

demo_iv_surface

September 1, 2025

0.1 Implied Volatility Surface — Financial Background

0.1.1 1. Why do we use IV (instead of raw prices)?

- **Standardized language:** Raw option prices vary wildly across strike K and maturity T and are not comparable. Quoting **implied volatility** puts everything on a common, annualized scale, so traders can compare across strikes, maturities, and even assets (e.g., “1M ATM IV = 18%”).
- **What you pay:** Markets **quote** in IV but **settle** in price via Black–Scholes (or your pricer): $\text{price} = \text{BS}(S_0, K, T, r, q, \sigma_{\text{quoted}})$. Bid/ask in IV \Rightarrow bid/ask in price.
- Only the market consensus discount factor e^{-rT} and forward F (both observable from bond/futures curves) are needed, thus avoiding the need to estimate r and q separately, also r, q are not arbitrary assumptions, but come from market curves (bonds, futures, forwards). There is a general consensus on these curves.
- **No “true IV”:** IV is **implied from prices** under a model; it’s not a physical parameter. A **mid-market IV surface** (from mid prices) reflects the market consensus; quotes carry a spread.
- **Economic intuition (time value):** $C = \max(S_0 - K, 0) + \text{time value}$. (Out-of-Money) OTM calls/puts are **all time value** and are most sensitive to tail probabilities; hence they reveal non-Gaussian features most clearly.
- **Practical uses:** Risk (Greeks like Vega/Vanna/Volga), scenario analysis, model calibration (local vol, Heston, SABR), and volatility/relative-value trading (e.g., skew trades). IV level/shape also acts as a sentiment/tail-risk barometer (e.g., VIX).

0.1.2 2. How IV surface maps to skewness & kurtosis

- **Skewness IV slope (in K or log-moneyness k):**
Left-skewed (crash-prone) equity indices \Rightarrow **downward skew** (low K IV high). Right-skewed commodities \Rightarrow **upward skew** (high K IV high).
- **Kurtosis IV curvature (“smile”):**
Fat tails make **both wings** (deep ITM/OTM) relatively expensive \Rightarrow IV **higher at wings, lower near ATM** (a smile).
- **Intuition without parity:**
Right tail thicker \Rightarrow OTM calls’ time value \uparrow ; left tail thicker \Rightarrow ITM calls are less “certain” than BS assumes, their residual time value \uparrow ; together this yields the smile.

Compact mapping:

$$\text{Skewness} \iff \frac{\partial \sigma_{\text{imp}}}{\partial k}, \quad \text{Kurtosis} \iff \frac{\partial^2 \sigma_{\text{imp}}}{\partial k^2} \text{ (or curvature of } w = \sigma^2 T \text{)}. \quad (1)$$

0.1.3 3. Mathematical background (key formulas)

- **Black–Scholes call:**

$$C_{\text{BS}}(S_0, K, T, r, q, \sigma) = S_0 e^{-qT} \Phi(d_1) - K e^{-rT} \Phi(d_2), \quad (2)$$

$$d_{1,2} = \frac{\ln(S_0/K) + (r - q \pm \frac{1}{2}\sigma^2)T}{\sigma\sqrt{T}}. \quad (3)$$

- **Implied volatility (per (K, T)):**

$$C_{\text{BS}}(\sigma_{\text{imp}}) = C_{\text{mkt}}. \quad (4)$$

- **Vega (price sensitivity to σ):**

$$\text{Vega} = \frac{\partial C_{\text{BS}}}{\partial \sigma} = S_0 e^{-qT} \phi(d_1) \sqrt{T}, \quad \Delta \sigma \approx \frac{\Delta C}{\text{Vega}}. \quad (5)$$

- **Forward & total variance (numerical stability):**

$$F = S_0 e^{(r-q)T}, \quad k = \ln(K/F), \quad w(k, T) = \sigma_{\text{imp}}(k, T)^2 T. \quad (6)$$

- **Put–call parity (consistency across strikes):**

$$C - P = S_0 e^{-qT} - K e^{-rT}. \quad (7)$$

- **Risk-neutral density (Breedon–Litzenberger):**

$$\frac{\partial^2 C(K, T)}{\partial K^2} = e^{-rT} f_{S_T}(K), \quad (8)$$

which links option prices (hence the IV surface) to the risk-neutral distribution's shape (skewness/kurtosis).

0.2 Implied Volatility Surface: Link to Skewness and Kurtosis – Mathematical Proof

We summarize the mathematical reasoning for why the slope and curvature of the IV surface encode the higher moments of the risk-neutral distribution.

0.2.1 1. Set-up

Forward price and log-moneyness:

$$F = S_0 e^{(r-q)T}, \quad k = \ln\left(\frac{K}{F}\right). \quad (9)$$

Total implied variance:

$$w(k, T) = \sigma_{\text{imp}}(k, T)^2 T. \quad (10)$$

Definition of implied volatility:

$$c_{\text{BS}}(k, w(k, T)) = c_{\text{mkt}}(k, T), \quad (11)$$

where c is the forward call price.

0.2.2 2. Implicit differentiation

Differentiate w.r.t. k :

$$c_k^{\text{BS}} + c_w^{\text{BS}} w_k = c_k^{\text{mkt}}, \quad \Rightarrow \quad w_k = \frac{c_k^{\text{mkt}} - c_k^{\text{BS}}}{c_w^{\text{BS}}}. \quad (12)$$

Differentiate again:

$$c_{kk}^{\text{BS}} + 2c_{kw}^{\text{BS}} w_k + c_{ww}^{\text{BS}} w_k^2 + c_w^{\text{BS}} w_{kk} = c_{kk}^{\text{mkt}}, \quad (13)$$

$$w_{kk} = \frac{c_{kk}^{\text{mkt}} - c_{kk}^{\text{BS}} - 2c_{kw}^{\text{BS}} w_k - c_{ww}^{\text{BS}} w_k^2}{c_w^{\text{BS}}}. \quad (14)$$

Thus IV slope and curvature are proportional to the difference between **true distribution** and **Gaussian BS** benchmarks.

0.2.3 3. Cumulant expansion

Let $X = \ln(S_T/F)$. Standardized cumulants:

$$\gamma_1 = \mathbb{E}[Z^3], \quad \gamma_2 = \mathbb{E}[Z^4] - 3, \quad Z = \frac{X - \mu}{\sigma_X}. \quad (15)$$

Edgeworth expansion:

$$f_X(x) \approx \frac{1}{\sigma_X} \phi(z) \left[1 + \frac{\gamma_1}{6} H_3(z) + \frac{\gamma_2}{24} H_4(z) + \frac{\gamma_1^2}{72} H_6(z) \right]. \quad (16)$$

Substitution into price derivatives gives, to leading order:

$$w_k \propto \gamma_1 + \mathcal{O}(\gamma_2), \quad w_{kk} \propto \gamma_2 + \mathcal{O}(\gamma_1^2). \quad (17)$$

0.2.4 4. ATM short-maturity asymptotics

At $k = 0$, small T :

$$\left. \frac{\partial \sigma_{\text{imp}}}{\partial k} \right|_{k=0} \approx -\frac{\gamma_1}{6} \frac{\sigma_{\text{imp}}}{\sqrt{T}}, \quad (18)$$

$$\left. \frac{\partial^2 \sigma_{\text{imp}}}{\partial k^2} \right|_{k=0} \approx \frac{\gamma_2}{24} \frac{\sigma_{\text{imp}}}{T} + \frac{\gamma_1^2}{72} \frac{\sigma_{\text{imp}}}{T}. \quad (19)$$

0.2.5 5. Conclusion

$$\text{Skewness} \iff \frac{\partial \sigma_{\text{imp}}}{\partial k}, \quad \text{Kurtosis} \iff \frac{\partial^2 \sigma_{\text{imp}}}{\partial k^2}. \quad (20)$$

- The **slope** of the IV smile reflects **skewness** (left/right tail asymmetry).
- The **curvature** of the IV smile reflects **kurtosis** (fat tails), plus a smaller skewness-squared term.

1 Option Mini-Lab — Implied Volatility Surface Demo

Goals 1. Generate a small grid of model-consistent option quotes. 2. Recover implied vols and build an IV surface. 3. Interpolate IV at arbitrary (K, T). 4. Visualize smiles, term structures, and the surface. 5. Stress-test solver stability (deep ITM/OTM, tiny T). 6. (Optional) Quick no-arbitrage sanity checks (butterfly/calendar, heuristic).

1.1 Conclusion

- We built an IV surface from synthetic quotes and recovered the ground-truth volatility.
- Interpolation (`iv_at`) provides smooth queries between grid nodes.
- Plot helpers generated smiles, term structures, and a 3D surface.
- The solver is numerically stable in edge cases (deep ITM/OTM, very short maturities).

Next steps (good first issues): - Add no-arbitrage smoothing / Hagan–West strike interpolation.
- Calibrate a SABR slice per maturity and compare to raw IV. - Add unit tests: round-trip pricing, interpolation edges, stress cases.

```
[1]: import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import sys, os
project_root = os.path.abspath(os.path.join(os.getcwd(), '..'))
sys.path.append(project_root)
```

```
[2]: %cd $project_root

/Users/zhaoyub/Documents/Tradings/option-mini-lab

/Users/zhaoyub/Library/Python/3.12/lib/python/site-
packages/IPython/core/magics/osm.py:417: UserWarning: This is now an optional
IPython functionality, setting dhyst requires you to install the `pickleshare`
library.
  self.shell.db['dhyst'] = compress_dhist(dhist)[-100:]
```

```
[3]: %load_ext autoreload
```

```
[4]: import numpy as np
import matplotlib.pyplot as plt

from src.iv_surface import (
    bs_price,
    implied_vol,
    Quote,
    IVSurface,
    build_surface,
    price_from_iv_grid,
    butterfly_violations,
    calendar_violations,
    implied_vol_trace
)
```

```
[5]: np.set_printoptions(precision=6, suppress=True)
rng = np.random.default_rng()
```

1.2 Market Setup

We create a synthetic market: - Spot $S=100$, rates $r=1\%$, dividend/borrow $q=0\%$. - Two maturities: $T \{0.5, 1.0\}$ years. - Strikes $K \{80, 90, 100, 110, 120\}$. - Ground-truth volatility $\sigma=20\%$.

We'll price European calls under Black–Scholes, then recover implied vols and build a rectangular surface.

1.2.1 Generating Synthetic Quotes:

```
[7]: S, r, q = 100.0, 0.01, 0.00
Ts = np.array([0.25, 0.5, 1.0, 2.0])
Ks = np.array([70, 80, 90, 100, 110, 120, 140])

TT, KK = np.meshgrid(Ts, Ks, indexing="ij")
TT = TT.ravel()
KK = KK.ravel()

def iv_true(K, T, S=S):
    # Toy smile + skew + mild term structure (smooth and positive)
    m = np.log(K / S) # log-moneyness
    base = 0.18 + 0.04*np.sqrt(T) # upward-sloping term structure
    wings = 0.12*(m**2) # symmetric smile (convex in log-moneyness)
    skew = -0.06*m # add a left skew (equity-style)
    sig = base + wings + skew
    return float(np.clip(sig, 0.05, 0.90))

sigma_true = np.array([iv_true(K, T) for K, T in zip(KK, TT)])
```

1.2.2 Build IV Surface

```
[8]: prices = np.array([bs_price(S, K, r, q, T, sig, call=True)
                        for (K, T, sig) in zip(KK, TT, sigma_true)])
calls = np.array([True] * len(prices))
surf = build_surface(S, r, q, KK, TT, prices, calls=True)

mad = float(np.nanmean(np.abs(surf.iv.ravel() - sigma_true)))
print("Mean |recovered IV - true IV|:", mad)
assert mad < 1e-8
```

Mean |recovered IV - true IV|: 1.5283065120980893e-09

1.2.3 Interpolation Sanity Checks

```
[9]: tests = [
      (100.0, 0.75), # between the two maturities, ATM
      (95.0, 0.5),  # between strikes at T=0.5
      (105.0, 1.00), # between strikes at T=1.0
    ]
for Kq, Tq in tests:
    v = surf.iv_at(Kq, Tq)
    print(f"IV_at(K={Kq:.1f}, T={Tq:.2f}) = {v:.6f}")
```

IV_at(K=100.0, T=0.75) = 0.214142

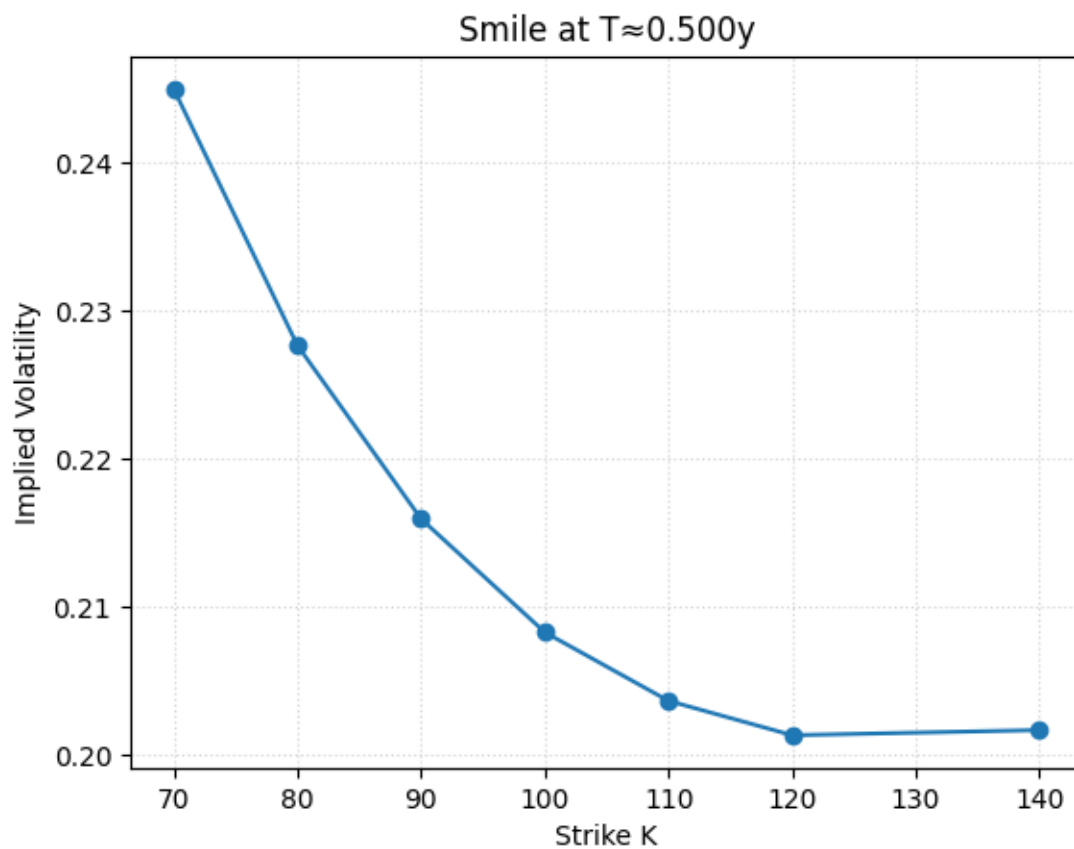
IV_at(K=95.0, T=0.50) = 0.212111

IV_at(K=105.0, T=1.00) = 0.217686

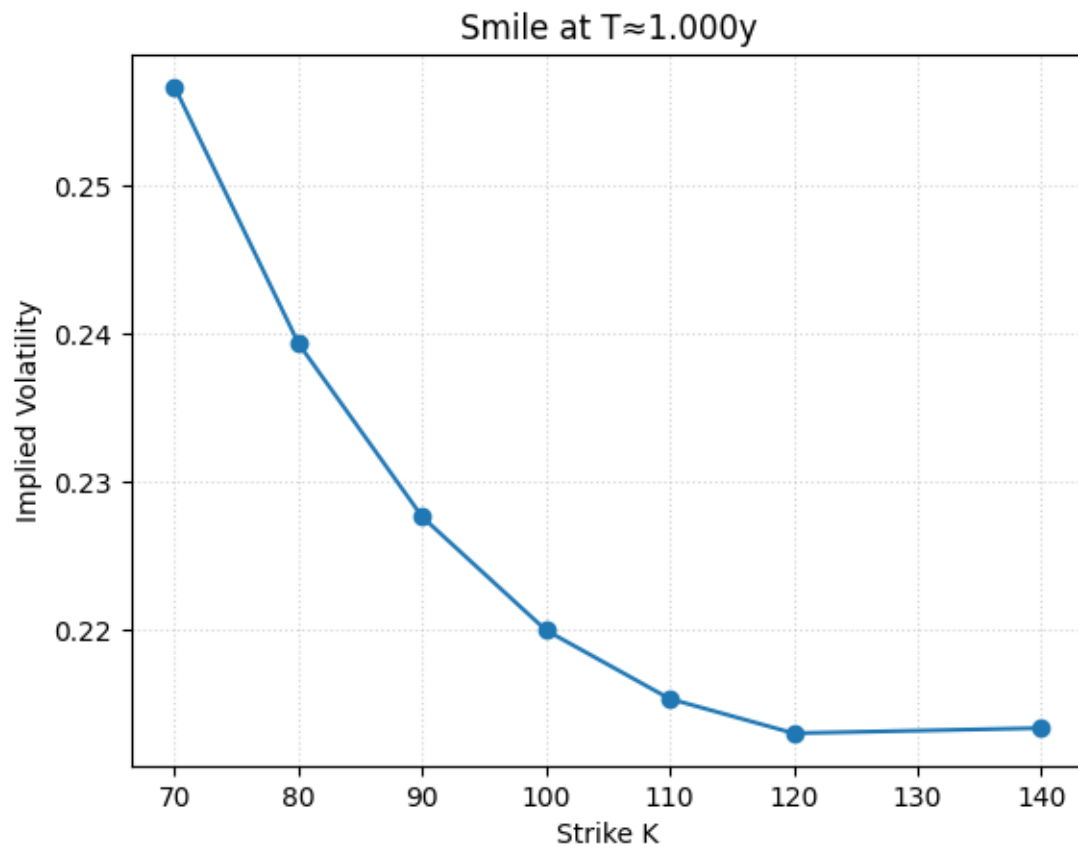
```
[10]: plt.figure(figsize=(6,4))
ax = surf.plot_smile(T=0.5)
plt.show()

plt.figure(figsize=(6,4))
ax = surf.plot_smile(T=1.0)
plt.show()
```

<Figure size 600x400 with 0 Axes>

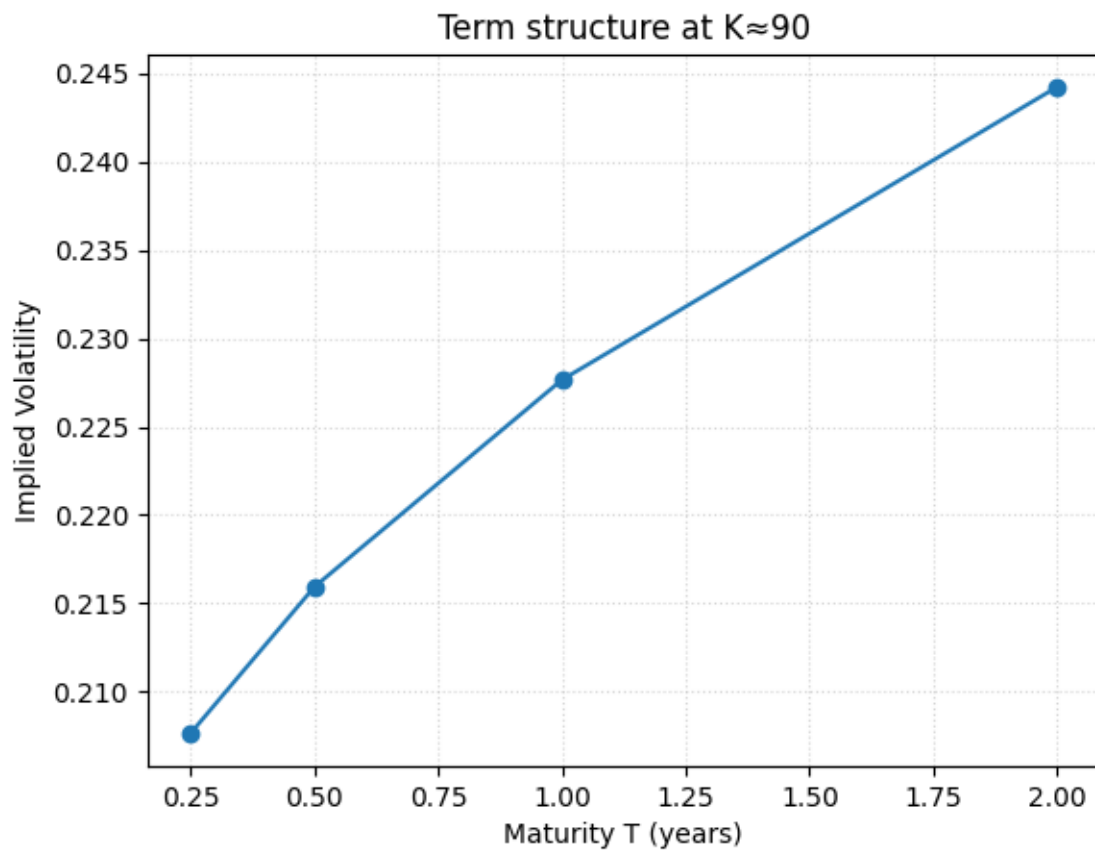


<Figure size 600x400 with 0 Axes>

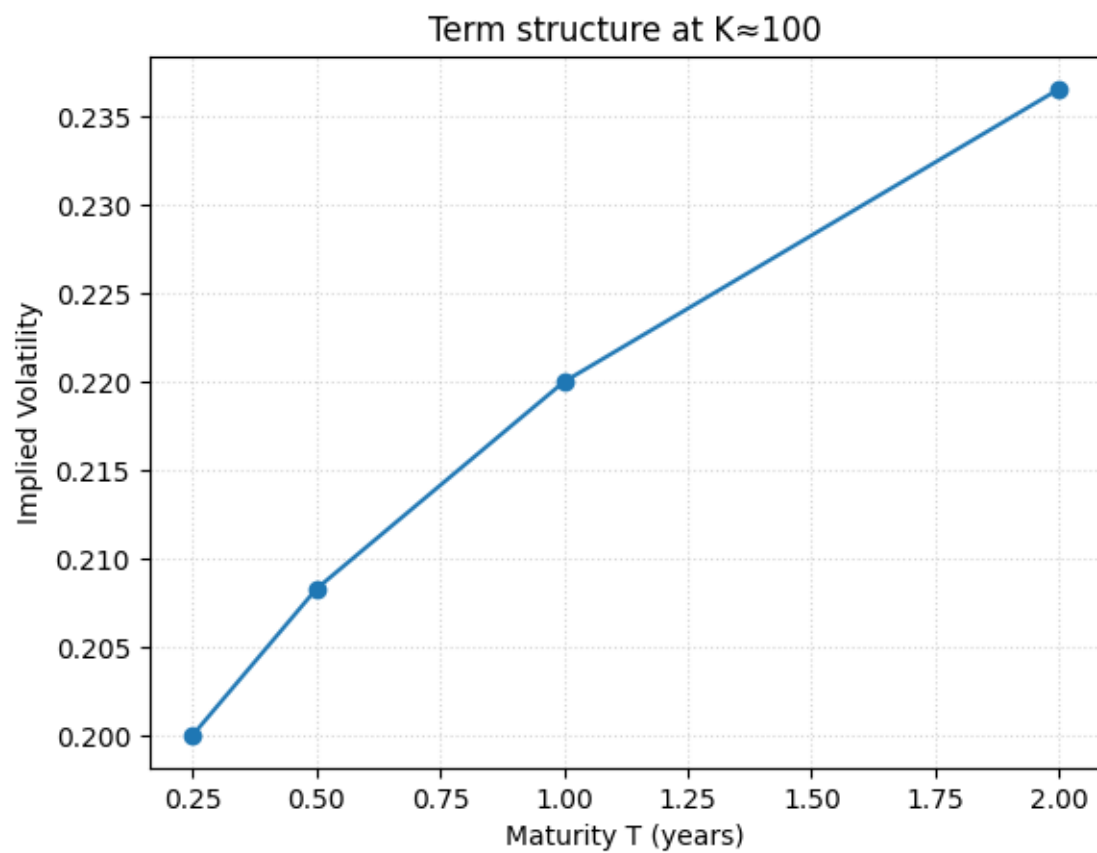


```
[11]: for K0 in [90, 100, 110]:  
    plt.figure(figsize=(6,4))  
    ax = surf.plot_term_structure(K=K0)  
    plt.show()
```

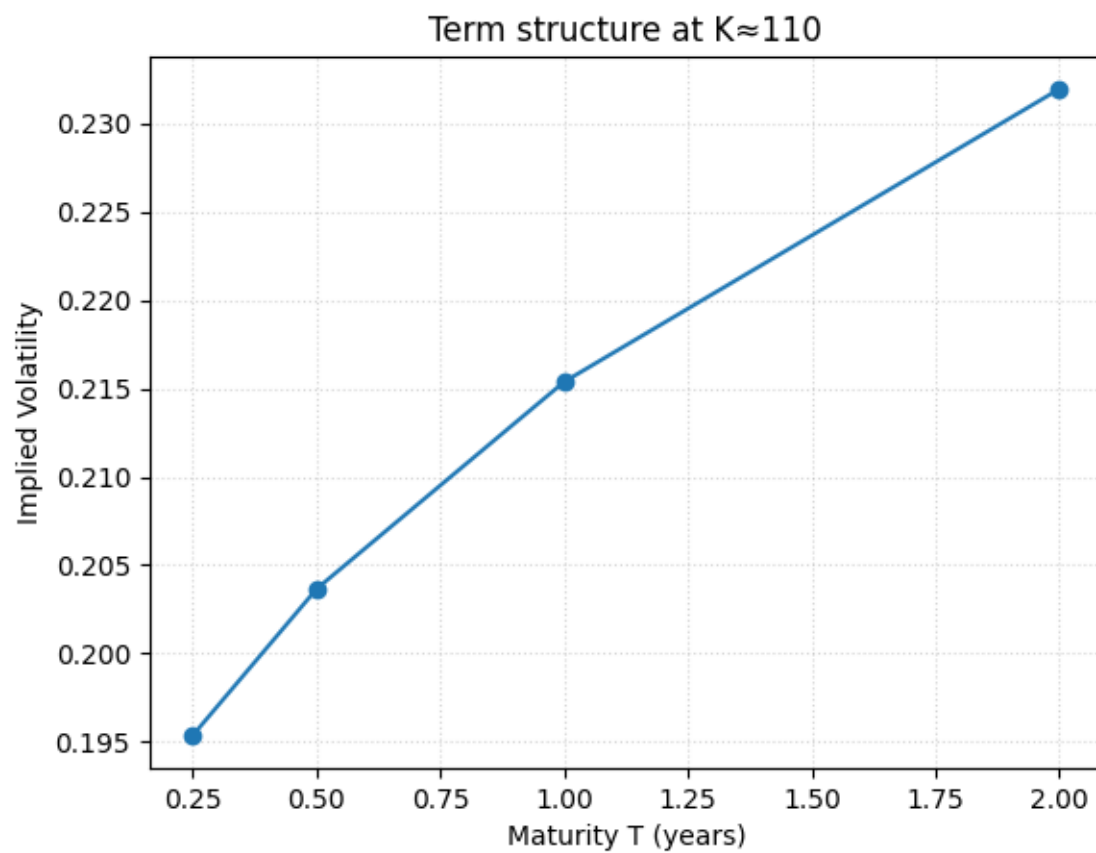
<Figure size 600x400 with 0 Axes>



<Figure size 600x400 with 0 Axes>

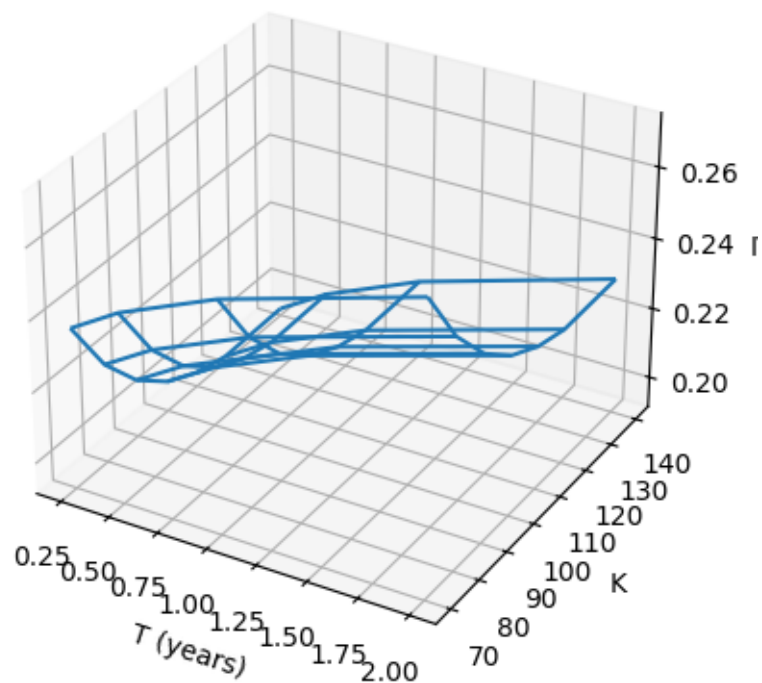


<Figure size 600x400 with 0 Axes>



```
[12]: ax = surf.plot_surface()  
      plt.show()
```

Implied Vol Surface



1.2.4 Pointwise IV Recovery vs Direct Solver

```
[14]: import pandas as pd

df = pd.DataFrame({
    "T": TT,
    "K": KK,
    "IV_true": sigma_true,
    "IV_rec": [surf.iv_at(K, T) for K, T in zip(KK, TT)]
}).sort_values(["T", "K"])

df.head(10), float(np.nanmean(np.abs(df.IV_true - df.IV_rec)))
```

```
[14]: (
      T      K  IV_true  IV_rec
0  0.25    70  0.236667  0.236667
1  0.25    80  0.219364  0.219364
2  0.25    90  0.207654  0.207654
3  0.25   100  0.200000  0.200000
4  0.25   110  0.195371  0.195371
5  0.25   120  0.193050  0.193050
6  0.25   140  0.193397  0.193397
```

```

7  0.50   70  0.244951  0.244951
8  0.50   80  0.227648  0.227648
9  0.50   90  0.215938  0.215938,
1.5283065120980893e-09)

```

1.3 Stress Test

We stress the solver with: - Deep ITM (K=60) and deep OTM (K=140) options. - Very small maturity $T=1/365$ 1 day.

We expect the solver to return a stable IV near the ground truth (here we price from $\sigma=0.20$ then invert).

1.3.1 Stress Test: Deep ITM/OTM & Small T

```

[17]: S, r, q = 100.0, 0.01, 0.0
Ks_extreme = [40, 60, 80, 100, 120, 150, 200]
Ts_extreme = [1/365, 7/365, 0.1, 0.5, 1.0, 3.0, 10.0]
sigma_true = 0.25

rows = []
for K in Ks_extreme:
    for T in Ts_extreme:
        p = bs_price(S, K, r, q, T, sigma_true, True)
        iv = implied_vol(p, S, K, r, q, T, True)
        rows.append((K, T, p, iv, iv - sigma_true))

import pandas as pd
df_extreme = pd.DataFrame(rows, columns=["K", "T", "Price", "IV_rec", "Error"])
display(df_extreme.head(10))
print("Max |error|:", df_extreme["Error"].abs().max())

```

	K	T	Price	IV_rec	Error
0	40	0.002740	60.001096	0.000001	-2.499990e-01
1	40	0.019178	60.007670	0.000001	-2.499990e-01
2	40	0.100000	60.039980	0.000001	-2.499990e-01
3	40	0.500000	60.199501	0.250011	1.074797e-05
4	40	1.000000	60.398403	0.250000	9.560407e-10
5	40	3.000000	61.317048	0.250000	8.326673e-15
6	40	10.000000	65.922693	0.250000	2.070566e-12
7	60	0.002740	40.001644	0.000001	-2.499990e-01
8	60	0.019178	40.011506	0.000001	-2.499990e-01
9	60	0.100000	40.059970	0.200000	-5.000000e-02

Max |error|: 0.249999

1.3.2 Near-intrinsic & near-zero prices (hard cases)

```
[18]: cases = []
# Deep ITM small T ~ intrinsic
for T in [1/365, 3/365, 5/365]:
    K = 60.0
    sig = 0.15
    p = bs_price(S, K, r, q, T, sig, True)
    # Push price slightly BELOW intrinsic to trigger lower bound behavior
    intrinsic = max(0.0, S*np.exp(-q*T) - K*np.exp(-r*T))
    p_minus = max(1e-12, intrinsic * 0.999999)
    p_plus = max(p, intrinsic + 1e-8)
    cases.append(("below_intrinsic", K, T, p_minus))
    cases.append(("at_model", K, T, p_plus))

# Deep OTM tiny T ~ price near 0
for T in [1/365, 2/365, 5/365]:
    K = 200.0
    sig = 0.2
    p = bs_price(S, K, r, q, T, sig, True)
    cases.append(("near_zero_price", K, T, p))

for name, K, T, p in cases:
    iv = implied_vol(p, S, K, r, q, T, True)
    print(f"{name:>18s} | K={K:6.1f} T={T:8.5f} price={p:.8g} IV={iv:.6f}")

below_intrinsic | K= 60.0 T= 0.00274 price=40.001604 IV=0.000001
at_model | K= 60.0 T= 0.00274 price=40.001644 IV=1.750110
below_intrinsic | K= 60.0 T= 0.00822 price=40.004891 IV=0.000001
at_model | K= 60.0 T= 0.00822 price=40.004931 IV=1.006934
below_intrinsic | K= 60.0 T= 0.01370 price=40.008179 IV=0.000001
at_model | K= 60.0 T= 0.01370 price=40.008219 IV=0.782972
near_zero_price | K= 200.0 T= 0.00274 price=0 IV=0.000001
near_zero_price | K= 200.0 T= 0.00548 price=0 IV=0.000001
near_zero_price | K= 200.0 T= 0.01370 price=0 IV=0.000001
```

1.3.3 Random grid w/ non-flat (K,T) + noise → inversion error stats

```
[19]: rng = np.random.default_rng(0)

def iv_true(K, T, S=100):
    m = np.log(K/S)
    base = 0.18 + 0.05*np.sqrt(T)
    wings = 0.10*(m**2)
    skew = -0.05*m
    return float(np.clip(base + wings + skew, 0.05, 0.9))

Ks = np.linspace(60, 160, 11)
```

```

Ts = np.linspace(1/365, 2.0, 10)

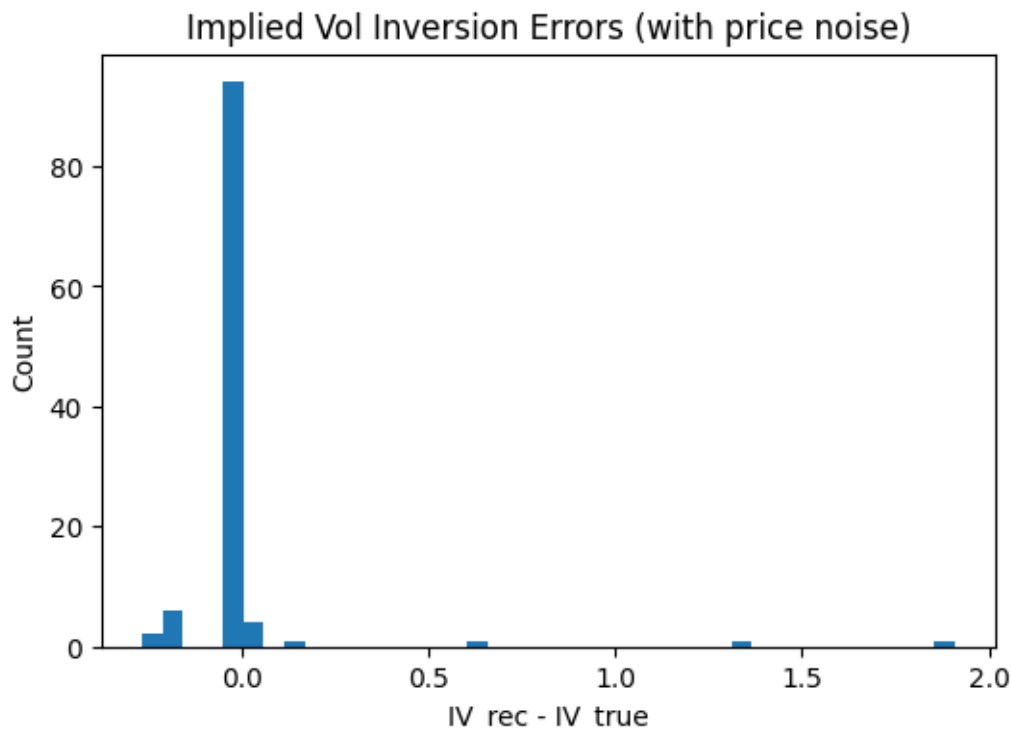
rows = []
for T in Ts:
    for K in Ks:
        sig = iv_true(K, T)
        p = bs_price(S, K, r, q, T, sig, True)
        # add small market micro-noise proportional to price level
        noise = (0.0005 + 0.0005*rng.random()) * p
        pmid = max(0.0, p + rng.normal(0.0, noise))
        iv = implied_vol(pmid, S, K, r, q, T, True)
        rows.append((K, T, sig, pmid, iv, iv - sig))

df_noise = pd.DataFrame(rows,
    columns=["K", "T", "IV_true", "Price_mid", "IV_rec", "Error"])
print("MAE:", df_noise["Error"].abs().mean(), " 95%|Error|:",
    df_noise["Error"].abs().quantile(0.95))

# Error histogram
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
plt.hist(df_noise["Error"], bins=40)
plt.xlabel("IV_rec - IV_true")
plt.ylabel("Count")
plt.title("Implied Vol Inversion Errors (with price noise)")
plt.show()

```

MAE: 0.05170201495639091 95%|Error|: 0.18011583043418106



1.4 Using Bid/Ask to Build Surface

In practice we often have bid/ask instead of a clean mid. Below we perturb model prices to synthetic bid/ask, build Quote objects with bid & ask, and confirm recovered IVs remain close to the truth.

```
[23]: IV_true = np.array([iv_true(K, T) for K, T in zip(KK, TT)])
mid_prices = np.array([bs_price(S, K, r, q, T, sig, True)
                        for (K, T, sig) in zip(KK, TT, IV_true)])

# --- build synthetic bid/ask around mid ---
# Spread in bps, gently increasing with moneyness to mimic wider wings.
bps = 50 + 50 * np.abs(np.log(KK / S)) / 0.5      # ~50-150 bps
spreads = bps * 1e-4
bids = np.maximum(0.0, mid_prices * (1.0 - spreads))
asks = mid_prices * (1.0 + spreads)

# Quotes with bid/ask (no direct `price`)
quotes_ba = [
    Quote(S=S, K=float(K), T=float(T), r=r, q=q, is_call=True, bid=float(b),
    ↪ask=float(a))
    for K, T, b, a in zip(KK, TT, bids, asks)
]
```



```

# Build surface from bid/ask -> (module computes mid internally)
surf_ba = IVSurface.from_quotes(quotes_ba)

# --- accuracy: recovered IV vs ground truth at grid nodes ---
IV_rec_nodes = surf_ba.iv.ravel()
err = IV_rec_nodes - IV_true
mae = float(np.mean(np.abs(err)))
p95 = float(np.quantile(np.abs(err), 0.95))
print(f"MAE |IV_rec - IV_true| = {mae:.6f}, 95% = {p95:.6f}")

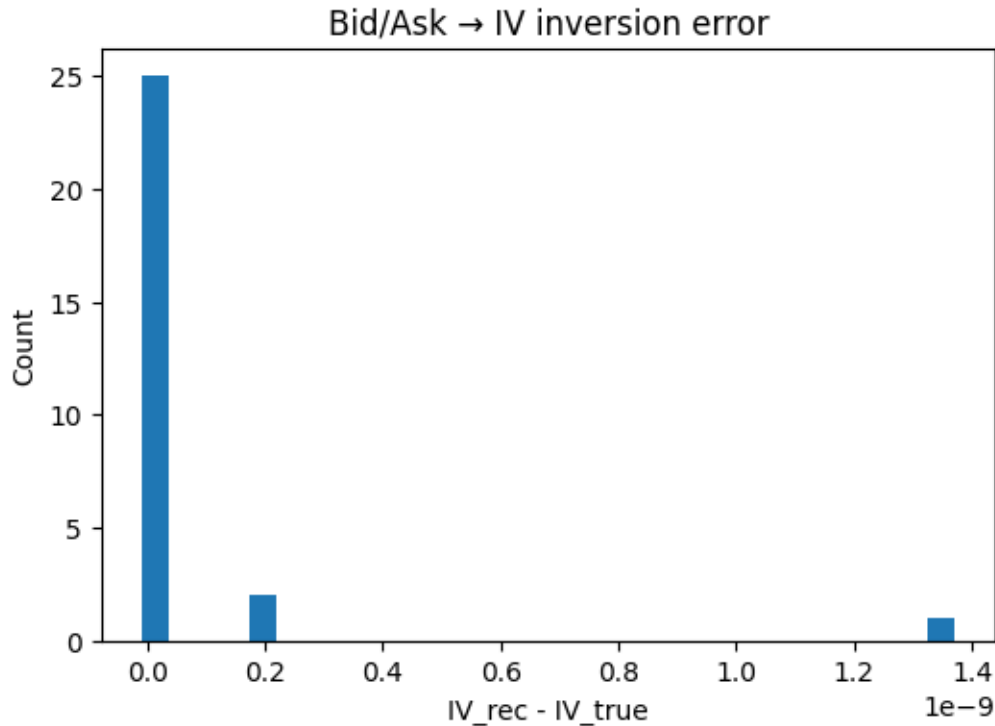
# Small table preview
df_chk = pd.DataFrame({
    "T": TT, "K": KK,
    "IV_true": IV_true,
    "IV_rec": IV_rec_nodes,
    "Error": err
}).sort_values(["T", "K"])
display(df_chk.head(10))

# Error histogram
plt.figure(figsize=(6,4))
plt.hist(err, bins=30)
plt.xlabel("IV_rec - IV_true")
plt.ylabel("Count")
plt.title("Bid/Ask → IV inversion error")
plt.show()

```

MAE |IV_rec - IV_true| = 0.000000, 95% = 0.000000

	T	K	IV_true	IV_rec	Error
0	0.25	70	0.235555	0.235555	3.154005e-12
1	0.25	80	0.221136	0.221136	8.192585e-12
2	0.25	90	0.211378	0.211378	2.570444e-13
3	0.25	100	0.205000	0.205000	1.942890e-16
4	0.25	110	0.201143	0.201143	1.646613e-11
5	0.25	120	0.199208	0.199208	2.096792e-10
6	0.25	140	0.199498	0.199498	1.370018e-09
7	0.50	70	0.245911	0.245911	1.890647e-10
8	0.50	80	0.231492	0.231492	4.246603e-14
9	0.50	90	0.221733	0.221733	2.445544e-13



1.5 Heuristic No-Arbitrage Checks

This is *not* a full no-arb enforcement. Quick checks only:

- **Butterfly (strike) convexity:** For fixed T , call price should be convex in K .
- **Calendar (term):** For fixed K , undiscounted call price is non-decreasing in T (under non-negative rates/dividends).

We'll test these on the **model prices** implied by `surf.iv` (by re-pricing with BS at the recovered IVs).

```
[20]: # Build a surface from the noisy quotes above
surf_noisy = build_surface(S, r, q, df_noise["K"], df_noise["T"],
    ↪df_noise["Price_mid"], calls=True)
C_noisy = price_from_iv_grid(surf_noisy, S, r, q)

butterfly_bad = butterfly_violations(C_noisy, surf_noisy.strikes)
calendar_bad = calendar_violations(C_noisy, surf_noisy.maturities)

print("Butterfly violations:", "None" if not butterfly_bad else
    ↪len(butterfly_bad))
print("Calendar violations :", "None" if not calendar_bad else
    ↪len(calendar_bad))
if butterfly_bad[:5]: print("Sample butterfly viols:", butterfly_bad[:5])
```

```
if calendar_bad[:5]: print("Sample calendar viols:", calendar_bad[:5])
```

Butterfly violations: 1

Calendar violations : None

Sample butterfly viols: [('butterfly', 0, 2, -0.0038668067156919506)]

1.5.1 Convergence Trace

```
[21]: K_star, T_star = 200.0, 1/365 # very far OTM & tiny T
p_star = bs_price(S, K_star, r, q, T_star, 0.25, True)
print("Target price:", p_star)
sigma_est = implied_vol_trace(p_star, S, K_star, r, q, T_star, True,
    ↪sigma_init=0.05)
print("Recovered :", sigma_est)
```

Target price: 0.0

Price intrinsic, returning lo

Recovered : 1e-06

1.6 Conclusion

- We built an IV surface from synthetic quotes and recovered the ground-truth volatility.
- Interpolation (iv_at) provides smooth queries between grid nodes.
- Plot helpers generated smiles, term structures, and a 3D surface.
- The solver is numerically stable in edge cases (deep ITM/OTM, very short maturities).

Next steps (good first issues): - Add no-arbitrage smoothing / Hagan–West strike interpolation.
- Calibrate a SABR slice per maturity and compare to raw IV. - Add unit tests: round-trip pricing, interpolation edges, stress cases.

```
[24]: ! jupyter nbconvert --to pdf notebooks/demo_iv_surface.ipynb
```

```
[NbConvertApp] Converting notebook notebooks/demo_iv_surface.ipynb to pdf
[NbConvertApp] Support files will be in demo_iv_surface_files/
[NbConvertApp] Making directory ./demo_iv_surface_files
[NbConvertApp] Writing 71428 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 352798 bytes to notebooks/demo_iv_surface.pdf
```