2026, giving us roughly 1 250 trading days, about five years of history. That's long enough to capture the full post-pandemic monetary policy cycle: the near-zero rate environment of 2021, the historic Fed hiking campaign of 2022–2023 (the fastest tightening cycle in four decades), and the stabilization phase we're living through now in early 2026. This period is particularly valuable for risk modelling because it includes both extreme volatility and quieter regimes.

Each CSV file from FRED arrives with two columns: `observation_date` and the ticker symbol (e.g., `DGS10` for the 10-year rate). We standardize these to `date` and `rate`, convert from percentage format (e.g., 3.45) to decimal (0.0345), and attach maturity labels. Any missing values, usually around federal holidays when the bond market is closed, are dropped. The result is a clean, tidy DataFrame ready for analysis.

```
import sys
from pathlib import Path

sys.path.append(str(Path().resolve().parent))

from src.rates import *

data = load_us_treasury_from_csv("../data")

latest_date = data["date"].max()
zero_rates = extract_zero_rates_at_date(data, latest_date)

print("Latest available date:", latest_date.date())
print("\nZero rates snapshot:")
for mat, rate in zero_rates.items():
    print(f"  {mat:>2}Y : {rate*100:.3f}%")
```

```
Latest available date: 2026-02-10

Zero rates snapshot:
   1Y : 3.400%
   2Y : 3.450%
   5Y : 3.700%
  10Y : 4.160%
```

What we see here is the current state of the market as of the most recent trading day in our dataset. Notice the upward slope: the 10-year rate sits 76 basis points above the 1-year rate. This is what bond traders call a "positive term premium", investors demand higher compensation for locking up their money longer, reflecting both inflation risk and uncertainty about future Fed policy.

It's worth pausing to appreciate how much information is embedded in these four numbers. The 1-year rate effectively tells us where the market thinks the Federal Funds rate will average over the next twelve months. The 10-year rate incorporates expectations about growth, inflation, and risk premiums stretching out to 2036. The slope between them, steepness or flatness, has historically been one of the best recession predictors: when the curve inverts (short rates above long rates), recessions tend to follow within 12–18 months. We're not inverted now, which is consistent with the consensus view that the U.S. economy is stabilizing rather than contracting.

> **Observation:** The yield curve is currently **upward sloping**, reflecting a typical term premium, investors demand higher rates for longer maturities. This is the "normal" shape in non-recessionary environments.

2.2 Data Limitations

While FRED provides high-quality, reliable data, several limitations should be acknowledged:

Limited maturity coverage. We observe only four discrete maturities. Pricing bonds at intermediate tenors requires interpolation, which introduces model risk. The choice between spline and Nelson–Siegel affects derived forward rates and valuations.

Constant maturity construction. FRED rates are synthetic "constant maturity" yields derived from actively traded securities with varying actual maturities, introducing small approximation errors compared to zero-coupon strip prices.

Missing data on non-trading days. The dataset excludes weekends and federal holidays. PCA daily returns are computed on business days only.

No credit risk or liquidity premiums. FRED yields reflect only the "risk-free" term structure. Actual portfolios contain corporate bonds and other securities with credit spreads not captured here.

Structural breaks and non-stationarity. The 2021–2026 period includes unprecedented monetary policy shifts. Models trained on this data may fail to generalize if future Fed policy differs structurally, a fundamental challenge in financial forecasting.

---

3 Theoretical Framework

3.1 Zero-Coupon Bond Pricing

> What is a discount factor?
>
> A discount factor $DF(T)$ answers a simple question: *how much is 1 dollar receivable in T years worth today?* If rates are 4%, a dollar in 10 years is worth roughly \$0.676 today. The lower the discount factor, the higher the implied interest rate.

The price of a coupon bond is the sum of all future cash flows, each discounted to today:

$$P = \sum_{t=1}^{T-1} C \cdot DF(t) + (C + F) \cdot DF(T)$$

where C is the annual coupon, F is the face value, and $DF(t)$ is the discount factor at maturity t.

The discount factor is linked to the **zero rate** r_T through:

$$DF(T) = \frac{1}{(1 + r_T)^T} \qquad \Longrightarrow \qquad r_T = DF(T)^{-1/T} - 1$$

---

4 Bootstrapping the Zero Curve

What is bootstrapping?

Bootstrapping is an iterative procedure: starting from the shortest maturity, we extract the discount factor that makes the bond price consistent with the market. We then use that result to move to the next maturity. The output is a **set of zero rates**, pure, coupon-free rates for each horizon, constructed to be arbitrage-free.

4.1 Mathematical Formulation

Consider a coupon-bearing bond with maturity T, annual coupon $C = c \cdot F$ (where c is the coupon rate and F is face value), and market price P. The no-arbitrage condition requires:

$$P = \sum_{t=1}^{T-1} C \cdot DF(t) + (C + F) \cdot DF(T)$$

If we have already bootstrapped discount factors $DF(1), DF(2), \ldots, DF(T-1)$ from shorter maturities, we can solve for $DF(T)$ by rearranging:

$$DF(T) = \frac{P - \sum_{t=1}^{T-1} C \cdot DF(t)}{C + F}$$

This is the **bootstrapping recursion**. Once we have $DF(T)$, we recover the zero rate via:

$$r(T) = DF(T)^{-1/T} - 1 = \left(\frac{1}{DF(T)}\right)^{1/T} - 1$$

Arbitrage-free property: If market prices are internally consistent, this procedure guarantees that no combination of bonds can generate riskless profit. Any deviation would create an arbitrage opportunity that would be instantly exploited away.

```
latest_date = data["date"].max()
zero_rates = extract_zero_rates_at_date(data, latest_date)

import pandas as pd

dfs = zero_to_discount_factors(zero_rates)

summary = pd.DataFrame({
    "Maturity (Y)": list(zero_rates.keys()),
```

```
    "Zero Rate (%)": [f"{r*100:.3f}" for r in zero_rates.values()],
    "Discount Factor": [f"{dfs[T]:.4f}" for T in zero_rates.keys()],
})
summary
```

	Maturity (Y)	Zero Rate (%)	Discount Factor
0	1	3.400	0.9671
1	2	3.450	0.9344
2	5	3.700	0.8339
3	10	4.160	0.6653

This procedure ensures **arbitrage-free construction** of the zero curve: no riskless profit is possible from combinations of these instruments.

---

5 Curve Interpolation

> Why interpolate?
>
> We only observe rates at 4 maturities (1Y, 2Y, 5Y, 10Y). But pricing a bond maturing in 3 or 7 years requires rates at those tenors too. Interpolation fills the gaps, the question is *how* to do it smoothly and economically.

5.1 Cubic Spline

A **natural cubic spline** fits a smooth, piecewise polynomial curve that passes exactly through each observed point. It minimises curvature (second-derivative) globally, producing the smoothest possible interpolation.

```
spline_curve = interpolate_zero_curve_spline(zero_rates)

# Demonstrate interpolation at intermediate maturities
print("Cubic Spline Interpolation:")
print(f"  Observed 2Y rate : {zero_rates[2]*100:.3f}%")
print(f"  Interpolated 3Y  : {spline_curve(3)*100:.3f}%")
print(f"  Observed 5Y rate : {zero_rates[5]*100:.3f}%")
print(f"  Interpolated 7Y  : {spline_curve(7)*100:.3f}%")
print(f"  Observed 10Y rate: {zero_rates[10]*100:.3f}%")
```

```
Cubic Spline Interpolation:
  Observed 2Y rate : 3.450%
  Interpolated 3Y  : 3.520%
  Observed 5Y rate : 3.700%
  Interpolated 7Y  : 3.886%
  Observed 10Y rate: 4.160%
```

5.2 Nelson–Siegel Parametric Curve

The **Nelson–Siegel** model imposes economic structure on the curve. Its four parameters each have a direct interpretation:

$$r(T) = \underbrace{\beta_0}_{\text{level}} + \underbrace{\beta_1 \frac{1 - e^{-T/\tau}}{T/\tau}}_{\text{slope}} + \underbrace{\beta_2 \left(\frac{1 - e^{-T/\tau}}{T/\tau} - e^{-T/\tau} \right)}_{\text{curvature}}$$

Parameter	Interpretation
β_0	Long-run level (rate at infinite maturity)
β_1	Slope: spread between short and long rates
β_2	Curvature: hump in the middle of the curve
τ	Speed at which short-term effects decay

```
params = fit_nelson_siegel(zero_rates)

print("Nelson-Siegel fitted parameters:")
print(f"    (level)     = {params[0]*100:.3f}%")
print(f"    (slope)     = {params[1]*100:.3f}%")
print(f"    (curvature) = {params[2]*100:.3f}%")
print(f"     (decay)     = {params[3]:.3f} years")

ns_curve = lambda T: nelson_siegel(T, *params)

fig = plot_yield_curve_interactive(zero_rates, spline_curve, ns_curve)
```

```
Nelson-Siegel fitted parameters:
    (level)     = 5.655%
    (slope)     = -2.271%
    (curvature) = -2.313%
     (decay)     = 4.008 years
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

Reading the chart: Both the spline and Nelson–Siegel fit the 4 observed points closely. The Nelson–Siegel curve is slightly smoother, while the spline captures local curvature more faithfully. The upward slope reflects current market expectations.

---

6 Forward & Swap Rates

What is a forward rate?

A forward rate $f(T_1, T_2)$ is the rate the market is *implicitly quoting today* for borrowing or lending between future dates T_1 and T_2. It is not a forecast, it is a no-arbitrage implication of the current yield curve.
A **swap rate** is the fixed rate that makes an interest rate swap fair at inception: it equals the weighted average of expected floating rates over the swap's life.

6.1 Forward Rate Derivation

A forward rate $f(T_1, T_2)$ is the rate agreed upon today for borrowing/lending between future dates T_1 and T_2. It is **not a forecast** — it is an arbitrage-free implication of the current zero curve.

No-arbitrage condition: Consider two investment strategies over horizon $[0, T_2]$:

1. **Strategy A:** Invest \$1 today at the zero rate $r(T_2)$ for T_2 years.
2. **Strategy B:** Invest \$1 at $r(T_1)$ for T_1 years, then roll over at the forward rate $f(T_1, T_2)$ for $(T_2 - T_1)$ years.

Both strategies must yield the same terminal value to prevent arbitrage:

$$(1 + r(T_2))^{T_2} = (1 + r(T_1))^{T_1} \cdot (1 + f(T_1, T_2))^{T_2 - T_1}$$

Solving for $f(T_1, T_2)$:

$$f(T_1, T_2) = \left(\frac{(1 + r(T_2))^{T_2}}{(1 + r(T_1))^{T_1}} \right)^{\frac{1}{T_2 - T_1}} - 1$$

Using the relationship $DF(T) = (1 + r(T))^{-T}$, this simplifies to:

$$f(T_1, T_2) = \left(\frac{DF(T_1)}{DF(T_2)} \right)^{\frac{1}{T_2 - T_1}} - 1$$

Economic interpretation: If the curve is upward sloping ($r(T_2) > r(T_1)$), forward rates lie **above** spot rates. This reflects the market's pricing of term premium and expectations.

```python
forwards = full_forward_curve(zero_rates)

print("Implied forward rates:")
for (T1, T2), f in forwards.items():
    print(f"  f({T1}Y → {T2}Y) = {f*100:.3f}%")

s10 = swap_rate(zero_rates, 10, curve_function=spline_curve)
print(f"\n10-year par swap rate: {s10*100:.3f}%")

fig_fwd = plot_forward_rates_interactive(zero_rates)
```

```
Implied forward rates:
  f(1Y → 2Y) = 3.500%
  f(2Y → 5Y) = 3.867%
  f(5Y → 10Y) = 4.622%

10-year par swap rate: 4.099%
```

```
Unable to display output for mime type(s): text/html
```

> **Key insight:** Forward rates are *above* spot zero rates, consistent with an upward-sloping curve. The market prices in rising rates over the medium term.

---

7 Bond Pricing & Risk Measures

Duration and Convexity, the two essential risk metrics

Duration measures how much a bond's price falls when rates rise by 1 basis point. A duration of 4.5 means the bond loses roughly 4.5% of its value if rates jump by 100 bps. **Convexity** adds a second-order correction: because the price–yield relationship is curved (not linear), duration alone underestimates price gains and overestimates losses. Higher convexity is generally desirable.

7.1 Bond Pricing from Zero Curve

```
price = bond_price_from_curve(5, 0.03, 100, spline_curve)
print(f"5Y bond (3% coupon, face 100) → Price = {price:.4f}")
print(f"Bond is trading {'at a discount' if price < 100 else 'at a premium'} "
      f"(market rates {'above' if price < 100 else 'below'} coupon rate)")
```

```
5Y bond (3% coupon, face 100) → Price = 96.9027
Bond is trading at a discount (market rates above coupon rate)
```

7.2 Modified Duration

Duration measures **first-order sensitivity** to yield changes. Consider a bond priced from the zero curve. The price is:

$$P = \sum_{t=1}^{T} CF_t \cdot (1 + r(t))^{-t}$$

where CF_t is the cash flow at time t (coupons plus principal at maturity). Modified duration is the derivative with respect to a **parallel shift** in all rates:

$$D_{\text{mod}} = -\frac{1}{P}\frac{\partial P}{\partial y}$$

For small rate changes Δy, the price change is approximately:

$$\Delta P \approx -D_{\text{mod}} \cdot P \cdot \Delta y$$

Numerical computation: We use finite differences. Compute the price under a small upward shock (+1 bp), then:

$$D_{\text{mod}} \approx -\frac{P(y+\epsilon) - P(y)}{P(y) \cdot \epsilon}, \qquad \epsilon = 0.0001$$

7.3 Convexity

Convexity captures the **second-order (curvature) effect**. The price–yield relationship is not linear — it's convex. Duration alone underestimates gains when rates fall and overestimates losses when rates rise. Convexity is the second derivative:

$$\mathcal{C} = \frac{1}{P}\frac{\partial^2 P}{\partial y^2}$$

Numerical computation: Using finite differences with three price evaluations:

$$\mathcal{C} \approx \frac{P(y+\epsilon) - 2P(y) + P(y-\epsilon)}{P(y) \cdot \epsilon^2}$$

For a rate shock Δy, the **Taylor expansion** gives the full price approximation:

$$\frac{\Delta P}{P} \approx -D_{\text{mod}} \cdot \Delta y + \frac{1}{2}\mathcal{C} \cdot (\Delta y)^2$$

The first term is the duration effect (linear). The second term is the convexity adjustment (quadratic). For large rate moves, ignoring convexity can lead to significant pricing errors.

```
dur = bond_duration(5, 0.03, 100, spline_curve)
cvx = bond_convexity(5, 0.03, 100, spline_curve)

print(f"Modified Duration : {dur:.4f} years")
print(f"Convexity         : {cvx:.4f}")

shock = 0.01
price_approx_change = (-dur * shock + 0.5 * cvx * shock**2) * 100
print(f"\nEstimated price change for +100 bps shock: {price_approx_change:.4f}%")
```

```
Modified Duration : 4.5419 years
Convexity         : 25.7583

Estimated price change for +100 bps shock: -4.4131%
```

---

8 Stress Testing

What is stress testing in fixed income?

Stress testing asks: *what happens to my portfolio if rates move in an extreme but plausible way?* Regulators (Basel, Solvency II) require institutions to demonstrate resilience under predefined shocks. The standard scenarios are parallel shifts (rates move uniformly) and curve reshapings (steepeners, flatteners).

8.1 Parallel Shift

A parallel shift moves all rates by the same amount across all maturities.

```
fig_stress = plot_stress_test_interactive(
    maturity=5,
    coupon_rate=0.03,
    face_value=100,
    curve=spline_curve,
    shock=0.01
)
```

```
Unable to display output for mime type(s): text/html
```

8.2 Scenario Grid

We test three scenarios on a mixed portfolio (2Y/5Y/10Y bonds):

```
fig_grid = plot_scenario_grid_interactive(portfolio, 100, spline_curve)
```

```
Unable to display output for mime type(s): text/html
```

Interpreting the results: A parallel +100 bps shock is the most severe scenario. The steepener is less damaging because short-end rates are unchanged, the portfolio's short-duration bonds act as a partial hedge.

---

9 Monte Carlo Value-at-Risk

> What is Value-at-Risk (VaR)?
>
> VaR answers: *what is the maximum loss I should expect to suffer in 95% of scenarios?* Here we run 2 000 random interest rate shocks drawn from a normal distribution, reprice the full portfolio under each shock, and read the 95th percentile of the loss distribution.

Interest rate shocks are simulated as:

$$\Delta r \sim \mathcal{N}(0,\, \sigma^2), \qquad \sigma = 100 \text{ bps}$$

Portfolio losses are computed as:

$$\text{Loss}_i = V_0 - V_{\text{shocked},i}$$

Assumption and limitations: The normal distribution assumption is a simplification widely used for tractability. However, empirical evidence shows that interest rate changes exhibit **fat tails** (extreme events occur more frequently than normal theory predicts) and **volatility clustering** (large moves tend to follow large moves). During the 2022–2023 Fed hiking cycle, weekly rate changes exceeding ±50 bps occurred multiple times, events the normal model would classify as 5-sigma outliers. More realistic alternatives include Student's t-distribution (captures fat tails), GARCH models (time-varying volatility), or empirical bootstrapping from historical shocks. We retain the normal assumption for pedagogical clarity, but acknowledge it underestimates tail risk.

```
np.random.seed(42)
var_95, losses = simulate_var(portfolio, 100, spline_curve, shock_std=0.01, n_sim=2000)

print(f"95% VaR (2 000 simulations,  = 100 bps): {var_95:.4f}")
print(f"Expected loss interpretation: in 95% of scenarios,")
print(f"portfolio loss does not exceed {var_95:.4f} per unit of face value.")

fig_var = plot_var_distribution_interactive(losses, var_95)
```

```
95% VaR (2 000 simulations,  = 100 bps): 7.0880
Expected loss interpretation: in 95% of scenarios,
portfolio loss does not exceed 7.0880 per unit of face value.
```

```
Unable to display output for mime type(s): text/html
```

> **Reading the histogram:** The loss distribution is approximately symmetric and bell-shaped, consistent with normally distributed rate shocks. The red dashed line marks the 95% VaR threshold.

10 Principal Component Analysis

Why PCA on yield curves?

Yield curve movements across maturities are highly correlated, when 2Y rates move, 5Y rates usually move too. PCA decomposes this complexity into a small number of **independent factors** that explain most of the variance. Decades of empirical research show that three factors capture 95–99% of yield curve variation: **level, slope, and curvature**.

10.1 Mathematical Framework

We observe daily yield changes across the four maturities over 5 years (~1 250 trading days). Let Δr_t denote the vector of rate changes on day t. We compute the **covariance matrix** of these changes, which captures how different maturities move together.

PCA decomposes this covariance matrix into orthogonal factors:

$$= \mathbf{V}\ \mathbf{V}^\top$$

where $\mathbf{V}$ contains the **eigenvectors** (principal components) and contains the **eigenvalues** (variance explained by each component).

What this means in practice:

- Each eigenvector is a pattern of co-movement. The first eigenvector (PC1) has all positive entries, meaning all rates move in the same direction — this is the **level factor**.
- The eigenvalues tell us how important each factor is. Typically, PC1 explains ~85–90% of all variation, PC2 explains ~5–10%, and PC3 explains ~2–5%.
- The three factors have clear economic interpretations: **level** (parallel shifts), **slope** (steepening/flattening), and **curvature** (humps in the middle of the curve).

This dimensional reduction is powerful: instead of tracking movements at hundreds of maturities, we can summarize almost everything with three numbers.

```
maturities_list = sorted([int(m) for m in data['maturity'].unique()])
print(f"Dataset loaded from local FRED CSV files")
print(f"Date range  : {data['date'].min().date()} → {data['date'].max().date()}")
print(f"Maturities  : {maturities_list}Y")
print(f"Observations: {data['date'].nunique()} trading days × {len(maturities_list)} maturit
```

```
Dataset loaded from local FRED CSV files
Date range  : 2021-02-10 → 2026-02-10
Maturities  : [1, 2, 5, 10]Y
Observations: 1250 trading days × 4 maturities
```

```
pca, maturities, returns = perform_pca(data)

total_var = pca.explained_variance_ratio_[:3].sum() * 100
print("PCA, Explained Variance:")
for i in range(min(4, len(pca.explained_variance_ratio_))):
    print(f"  PC{i+1}: {pca.explained_variance_ratio_[i]*100:.1f}%")
print(f"\nFirst 3 components explain: {total_var:.1f}% of total variance")

fig_pca1 = plot_pca_components_interactive(pca, maturities)
```

```
PCA, Explained Variance:
  PC1: 87.8%
  PC2: 9.3%
  PC3: 2.2%
  PC4: 0.7%

First 3 components explain: 99.3% of total variance
```

```
Unable to display output for mime type(s): text/html
```

```
fig_pca2 = plot_pca_variance_interactive(pca)
```

```
Unable to display output for mime type(s): text/html
```

Interpreting the factors:

- **PC1 (Level)**, all loadings have the same sign. When this factor moves, all rates rise or fall together. It is by far the dominant driver of yield curve risk, capturing the 2022–2023 Fed hiking cycle in full.
- **PC2 (Slope)**, loadings are positive at long maturities and negative at short ones. This captures steepening/flattening dynamics.
- **PC3 (Curvature)**, a hump-shaped pattern, reflecting movements in intermediate maturities relative to the short and long ends.

---

11 Extension: Machine Learning Yield Curve Forecasting

> Can we predict where rates are heading?
>
> Traditional yield curve models are *descriptive*: they tell us what the curve looks like today, and how sensitive it is to shocks. But a natural next question emerges: can we build a model that *predicts* where rates will be in 1 month, 3 months, or 6 months? Machine learning offers a data-driven approach to this problem.

We've already prepared everything we need for this extension. Our PCA analysis computed **daily yield changes** across all four maturities over 5 years, that's 1 249 observations of how the curve evolves day-to-day. We can now use this time series to train a supervised learning model.

The idea is simple but powerful. Today's short-term rates often contain information about where longer-term rates are heading. If the 1-year rate jumps sharply, the 10-year rate tends to follow, though with a lag and a dampening effect. Machine learning can learn these patterns automatically from historical data, without requiring us to specify the functional form by hand.

We use a **Random Forest regressor**, an ensemble method that fits multiple decision trees and averages their predictions. Random Forests are robust to overfitting, handle non-linear relationships naturally, and require minimal hyperparameter tuning. We train the model to predict the **1-month forward change** in the 10-year zero rate, using as features: (1) today's level of all four rates, (2) yesterday's changes in all four rates, and (3) the rolling 20-day volatility of rate changes.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, r2_score

# Build ML dataset from PCA returns
def build_ml_features(returns):
    """
    Create supervised learning dataset:
      - Features: today's rate levels + yesterday's changes + 20d vol
      - Target: 1-month forward change in 10Y rate
    """
    df = returns.copy()
    df.columns = [f"rate_{int(c)}Y" for c in df.columns]

    # Lagged changes
    for col in df.columns:
        df[f"{col}_lag1"] = df[col].shift(1)
```

```
    # Rolling volatility (20-day window)
    for col in [c for c in df.columns if c.startswith("rate_")]:
        df[f"{col}_vol20d"] = df[col].rolling(20).std()

    # Target: 1-month forward change in 10Y
    df["target"] = df["rate_10Y"].shift(-21)  # ~1 month = 21 trading days

    # Drop rows with NaN (edges of rolling windows and lags)
    df = df.dropna()

    return df

ml_data = build_ml_features(returns)

# Features: all columns except target
X = ml_data.drop(columns=["target"])
y = ml_data["target"]

# Train/test split: 80% train, 20% test (chronological)
split_idx = int(len(X) * 0.8)
X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]

print(f"Training set  : {len(X_train)} observations")
print(f"Test set      : {len(X_test)} observations")
print(f"Features used : {X.shape[1]}")
```

```
Training set  : 966 observations
Test set      : 242 observations
Features used : 16
```

```
# Train Random Forest
rf = RandomForestRegressor(
    n_estimators=100,
    max_depth=8,
    min_samples_split=10,
    random_state=42,
    n_jobs=-1
)

rf.fit(X_train, y_train)

# Predictions
y_pred_train = rf.predict(X_train)
y_pred_test = rf.predict(X_test)

# Evaluation metrics
mae_train = mean_absolute_error(y_train, y_pred_train)
mae_test = mean_absolute_error(y_test, y_pred_test)
r2_train = r2_score(y_train, y_pred_train)
```

```
r2_test = r2_score(y_test, y_pred_test)

print("\nModel Performance:")
print(f"  Train MAE : {mae_train*10000:.2f} bps")
print(f"  Test MAE  : {mae_test*10000:.2f} bps")
print(f"  Train R²  : {r2_train:.3f}")
print(f"  Test R²   : {r2_test:.3f}")
```

```
Model Performance:
  Train MAE : 4.20 bps
  Test MAE  : 3.86 bps
  Train R²  : 0.343
  Test R²   : -0.068
```

```
# Feature importance
import pandas as pd

importances = pd.DataFrame({
    "Feature": X.columns,
    "Importance": rf.feature_importances_
}).sort_values("Importance", ascending=False).head(10)

print("\nTop 10 Most Important Features:")
print(importances.to_string(index=False))
```

```
Top 10 Most Important Features:
             Feature  Importance
     rate_10Y_vol20d    0.092998
      rate_2Y_vol20d    0.075615
      rate_1Y_vol20d    0.073198
       rate_10Y_lag1    0.071506
 rate_5Y_lag1_vol20d    0.070160
rate_10Y_lag1_vol20d    0.069937
 rate_1Y_lag1_vol20d    0.068644
       rate_2Y_lag1    0.067967
 rate_2Y_lag1_vol20d    0.063039
       rate_5Y_lag1    0.062279
```

```
# Visualize: Actual vs Predicted on test set
import plotly.graph_objects as go

fig = go.Figure()

test_dates = ml_data.index[split_idx:]

fig.add_trace(go.Scatter(
    x=test_dates,
```

```
    y=y_test * 10000,
    mode="lines",
    name="Actual 1M Δ (10Y)",
    line=dict(color="#457B9D", width=1.5),
))

fig.add_trace(go.Scatter(
    x=test_dates,
    y=y_pred_test * 10000,
    mode="lines",
    name="Predicted 1M Δ (10Y)",
    line=dict(color="#E63946", width=1.5, dash="dot"),
))

fig.update_layout(
    title=dict(
        text="ML Forecast: 1-Month Forward Change in 10Y Rate",
        font=dict(size=16),
        y=0.95,
        x=0.5,
        xanchor='center',
        yanchor='top'
    ),
    xaxis_title="Date",
    yaxis_title="Rate Change (bps)",
    template="plotly_white",
    height=500,
    hovermode="x unified",
    margin=dict(t=80, b=80, l=60, r=40),
    legend=dict(orientation="h", yanchor="bottom", y=-0.2, xanchor="center", x=0.5),
)

fig.show()
```

```
Unable to display output for mime type(s): text/html
```

The results reveal an important lesson about machine learning in finance. On the test set, we achieve a Mean Absolute Error (MAE) of roughly 4 basis points, which sounds excellent. However, the R^2 score is **negative** (around -0.07), meaning the model performs **worse than a naive baseline** that simply predicts zero change every day.

What went wrong? The test period likely contains structural breaks or regime shifts that didn't exist in the training data. Interest rate movements from 2021–2024 (our training set) were dominated by the Fed's unprecedented hiking cycle. The test period (late 2024–2026) represents a different regime: rates stabilizing, inflation moderating, policy uncertainty. The model learned patterns that don't generalize.

This is a **classic overfitting problem** in financial forecasting. The Random Forest captured noise rather than signal. The feature importances show that volatility measures (rolling 20-day standard deviations) rank highest, but these are inherently unstable and regime-dependent.

What we learn from this failure:

- **Time series forecasting is hard.** Unlike image classification or NLP, financial markets are non-stationary. Patterns that worked yesterday can fail tomorrow when the macroeconomic regime shifts.
- **Out-of-sample testing is critical.** In-sample performance (Train $R^2 = 0.34$) looked respectable, but it was meaningless. Only the test set revealed the truth.
- **Simple models can outperform complex ones.** A naive "no-change" forecast beats our Random Forest here. Sometimes the best prediction is humility.

How to improve:

- Incorporate macro features: Fed Funds futures, CPI expectations, unemployment rates — variables that capture regime shifts.
- Use regime-switching models (Markov-switching, HMM) that explicitly model structural breaks.
- Shorten the forecast horizon: predicting 1-week changes instead of 1-month might work better.
- Ensemble with economic models: blend ML forecasts with term structure models (Vasicek, CIR).

This extension demonstrates both the **potential and the limits** of machine learning in finance. The infrastructure is there (PCA time series, clean data, feature engineering), but prediction requires more than just throwing algorithms at historical patterns. It requires understanding the economic drivers and respecting the non-stationarity of markets.

---

12 Conclusion

This project delivers a complete, reproducible framework for interest rate curve modelling and risk analysis. Starting from four observable Treasury yields published daily by FRED, we've built a zero-coupon curve, stress-tested it under extreme scenarios, decomposed historical movements into fundamental risk factors, and extended the toolkit with machine learning forecasting.

The results reveal several key insights. First, the current U.S. yield curve is upward sloping (3.40% at 1Y, 4.16% at 10Y), reflecting a 76 basis point term premium, the market's compensation for inflation risk and uncertainty about future Fed policy. Second, our PCA analysis shows that 99.3% of yield curve variation over five years is explained by just three factors: level shifts, slope changes, and curvature adjustments. This is why professional risk managers focus on "level-duration" and "slope-duration", it's not simplification, it's empirical reality.

Third, duration is the primary risk driver: a +100 bp parallel shock drops our portfolio value by roughly 4.5%, consistent with modified duration. But convexity matters too, the price-yield relationship is curved, not linear, meaning gains from rate falls exceed losses from rate rises by a margin that can be worth millions at institutional scale. Fourth, Monte Carlo VaR gives us the full loss distribution, not just a single number. We see the clustering around small moves, the fat left tail of severe downside scenarios, and the symmetric shape that validates our normal shock assumption.

Finally, the Random Forest forecasting module reveals an important lesson about machine learning in finance. Despite having proper infrastructure (PCA time series, clean data, feature engineering), the model failed to generalize to the test period (R^2 = -0.068), performing worse than a naive baseline. This failure demonstrates that historical patterns alone cannot capture regime shifts in non-stationary markets. The extension highlights both the potential of ML tools and the critical need to combine them with economic understanding and respect for structural breaks.

The framework is extensible. Natural next steps include adding more maturities (3M bills, 30Y bonds), incorporating credit spreads, building full term structure models (Vasicek, CIR, Hull-White), and feeding macro indicators into the ML module. But even as it stands, this is a production-ready toolkit. The code runs. The data is real. The methods are industry-standard.

That's the goal of quantitative finance: turning abstract concepts, discount factors, eigenvalues, convexity, into tools that make better decisions. This project bridges that gap.

Code and data available on GitHub. Built with Python, Quarto, NumPy, pandas, scikit-learn, and Plotly.

---

13 Bibliography

- Ang, A., & Piazzesi, M. (2003). A no-arbitrage vector autoregression of term structure dynamics with macroeconomic and latent variables. *Journal of Monetary Economics*, 50(4), 745–787.
- Bianchi, D., Büchner, M., & Tamoni, A. (2021). Bond risk premiums with machine learning. *Review of Financial Studies*, 34(2), 1046–1089.
- Diebold, F. X., & Li, C. (2006). Forecasting the term structure of government bond yields. *Journal of Econometrics*, 130(2), 337–364.
- Fabozzi, F. J. (2007). *Fixed Income Analysis* (2nd ed.). John Wiley & Sons.
- Hicks, J. R. (1939). *Value and Capital*. Oxford University Press.
- Jarrow, R. A. (2009). The term structure of interest rates. *Annual Review of Financial Economics*, 1(1), 69–96.
- Litterman, R., & Scheinkman, J. (1991). Common factors affecting bond returns. *Journal of Fixed Income*, 1(1), 54–61.
- Macaulay, F. R. (1938). *The Movements of Interest Rates, Bond Yields and Stock Prices in the United States Since 1856*. National Bureau of Economic Research.
- McCulloch, J. H. (1971). Measuring the term structure of interest rates. *Journal of Business*, 44(1), 19–31.
- Nelson, C. R., & Siegel, A. F. (1987). Parsimonious modeling of yield curves. *Journal of Business*, 60(4), 473–489.