

Mémoire — PFE d'un IRS sous Hull–White - Démo illustrative

Jugurtha Boudarene

Janvier 2026

Table des matières

1	Présentation rapide du projet et exécution locale	1
1.1	Objectif	1
1.2	Pré-requis	2
1.3	Installation (machine quelconque)	2
1.4	Lancer l'application Streamlit	2
1.5	Lancer un run en ligne de commande (sans UI)	2
2	Structure du projet (arborescence)	2
3	Guide utilisateur (workflow)	3
3.1	Workflow recommandé (via l'UI Streamlit)	3
4	Fichiers importants (référéncés dans docs_registry.json)	4
4.1	Modèles (briques analytiques / simulation de taux)	4
4.2	Pricers (caplets/caps, swaps, swaptions)	4
4.3	Calibration (HW1F / HW2F) et volatilité implicite	4
4.4	Risque PFE/EPE (simulation de courbe, exposition swap, reporting)	5
4.5	Couche “instruments” (API POO pour unifier l'usage des pricers)	5

[Cliquez pour accéder à la démo Streamlit \(PFE\)](#)

1 Présentation rapide du projet et exécution locale

1.1 Objectif

Ce projet est une **démo technique** visant à reproduire une chaîne de calcul **PFE** (*Potential Future Exposure*) sur un **swap de taux (IRS)** dans un cadre **reproductible** :

- **Courbe** : chargement de données (Excel/template) et construction d'une structure par terme (DF/zc/forwards).
- **Modèle de taux** : simulation de scénarios via **Hull–White** (**1F** et/ou **2F** selon la démo), calibré sur un panel d'instruments (caps/floors, swaptions, etc. selon disponibilité).
- **Pricer** : calcul de la **MTM** du swap sur la grille de temps, scénario par scénario.
- **Exposition** : conversion MTM \rightarrow exposition positive $E(t) = \max(V(t), 0)$, puis indicateurs :

$$\text{EPE}(t) = E[E(t)], \quad \text{PFE}_\alpha(t) = \inf\{x : P(E(t) \leq x) \geq \alpha\}.$$

- **Visualisation / traçabilité** : graphiques (courbes, volatilités, PFE), exports de runs, comparaisons et logs.

1.2 Pré-requis

- Python **3.10+** recommandé.
- Dépendances principales : **streamlit**, **numpy**, **pandas**, **matplotlib** (et éventuellement **scipy** selon calibration).
- (Optionnel) Lecture Excel : **openpyxl**.

1.3 Installation (machine quelconque)

Dans un terminal à la racine du projet :

```
# 1) (Optionnel) créer et activer un environnement virtuel
python -m venv .venv

# Windows
.venv\Scripts\activate

# 2) installer les dépendances
pip install -r requirements.txt
```

1.4 Lancer l'application Streamlit

Point d'entrée typique :

- **App multi-pages** : **app.py**

```
# App Streamlit
streamlit run .\streamlit_app\app.py
```

1.5 Lancer un run en ligne de commande (sans UI)

Notebook disponible à la racine du projet **test.ipynb** pour exécuter :

- construction courbe,
- calibration HW,
- simulation des scénarios,
- calcul PFE,
- exports (CSV/PNG).

Où sont les résultats ? Chaque run crée un dossier **data/run_...** contenant typiquement :

- **curves/** : DF/zc/forwards, inputs et paramètres.
- **calibration/** : erreurs, vols cibles vs vols model, paramètres HW.
- **exposure/** : MTM paths (optionnel), expositions $E(t)$, EPE, PFE.
- **figs/** : figures PNG (si activé).
- **run_meta.json** : paramètres du run (seed, modèle, α , grille, etc.).

2 Structure du projet (arborescence)

```
.
├── app.py
├── test.ipynb
├── requirements.txt
├── streamlit_app/
│   ├── app.py
│   └── pages/
│       └── 1_Overview.py
```

```

2_Calibration_HW1F.py (HW 1F : fit)
3_Calibration_HW2F.py (HW 2F : fit)
4_PFE_Swap.py
5_Portfolio_Tracking.py
6_Documentation.py
data/
docs_registry.json
irlab.db
runs.sqlite
ui/
capture.py
code_docs.py
db.py
io.py
plotting.py
ir/
Calibration_Templates/
SWPN_Calibration_Template_30092025_USD.xlsx
market/
curve.py (load courbes / templates)
loaders_excel.py (load courbes / templates)
plots.py (plots courbe / vols)
models/
hw1f.py (Hull--White 1F)
hw2f.py (Hull--White 2F)
pricers/
hw1f_pricer.py
hw2f_pricer.py
calibration/
hw1f_calibration.py
vol.py
hw2f_profile.py
instruments/
base.py
rates.py
instruments/
hw2f_sim.py
pfe_plot.py
pfe_swap.py

```

3 Guide utilisateur (workflow)

3.1 Workflow recommandé (via l'UI Streamlit)

1. Démarrer l'app :

```
streamlit run .\streamlit_app\app.py
```

2. Calibrer le marché (page *Calibration HW1F/HW2F*) :

- Importer / sélectionner une **courbe** (DF/zc) et vérifier les plots.
- (Optionnel) charger un template de calibration (un fichier est déjà chargé par défaut).
- Choisir le modèle (**HW 1F** ou **HW 2F**).
- Lancer la calibration (erreurs par maturité/tenor, paramètres).
- Vérifier la cohérence : **vols/prix target vs vols/prix modèle**.

3. Calcul PFE (page *PFE*) :

- Choisir **N scénarios**, **seed**, **grille de temps** (horizon).

- Fixer le niveau α (ex : 95% ou 99%).
 - Lancer la simulation, puis afficher **EPE** et **PFE $_{\alpha}$** (de préférence choisir un run déjà présent dans la base de données et plutôt un run HW2F car il est très rapide).
 - Exporter les résultats (CSV/PNG) via les boutons **download**.
4. **Lire la documentation du code** (page *Documentation*) :
- La navigation est restreinte aux fichiers listés dans `docs_registry.json`.
 - Le panneau de droite affiche une **fiche manuelle** (titre, résumé, usage, notes).

4 Fichiers importants (référéncés dans `docs_registry.json`)

Principe. La page *Documentation* lit `pages/docs_registry.json` : chaque clé correspond au chemin relatif d'un fichier, et chaque entrée fournit `title`, `tags`, `summary`, `usage`, `notes`. Dans l'UI, ces fiches sont rendues par `pages/code_docs.py`.

Ci-dessous, la **liste des modules effectivement présents** dans le `docs_registry.json` fourni, regroupés par rôle dans le projet.

4.1 Modèles (briques analytiques / simulation de taux)

- `ir/models/hw1f.py`
Rôle : implémente le **Hull–White 1 facteur** (formules fermées) + **une couche Monte Carlo**.
- `ir/models/hw2f.py`
Rôle : implémente le **Hull–White 2 facteurs (G2++)** en **analytique** (sans MC) : ingrédients fermés pour caplets/swaptions.

4.2 Pricers (caplets/caps, swaps, swaptions)

- `ir/pricers/hw1f_pricer.py`
Rôle : pricer **Hull–White 1F** orienté **pricing analytique** : options sur ZC, caplets/-caps/floors et swaptions.
- `ir/pricers/hw2f_pricer.py`
Rôle : pricer **Hull–White 2F (G2++)** pour : **caplets** (via option sur ZC) et **swaptions** (via **approximation Rebonato gaussienne** du swap rate + pricing **Bachelier**).

4.3 Calibration (HW1F / HW2F) et volatilité implicite

- `ir/calibration/hw1f_calibration.py`
Rôle : calibrateur HW1F sur (**a**, **sigma**) à partir de caplets ou swaptions, avec objectif **RMSRE**.
- `ir/calibration/hw2f_profile.py`
Rôle : calibration HW2F (G2++) sur swaptions via approche **profile** : **grille** sur (a, b, ρ) + **optimisation** sur (σ, η) pour chaque point de grille.
- `ir/calibration/vol.py`
Rôle : utilitaire d'**inversion prix** \rightarrow **vol** en **Bachelier** (vol normale en bps) via **Brent**.

4.4 Risque PFE/EPE (simulation de courbe, exposition swap, reporting)

— `ir/risk/hw2f_sim.py`

Rôle : simulateur Monte Carlo “courbe” en HW2F : génère des distributions de **zéro-coupons** $P(t, T)$ en simulant exactement des facteurs gaussiens corrélés.

— `ir/risk/pfe_swap.py`

Rôle : calcule la **distribution de MTM** d’un swap vanilla à des dates t , puis agrège en **EPE** et **PFE_q** sur une grille.

— `ir/risk/pfe_plot.py`

Rôle : reporting : fonction de plotting `matplotlib` pour profils PFE (+ EPE optionnel), annotation du pic, formatage des montants et export PNG.

4.5 Couche “instruments” (API POO pour unifier l’usage des pricers)

— `ir/instruments/rates.py`

Rôle : définit des wrappers POO (Caplet/Cap/Floor/Swap/Swaption, etc.) avec une API homogène `price(pricer)`.