Challenges in Pricing Convertible Bonds and the TF Model

Convertible Bonds (CBs) are hybrid instruments:

- Consisting of a **straight bond** + **call option on the underlier**.
- Include embedded options: **early conversion**, **callability**, **putability**.

Key Challenge:

- The **cash payment component** (e.g., coupons, principal) is subject to **default risk**.
- Requires separation of the CB into two parts with **different discount rates**.

Tsiveriotis–Fernandes (TF) Model: Splits CB value U into:(detailed solution in Appendix)

- V: **Debt component** – discounted at $r + r_c$ (r_c: credit spread).
- B: **Equity component** – discounted at risk-free rate r.

$$\text{CB (full)}: \frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 U}{\partial S^2} + rS\frac{\partial U}{\partial S} + r(U - V) - (r + r_c)V + \text{accrued coupon} = 0$$

$$\text{COCB (cash-only CB, V)}: \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - (r + r_c)V + \text{accrued coupon} = 0$$

Project Overview

Problem: PDE-based CB pricers (e.g., TF Model) are computationally slow.

Objective: Accelerate pricing using ML/ RL while maintaining high accuracy (we need a fast and accurate replication of the TF model).

Key Requirement: Accurate PDE prices for ML training. Two ways to improve accuracy:

- **Calibration:** Tune model using **adjusted implied volatilities (IV)**.
- **Numerical Method:** Use **Crank-Nicolson (CN)** scheme with **optimal grid relation:** $dt = const \times dS$ to reduce numerical errors.

Workflow Summary:

1. Calibrate parameters (IV)
2. Implement Crank-Nicolson method with optimal dt/dS
3. Generate accurate training data from PDE solver
4. Train and test ML/RL models for Computational speed & Pricing accuracy.

TF Model IV Calibration — Summary of Work

	Pointwise Best	Global Single-Factor
Method	• **Grid search** to find optimal IV factors for **each data point**. • Then **average** or run a **regression** to derive a unified factor.	• Use **ML** to minimize total pricing error across all bonds/dates simultaneously. • Impose global form: $\sigma_i = \beta \cdot IV_{\text{Bloomberg},i}$.
Advantages	• **Near-perfect local fit** for each point.	• **Strong overall consistency** with a single global factor.
Concerns	• **Lacks global coherence**: β may vary by date. • Feasibility of a global factor depends on whether a **high** R^2 regression can be found.	• Time-consuming to optimize. • Cannot perfectly fit every data point. • Some dates show significant deviations.

TF Model IV Calibration — Regression for A Global Adjustment Factor

Goal: Find a unified factor explaining pointwise optimal adjustments.

Methodology:

- Regress `log(bestIV)` on `log(IVOL)`, `moneyness`, and `TTM`.
- Train-test split: 80% train, 20% test. Total: 43,748 samples.

Regression Equation:

$$\log(\text{bestIV}) = 0.7827 \cdot \log(\text{IVOL}) + 0.092 \cdot \text{moneyness} - 0.026 \cdot \text{TTM}, \quad R^2 = 0.944$$

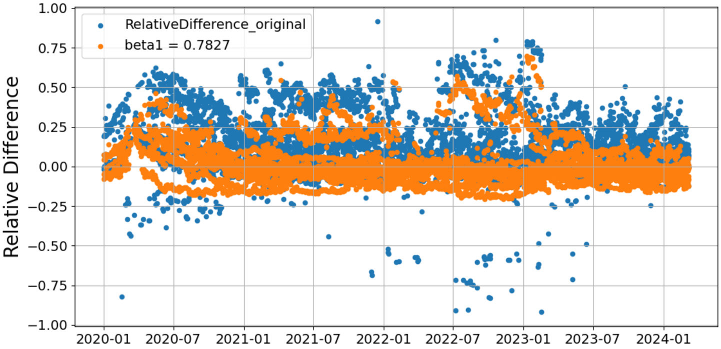
$$\Rightarrow \text{bestIV} = (\text{IVOL})^{0.7827}$$

Houlihan Lokey

TF Model IV Calibration — Regression for A Global Adjustment Factor

Test Results:

- Used bestIV $= (\text{IVOL})^{0.7827}$ to price test dataset.
- Plotted pricing errors under both IVOL and bestIV assumptions.
- Accuracy improved by approx. **25%**.



Crank-Nicolson (CN) + Iterative Methods — Solving the TF Model

- An average of explicit & implicit methods. **Unconditionally stable.** $\mathcal{O}(\Delta t^2 + \Delta S^2)$.
- In the TF model, due to **constraints (conversion, call, put)**, the system becomes **non-linear**.
- Requires iterative solvers to ensure stability and convergence.

Iterative Methods: starts with an initial guess and successively improves it until convergence.

- **PSOR (Projected SOR)** — constraints applied **explicitly**. Iterate untill convergence:
  - Compute Gauss-Seidel step: $gs\_U_i = \frac{rhs_i - \ell_{i-1} \cdot x_{i-1}^{(cur)} - u_i \cdot x_{i+1}^{(old)}}{d_i}$
  - Update: $U_i^{(cur)} \leftarrow (1-\omega)U_i^{(old)} + \omega \cdot gs\_U_i$
- **Penalty Method** — constraints applied **implicitly**:
  - Add penalties to diagonal for $U_i^{(k)} < \max(B_p, \text{conversion})$, and solve:

$$(d_i + P_i)U_i = rhs_i + P_i \cdot U^*$$

  - Iterate until:

$$\max_i \frac{|U_i^{(k+1)} - U_i^{(k)}|}{\max(1, |U_i^{(k+1)}|)} < \text{tol} \quad \text{or} \quad P^k = P^{k+1}$$

CN+PSOR vs CN+Penalty– 3D U Surface

$P_r = 100$, $T = 5$, $\sigma = 0.2$, $r = 0.05$, $d = 0.0$, $r_c = 0.02$, $c_v = 120$, $c_p = 0.08$, $c_{fq} = 2$, $dt = 0.0031$, $dS = 0.0417$, $S_{\text{upper}} = 160$, callable after $t = 2y$ with price 110, putable in $t = 3y$ with price 105.

At **T=0**, **Penalty** gives a **smoother payoff**, more obvious when stock price approaches to **S_upper**. Could because: 1) **Penalty converges better** or 2) some boundary condition errors at S_upper in the PSOR.

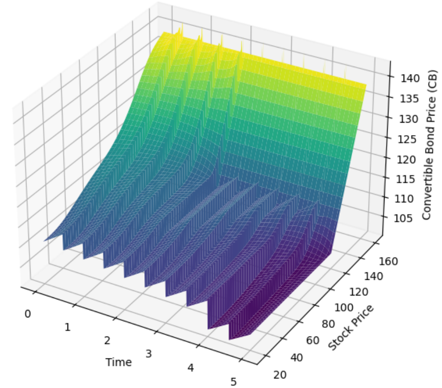


Figure: PSOR

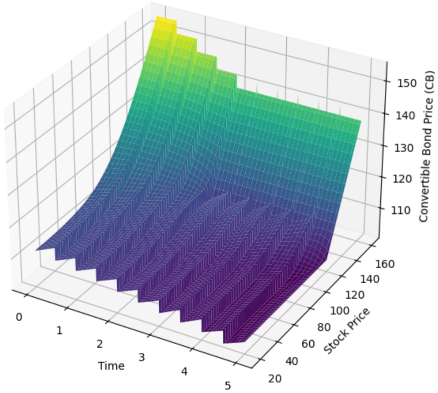


Figure: Penalty

Convergence Analysis — CN + PSOR vs CN + Penalty

- **PSOR is extremely slow and fails to converge.**
- **Penalty method converges instantly — usually within 1 iteration.**
- **Explanation:**
  - Penalty method enables **finite termination**, embedding constraints directly into the system.
  - PSOR applies projections, requiring many iterations without guaranteed convergence.

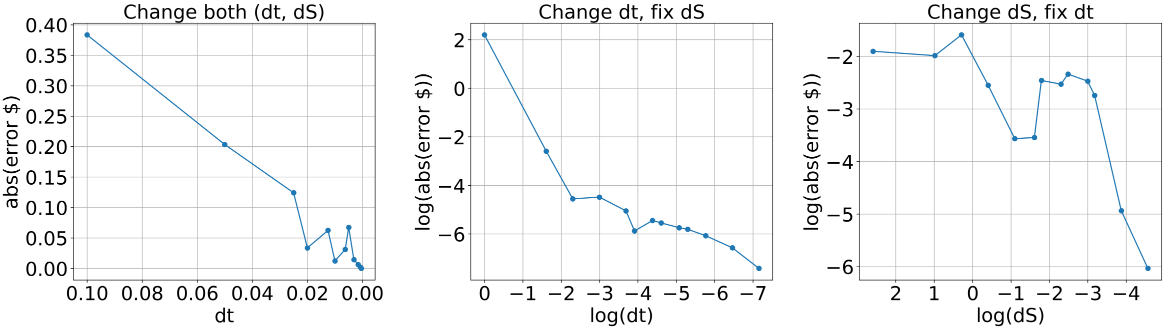
Setup: $S_{\text{upper}} = 160$, $T = 5$. Model price evaluated at $S = 100$.

dt	dS	Price		Price Diff.		Max Iterations		Time	
		PSOR	Penalty	PSOR	Penalty	PSOR	Penalty	PSOR	Penalty
0.05	0.6667	118.15	119.30	–	–	100	1	5.58 s	6.09 s
0.025	0.3333	118.02	119.38	-0.13	0.079	100	1	38.54 s	0.01 s
0.0125	0.1667	117.98	119.44	-0.040	0.062	400	1	2.68 m	0.04 s
0.0063	0.0833	117.95	119.47	-0.034	0.031	600	1	31.8 m	0.25 s
0.0031	0.0417	117.99	119.49	0.046	0.017	600	1	4.72 h	0.99 s
0.0016	0.0208	119.30	119.50	1.309	0.008	600	1	6.63 h	4.12 s
0.00078	0.01042	–	119.50	–	0.004	–	1	–	17.81 s
0.00039	0.00521	–	119.51	–	0.002	–	1	–	74.90 s

Convergence Analysis — Penalty Method Results

Objective: Determine optimal Δt and ΔS to ensure pricing accuracy to generate ML training data. **Steps:**

- ML error tolerance: 1%. Set FD tolerance to 0.01%.
- Reference price: 119.5056 (from $dt = 0.00039$, $dS = 0.00521$).
- With $dt = 0.00078$, $dS = 0.01042$, error is 0.0018\$ (1.5e-3%).
- Fix dS, vary dt: verify convergence rate $\mathcal{O}(\Delta t^2)$.
- Fix dt, vary dS: verify convergence rate $\mathcal{O}(\Delta S^2)$.
- Once verified, we adopt $dt = c \cdot dS$ going forward.



Appendix

Boundary Conditions and Constraints applied for the TF Model

Boundary Conditions at t = T: If not converted U=V=Par+c. If converted: U=conv_val, V=0.

Boundary Conditions at S=0: U=V=discouned (Par+c); S_upper: U=conv_val, V=0.

Backward Induction

- Apply constraints: during **call** period: $U \leq max(B_c, conv\_val), V = 0 \ if \ U \geq B_c$; **put** period:$U \geq B_p, V = B_p \ if \ U \leq B_p$; **conversion**: $U \geq conv\_val, V = 0 \ if \ U \leq conv\_val$
- Add accrued interest at each time step and coupons at coupon dates.

Crank-Nicolson Algorithm in details

- Finite Difference Equation for CB value U (similar for the bond part V):

$$\frac{U_i^{j-1} - U_i^j}{\Delta\tau} = (1-\theta)\left(\frac{\sigma^2 S_i^2 U_{i+1}^j - 2U_i^j + U_{i-1}^j}{\Delta S^2} + rS_i \frac{U_{i+1}^j - U_{i-1}^j}{2\Delta S} - rU_i^j\right)$$

$$+\theta\left(\frac{\sigma^2 S_i^2 U_{i+1}^{j-1} - 2U_i^{j-1} + U_{i-1}^{j-1}}{\Delta S^2} + rS_i \frac{U_{i+1}^{j-1} - U_{i-1}^{j-1}}{2\Delta S} - rU_i^{j-1}\right) - 0.5 * r_c(V_i^{j-1} + V_i^j),\ (0 \le \theta \le 1).$$

- CN: an average of the explicit and implicit methods, $\theta = 1/2$. Derives PDEs for U and V.
- Boundary conditions: S=0: $U = V = \textit{discounted Par Value}$; S_max: $U = \textit{conversion value}, V = 0$
- Solve U and V **($\star$)**: (Matrix Formulation: M_U, M_V, (3.44, 3.45) (in the next slide))

$$(\mathbf{I} - \theta \mathbf{M_V})V^{j-1} = (\mathbf{I} + (1-\theta)\mathbf{M_V})\, V^j$$

$$(\mathbf{I} - \theta \mathbf{M_U})U^{j-1} = (\mathbf{I} + (1-\theta)\mathbf{M_U})\, U^j - 0.5 * r_c \Delta\tau (V^{j-1+V^j})$$

- Unconditionally stable and convergent.
- Second order measure w.r.t. both $\Delta t \textit{ and } \Delta S$. Truncation error is O($\Delta t^2 + \Delta S^2$).

M_U and M_V

$$\alpha_i = \left(\frac{\sigma^2 S_i^2}{2\Delta S^2} - \frac{r S_i}{2\Delta S} \right) \Delta\tau,$$

$$\beta_i = \left(\frac{\sigma^2 S_i^2}{2\Delta S^2} + \frac{r S_i}{2\Delta S} \right) \Delta\tau.$$

$$\mathbf{M_U} = \begin{pmatrix} -r\Delta\tau & 0 & 0 & \cdots & 0 \\ \alpha_1 & -(r\Delta\tau + \alpha_1 + \beta_1) & \beta_1 & \cdots & 0 \\ 0 & \alpha_2 & -(r\Delta\tau + \alpha_2 + \beta_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \beta_{m-1} \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$\mathbf{M_V} = \begin{pmatrix} -(r + r_c)\Delta\tau & 0 & 0 & \cdots & 0 \\ \alpha_1 & -((r + r_c)\Delta\tau + \alpha_1 + \beta_1) & \beta_1 & \cdots & 0 \\ 0 & \alpha_2 & -((r + r_c)\Delta\tau + \alpha_2 + \beta_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \beta_{m-1} \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$