Simulateur de Prix CRU

Guide Technique Simplifié

Mathématiques, Code & Interface

Expliqué comme si vous aviez 5 ans
(bon, peut-être 15 ans avec des notions de stats)

Département Data & Analytics

Version 2.0 — Janvier 2026

Ce document explique :

1. Comment le modèle mathématique fonctionne (en termes simples)
2. Ce que fait chaque ligne de code importante
3. À quoi sert chaque slider/paramètre dans l'interface

Table des matières

1 L'Idée Générale (En 30 Secondes)

Imaginez que vous voulez savoir combien coûtera l'acide sulfurique dans 3 ans.

Option 1 : Regarder la prévision CRU → « 140 $/t »

Problème : Le marché ne suit jamais exactement les prévisions.

Option 2 (ce qu'on fait) :

1. Prendre la prévision CRU comme « tendance de base »
2. Ajouter du « bruit » aléatoire (volatilité du marché)
3. Ajouter des « chocs » possibles (spikes)
4. Répéter 500 fois pour voir toutes les possibilités

Résultat : Au lieu d'UN chiffre, on a une PLAGE de prix possibles.

2 Le Modèle Mathématique

2.1 La Formule Magique

Tout le simulateur repose sur UNE formule :

$$\boxed{Prix(t) = Tendance(t) \times e^{\sigma \cdot W(t)} \times (1 + Spike(t))} \quad (1)$$

Décomposons ça :

Terme	Signification en français
$Prix(t)$	Le prix simulé au mois t
$Tendance(t)$	La prévision CRU interpolée (la ligne rouge pointillée)
$e^{\sigma \cdot W(t)}$	Le facteur de volatilité : fait varier le prix autour de la tendance
$(1 + Spike(t))$	Le multiplicateur de spike : +30% si choc, sinon ×1

2.2 Composant 1 : La Tendance $T(t)$

D'où ça vient ? Du fichier Excel CRU Outlook.

Le problème : Les données sont annuelles (2025, 2026, 2027...)

La solution : On interpole avec une courbe lisse (spline cubique)

En code Python :

```
from scipy.interpolate import interp1d
f = interp1d(years, prices, kind='cubic')
monthly_trend = f(monthly_dates)
```

Résultat : On passe de 6 points annuels à 72 points mensuels.

2.3 Composant 2 : La Volatilité $\sigma \cdot W(t)$

C'est quoi ? Du « bruit » aléatoire qui fait bouger le prix.

Le problème initial : Le bruit classique (Brownien simple) fait des courbes « en dents de scie » — pas réaliste.

La solution : On « lisse » le bruit avec un processus AR(1) :

$$X(t) = \underbrace{\alpha \cdot X(t-1)}_{\text{mémoire du passé}} + \underbrace{(1-\alpha) \cdot \epsilon(t)}_{\text{nouveau choc}} \tag{2}$$

— α = facteur de lissage (0.7 par défaut)
— Plus α est grand, plus la courbe est « douce »
— $\epsilon(t)$ = choc aléatoire normal $\sim N(0, \sigma)$

En code Python :

```
smoothed[0] = raw_shocks[0]
for t in range(1, n_steps):
    smoothed[t] = alpha * smoothed[t-1] + (1-alpha) * raw_shocks[t]
```

En image :

Sans lissage : $\sim\sim\sim\sim$ (electrocardiogramme)
Avec lissage : $\smile\frown\smile$ (vagues douces)

2.4 Composant 3 : Les Spikes $S(t)$

C'est quoi ? Des chocs de prix soudains (grèves, ouragans, sanctions...).
Comment ça marche ?

1. On tire au hasard si un spike arrive (probabilité = fréquence/12)
2. Si spike : le prix monte de $I\%$ (intensité)
3. Puis le prix redescend progressivement sur P mois (persistance)

La formule de décroissance :

$$Spike(t+k) = I \times e^{-3 \cdot k/P} \tag{3}$$

$k = 0$ (jour du spike)	$Spike = I$ (ex : $+50\%$)
$k = P/3$	$Spike \approx 0.37 \times I$ ($+18\%$)
$k = P$ (fin de persistance)	$Spike \approx 0.05 \times I$ ($+2.5\%$)

Forme visuelle : « Aileron de requin » (Shark Fin) — montée brutale, descente progressive.

En code Python :

```
for k in range(persistence):
    progress = k / persistence
    decay = exp(-3.0 * progress)  # Decroissance exponentielle
    spike_contribution[t + k] += intensity * decay
```

2.5 Composant 4 : Monte Carlo (500 simulations)

Pourquoi 500 ? Pour avoir assez de scénarios et calculer des statistiques fiables.
Ce qu'on calcule :
— **Moyenne** : le scénario « central »
— **Percentile 5%** : pire cas (prix bas)
— **Percentile 95%** : pire cas (prix haut)

En code Python :

```
all_paths = np.zeros((500, n_months))  # 500 simulations x 72 mois

for i in range(500):
    brownian = generate_brownian_motion(...)
    spikes = generate_spike_process(...)
    all_paths[i] = trend * exp(brownian) * (1 + spikes)

mean = np.mean(all_paths, axis=0)
p5 = np.percentile(all_paths, 5, axis=0)
p95 = np.percentile(all_paths, 95, axis=0)
```

3 Guide Complet des Paramètres de l'Interface

Voici TOUS les sliders et options de l'application, avec leur effet exact.

3.1 Section : Commodity Selection

Paramètre	Valeurs	Ce que ça change
`Commodity`	Sulfuric Acid / Sulphur	Change le fichier source (feuille Excel différente)
`Market / Region`	CFR US Gulf, FOB Japan...	Sélectionne la colonne de prix dans l'Excel

3.2 Section : Projection Horizon

Paramètre	Valeurs	Ce que ça change
`Start Year`	2025, 2026, 2027	Année de début de la simulation
`End Year`	2028, 2029, 2030	Année de fin → détermine le nombre de mois

Exemple : Start=2025, End=2030 → 6 ans = 72 mois de simulation.

3.3 Section : Volatility Settings

Paramètre	Valeurs	Ce que ça change
`Use Historical Volatility`	Oui / Non	Si Oui : calcule σ depuis l'historique
`Volatility Multiplier`	0.5 à 2.0	Multiplie la volatilité historique (1.5 = 50% plus volatile)
`Base Volatility (%)`	5% à 50%	Volatilité manuelle si historique désactivé

Formule utilisée :

$$\sigma_{\text{final}} = \sigma_{\text{historique}} \times \text{Multiplier} \times 0.6$$

Le $\times 0.6$ est un « dampening » pour éviter les courbes trop bruyantes.

3.4 Section : Market Spikes (Black Swans)

Paramètre	Valeurs	Ce que ça change
`Spike Frequency`	0 à 3/an	Probabilité $p = freq/12$ par mois
`Spike Intensity`	0% à 100%	Amplitude du spike (ex : 50% = prix $\times$ 1.5)
`Spike Persistence`	1 à 12 mois	Durée avant retour à la normale
`Decay Type`	Exponential / Linear	Forme de la décroissance

Conseil pratique :
— Scénario calme : Freq=0.5, Intensity=20%, Persistence=3
— Scénario stress : Freq=2.0, Intensity=50%, Persistence=6

3.5 Section : Simulation Settings

Paramètre	Valeurs	Ce que ça change
`Monte Carlo Paths`	100 à 2000	Nombre de scénarios simulés
`Curve Smoothing`	0.3 à 0.9	Coefficient AR(1) α : plus haut = plus lisse
`Random Seed`	0 à 99999	Graine aléatoire (même seed = même résultat)

3.6 Section : Scenario Explorer (Nouveau !)

Paramètre	Valeurs	Ce que ça change
`Select Scenario`	1 à N	Affiche le scénario #X sur le graphique

Ce slider permet de « naviguer » dans les 500 (ou plus) scénarios simulés pour voir différentes réalisations possibles du marché.

4 Résumé du Code Principal

4.1 Architecture Simplifiée

```
app.py
  |
  +-- parse_outlook_data()      # Lit l'Excel CRU
  +-- interpolate_trend()       # Interpole en mensuel
  +-- calculate_volatility()    # Calcule sigma historique
  |
  +-- generate_brownian_motion()  # Genere le bruit lisse
  +-- generate_spike_process()    # Genere les spikes
  +-- simulate_prices()           # Combine tout
  |
  +-- create_projection_chart()   # Graphique Plotly
  +-- main()                      # Interface Streamlit
```

4.2 Fonctions Clés Expliquées

4.2.1 generate_brownian_motion()

```
def generate_brownian_motion(n_steps, volatility, n_sims,
                             smoothing_factor=0.7, seed=None):
    # 1) Convertir volatilite annuelle en mensuelle
    monthly_vol = (volatility / sqrt(12)) * 0.6  # Dampen

    # 2) Generer chocs aleatoires
    raw_shocks = normal(0, monthly_vol, (n_sims, n_steps))

    # 3) Appliquer lissage AR(1)
    smoothed = zeros((n_sims, n_steps))
    smoothed[:, 0] = raw_shocks[:, 0]
    for t in range(1, n_steps):
        smoothed[:, t] = (smoothing_factor * smoothed[:, t-1] +
                          (1-smoothing_factor) * raw_shocks[:, t])

    # 4) Somme cumulative
    return cumsum(smoothed, axis=1)
```

4.2.2 generate_spike_process()

```
def generate_spike_process(n_steps, freq, intensity,
                           persistence, decay_type):
    # 1) Probabilite de spike par mois
    p_spike = min(freq / 12, 0.3)

    # 2) Tirer les spikes (bernoulli)
    spikes = binomial(1, p_spike, n_steps)

    # 3) Appliquer decroissance
    contribution = zeros(n_steps)
    for t in range(n_steps):
        if spikes[t] == 1:
            for k in range(persistence):
                if t + k < n_steps:
                    decay = exp(-3.0 * k / persistence)
                    contribution[t + k] += intensity * decay

    return contribution
```

5 Conclusion : Comment Tout S'Assemble

1. L'utilisateur choisit ses paramètres dans la sidebar
2. On lit la tendance CRU depuis Excel
3. On génère 500 trajectoires avec bruit + spikes

4. On calcule moyenne et percentiles
5. On affiche le graphique avec :
   — Bande grise = zone de risque (5%-95%)
   — Ligne rouge = prévision officielle
   — Ligne bleue = scénario sélectionné
6. L'utilisateur peut explorer chaque scénario avec le slider

En résumé :
Prix = Tendance CRU × Bruit lissé × Spikes avec décroissance
Répété 500 fois → Distribution de prix possibles

Document généré le 28 janvier 2026 — Pour toute question, contacter l'équipe Data & Analytics.