Projet Finance de marché (Sujet N)

Question 1 :

Soit la dynamique $dr_t = a(b - r_t)dt + \gamma dW_t$. On discrétise le temps sur l'intervalle $[0, T]$ en $N + 1$ valeurs $(t_i)_{i \in [0,N]}$.

Voici le schéma d'Euler correspondant à cette dynamique :

$$\begin{cases} r_{t_0} = r_0 \\ r_{t_{i+1}} = r_{t_i} + a(b - r_{t_i})(t_{i+1} - t_i) + \gamma(W_{t_{i+1}} - W_{t_i}) \end{cases}$$

Avec r_0 donné et $(W_{t_i})_{i \in [0,N]}$ Mouvement Brownien.

Question 2 :

On utilise dans notre cas un pas constant $\Delta_{t_i} = t_{i+1} - t_i$ d'où la variable $step$. De même, étant donné que :

$$(W_{t_{i+1}} - W_{t_i}) \overset{loi}{\rightarrow} N(0, t_{i+1} - t_i)$$

On simule donc $W_{t_{i+1}} - W_{t_i}$ par $\sqrt{\Delta_{t_i}} G_i$ avec $G_i \overset{loi}{\rightarrow} N(0,1)$ pour tout $i \in [0, N]$. On précise que tous les G_i sont indépendants.

```
[16] r0 = 0.04
     a = 0.1
     b = 0.02
     gamma = 0.12
     T = 2
     N = 1000
     step = T/N

     ti = np.linspace(0, T, N)
```

```
[3]  def G(size):
       res = []
       for _ in range(size):
         res.append(np.random.normal())
       return res

     def r_t(G):
       res = [r0]
       for i in range(N):
         res.append(res[-1] + a*(b-res[-1]) * step + gamma * np.sqrt(step) * G[i-1])
       return res
```

```
[4]  G_sample = G(N)
     r_t_sample = r_t(G_sample)

     plt.figure()
     plt.plot(r_t_sample)
     plt.show()
```

Fig. 1 : Code question 2

Après import des modules *numpy* et *matplotlib.pyplot* et exécution du code ci-dessus tout en prenant compte l'aléa on obtient une courbe de cet acabit :

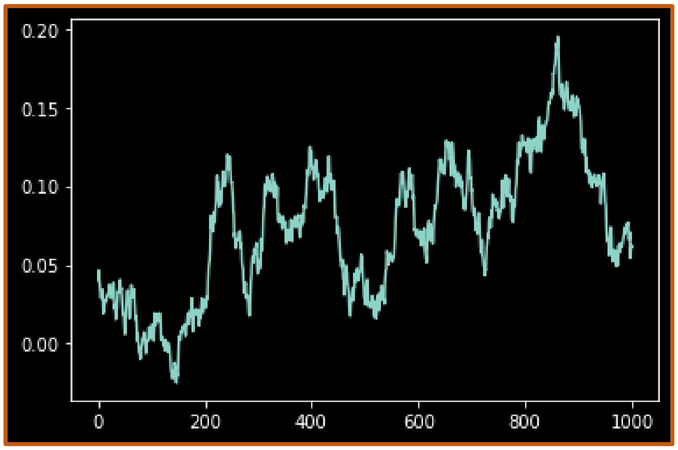

Fig. 2 : Exemple de résultat de la question 2

Question 3 :

$$\int_0^T r_u du = \int_0^{t_1} r_u du + \int_{t_1}^{t_2} r_u du + \cdots + \int_{t_{n-1}}^{t_n} r_u du$$

Or pour $t_{i-1} < u < t_i$, on a $r_u \sim r_{t_{i-1}}$ donc :

$$\int_{t_{i-1}}^{t_i} r_u du \sim r_{t_{i-1}} \int_{t_{i-1}}^{t_i} 1\, du \sim r_{t_{i-1}} \Delta_{t_i}$$

Donc :

$$\int_0^T r_u du = \lim_{n \to +\infty} \sum_{i=1}^{n} r_{t_{i-1}}^n \Delta_{t_i}^n$$

Question 4 :

Soit $B(0,T)$ le prix du produit à valeur de 1 en T. Par définition, dans le cas r_t est déterministe on a :

$$B(0,T) = e^{-\int_0^T r_u du}$$

Or dans notre cas, r_t est de nature stochastique. Donc selon la loi des grands nombres :

$$B(0,T) = E_Q\left[e^{-\int_0^T r_u du}\right]$$

```
[15] K = 10000
     B_list = []
     for _ in range(K):
       B_list.append(np.exp(- np.sum(np.array(r_t(G(N))) * step)))
     B = np.mean(B_list)
     print('Prix ZC = {}'.format(B))

     Prix ZC = 0.9231838285626114
```

Fig. 3 : Code question 4

Question 5 :

Non ils ne sont pas corrélés, le marché obligataire représenté ici par l'actif sans risque S^0 semble ne varier, d'après sa dynamique $dS_t^0 = r_t S_t^0 dt$, qu'en fonction du temps (drift). Alors que le drift de la dynamique de l'actif S est nul, ne dépendant donc que de la volatilité (B_t).

Question 6 :

Etant donné le processus stochastique S avec $d\widetilde{S_t} = \sigma\left(t, \widetilde{S_t}\right)\widetilde{S_t} dB_t$, voici son schéma d'Euler correspondant :

$$\begin{cases} \widetilde{S_0} = 30 \\ \widetilde{S_{t_{i+1}}} = \widetilde{S_{t_i}} + \sigma\left(t_i, \widetilde{S_{t_i}}\right) \widetilde{S_{t_i}} \left(\frac{1}{3}\left(W_{t_{i+1}} - W_{t_i}\right) + \frac{2\sqrt{2}}{3}\left(C_{t_{i+1}} - C_{t_i}\right)\right) \end{cases}$$

Les trajectoires calculées ne sont pas solutions exactes de l'équation différentielle stochastique. C'est en ayant recourt à la méthode de Monte-Carlo qu'on converge vers une solution.

Question 7 :

Similairement à la question 2, on simule $W_{t_{i+1}} - W_{t_i}$ par $\sqrt{\Delta_{t_i}} {G_i}^1$ et $C_{t_{i+1}} - C_{t_i}$ par $\sqrt{\Delta_{t_i}} {G_i}^2$, à la différence qu'ici ${G_i}^1 \xrightarrow{loi} N(0,1)$ et ${G_i}^2 \xrightarrow{loi} N(0,1)$ sont indépendants pour tout i.

Cela se traduit informatiquement par la génération pseudo-aléatoire de deux vecteurs gaussiens, indépendamment l'un de l'autre.

```
[65] S0 = 30

[78] def sigma(t, x):
       return 0.15 * (1 + np.sqrt(t) + (x+1)/(1+x**2))

     def S_t(G1, G2):
       res = [S0]
       for i in range(N):
         res.append(res[-1] + sigma(ti[i-1], res[-1]) * res[-1] * ((1/3) * np.sqrt(step) * G1[i] + (2*np.sqrt(2)/3) * np.sqrt(step) * G2[i]))
       return res

[83] plt.figure()

     for _ in range(10):
       aux = S_t(G(N), G(N))
       plt.plot(aux)
     plt.show()
```

Fig. 4 : Code question 7

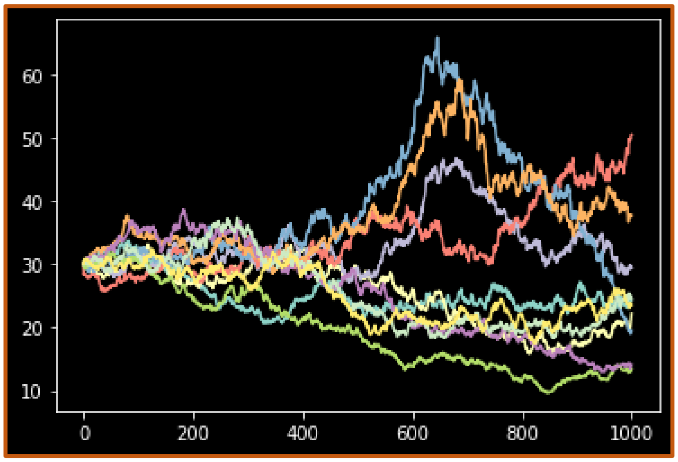

Fig. 5 : Exemple de résultat de la question 7 (trajectoires)

Question 8 :

Etant donnée que le marché est complet et en situation d'absence d'opportunité d'arbitrage, il existe un portefeuille V couvrant le produit de payoff terminal ξ_T, et ce portefeuille est une martingale sous Q. Le prix de ce produit, à la date $t = 0$, est l'espérance sous la probabilité risque-neutre du payoff-actualisé :

$$V_0 = E_Q[\xi_T \, e^{-r_T T}]$$

Cela se traduit numériquement par la génération pseudo aléatoire des vecteurs gaussiens nécessaires puis le calcul des trajectoires de sous-jacents. On récupère ainsi S_T afin de calculer le payoff ξ_T. On réitère ce procédé un nombre conséquent de fois afin d'estimer V_0 par une moyenne des simulations actualisés (Monte-Carlo).

Question 9 :

Pour l'option européenne, on a besoin de la valeur du sous-jacent à mi-temps. Cela se traduit numériquement par la valeur d'indice $\lfloor T/2 \rfloor$ du vecteur S, afin d'éviter toute erreur. On actualise ensuite le payoff au taux r_T.

```
[85] n = 1000
     payoff_act_list = []
     for _ in range(n):
       G1 = G(N)
       G2 = G(N)
       S = S_t(G1, G2)
       rt = r_t(G1)
       payoff_act_list.append(max(0, S[int(np.ceil(T/2))] - S[-1]) * np.exp(-rt[-1] * T))
     print('V0 = {}'.format(np.mean(payoff_act_list)))

     V0 = 5.571181238956342
```

Fig. 6 : Estimation du prix de l'option Européenne

Pour l'option asiatique, similairement à la question 3, on estime $\frac{1}{T}\int_0^T S_u du$ par $moy\ (S_{t_i})_{i\in[0,N]}$. On actualise de même le payoff au taux r_T. On teste ici avec des strikes K de valeurs oscillant autour de S_0 (=$10, 20, \ldots, 70$). On remarque que le prix augmente avec le strike, l'option ayant plus de chance de s'avérer utile.

```
[95] n = 1000
     K = np.linspace(0, 70, 8)
     for strike in K:
       payoff_act_list = []
       for _ in range(n):
         G1 = G(N)
         G2 = G(N)
         S = S_t(G1, G2)
         rt = r_t(G1)
         payoff_act_list.append(max(0, (strike - np.mean(S))) * np.exp(-rt[-1] * T))
       print('V0(K={}) = {}'.format(strike, np.mean(payoff_act_list)))

     V0(K=0.0) = 0.0
     V0(K=10.0) = 0.0
     V0(K=20.0) = 0.0465592150432
     V0(K=30.0) = 2.586873630446629
     V0(K=40.0) = 10.541835627451775
     V0(K=50.0) = 20.20267047912313
     V0(K=60.0) = 29.625920311496614
     V0(K=70.0) = 39.826071019362466
```

Fig. 7 : Estimation du prix de l'option Asiatique