Documento Principal do Projeto
Modelo de Dois Fatores de Schwartz–Smith, Kalman (como ferramenta), Inferência do Forward e Estratégia de Trading

1 Teoria do Modelo de Dois Fatores (Schwartz–Smith)

Princípios e ideias centrais

Modelamos o **log-preço spot** S_t de uma commodity como a soma de dois fatores estocásticos:

$$\ln S_t = X_t + Y_t,$$

onde:

- X_t é o **componente de curto prazo** (desvio transitório com *reversão à média* para 0);
- Y_t é o **componente de longo prazo** (nível de equilíbrio, um passeio aleatório com deriva).

Assim, o preço *observado* pode estar temporariamente acima/abaixo do nível de equilíbrio (X_t), enquanto o próprio equilíbrio (Y_t) pode se deslocar ao longo do tempo.

Equações diferenciais estocásticas (SDEs)

A dinâmica contínua é:

$$dX_t = -\kappa\, X_t\, dt + \sigma_X\, dW_t^X, \tag{1}$$

$$dY_t = \mu\, dt + \sigma_Y\, dW_t^Y, \tag{2}$$

$$S_t = \exp\big(X_t + Y_t\big), \tag{3}$$

com dW_t^X e dW_t^Y movimentos Brownianos (possivelmente correlacionados, $\mathrm{Corr}(dW_t^X, dW_t^Y) = \rho$). Os parâmetros são:

- $\kappa > 0$: velocidade de reversão de X_t para 0;
- $\sigma_X, \sigma_Y > 0$: volatilidades dos fatores curto e longo;
- μ: deriva do fator de longo prazo Y_t;
- $\rho \in [-1, 1]$: correlação instantânea entre choques de X e Y.

Soluções condicionais e estatísticas úteis

Para X_t (processo de Ornstein–Uhlenbeck):

$$X_T = X_t e^{-\kappa(T-t)} + \sigma_X \int_t^T e^{-\kappa(T-s)}\, dW_s^X,$$

logo, com $\Delta := T - t$,

$$\mathbb{E}[X_T \mid X_t] = X_t e^{-\kappa\Delta}, \qquad \mathrm{Var}[X_T \mid X_t] = \frac{\sigma_X^2}{2\kappa}\big(1 - e^{-2\kappa\Delta}\big).$$

Para Y_t (Browniano aritmético com deriva):

$$Y_T = Y_t + \mu\,\Delta + \sigma_Y(W_T^Y - W_t^Y), \quad \mathbb{E}[Y_T \mid Y_t] = Y_t + \mu\Delta, \quad \mathrm{Var}[Y_T \mid Y_t] = \sigma_Y^2\Delta.$$

A covariância condicional é

$$\mathrm{Cov}[X_T, Y_T \mid X_t, Y_t] = \rho\,\sigma_X\sigma_Y\,\frac{1 - e^{-\kappa\Delta}}{\kappa}.$$

Nota rápida sobre a notação condicional. Expressões como $\mathbb{E}[X_T \mid X_t]$ e $\mathrm{Var}[X_T \mid X_t]$ significam, respectivamente, **média** e **variância** de X_T *condicionadas* ao valor conhecido de X_t no tempo t. Lê-se "a expectativa de X_T dado (sabendo) X_t". É uma forma padrão de explicitar que estamos tratando a incerteza futura levando em conta a *informação disponível* no presente.

Forward $F(t,T)$ (expressão fechada)

Sob medida neutra ao risco,

$$F(t,T) = \mathbb{E}[S_T \mid \mathcal{F}_t] = \mathbb{E}\big[\exp(X_T + Y_T) \mid X_t, Y_t\big].$$

Como $U := X_T + Y_T$ é normal condicionalmente a (X_t, Y_t), vale $\mathbb{E}[e^U] = \exp\big(\mathbb{E}[U] + \frac{1}{2}\mathrm{Var}(U)\big)$. Combinando as médias/variâncias acima (com $\Delta = T - t$),

$$\boxed{F(t,T) = \exp\Big(X_t e^{-\kappa\Delta} + Y_t + \mu\Delta + \tfrac{1}{2}\Big[\tfrac{\sigma_X^2}{2\kappa}\big(1 - e^{-2\kappa\Delta}\big) + \sigma_Y^2\Delta + 2\rho\,\sigma_X\sigma_Y\,\tfrac{1-e^{-\kappa\Delta}}{\kappa}\Big]\Big)}.$$

Para prazos longos (Δ grande), o termo $e^{-\kappa\Delta}$ se anula e o forward reflete sobretudo Y_t e a variância/deriva de longo prazo.

2 Filtro de Kalman (visão-ferramenta)

Uso conceitual

Veremos o KF como uma **ferramenta** para:

- estimar *estados latentes* $\{\hat{X}_t, \hat{Y}_t\}$ recursivamente;

- ajustar *parâmetros* $\Theta = \{\kappa, \sigma_X, \sigma_Y, \rho, \mu\}$ via MLE ou EM.

A modelagem é linear-gaussiana em espaço de estados: transição para $[X\ Y]^\top$ a cada Δt e observações (spot e/ou $\ln F_{\text{mkt}}(t, T_i)$) lineares nos estados. A seção de *fluxograma* a seguir assume o uso de uma **biblioteca Python de espaço de estados** com KF e estimação por **MLE** (p. ex., `statsmodels.tsa.statespace`) e/ou **EM** (p. ex., `pykalman`) como *bloco fechado*, encapsulada em um `KalmanEngine`.

3 Fluxograma — Inferência de $F(t^*, T)$ via `KalmanEngine` (bloco fechado)

O que esta ferramenta faz

Dado histórico até um instante t^* e um conjunto de maturidades $\{T_i\}_{i=1}^{M}$, a `KalmanEngine`:

1. recebe **arrays** padronizados (preparados) de observações e prazos;
2. ajusta Θ por *MLE* (ou *EM*);
3. filtra/suaviza estados e retorna $(\hat{X}_{t^*}, \hat{Y}_{t^*})$;
4. calcula o vetor $F_{\text{modelo}}(t^*, T_i)$ usando a expressão fechada.

Especificação dos *inputs* (formatos)

Considere $t = 0, 1, \ldots, T-1$ como datas úteis (ordenadas) e $i = 1, \ldots, M$ como *tenores* ou *maturidades* fixas.

- `time`: vetor de datas (tamanho T).
- `F_mkt`: matriz $T \times M$ com os **settlements** (preços a termo) alinhados por *tenor* (não por código de contrato). Cada coluna é um tenor, cada linha é uma data.
- `ttm`: matriz $T \times M$ com $\Delta_{t,i}$ = time-to-maturity em anos do tenor i na data t.
- `S` (opcional): vetor $T \times 1$ com spot (ou proxy), se disponível.
- `mask` (opcional): matriz booleana $T \times M$ indicando *faltantes* em `F_mkt`.
- `cfg`: dicionário de configuração com:
  - `method` $\in$ {`"MLE"`, `"EM"`};
  - chutes iniciais de Θ (opcional);
  - `R` (ruído de medição): escalar ou matriz diagonal por observável (pode ser *auto-estimado* pela engine).

Tamanhos: `F_mkt` e `ttm` são $[T, M]$; `S` é $[T]$; t^* é um índice ou data dentro de `time`.

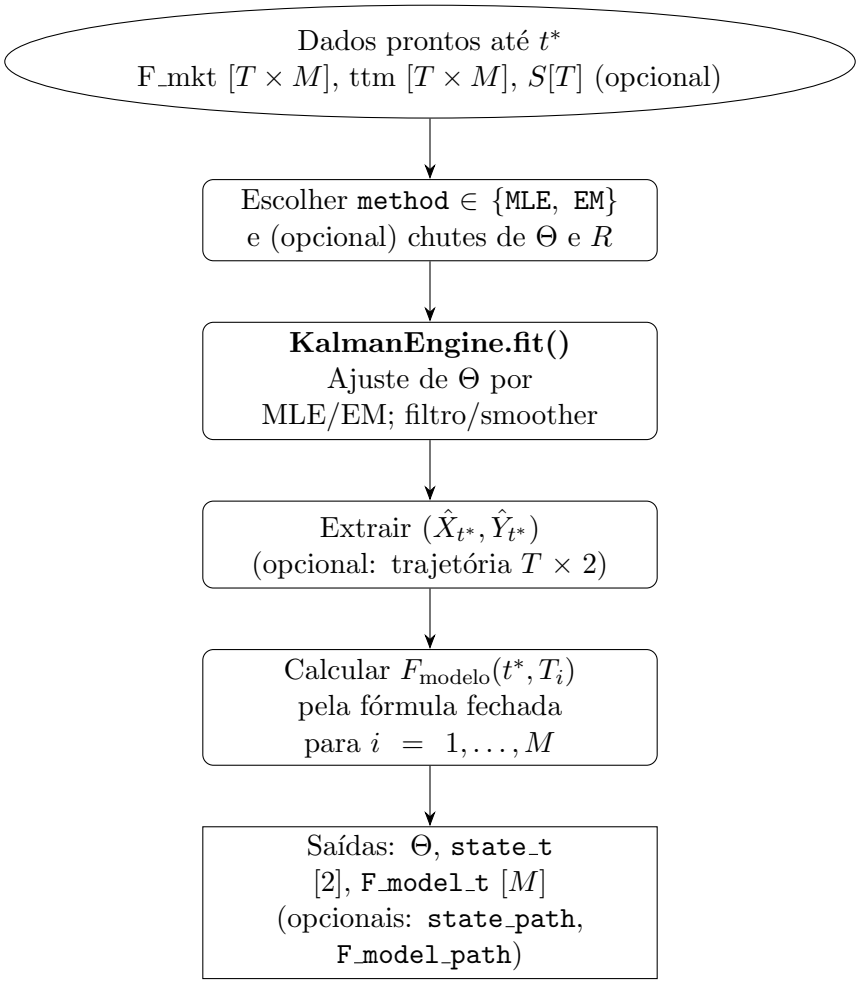
Especificação dos *outputs* (formatos)

- `Theta`: tupla/dict com $\{\kappa, \sigma_X, \sigma_Y, \rho, \mu\}$.
- `state_t`: vetor [2] com $(\hat{X}_{t^*}, \hat{Y}_{t^*})$.
- `state_path` (opcional): matriz $T \times 2$ com a trajetória filtrada/suavizada.
- `F_model_t`: vetor $[M]$ com $F_{\text{modelo}}(t^*, T_i)$ para todos os tenores carregados.
- `F_model_path` (opcional): matriz $T \times M$ com $F_{\text{modelo}}(t, T_i)$ ao longo do histórico.

Pré-processamento mínimo antes da `KalmanEngine`

1. **Mapeamento de tenores**: para cada data, mapear contratos listados para um *grid fixo* de tenores $\{T_i\}$; preencher `F_mkt`[t,i] e `ttm`[t,i].
2. **Limpeza**: remover valores inválidos; manter `mask` para faltantes.
3. **Coerência**: `ttm` positiva e decrescente com o calendário; usar convenção ACT/365 (ou outra *fixa*).

Fluxograma (engine como caixa-preta)



4 Estratégia de Trading (como função/caixa-preta)

Ideia operacional

Transformar o desvio entre o forward do *modelo* e o de *mercado* em **ordens explícitas** para uso direto em backtesting:

$$\Delta(t, T_i) := F_{\text{modelo}}(t, T_i) - F_{\text{mkt}}(t, T_i).$$

Padronizamos por risco a termo (ou erro-padrão do modelo) para obter *z-scores*. A função aplica **limiares de entrada/saída** (banda de inação), dimensiona posições com *vol-target* ou *QP mean–variance* e retorna **ordens** bem definidas.

Especificação dos *inputs* (formatos)

Considere um único tempo t (aplicável *rolling* ao longo do histórico):

- `F_model_t`: vetor $[M]$ com $F_{\text{modelo}}(t, T_i)$.

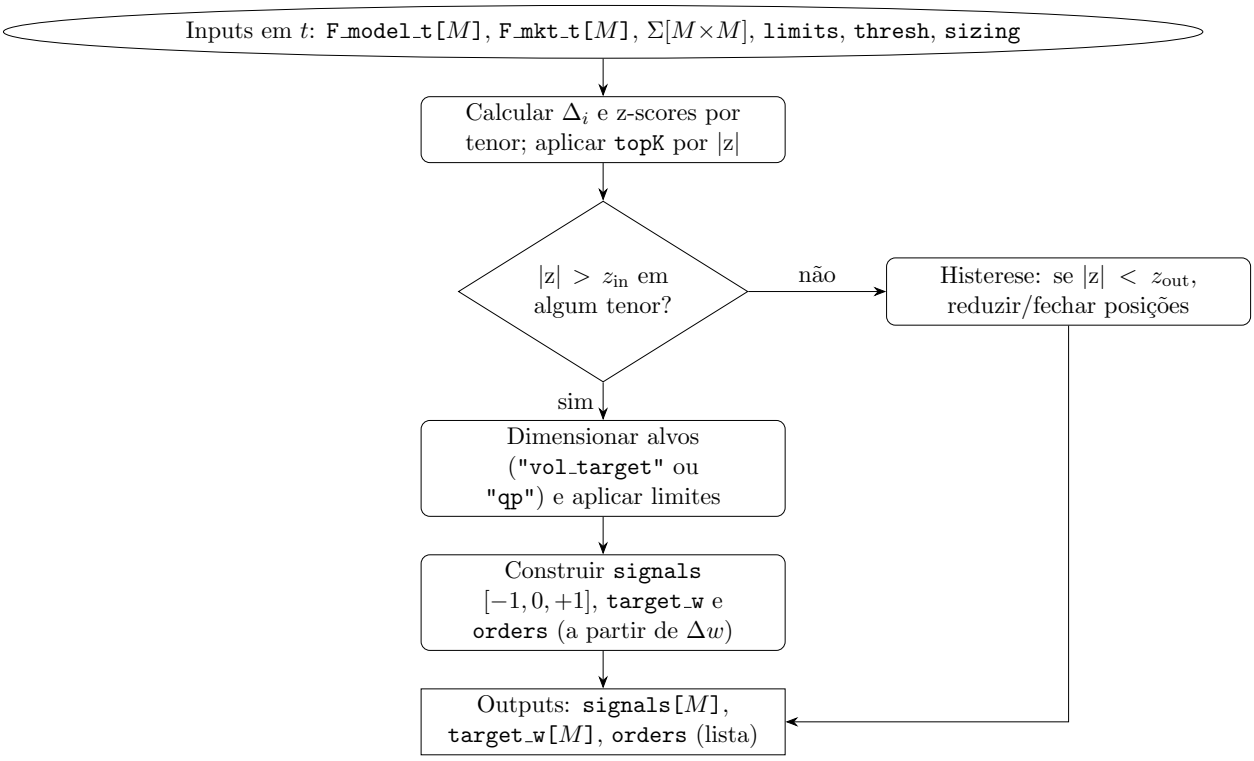
- `F_mkt_t`: vetor $[M]$ com $F_{\text{mkt}}(t, T_i)$ (mesmos tenores).
- `risk`: matriz Σ $[M \times M]$ de covariâncias dos retornos dos futuros (estimada por janela).
- `limits`: parâmetros operacionais (alavancagem, limites por tenor, *position caps*).
- `thresh`: limiares z_{in} e z_{out} (histerese).
- `sizing`: { `"vol_target"` com `vol_target`, `"qp"` com $\gamma, \lambda_1, \lambda_2$ }.
- `topK` (opcional): número máximo de tenores a operar por decisão (seleção pelos maiores |z-score|).
- `w_prev` (opcional): vetor $[M]$ com posições vigentes (para custos de troca no caso `"qp"`).

Especificação dos *outputs* (formatos)

- `signals`: vetor $[M]$ em $\{-1, 0, +1\}$ (vender, nada, comprar) por tenor.
- `target_w`: vetor $[M]$ com posições-alvo normalizadas (após sizing e limites).
- `orders`: *lista* de estruturas do tipo `(tenor_id, side, size)` para execução no back-test. Pode ser derivada de $\Delta w =$ `target_w` $-$ `w_prev`.

Observação: ao rodar *rolling*, agregamos `signals` em uma matriz $[T \times M]$ e `orders` em uma lista de listas (uma por tempo).

Fluxograma — `TradeEngine` (caixa-preta)



Comportamento esperado do output. Para cada t, `orders` é uma lista de instruções claras (tenor, lado, tamanho). Se for preferível ao backtester, também é possível consumir diretamente `target_w` (posições alvo) ou `signals` (sinais discretos).

5 Arquitetura do Código — Fluxograma Geral (módulos e I/O)

Propósito. Esta seção organiza todo o pipeline em **blocos modulares** com contratos claros de *entrada/saída* (I/O), para que o `main.py` possa orquestrar o processo sem expor detalhes matemáticos internos. A ideia é plugar três caixas–pretas: (i) inferência do forward teórico do modelo, (ii) preparação dos insumos de risco/limites para trading e (iii) motor de decisão (Trade Engine) que retorna ordens.

Entradas e saídas (nível do programa)

- **Entradas globais (dados no tempo $1{:}T$):**
  - `F_mkt` $[T \times M]$: matriz de preços de futuros observados, uma linha por data, uma coluna por maturidade.
  - `ttm` $[T \times M]$: time-to-maturity (em anos) correspondente a cada célula de `F_mkt`.
  - `S` $[T]$ (opcional): spot histórico (se disponível).
  - `cost` $[M]$: custo por contrato (tick/fee) para execução.

  - `cfg`: configurações (janelas de cálculo, método de ajuste `MLE/EM`, limites padrão etc.).

- **Saída final (no instante t^* selecionado):**
  - `orders_t`: lista estruturada de ordens no tempo t^* (por exemplo, tuplas `(maturity_idx, side, qty)`), já respeitando custos, limites e restrições.

Blocos (contratos I/O)

`ComputeModelForward()`. *Função caixa–preta que usa a engine de Kalman (MLE/EM) + fórmula fechada do modelo.*

- **Input:** `F_mkt` $[T \times M]$, `ttm` $[T \times M]$, `S` $[T]$ (opc.), `cfg.method`$\in \{$`MLE`, `EM`$\}$, chutes iniciais (opcionais) de Θ.
- **Output (no t^*):**
  - `Theta`: parâmetros calibrados do modelo.
  - `state_t` [2]: estado filtrado $(\hat{X}_{t^*}, \hat{Y}_{t^*})$.
  - `F_model_t` $[M]$: vetor do forward teórico $F_{\text{modelo}}(t^*, T_i)$, $i = 1{:}M$.
  - (*opcional*) `state_path` $[T \times 2]$, `F_model_path` $[T \times M]$.

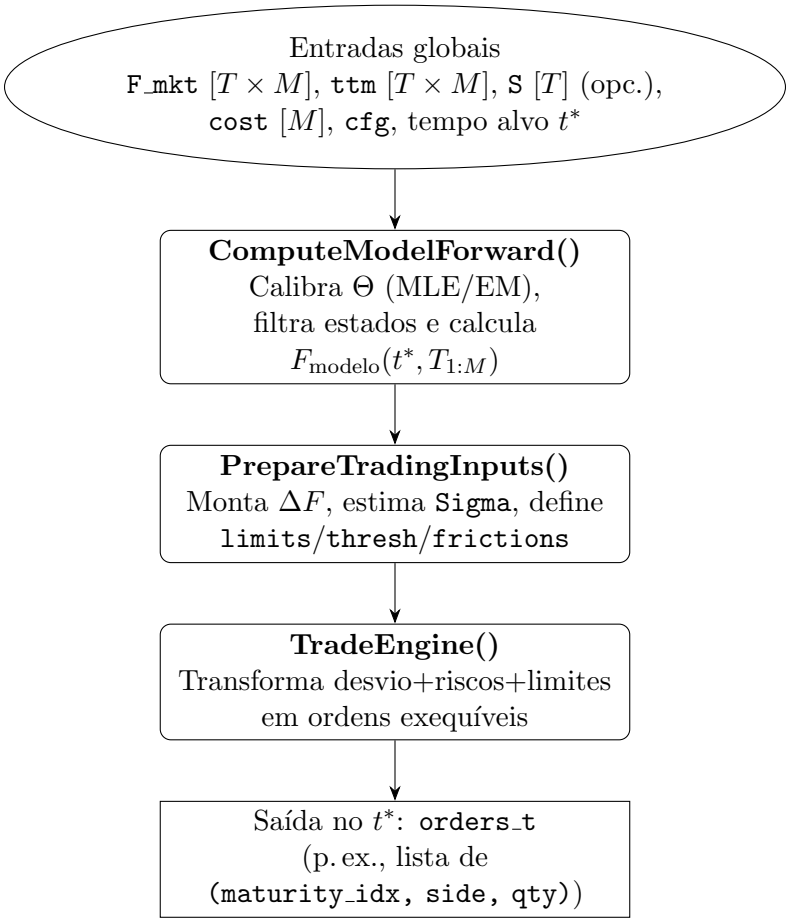
`PrepareTradingInputs()`. *Pré-processa insumos para o motor de decisão.*

- **Input (no t^*):** `F_mkt_t` $[M]$ (linha t^* de `F_mkt`), `F_model_t` $[M]$, `ttm_t` $[M]$, `cost` $[M]$, `cfg`.
- **Output:**
  - `mispricing` $[M]$: vetor $\Delta F = F_{\text{modelo}}(t^*, \cdot) - F_{\text{mkt}}(t^*, \cdot)$.
  - `Sigma` $[M \times M]$: matriz de risco/covariância (estimada via janela rolling de retornos de futuros ou do erro de modelo).
  - `limits` $[M]$: limites por maturidade (posição máxima, notional, VaR aproximado etc.).
  - `thresh` $[M]$: limiares mínimos de desvio (em preço ou em Z-score) para acionar ordens.
  - `frictions`: estrutura com custos efetivos (tick/fee/slippage) agregados.

`TradeEngine()`. *Gera decisões táticas a partir de desvio/risco/limites.*

- **Input (no t^*):** `mispricing` $[M]$, `Sigma` $[M \times M]$, `limits` $[M]$, `thresh` $[M]$, `frictions`.
- **Output:** `orders_t`: lista de ordens (por maturidade) com `side` $\in \{$`BUY`, `SELL`$\}$ e `qty` viável (respeitando risco/limites/custos).

Fluxograma (visão fim-a-fim)



Observações práticas

- O `main.py` apenas define o t^*, carrega dados e chama `ComputeModelForward` → `PrepareTradingInputs` → `TradeEngine`, persistindo `orders_t`.
- A matriz `Sigma` pode ser calculada com uma janela rolling (ex.: 60 dias) de retornos das séries de futuros por maturidade ou a partir dos resíduos `F_mkt` − `F_model` para refletir risco de erro do modelo.
- `limits` e `thresh` podem vir de `cfg` (valores fixos) ou ser derivados de métricas (p. ex., VaR aproximado com `Sigma`, P&L por tick com `cost`).
- `frictions` agrega custos fixos/lineares (fee, spread médio, slippage) a serem comparados ao benefício esperado do desvio.

6 Dados e API (CME Group) — o que coletar e como usar

Universo e campos

- **Universo**: escolha uma família (ex.: WTI, Henry Hub, etc.) e um conjunto fixo de tenores $\{T_i\}_{i=1}^{M}$ a acompanhar.
- **Campos por data/tenor**: `settlement_price`, `expiration_date`, `trade_date`. Organizar por *tenor* (não por símbolo específico).

Arranjos para as engines

- `F_mkt[`$T\times M$`]` e `ttm[`$T\times M$`]` alimentam `KalmanEngine`.
- Saídas `F_model_t[`M`]` e `F_mkt_t[`M`]` alimentam `TradeEngine`.

Referências (links por conceito)

Modelo de dois fatores (paper original)
`https://www.anderson.ucla.edu/faculty/eduardo.schwartz/articles/72.pdf`

Conceitos de forwards/futuros e precificação
`https://www.columbia.edu/~mh2078/FoundationsFE/for_swap_fut-options.pdf`

Introdução prática ao Filtro de Kalman
`https://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`

Dados de mercado (CME Group)
`https://www.cmegroup.com/market-data.html`