Convexity Hedging with Two Swaps

Table of contents

Duration matching eliminates first-order interest-rate risk, but the hedge breaks down for larger rate moves because assets and liabilities have different *convexity*. This tutorial extends the duration-hedging example by matching **both** duration and convexity using two swap tenors and the `immunize()` solver.

```
import os
import sys
from pathlib import Path

# make the alm package importable when running from docs/tutorial/
sys.path.insert(0, str(Path.cwd().parents[1] / "src"))

import polars as pl
from great_tables import GT

from alm.asset import Bond
from alm.liability import SPIA, Term
from alm.core import InterestRateSwap, dv01, dollar_convexity, immunize

# Detect output format - use matplotlib for PDF, plotly for HTML
```

```
IS_PDF = os.environ.get("QUARTO_FMT", "html") == "pdf"

if IS_PDF:
    import matplotlib.pyplot as plt
    import matplotlib.ticker as mticker
else:
    import plotly.graph_objects as go
```

Step 1 — Build the portfolio

We reuse the same portfolio from the duration-hedging tutorial.

Assets

Bond	Face	Coupon	Maturity	Frequency
A	\$5M	3.5 %	5 yr	Semi-ann
B	\$3M	4.5 %	10 yr	Semi-ann

```
bond_a = Bond(face_value=5_000_000, coupon_rate=0.035, maturity=5, frequency=2)
bond_b = Bond(face_value=3_000_000, coupon_rate=0.045, maturity=10, frequency=2)
```

Liabilities

```
qx = [0.01] * 30

spia = SPIA(
    premium=4_000_000,
    annual_payout=300_000,
    qx=qx,
    frequency=12,
    certain_period=0,
)

term = Term(
    face_value=1_000_000,
    annual_premium=5_000,
    term=20,
```

```
    qx=qx,
    frequency=12,
)
```

Step 2 — Measure the duration *and* convexity gaps

```
rate = 0.04

# --- Asset side ---
pv_a = bond_a.present_value(rate)
pv_b = bond_b.present_value(rate)
dv01_a = dv01(bond_a.present_value, rate) + dv01(bond_b.present_value, rate)
dc_a = dollar_convexity(bond_a.present_value, rate) + dollar_convexity(bond_b.present_value,

# --- Liability side ---
pv_spia = spia.present_value(rate)
pv_term = term.present_value(rate)
dv01_l = dv01(spia.present_value, rate) + dv01(term.present_value, rate)
dc_l = dollar_convexity(spia.present_value, rate) + dollar_convexity(term.present_value, rat

dv01_gap = dv01_l - dv01_a
dc_gap = dc_l - dc_a

gap_df = pl.DataFrame({
    "Side": ["Assets", "Assets", "Liabilities", "Liabilities", "Gap", "Gap"],
    "Metric": ["DV01", "Dollar Convexity", "DV01", "Dollar Convexity",
               "DV01 Gap", "Dollar Convexity Gap"],
    "Value": [dv01_a, dc_a, dv01_l, dc_l, dv01_gap, dc_gap],
})

(
    GT(gap_df, rowname_col="Metric", groupname_col="Side")
    .tab_header(title="Duration & Convexity Gaps", subtitle=f"Market rate: {rate:.2%}")
    .fmt_number(columns="Value", decimals=2)
)
```

Duration & Convexity Gaps Market rate: 4.00%	Value
Assets	
DV01	4,728.16
Dollar Convexity	357,171,001.66
Liabilities	
DV01	5,351.69
Dollar Convexity	925,180,322.03
Gap	
DV01 Gap	623.53
Dollar Convexity Gap	568,009,320.37

The liability side has both higher DV01 (duration gap) *and* higher dollar convexity (convexity gap). A duration-only hedge would leave the convexity mismatch unaddressed.

Step 3 — Why duration-only hedging is imperfect

A single swap can close the DV01 gap, but the remaining convexity mismatch causes hedging error for larger rate moves. We demonstrate this by building a duration-only hedge and testing it across a range of rate shocks.

```
# --- Duration-only hedge: single 10-year receive-fixed swap ---
unit_swap_10y = InterestRateSwap(
    notional=1, fixed_rate=rate, tenor=10, frequency=2, pay_fixed=False,
)

def swap_pv_10y(r: float) -> float:
    return unit_swap_10y.present_value([r] * unit_swap_10y.n_periods, r)

dv01_per_dollar_10y = dv01(swap_pv_10y, rate)
notional_dur_only = dv01_gap / dv01_per_dollar_10y
notional_dur_only = round(notional_dur_only / 100_000) * 100_000

swap_dur_only = InterestRateSwap(
    notional=notional_dur_only, fixed_rate=rate, tenor=10,
    frequency=2, pay_fixed=False,
)
```

Test across parallel rate shocks from −200 to +200 bps:

```
def portfolio_pv(r: float) -> float:
    """Total asset PV (bonds only, no hedge)."""
    return bond_a.present_value(r) + bond_b.present_value(r)

def liability_pv(r: float) -> float:
    return spia.present_value(r) + term.present_value(r)

shocks_bp = list(range(-200, 201, 25))
rows = []
for shock in shocks_bp:
    r2 = rate + shock / 10_000
    a_pv = portfolio_pv(r2)
    l_pv = liability_pv(r2)
    swap_pv = swap_dur_only.present_value([r2] * swap_dur_only.n_periods, r2)

    surplus_unhedged = a_pv - l_pv
    surplus_dur_only = a_pv + swap_pv - l_pv
    rows.append({
        "shock_bp": shock,
        "surplus_unhedged": surplus_unhedged,
        "surplus_dur_only": surplus_dur_only,
    })

shock_df = pl.DataFrame(rows)
```

```
if IS_PDF:
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.plot(shock_df["shock_bp"], shock_df["surplus_unhedged"], label="Unhedged")
    ax.plot(shock_df["shock_bp"], shock_df["surplus_dur_only"], label="Duration-only hedge")
    ax.set_title("Surplus Under Parallel Rate Shocks")
    ax.set_xlabel("Rate shock (bps)")
    ax.set_ylabel("Surplus ($)")
    ax.yaxis.set_major_formatter(mticker.StrMethodFormatter("${x:,.0f}"))
    ax.legend()
    plt.tight_layout()
    plt.show()
else:
    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=shock_df["shock_bp"].to_list(),
        y=shock_df["surplus_unhedged"].to_list(),
        mode="lines", name="Unhedged",
```

```
    ))
    fig.add_trace(go.Scatter(
        x=shock_df["shock_bp"].to_list(),
        y=shock_df["surplus_dur_only"].to_list(),
        mode="lines", name="Duration-only hedge",
    ))
    fig.update_layout(
        title="Surplus Under Parallel Rate Shocks",
        xaxis_title="Rate shock (bps)",
        yaxis_title="Surplus ($)",
        yaxis_tickformat="$,.0f",
        template="plotly_white",
    )
    fig.show()
```

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

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

The duration-only hedged surplus is approximately flat near the current rate but curves away for larger moves — this curvature is the **convexity mismatch**.

Step 4 — Convexity hedge with two swaps

To close *both* the duration and convexity gaps, we need two hedging instruments with different duration/convexity profiles. We use a **5-year** and a **20-year** receive-fixed swap.

```
# --- Unit-notional swaps ---
unit_swap_5y = InterestRateSwap(
    notional=1, fixed_rate=rate, tenor=5, frequency=2, pay_fixed=False,
)
unit_swap_20y = InterestRateSwap(
    notional=1, fixed_rate=rate, tenor=20, frequency=2, pay_fixed=False,
)

def swap_pv_5y(r: float) -> float:
    return unit_swap_5y.present_value([r] * unit_swap_5y.n_periods, r)

def swap_pv_20y(r: float) -> float:
    return unit_swap_20y.present_value([r] * unit_swap_20y.n_periods, r)
```

```
dd1 = dv01(swap_pv_5y, rate)
dc1 = dollar_convexity(swap_pv_5y, rate)
dd2 = dv01(swap_pv_20y, rate)
dc2 = dollar_convexity(swap_pv_20y, rate)

n1, n2 = immunize(
    dd_gap=dv01_gap, dc_gap=dc_gap,
    dd_per_unit_1=dd1, dc_per_unit_1=dc1,
    dd_per_unit_2=dd2, dc_per_unit_2=dc2,
)

# Round to nearest $100k
n1 = round(n1 / 100_000) * 100_000
n2 = round(n2 / 100_000) * 100_000

hedge_df = pl.DataFrame({
    "Instrument": ["5-yr swap", "20-yr swap"],
    "DV01 per $1": [dd1, dd2],
    "Dollar Cvx per $1": [dc1, dc2],
    "Notional": [n1, n2],
})

(
    GT(hedge_df, rowname_col="Instrument")
    .tab_header(title="Convexity Hedge - Swap Sizing")
    .fmt_number(columns="DV01 per $1", decimals=10)
    .fmt_number(columns="Dollar Cvx per $1", decimals=6)
    .fmt_currency(columns="Notional", decimals=0)
)
```

Convexity Hedge — Swap Sizing			
	DV01 per $1	Dollar Cvx per $1	Notional
5-yr swap	0.0004491293	23.498866	−$8,300,000
20-yr swap	0.0013677747	239.876848	$3,200,000

```
# --- Build sized swaps ---
swap_5y = InterestRateSwap(
    notional=n1, fixed_rate=rate, tenor=5, frequency=2, pay_fixed=False,
)
swap_20y = InterestRateSwap(
```

```
    notional=n2, fixed_rate=rate, tenor=20, frequency=2, pay_fixed=False,
)
```

Verify gaps are closed

```
def hedge_pv(r: float) -> float:
    return (
        swap_5y.present_value([r] * swap_5y.n_periods, r)
        + swap_20y.present_value([r] * swap_20y.n_periods, r)
    )

dv01_hedge = dv01(hedge_pv, rate)
dc_hedge = dollar_convexity(hedge_pv, rate)

verify_df = pl.DataFrame({
    "Metric": [
        "DV01 (Assets + Hedge)",
        "DV01 (Liabilities)",
        "Residual DV01 Gap",
        "Dollar Cvx (Assets + Hedge)",
        "Dollar Cvx (Liabilities)",
        "Residual Cvx Gap",
    ],
    "Value": [
        dv01_a + dv01_hedge,
        dv01_l,
        (dv01_a + dv01_hedge) - dv01_l,
        dc_a + dc_hedge,
        dc_l,
        (dc_a + dc_hedge) - dc_l,
    ],
})

(
    GT(verify_df, rowname_col="Metric")
    .tab_header(title="Hedged Portfolio - Gap Verification")
    .fmt_number(columns="Value", decimals=2)
)
```

Hedged Portfolio — Gap Verification	Value
DV01 (Assets + Hedge)	5,377.27
DV01 (Liabilities)	5,351.69
Residual DV01 Gap	25.57
Dollar Cvx (Assets + Hedge)	929,736,325.39
Dollar Cvx (Liabilities)	925,180,322.03
Residual Cvx Gap	4,556,003.36

Step 5 — Compare hedges under rate shocks

```
rows2 = []
for shock in shocks_bp:
    r2 = rate + shock / 10_000
    a_pv = portfolio_pv(r2)
    l_pv = liability_pv(r2)
    swap_d = swap_dur_only.present_value([r2] * swap_dur_only.n_periods, r2)
    swap_c = hedge_pv(r2)

    rows2.append({
        "shock_bp": shock,
        "surplus_unhedged": a_pv - l_pv,
        "surplus_dur_only": a_pv + swap_d - l_pv,
        "surplus_dur_cvx": a_pv + swap_c - l_pv,
    })

compare_df = pl.DataFrame(rows2)
```

```
if IS_PDF:
    fig2, ax2 = plt.subplots(figsize=(10, 5))
    ax2.plot(compare_df["shock_bp"], compare_df["surplus_unhedged"], label="Unhedged")
    ax2.plot(compare_df["shock_bp"], compare_df["surplus_dur_only"], label="Duration-only")
    ax2.plot(compare_df["shock_bp"], compare_df["surplus_dur_cvx"], label="Duration + Convexi
    ax2.set_title("Surplus Comparison: Three Hedging Strategies")
    ax2.set_xlabel("Rate shock (bps)")
    ax2.set_ylabel("Surplus ($)")
    ax2.yaxis.set_major_formatter(mticker.StrMethodFormatter("${x:,.0f}"))
    ax2.legend()
    plt.tight_layout()
```

```
    plt.show()
else:
    fig2 = go.Figure()
    fig2.add_trace(go.Scatter(
        x=compare_df["shock_bp"].to_list(),
        y=compare_df["surplus_unhedged"].to_list(),
        mode="lines", name="Unhedged",
    ))
    fig2.add_trace(go.Scatter(
        x=compare_df["shock_bp"].to_list(),
        y=compare_df["surplus_dur_only"].to_list(),
        mode="lines", name="Duration-only",
    ))
    fig2.add_trace(go.Scatter(
        x=compare_df["shock_bp"].to_list(),
        y=compare_df["surplus_dur_cvx"].to_list(),
        mode="lines", name="Duration + Convexity",
    ))
    fig2.update_layout(
        title="Surplus Comparison: Three Hedging Strategies",
        xaxis_title="Rate shock (bps)",
        yaxis_title="Surplus ($)",
        yaxis_tickformat="$,.0f",
        template="plotly_white",
    )
    fig2.show()
```

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

```
# Summary table at key shock levels
key_shocks = [-200, -100, -50, 0, 50, 100, 200]
summary = compare_df.filter(pl.col("shock_bp").is_in(key_shocks))

(
    GT(summary, rowname_col="shock_bp")
    .tab_header(title="Surplus at Key Rate Shocks")
    .tab_stubhead(label="Shock (bps)")
    .cols_label(
        surplus_unhedged="Unhedged",
        surplus_dur_only="Duration-only",
        surplus_dur_cvx="Duration + Convexity",
```

```
    )
    .fmt_currency(
        columns=["surplus_unhedged", "surplus_dur_only", "surplus_dur_cvx"],
        decimals=0,
    )
)
```

Surplus at Key Rate Shocks			
Shock (bps)	Unhedged	Duration-only	Duration + Convexity
-200	$3,031,669	$3,176,033	$3,296,261
-100	$3,200,632	$3,269,307	$3,296,565
-50	$3,255,600	$3,289,106	$3,295,504
0	$3,294,220	$3,294,220	$3,294,220
50	$3,318,618	$3,286,690	$3,293,044
100	$3,330,669	$3,268,312	$3,292,235
200	$3,324,137	$3,205,117	$3,292,471

Key takeaways

- **Duration-only** hedging eliminates first-order rate sensitivity but leaves a convexity mismatch — surplus curves away from target for larger moves.
- **Duration + convexity** hedging requires **two** instruments with different duration/convexity profiles (here, 5-yr and 20-yr swaps). The `immunize()` function solves the 2×2 system for optimal notionals.
- The convexity-hedged surplus stays **much flatter** across a wide range of rate shocks, providing better protection against large parallel moves.
- In practice, residual risk remains from non-parallel shifts (key-rate risk), higher-order terms, and rounding of notionals.