Projet de Mathématiques 2021-2022 (Deuxième Partie)

Ngo Duong, Thibaud Reigner, Seydina Dieng

Avril 2022

1 Questions 11-20

1.1 Question 11

En utilisant la formule d'Itô

$$dg(S_t) = g'(S_t)dS_t + \frac{|\sigma S_t|^2}{2}|g''(St)dt$$

Avec : $g : \mathbb{R} \to \mathbb{R}$ de régularité C2. On applique la formule d'Itô à $g = ln$ car $x \mapsto lnx$ est C2 sur $\mathbb{R}_+^*$

On a :

$$dg(S_t) = g'(S_t)dS_t + \frac{|\sigma S_t|^2}{2}|g''(St)dt$$

et

$$dS_t = S_t * (rdt + \sigma dB_t)$$

$$dlnS_t = \frac{dS_t}{S_t} + \frac{|\sigma|^2}{2}(-\frac{|S_t|^2}{S_t^2})dt$$

$$= \frac{dS_t}{S_t} - \frac{|\sigma|^2}{2}$$

$$= \frac{S_t * (rdt + \sigma dB_t)}{S_t} - \frac{|\sigma|^2}{2}dt$$

$$= (r - \frac{|\sigma|^2}{2})dt + \sigma dB_t$$

Par intégration par rapport au temps sur [0, t], $t \in \mathbb{R}_+^*$ on a :

$$\int_0^t dlnS_u = \int_0^t (r - \frac{|\sigma|^2}{2})du + \int_0^t \sigma dBu \iff lnS_t - lnS_0 = (r - \frac{|\sigma|^2}{2})t + \sigma(B_t - B_0)$$

Or $B_0 = 0$ (par définition)

$$\iff ln\frac{S_t}{S_0} = (r - \frac{|\sigma|^2}{2})t + \sigma B_t$$

$$\iff S_t = s * exp(rt - \frac{t|\sigma|^2}{2})exp(\sigma B_t)$$

1.2 Question 12

Le pseudo-code du pricer_MC est le suivant :

Algorithm 1 *Pricer_MC*

Require: $T, s \geq 0, r, f : R_+ \to R_+$ continue
Ensure: Renvoie $Prix_{mc}$

```
function PRICER_3(s, r, σ, T, f)
    ksi ← echantillon_gaussien(0, 1, n)
    Prix_mc ← 0
    for i ← 0 to n − 1 do
        Prix_mc ← Prix_mc + exp(−t ∗ T) ∗ f(s ∗ exp(r − σ²/2 ∗ T ∗ σ ∗ √T ∗ ksi[i]))
    end for
    return Prix_mc ∗ 1/n
end function
```

1.3 Question 13

On trace le graphique pour le prix avec $r = 0.01, \sigma = 0.1, s = 100, T = 1$ et $f(x) = max(x-100, 0)$ pour $n = 10^5 k, \forall k \in [|1, 10|]$:

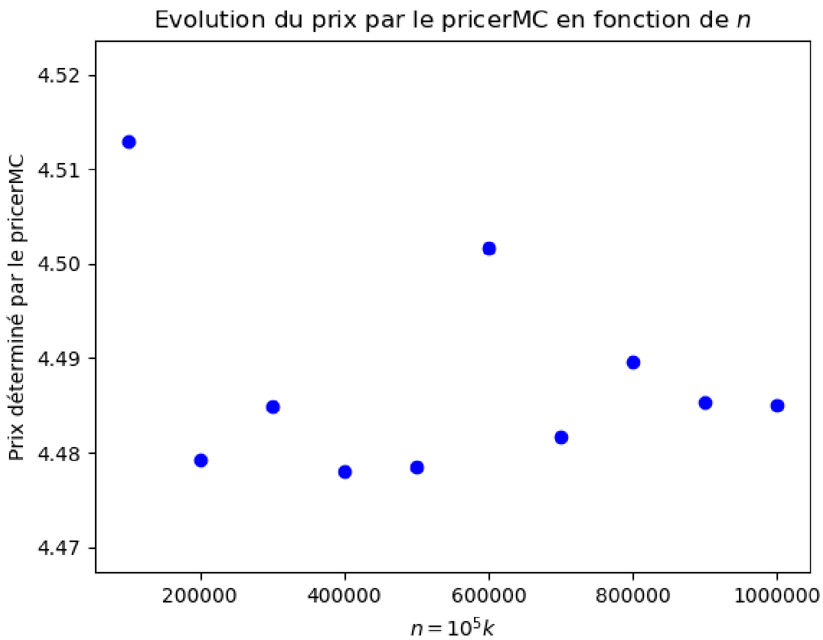

FIGURE 1 – Comparaison du put et du troisième pricer pour plusieurs valeurs de n

1.4 Question 14

Prenons

$$\mathbb{P} := \mathbb{E}[\exp(-rT)f(S_T)]$$

et

$$\widehat{PrixMC}(n) := \frac{1}{n}\sum_{1}^{n}\exp(-rT)f(s)\exp((r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}\xi_i)$$

On pose $\forall a \in [1,n]$

$$X_i = exp(-rT)f(s)\exp((r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}\xi_i)$$

On va montrer que les X_i sont indentiquement et indépendamment distribués (iid). On voit directement que $\forall i \in [1,n]$, $X_i = \phi(\xi_i)$ où

$$\phi : R \to R$$

$$x \to exp(-rT)f(s)\exp((r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}x)$$

De plus, la fonction ϕ est strictement monotone croissante. Soient (i, j) $\in [1,n]^2$, $i \neq j$. On a $(\star)$:

$$\mathbb{Q}[\xi_i \leq x_i \bigcap \xi_j \leq x_j] = \mathbb{Q}[\xi_i \leq x_i]\mathbb{Q}[\xi_j \leq x_j]$$

car (ξ_i), $i \in [1,n]$ est iid d'après l'énoncé. Et,

$$\mathbb{Q}[\xi_i \leq x_i \bigcap \xi_j \leq x_j] = \mathbb{Q}[X_i \leq \phi(x_i) \bigcap X_j \leq \phi(x_j)] = \mathbb{Q}[X_i \leq \phi(x_i)]\mathbb{Q}[X_j \leq \phi(x_j)]$$

en composant $(\star)$ par ϕ. Les Xi sont donc iid. On peut utiliser la Loi Forte des Grands Nombres :

$$\widehat{PrixMC}(n) = \frac{1}{n}\sum_{i=0}^{n} Xi \to^{ps} \mathbb{E}[X_1]$$

où

$$\mathbb{E}[X_1] = \mathbb{E}[\exp((r-\frac{\sigma^2}{2})T+\sigma\sqrt{T}\xi_1)]$$

On reconnaît une partie de l'expression de $f(S_T)$ de la question 11.

$$f(S_T) = f(S_0\exp((rT-\frac{T|\sigma^2|}{2})T+\exp\sigma B_T)$$

Par définition, on a

$$B_T \sim \mathcal{N}(0,T)$$

De plus,

$$\xi_1 \sim \mathcal{N}(0,1) \to \xi_1\sqrt{T} \sim \mathcal{N}(0,T)$$

On en conclut donc que les 2 expressions suivent la même loi. Donc,

$$(\widehat{PrixMC}(n))_{n\in N} \to^{ps} \mathbb{P} := \mathbb{E}[\exp(-rT)f(S_T)]$$

1.5 Question 15

Voici le pseudo-code de la fonction put_BS :

Algorithm 2 *Put_BS*

Require: $T, s \geq 0, r, f : R_+ \rightarrow R_+$ continue
Ensure: Renvoie $Prix_{bs}$

```
function PRICER_3(s, r, σ, T, f)
    d← 1/(σ*√T) * (log(s/K) + (r + (σ²/2) * T))
    function F(x)
        Retourne la probabilité d'avoir x
    end function
    Prix_bs←−s * F(−d) + K * exp(−r * T) * F(−d + σ * T)
    return Prix_bs
end function
```

1.6 Question 16

Application Numérique : $r = 0.01, \sigma = 0.1, s = 100, T = 1, K = 90$ ce qui donne : 0.5815

1.7 Question 17

Nous souhaitons afficher un graphique comparant les valeurs du pricer3 et du put pour plusieurs valeurs de n

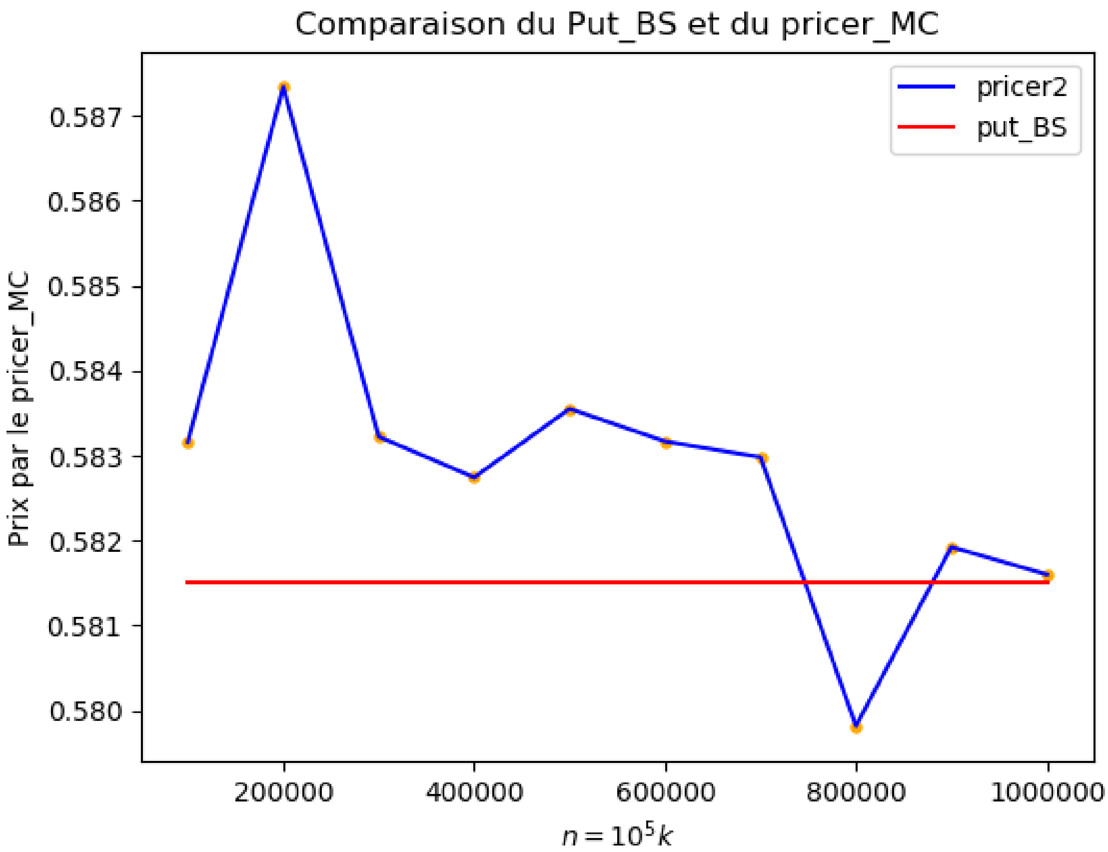

Figure 2 – Comparaison du put et de PricerMC pour plusieurs valeurs

1.8 Question 18

On trace sur un graphique en 3 dimensions le prix de l'option en utilisant la fonction put_BS pour $r = 0.01, \sigma = 0.1, s = 20k, K = 100, \forall k \in [|1, 10|], \forall T \in \frac{1}{12}, \frac{1}{6}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, 1$

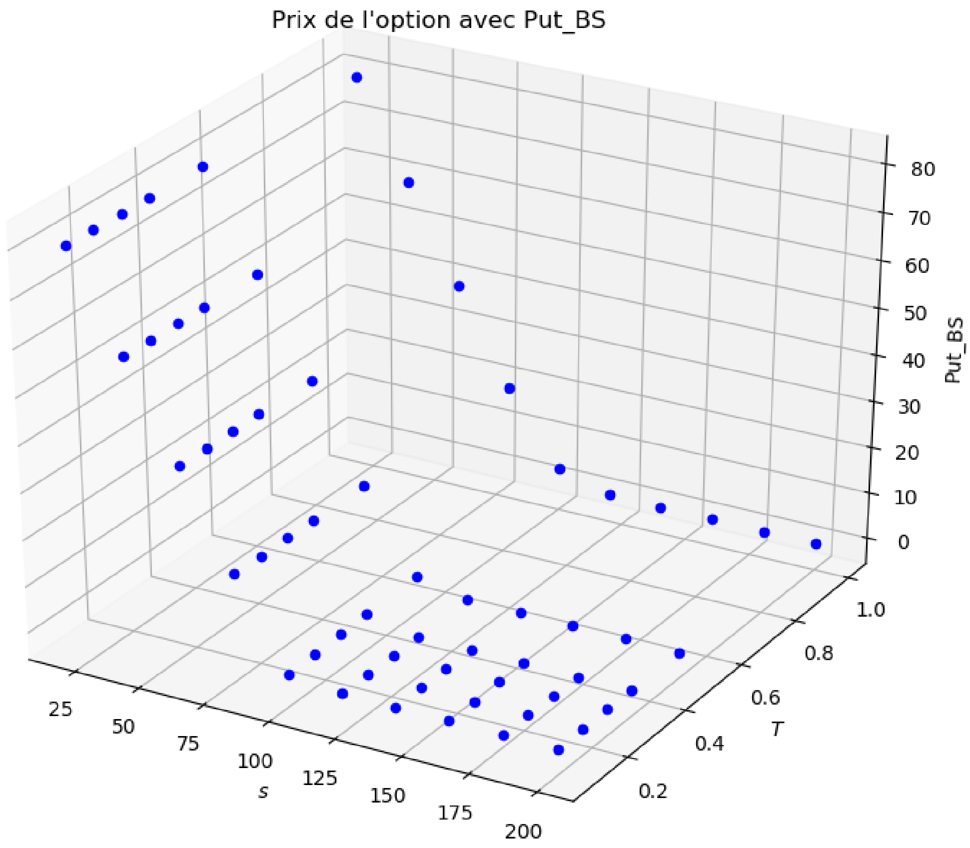

FIGURE 3 – Prix de l'option en fonction de s et de T

Remarques : le put de l'option est très sensible aux évolutions des valeurs de s : Plus s décroit, plus la valeur du put augmente fortement. Les variations de T semblent n'avoir aucun(ou peu) d'effet sur l'évolution du put. En conclusion, le prix calculé par le pricer par formule fermée est très sensible à la valeur de s.

1.9 Question 19

On trace sur un même graphique le prix en utilisant la fonction pricer_2 d'une option qui paye $r = 0.03, \sigma = 0.2, s = 100, T = 1$ et $f(x) = max(100 - S_T, 0)$ pour $N = 10k, \forall k \in [|1, 100|]$. Puis on trace sur le même graphe la droite qui a pour ordonnée p en utilisant la fonction put_BS pour le calcul de p.

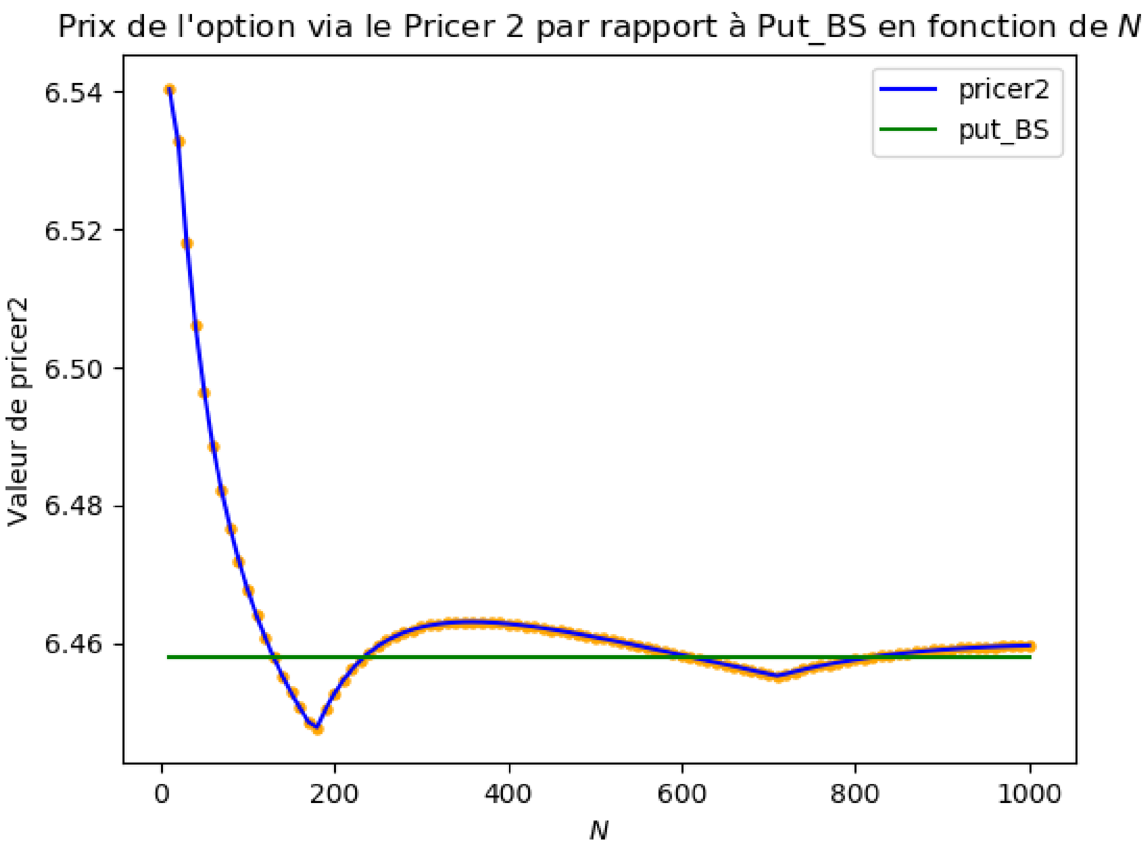

FIGURE 4 – Etude de la convergence de Cox-Ross-Rubinstein vers Black-Scholes

Remarques : Le pricer effectue un premier dépassement du put quand N est encore assez faible, puis converge vers le put quand N augmente : le modèle de Cox-Ross-Rubinstein converge donc vers le modèle de Black-Scholes lorsque le pas de discrétisation N tend est assez grand. Voici le pseudo-code de la question 19 :

Algorithm 3 *Put*

Require: $T, s \geq 0, r, f : R_+ \rightarrow R_+$ continue
Ensure: Trouve les valeurs de $r_N, h_N, b_N, p, p2$

function F(x)
  Retourne $max(100 - x, 0)$
end function
$s \leftarrow 100$
$\sigma \leftarrow 0.3$
$r \leftarrow 0.02$
$T \leftarrow 1$
for $i \leftarrow$1 to 100 **do**
  $Nliste[i] \leftarrow 10 * i$
end for
$K \leftarrow 100$
$p2 \leftarrow liste\_vide$
for $N \in Nliste$ **do**
  $r_N \leftarrow r * \frac{T}{n}$
  $h_N \leftarrow (1 + r_N) * \exp(\sigma * \sqrt{\frac{1}{n}}) - 1$
  $b_N \leftarrow (1 + r_N) * \exp(-\sigma * \sqrt{\frac{1}{n}}) - 1$
  $p \leftarrow price2(N, r_N, h_N, b_N, s, f)$
  $p2 \leftarrow p2 \bigcup p$
end for

2 Question 20 - EDP de Black Scholes

2.1 EDP de BS

On considère l'équation aux dérivées partielles suivante qu'on appelle EDP de Black Scholes :

$$\forall (t,x) \in]0,T] \times (x_{min}, x_{max}), \frac{\partial p}{\partial t}(t,x) - \frac{1}{2}\sigma^2 \frac{\partial^2 p}{\partial x^2}(t,x) - (r - \frac{\sigma^2}{2})\frac{\partial p}{\partial x}(t,x) + rp(t,x) = 0 \quad (BS)$$

$$\begin{cases} \text{p est le prix du put en fonction du temps jusqu'à maturité t et log prix de l'actif x=log(S)} \\ \sigma \text{ est la volatilité du prix de l'action} \\ \text{r est le taux d'intérêt sans risque} \end{cases}$$

Il faut maintenant la résoudre pour trouver p. Pour cela, nous allons approcher cette EDP avec 3 méthodes différentes :

- les différences finies explicites
- les différences finies implicites
- méthode de Crank-Nicholson

2.2 Discrétisation du temps et de l'espace

Soit $M \in \mathbb{N}$ le nombre de points en temps à calculer, $N \in \mathbb{N}$ le nombre de points en espace à calculer. On va poser :

$$\begin{cases} \forall j \in 0, N, x_j = x_{min} + jh \text{ où } h = \frac{x_{max} - x_{min}}{N} \\ \forall m \in 0, M, t_m = m\Delta t \text{ où } \Delta t = \frac{T}{M} \end{cases}$$

Ainsi pour que ce soit plus lisible, on pose :

$$\begin{cases} p(t_m, x_j) = p_{m,j} \\ \mathbf{P}_m = \begin{pmatrix} p_{m,0} & p_{m,1} & \dots & p_{m,N} \end{pmatrix} \end{cases}$$

2.3 Différences finies explicites

Selon le schéma **explicite**, on peut approcher les dérivées partielles par les quotients différentiels suivant :

$$\begin{cases} \frac{\partial p}{\partial t}(t_m, x_j) = \frac{1}{\Delta t}(p_{m+1,j} - p_{m,j}) \\ \frac{\partial p}{\partial x}(t_m, x_j) = \frac{1}{2h}(p_{m,j+1} - p_{m,j-1}) \\ \frac{\partial^2 p}{\partial x^2}(t_m, x_j) = \frac{1}{h^2}(p_{m,j+1} - 2p_{m,j} + p_{m,j-1}) \end{cases}$$

On injecte ensuite dans (BS) :

$$\begin{aligned}
(BS) &\Longleftrightarrow \frac{1}{\Delta t}(p_{m+1,j} - p_{m,j}) - \frac{\sigma^2}{2h^2}(p_{m,j+1} - 2p_{m,j} + p_{m,j-1}) - \frac{(r - \frac{\sigma^2}{2})}{2h}(p_{m,j+1} - p_{m,j-1}) + rp_{m,j} = 0 \\
&\Longleftrightarrow p_{m+1,j} = \frac{\Delta t \sigma^2}{2h^2}(p_{m,j+1} - 2p_{m,j} + p_{m,j-1}) + \frac{\Delta t(r - \frac{\sigma^2}{2})}{2h}(p_{m,j+1} - p_{m,j-1}) + (1 - \Delta t * r)p_{m,j} \\
&\Longleftrightarrow p_{m+1,j} = (\frac{\Delta t \sigma^2}{2h^2} + \frac{\Delta t(r - \frac{\sigma^2}{2})}{2h})p_{m,j+1} + ((1 - \Delta t * r) - \frac{\Delta t \sigma^2}{h^2})p_{m,j} + (\frac{\Delta t \sigma^2}{2h^2} - \frac{\Delta t(r - \frac{\sigma^2}{2})}{2h})p_{m,j-1}
\end{aligned}$$

D'où $\boxed{p_{m+1,j} = (\frac{\Delta t\sigma^2}{2h^2} + \frac{\Delta t(r-\frac{\sigma^2}{2})}{2h})p_{m,j+1} + ((1-\Delta t * r) - \frac{\Delta t\sigma^2}{h^2})p_{m,j} + (\frac{\Delta t\sigma^2}{2h^2} - \frac{\Delta t(r-\frac{\sigma^2}{2})}{2h})p_{m,j-1}}$

Implémentation : On a une relation (de récurrence) de p à l'instant t_{m+1} qui dépend des différentes positions de p à l'instant t_m. Autrement dit, on peut calculer $\mathbf{P}_{m+1}$ à partir des coefficients de $\mathbf{P}_m$ jusqu'à M.
Pour cela, il faut au préalable mettre en place les conditions aux limites suivantes :

$$\begin{cases} \mathbf{P}_0 = \begin{pmatrix} max(K-\exp(x_0),0) & max(K-\exp(x_1),0) & \dots & max(K-\exp(x_j),0) & \dots & max(K-\exp(x_{N-1}),0) \end{pmatrix} \\ \mathbf{P}_i = \begin{pmatrix} K\exp(-r*t_i)-\exp(x_{min}) & \dots & \dots & \dots & \dots & 0 \end{pmatrix}, \forall i \in \{0,...,M-1\} \end{cases}$$

Notre implémentation Python utilise une matrice pour stocker les vecteurs $\mathbf{P}_m$ dans ces lignes.

Algorithm 4 Prix du put - Méthode explicite à implémenter

Require: σ, $M \in \mathbb{N}$, $N \in \mathbb{N}$, r, K, T, x_{min}, x_{max}
Ensure: La i^e ligne de P représente les prix possibles du put à l'instant t_i

$P \leftarrow$ zeros(M, N) ▷ //Créer une matrice (M)x(N)

//Discrétisation de l'espace-temps
$\Delta t = T/M$
$h = (x_{max} - x_{min})/N$
//Initialisation des conditions aux limites
for $k \leftarrow 2$ to $N-1$ **do**
  $P_{1,k} = max(K - exp(x_{min} + k * h), 0)$
end for
for $j \leftarrow 1$ to M **do**
  $P_{j,1} = K * exp(-r * \Delta t * j) - exp(x_{min})$
  $P_{j,M} = 0$
end for
//Construction de la matrice P
$a_1 \leftarrow \Delta t * (1/\Delta t - r - (sigma/h)^2)$
$a_2 \leftarrow \Delta t * ((1/2) * (sigma/h)^2 + (r - (sigma^2/2)) * (1/(2 * h)))$
$a_3 \leftarrow \Delta t * ((1/2) * (sigma/h)^2 - (r - (sigma^2/2)) * (1/(2 * h)))$
for $m \leftarrow 2$ to M **do**
  for $n \leftarrow 2$ to $N-2$ **do**
    $P_{m,n} = a1 * P_{m-1,n} + a2 * P_{m-1,n+1} + a3 * P_{m-1,n-1}$
  end for
end for

Stabilité : On doit maintenant chercher quelle valeur de M fixer afin de garantir la stabilité de la méthode.
Si on pose : $k = \frac{\Delta t}{h}$ alors le schéma explicite est stable si $k \leq \frac{1}{2} \iff \frac{2T}{h^2} \leq M$.

$$\boxed{\text{Pour garantir la stabilité du schéma explicite, il faut prendre } M \geq \frac{2TN^2}{(x_{max}-x_{min})^2}}$$

2.4 Différences finies implicites