derivx — Precificador de Derivativos por *Building Blocks*

MC/LSMC, PDE Crank–Nicolson, FFT (Heston) e IR Black–76 com DSL Declarativa

Walter C. Neto (repo: https://github.com/walterCNeto/precificador)

September 4, 2025

Abstract

`derivx` é uma biblioteca leve e extensível para precificação de derivativos em Python. A arquitetura separa: (i) *modelo sob risco-neutro* (gerador de trajetórias), (ii) *numerário/curva* (desconto determinístico), (iii) *payoffs* como funcionais de trajetória (PF) componíveis, e (iv) *política de exercício* (Européias/Bermudas/Americanas via LSMC). Uma DSL declarativa (JSON/dict) permite especificar produtos sem escrever código novo. Inclui backends MC (GBM/Heston), PDE 1D (Crank–Nicolson) para vanillas, FFT (Carr–Madan) para Heston europeu, e **módulo de Juros (IR) com fórmulas fechadas Black–76**: ZCB, FRA, Swap (PV e taxa par), Cap/Floor (caplets/floorlets) e Swaption pagador/recebedor. Exemplos, testes e CI acompanham o pacote. Este documento traz teoria, API, guias de uso e exemplos.

Contents

1 Instalação e *quick start*

```
# Windows PowerShell
python -m venv .venv
.\.venv\Scripts\Activate.ps1
pip install -e ".[dev]"
pytest -q
python examples/smoke.py
```

Exemplo mínimo (via DSL).

```python
from derivx import price_from_spec, bs_call_price

spec = {
  "engine": "mc",
  "model": {"name":"gbm","r":0.05,"q":0.0,"sigma":0.2},
  "grid": {"T": 1.0, "steps": 128},
  "S0": [100.0],
  "product": {"style":"european","type":"european_call","asset":0,"K"
     :100.0},
  "n_paths": 80_000, "seed": 42,
}
p, se = price_from_spec(spec)
print("MC:", p, "  ", 1.96*se)
print("BS:", bs_call_price(100, 100, r=0.05, q=0.0, sigma=0.2, T=1.0))
```

2 Arquitetura (visão geral)

- **Curva (numerário).** `PiecewiseFlatCurve` implementa $r(t)$ *piecewise-flat* com desconto $DF(t_0, t_1) = \exp\{-\int_{t_0}^{t_1} r(u)\,du\}$ exato por trechos.

- **Modelos sob $\mathbb{Q}$.** GBM multiativo (`RiskNeutralGBM`) com $q_i(t)$, $\sigma_i(t)$ e correlação; Heston (MC/FFT) para europeias.
- **Funcionais de trajetória (PF).** Utilitários vetorizados: S_{t_k}, máximos/mínimos corridos, médias, barreiras, cestas. Payoffs são funções `paths -> np.ndarray`.
- **Motores numéricos.** *MC*: simulação e estimação $E_{\mathbb{Q}}[X]$ (antitético e CV opcionais). *PDE CN*: 1D para calls/puts; exercício americano via projeção max por passo de tempo. *FFT (Heston)*: Carr–Madan com parâmetros α, N, η.
- **Exercício (LSMC).** `ExerciseSpec` + `lsmc.price`: regressão do valor de continuação em features (por padrão $\log S$, base polinomial até grau 2 + cruzados).
- **IR Black–76 (analítico).** Módulo `ir.black76`: ZCB, FRA, Swap (PV e taxa par), Cap/Floor (soma de caplets/floorlets), Swaption pagador/recebedor (Black–76 sobre a taxa de swap). Integração direta na DSL via `engine:"analytic"`.
- **DSL.** `price_from_spec(spec)` mapeia um dicionário JSON para engine, grade temporal e produto.

3 Base teórica

3.1 Precificação sob $\mathbb{Q}$ e numerário

Seja $B(t) = \exp\{\int_0^t r(u)\,du\}$ o banco. Para payoff X em T:

$$\Pi_0 = \mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B(T)}\right] = DF(0,T)\,\mathbb{E}^{\mathbb{Q}}[X].$$

No MC estimamos $\hat{\Pi}_0 = DF(0,T)\,\overline{X}$ com erro-padrão $\mathrm{SE} = s_X/\sqrt{N}$ e IC 95% como $\pm 1.96\,\mathrm{SE}$.

3.2 GBM multiativo e simulação exata

Para $i = 1, \dots, D$:

$$\frac{dS_i}{S_i} = (r(t) - q_i(t))\,dt + \sigma_i(t)\,dW_i^{\mathbb{Q}}(t), \quad \mathrm{Cov}(dW_i, dW_j) = \rho_{ij}\,dt.$$

No passo Δt:

$$S_{t+\Delta t} = S_t \exp\Big((r - q - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\,Z\Big),$$

com $Z \sim \mathcal{N}(0,\rho)$ via Cholesky. Antitético usa Z e $-Z$.

3.3 LSMC (Longstaff–Schwartz)

Datas de exercício $\mathcal{T} = \{t_{k_j}\}$; payoff imediato $g(S_{t_k})$. Seja V_k o valor ótimo em t_k e τ a *stopping time* ótima. Define-se o valor de continuação:

$$C_k(s) = \mathbb{E}[\,DF(t_k, t_{k+1})\,V_{k+1} \mid S_{t_k} = s].$$

Algoritmo em *backward induction* com regressão (ITM) e regra de parada $g \geq \widehat{C}$.

3.4 PDE 1D Crank–Nicolson (vanillas)

PDE de Black–Scholes:

$$\partial_t V + (r-q)S\,\partial_S V + \tfrac{1}{2}\sigma^2 S^2 \partial_{SS}^2 V - rV = 0.$$

CN $\Rightarrow$ sistema tridiagonal por passo; Americano: projeção $V \leftarrow \max(V, \mathrm{payoff})$.

3.5 FFT (Heston europeu)

Carr–Madan: $C(K) = \frac{e^{-\alpha k}}{\pi} \int_0^\infty \Re(\mathrm{e}^{-iuk}\psi(u))\, du$, $k = \ln K$; discretização uniforme em frequência (η) e FFT de tamanho N.

3.6 Módulo de Juros (IR) — Black–76 e curva *piecewise-flat*

Curva e fatores de desconto. Com $r(t)$ *piecewise-flat*, o fator de desconto $D(0,T) = \exp\{-\int_0^T r(u)\, du\}$ é exato por trechos. Para prazos discretos $0 = T_0 < T_1 < \cdots < T_m$, o *forward simples* sobre $[T_i, T_{i+1}]$ é

$$F_i = \frac{1}{\tau_i}\left(\frac{D(0,T_i)}{D(0,T_{i+1})} - 1\right), \quad \tau_i = T_{i+1} - T_i.$$

ZCB e FRA. Um zero cupom de nominal N com vencimento T: $\mathrm{PV}_0 = N\, D(0,T)$. Um FRA que liquida em T_{i+1} com taxa fixa K tem payoff $N\, \tau_i\, (L_i - K)$ no tempo T_{i+1} e valor hoje

$$\mathrm{PV}_0^{\mathrm{FRA}} = N\, D(0, T_{i+1})\, \tau_i\, (F_i - K).$$

Swap: taxa par e PV. Para pagamentos fixos em $\{T_1, \ldots, T_m\}$ com frações $\{\tau_i\}$, o *anuidade* é $A = \sum_{i=1}^m \tau_i D(0,T_i)$. A taxa *par* é

$$K^\star = \frac{D(0,T_0) - D(0,T_m)}{A}.$$

O PV de um *payer swap* (paga fixo K, recebe flutuante) é

$$\mathrm{PV}_0^{\mathrm{swap}} = N\Big(D(0,T_0) - D(0,T_m) - K\, A\Big).$$

Cap/Floor (Black–76). Um *caplet* sobre $[T_i, T_{i+1}]$ tem payoff $N\, \tau_i \max(L_i - K, 0)$ em T_{i+1}. No modelo Black–76 com volatilidade σ sobre F_i até o tempo de *reset* T_i,

$$\mathrm{PV}_0^{\mathrm{caplet}} = N\, D(0, T_{i+1})\, \tau_i \big(F_i \Phi(d_1) - K\Phi(d_2)\big), \quad d_{1,2} = \frac{\ln(F_i/K) \pm \frac{1}{2}\sigma^2 T_i}{\sigma\sqrt{T_i}}.$$

Um *cap* é a soma dos caplets; *floor* é análogo com K e F_i trocados de sinal.

Swaption (Black–76). Para uma swaption *pagadora* com exercício em T_0 sobre o swap que paga fixo K nos tempos $\{T_1, \ldots, T_m\}$, define-se o *anuidade* $A = \sum_{i=1}^m \tau_i D(0,T_i)$ e a taxa de *swap a termo* $F_{\mathrm{swap}} = \frac{D(0,T_0) - D(0,T_m)}{A}$. Sob Black–76 com volat. σ da taxa de swap até T_0,

$$\mathrm{PV}_0^{\text{payer swpt}} = N\, A\big(F_{\mathrm{swap}} \Phi(d_1) - K\Phi(d_2)\big), \quad d_{1,2} = \frac{\ln(F_{\mathrm{swap}}/K) \pm \frac{1}{2}\sigma^2 T_0}{\sigma\sqrt{T_0}}.$$

A *receiver* troca F_{swap} e K.

4 Como usar a biblioteca

4.1 Camada 1 — API direta (curvas, modelos, motores)

```python
import numpy as np
from derivx import PiecewiseFlatCurve, RiskNeutralGBM, MonteCarloEngine

# Curva "piecewise-flat": 5% a.a.
r_curve = PiecewiseFlatCurve(np.array([1e-8]), np.array([0.05]))

# Modelo GBM monofator
model = RiskNeutralGBM(r_curve, q_funcs=[0.0], sigma_funcs=[0.2], corr=
    None)
eng = MonteCarloEngine(model)

# Grade temporal e S0
times = np.linspace(0.0, 1.0, 128+1)
S0 = [100.0]

# Payoff europeu: call
from derivx import PF, relu
payoff = lambda paths: relu(PF.terminal(paths, asset=0) - 100.0)

price, se = eng.price(payoff, S0, times, n_paths=80_000, seed=42)
print(price, "  ", 1.96*se)
```

Exercício via LSMC.

```python
from derivx import ExerciseSpec

def imm_put(paths, k):
    St = PF.at_time(paths, asset=0, idx=k)
    return relu(100.0 - St)

exercise_idx = [16, 32, 48, 64]  # inclui vencimento
spec = ExerciseSpec(exercise_idx=exercise_idx, immediate_payoff=imm_put
    )

price, se = eng.price_exercisable(spec, S0, times, n_paths=150_000,
    seed=7)
```

4.2 Camada 2 — DSL declarativa (price_from_spec)

```python
from derivx import price_from_spec

spec = {
  "engine":"mc",  # "mc" | "pde" | "fft" | "analytic" | "auto"
  "model":{"name":"gbm","r":0.05,"q":0.0,"sigma":0.2},
  "grid":{"T":1.0,"steps":128},   # PDE/FFT usam s  T
  "S0":[100.0],
  "product":{"style":"european","type":"european_call","asset":0,"K"
    :100.0},
  "n_paths":80_000,"seed":7
}
price, se = price_from_spec(spec)
```

IR via DSL (analítico)

```python
# (IR1) ZCB
spec_zcb = {
  "engine":"analytic",
  "model":{"r_curve":{"times":[0.0, 1.0, 2.0], "rates":[0.05, 0.05,
     0.05]}},
  "grid":{"T":2.0},
  "S0":[],
  "product":{"type":"zcb","notional":1.0,"T":2.0}
}

# (IR2) FRA (T1->T2)
spec_fra = {
  "engine":"analytic",
  "model":{"r_curve":{"times":[0.0,1.0,2.0],"rates":[0.05,0.05,0.05]}},
  "product":{"type":"fra","T1":1.0,"T2":2.0,"tau":1.0,"K":0.05,"
     notional":1e6}
}

# (IR3) Swap PV e taxa par
spec_swap = {
  "engine":"analytic",
  "model":{"r_curve":{"times":[0,1,2,3,4,5],"rates":[0.05]*5}},
  "product":{"type":"swap","T0":0.0,"payment_times":[1,2,3,4,5],
             "tau":1.0,"fixed_rate":0.052,"notional":1e6}
}
# Para taxa par, defina "par":true e omita "fixed_rate" => PV ~ 0.

# (IR4) Cap (soma de caplets Black-76)
spec_cap = {
  "engine":"analytic",
  "model":{"r_curve":{"times":[0,1,2,3],"rates":[0.05,0.05,0.05]},
           "sigma":0.20},  # Black vol
  "product":{"type":"cap","payment_times":[1,2,3],"tau":1.0,"K":0.05,
             "reset_times":[0,1,2], "notional":1e6}
}

# (IR5) Swaption pagadora Black-76
spec_swpt = {
  "engine":"analytic",
  "model":{"r_curve":{"times":[0,1,2,3,4,5],"rates":[0.05]*5},"sigma"
     :0.25},
  "product":{"type":"payer_swaption","expiry":2.0,"payment_times"
     :[3,4,5],
             "tau":1.0,"K":0.05,"notional":1e6}
}
```

4.3 Precisão & Performance: *knobs* práticos

Backend	Parâmetro	Efeito prático
MC	`n_paths`	SE $\propto 1/\sqrt{N}$; mais paths $\Rightarrow$ erro menor (custo linear).
MC	`steps`	Menor bias temporal em path-dep./LSMC; custo $\propto$ paths $\times$ steps.
PDE	`NS, NT`	Convergência $\mathcal{O}(\Delta t + \Delta S^2)$; aumente malha até estabilizar.
PDE	`Smax_mult`	Domínio $[0, S_{\max}]$; use 5–7 vezes o strike inicial.
FFT	α, N, η	Damping e resolução em frequência/strike (ver seção Heston).
IR	`sigma`	Vol Black para cap/floor/swaption. Maturidade usada no $d_{1,2}$ é o tempo até reset/expiração.

5 Validação, paridades e reprodutibilidade

- **Consistência BS**: MC (GBM) $\approx$ Black–Scholes (dentro de $k \cdot \text{SE}$).
- **PDE vs BS**: put europeu CN $\approx$ BS; americano PDE $\geq$ europeu.
- **Propriedades path-dep.**: up&out $\leq$ vanilla; Asiática $\leq$ vanilla (mesma K).
- **LSMC**: Bermudas com janelas mais densas $\nearrow$ Americana.
- **Heston**: MC $\approx$ FFT (tolerância baseada em SE).
- **IR (paridades)**:
  - *Cap–Floor parity*: $\text{Cap}(K) - \text{Floor}(K) = \text{PV}(\text{payer swap com taxa } K)$.
  - *Swap par*: para $K = K^\star$, $\text{PV}^{\text{swap}} \approx 0$.
  - *FRA*: usando F_i da curva, $\text{PV}^{\text{FRA}} = N D(0, T_{i+1}) \tau_i (F_i - K)$.

Execução dos testes.

```
# Windows
$env:PYTEST_DISABLE_PLUGIN_AUTOLOAD="1"
python -m pytest -q
```

6 Exemplos práticos (fim-a-fim)

6.1 (E1) Europeu: MC vs Black–Scholes

```python
from derivx import price_from_spec, bs_call_price
S0=K=100.0; r=0.05; q=0.0; sigma=0.2; T=1.0
spec = {"engine":"mc","model":{"name":"gbm","r":r,"q":q,"sigma":sigma},
        "grid":{"T":T,"steps":128},"S0":[S0],
        "product":{"style":"european","type":"european_call","asset":0,
            "K":K},
        "n_paths":80_000,"seed":42}
pmc,se = price_from_spec(spec)
pbs = bs_call_price(S0,K,r,q,sigma,T)
print(f"MC={pmc:.4f}    {1.96*se:.4f} | BS={pbs:.4f}")
```

6.2 (E2) PDE: put europeu vs americano

```
from derivx import price_from_spec
euro={"engine":"pde","model":{"name":"gbm","r":0.05,"q":0.0,"sigma"
    :0.2},
      "grid":{"T":1.0},"S0":[100.0],
      "product":{"style":"european","type":"european_put","asset":0,"K"
         :100.0},
      "NS":800,"NT":800,"Smax_mult":5.0}
amer={**euro,"product":{"style":"american","type":"european_put","asset
    ":0,"K":100.0}}
pe,_ = price_from_spec(euro); pa,_ = price_from_spec(amer)
print(f"PDE euro={pe:.4f}  PDE amer={pa:.4f}  (amer >= euro)")
```

6.3 (E3) Barreira up&out e monotonia

```
base={"engine":"mc","model":{"name":"gbm","r":0.05,"q":0.0,"sigma"
    :0.2},
      "grid":{"T":1.0,"steps":128},"S0":[100.0],"n_paths":100_000,"seed
         ":7}
van,_  = price_from_spec({**base,"product":{"style":"european","type":"
    european_call","asset":0,"K":100.0}})
uo130,_= price_from_spec({**base,"product":{"style":"european","type":"
    up_and_out_call","asset":0,"K":100.0,"barrier":130.0}})
uo140,_= price_from_spec({**base,"product":{"style":"european","type":"
    up_and_out_call","asset":0,"K":100.0,"barrier":140.0}})
print(f"UO130={uo130:.4f} <= UO140={uo140:.4f} <= Vanilla={van:.4f}")
```

6.4 (E4) Asiática aritmética $\leq$ vanilla

```
asian,_= price_from_spec({**base,"product":{"style":"european","type":"
    asian_arith_call","asset":0,"K":100.0}})
print(f"Asian={asian:.4f} <= Vanilla={van:.4f}")
```

6.5 (E5) Bermudas (LSMC): frequência de exercício

```
def berm(ex_every):
  spec={"engine":"mc","model":{"name":"gbm","r":0.05,"q":0.0,"sigma"
      :0.2},
        "grid":{"T":1.0,"steps":256},"S0":[100.0],
        "product":{"style":"bermudan","type":"european_put","asset":0,"
           K":100.0,"exercise_every":ex_every},
        "n_paths":120_000,"seed":7}
  return price_from_spec(spec)
for ex in (8,16,32):
  p,se = berm(ex)
  print(f"ex_every={ex:>2}: {p:.4f}    {1.96*se:.4f}")
```

6.6 (E6) Heston: FFT vs MC

```
common={"name":"heston","r":0.05,"q":0.0,"kappa":1.5,"theta":0.04,"xi"
    :0.5,"rho":-0.7,"v0":0.04}
fft={"engine":"fft","model":common,"grid":{"T":1.0},"S0":[100.0],
```

```
     "product":{"style":"european","type":"european_call","asset":0,"K"
        :100.0},
     "alpha":1.5,"N":4096,"eta":0.25}
mc={"engine":"mc","model":common,"grid":{"T":1.0,"steps":512},"S0"
   :[100.0],
    "product":{"style":"european","type":"european_call","asset":0,"K"
       :100.0},
    "n_paths":200_000,"seed":7}
p_fft,_=price_from_spec(fft); p_mc,se=price_from_spec(mc)
print(f"FFT={p_fft:.4f} | MC={p_mc:.4f}    {1.96*se:.4f}")
```

6.7 (E7) Basket 2D (GBM correlacionado)

```
spec={"engine":"mc",
      "model":{"name":"gbm","r":0.05,"q":[0.01,0.03],"sigma"
         :[0.20,0.30],
               "corr":[[1.0,0.5],[0.5,1.0]]},
      "grid":{"T":2.0,"steps":80},"S0":[100.0,120.0],
      "product":{"style":"european","type":"basket_call","weights"
         :[0.5,0.5],"K":110.0},
      "n_paths":100_000,"seed":3}
p,se=price_from_spec(spec)
print(p, "  ", 1.96*se)
```

6.8 (E8) IR: paridade Cap–Floor e taxa par de swap

```
from derivx import price_from_spec
r_curve = {"times":[0,1,2,3,4,5], "rates":[0.05]*5}
base = {"engine":"analytic","model":{"r_curve":r_curve}}

cap = {**base, "product":{"type":"cap","payment_times":[1,2,3,4,5], "
   tau":1.0,
                          "reset_times":[0,1,2,3,4], "K":0.05, "
                             notional":1e6},
       "model":{**base["model"], "sigma":0.20}}
floor= {**base, "product":{"type":"floor","payment_times":[1,2,3,4,5],
   "tau":1.0,
                          "reset_times":[0,1,2,3,4], "K":0.05, "
                             notional":1e6},
        "model":{**base["model"], "sigma":0.20}}
swap = {**base, "product":{"type":"swap","T0":0.0,"payment_times"
   :[1,2,3,4,5],
                          "tau":1.0, "fixed_rate":0.05, "notional":1e6
                             }}

pc,_ = price_from_spec(cap)
pf,_ = price_from_spec(floor)
ps,_ = price_from_spec(swap)
print("Cap-Floor ~ Swap PV:", pc - pf, "~", ps)

swap_par = {**base, "product":{"type":"swap","T0":0.0,"payment_times"
   :[1,2,3,4,5],
                              "par":True, "tau":1.0, "notional":1e6}}
p_par,_ = price_from_spec(swap_par)
print("Swap par PV ~ 0:", p_par)
```

7 Estrutura do repositório (resumo)

```
src/derivx/
  curves.py           # PiecewiseFlatCurve (df, ints)
  models/gbm.py       # RiskNeutralGBM (paths; df via curva)
  engine/montecarlo.py
  engine/pde.py       # CN 1D vanillas (euro/amer)
  engine/fft.py       # Heston FFT (Carr-Madan)
  exercise/lsmc.py    # LSMC genérico
  payoffs/core.py     # PF utilitários (terminal, média, etc.)
  payoffs/extra.py    # Digitais/Gap/Exchange (PF)
  ir/black76.py       # ZCB, FRA, Swap, Cap/Floor, Swaption (Black-76)
  analytic/bs.py      # , d1d2, BS
  analytic/haug.py    # Digitais, Gap, Margrabe (closed-form)
  dsl/spec.py         # mapeia dict -> engine/produto
tests/                # sanity, propriedades e referências (BS/Haug/IR)
examples/             # scripts reprodutíveis
```

8 Licença e aviso

MIT. Uso acadêmico/educacional; valide premissas, calibração e risco de modelo antes de produção. Notas: módulo IR assume *single-curve* (mesma curva para desconto e forwards) e convenções simples (τ como ano de ACT/360=1.0, etc.). Agendas e day-count mais realistas podem ser conectadas externamente; multi-curve e convexity adjustments não estão incluídos.

Referências

Black, F.; Scholes, M. (1973) *The Pricing of Options and Corporate Liabilities.*
Black, F. (1976) *The Pricing of Commodity Contracts.*
Brigo, D.; Mercurio, F. (2006) *Interest Rate Models – Theory and Practice.*
Hull, J. (2018) *Options, Futures, and Other Derivatives.*
Longstaff, F.; Schwartz, E. (2001) *Valuing American Options by Simulation.*
Carr, P.; Madan, D. (1999) *Option Valuation Using the FFT.*
Heston, S. (1993) *A Closed-Form Solution for Options with Stochastic Volatility.*
Margrabe, W. (1978) *The Value of an Option to Exchange One Asset for Another.*
Haug, E. G. (2006) *The Complete Guide to Option Pricing Formulas.*