**Une image contenant tableau blanc

Description générée automatiquement**

**Methodes numériques pour le pricing d’option :**

DELHOMME Marco – DELOR Victor

MAM5 - IMAFA

Superviseur : Didier AUROUX

Date : Mardi 10 Novembre 2020

Polytech Nice-Sophia 2020 – 2021

Sommaire :

1. Introduction…………………………………………………………………………….2
2. Choix de l’équation à utiliser : Équation de Black-Scholes………..2

2.1) Rappel et définition……………………………………………………………

2.2) Étude de l’équation……………………………………………………………

2.2.1) Conditions aux bords………………………………………………..

2.2.2) Passage à une équation de la chaleur……………………….

3) Choix des schémas numériques pour l’équation de la chaleur……

3.1) Discrétisation ………………………………………………………………………

3.2) Schéma d’Euler explicite……………………………………………………..

3.3) Schéma d’Euler implicite……………………………………………………..

4) Analyse des résultats………………………………………………………………..

4.1) Résultats obtenus par Euler explicite…………………………………..

4.2) Résultats obtenus par Euler implicite…………………………………..

4.3) Comparaison des deux schémas utilisés……………………………….

5) Conclusion

1. Introduction :

Le but de ce projet est de résoudre l’équation de Black-Scholes afin de modéliser le pricing d’une option.

Pour ce faire, nous procéderons tout d’abord à des changements de variables pour nous ramener à une équation de la chaleur, à partir de notre équation de Black-Scholes. Par la suite, nous discrétiserons le temps et l’espace pour pouvoir y appliquer une méthode numérique. Une fois réalisé, nous pourrons utiliser les schémas d’Euler explicite et implicite et nous analyserons les résultats obtenus.

1. Equation de Black-Scholes :

**Rappel :**

On note S(t) le prix d’un actif à l’instant t, V(t,S) la valeur d’une option sur l’actif S, s la volatilité ainsi que r un rendement déterministe.

On rappelle la définition de l’équation de Black-Scholes :

(\*)

**Etude théorique :**

Par le biais d’un changement de variable, on va transformer l’équation de Black-Scholes en équation de la chaleur.

Tout d’abord, on pose

On a alors :

Puis, on pose . Ce changement de sens du temps nous permet de passer en temps croissant et nous verrons par la suite que cette étape nous permet d’avoir une condition initiale dans nos schémas numériques.

On a alors :

Par la suite, on pose

Ces changements de variable nous permettent d’obtenir d’ores et déjà à partir de  :

Enfin, on procède à un ultime changement de variable en posant : avec quelconques.

Puisque ces derniers sont quelconques, on peut les choisir comme on le souhaite et on remarque qu’en prenant on a qui devient :

On retrouve donc bien une relation qui correspond à l’équation de la chaleur.

**Conditions aux bords :**

Soient **T l’échéance** et **K le strike** de l’option.

L’équation de Black-Scholes est une équation aux dérivées partielles hyperbolique. Elle possède par conséquent des conditions aux bords, qui interviennent lorsque (à maturité) et lorsque ou

On note : •

•

Dans le cas d’un **Call** (européen), on a : •

•

• équivaut à en l’infini

Dans le cas d’un **Put** (européen), on a : •

•

•

**Condition initiale :**

Dans le **cas d’un Call** (européen) :

La condition initiale en temps est ici donnée par :

De manière symétrique, dans le **cas d’un Put** (européen), on aura alors comme condition initiale en temps :

Une image contenant texte

Description générée automatiquement

Finalement, nous devons résoudre l’EDP

Une image contenant texte

Description générée automatiquement

Avec

D’après le cours, la solution à cette EDP est donnée par :

Par calcul de l’intégrale et changements de variable inverses, on a ainsi :

Avec :

, et

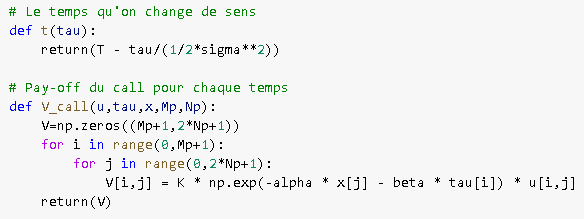
1. Schémas numériques pour l’équation de la chaleur :

Afin de résoudre numériquement l’EDP , nous avons implémenté l’étude théorique vu précédemment sur Python puis nous avons discrétisé notre domaine et enfin avons implémenté des schémas numériques ce qui nous a permis de réaliser une analyse pertinente sur le sujet.

Pour commencer, il a été nécessaire de réaliser le changement de variable vu ci-dessus :



Ensuite, nous avons implanté le changement de sens du temps ainsi que notre fonction nous permettant de calculer le prix de l’option (nous avons fait les analyses dans le cas d’un Call) :

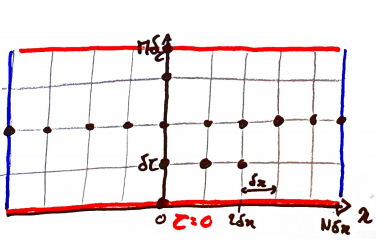


La modélisation du prix de notre actif S est également intervenue dans notre programme. Nous nous sommes appuyés sur la définition du cours qui est la suivante :



Par la suite, nous avons procédé à la discrétisation du domaine de définition de l’équation de Black-Scholes pour pouvoir approcher numériquement notre solution.

**Discrétisation :**

Afin de résoudre numériquement l’EDP , nous allons discrétiser notre domaine en espace et en temps comme cela :

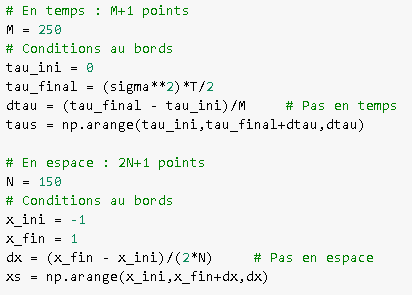
En temps, nous prenons M+1 points de discrétisation (0 et M points vers le haut).

En espace, nous prenons 2N+1 points de discrétisation (0, N points vers la gauche et N vers la droite).

Ainsi, les domaines de définitions du temps et de l’espace peuvent se discrétiser tel que :

avec

D’un point de vue programmation, cette discrétisation donne :



Soit

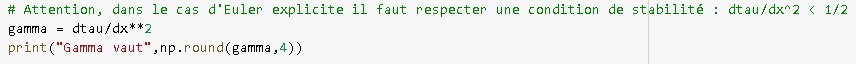
On souhaite désormais approximer . Pour cela, nous avons implémenté 2 schémas numériques, **Euler explicite** et **Euler implicite**. Par la suite nous allons analyser les résultats que nous avons obtenu et que nous allons vous présenter ci-après dans ce document.

**Euler explicite :**

Nous rappelons la définition du schéma numérique d’Euler explicite :

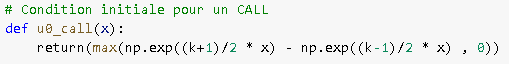
En réécrivant l’équation comme ci-après : (1) avec , nous pouvons calculer la solution à l’instant .

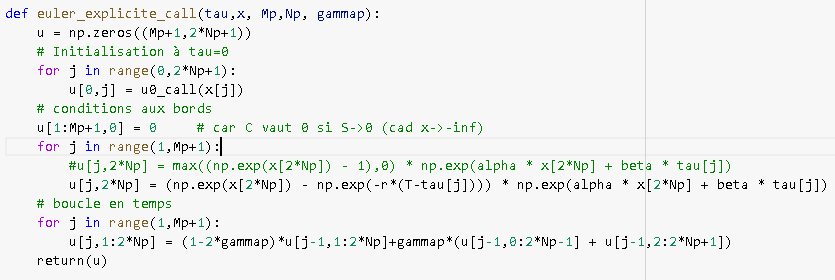
On note que ce schéma comporte une condition de stabilité. En effet, il est stable si et seulement si  :



Il est également nécessaire de considérer des conditions aux bords pour la résolution de notre EDP.

Lorsque vaut zéro donc il n’y a pas de problème.

Pour les cas où , nous avons implémenté des formules du cours nous permettant d’obtenir les conditions aux limites dans ces cas-là, comme ci-après :



4. / Interprétation

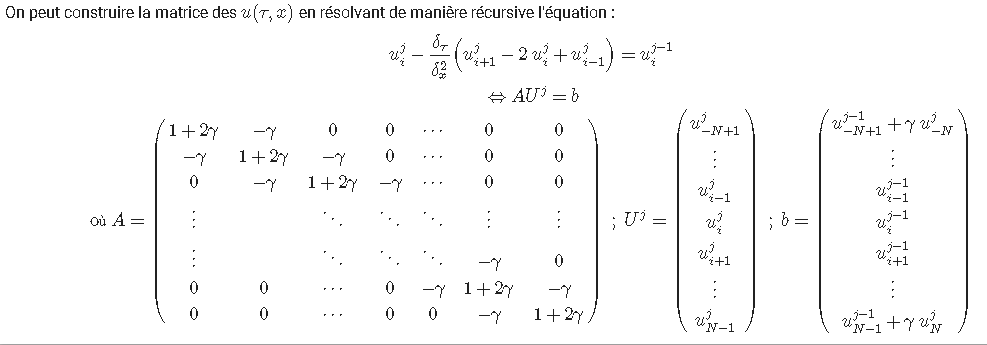
On précise par ailleurs que pour pouvoir calculer la fonction pay-off, on calcule une matrice u de taille où vont se placer nos conditions aux bords. On remplit cette matrice ligne par ligne par le biais de l’équation (1) ci-dessus.

L’erreur pour ce schéma est de l’ordre de .

**Euler implicite :**

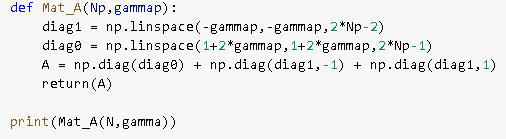
Le schéma d’Euler implicite est donné par :

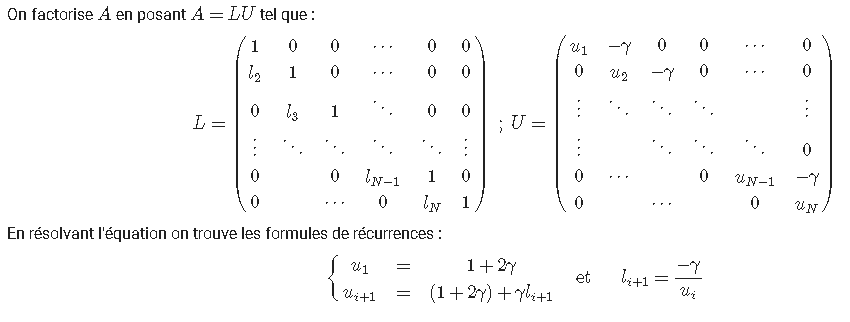
et équivaut à l’équation suivante :

On en déduit ainsi que :

En effet, on va se servir de cette forme matricielle et d’une factorisation LU pour pouvoir résoudre notre schéma numérique d’Euler implicite.

On définit ainsi pour commencer notre matrice A dans python :

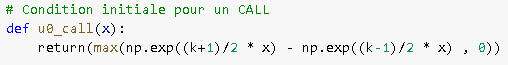


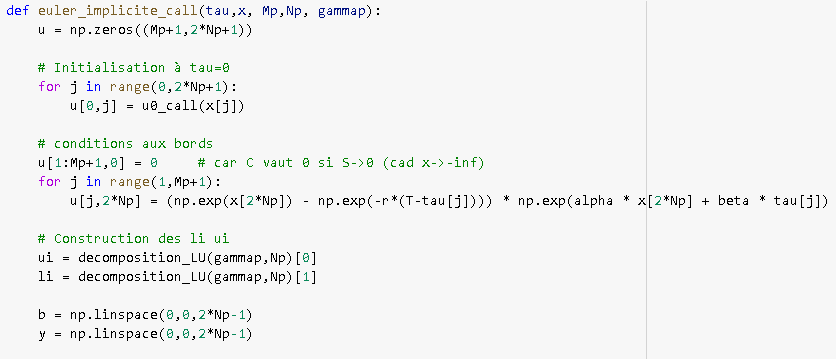
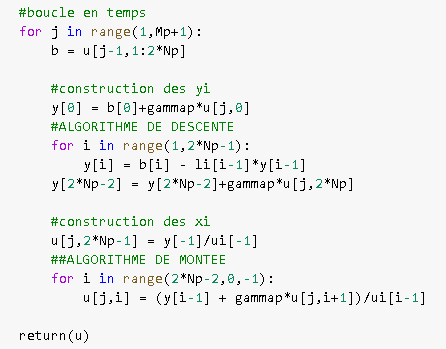
Puis,

On implémente alors une méthode de décomposition LU dans python :



Et enfin on implémente le schéma numérique, avec des conditions aux bords trouvées de même qu’avec Euler explicite :



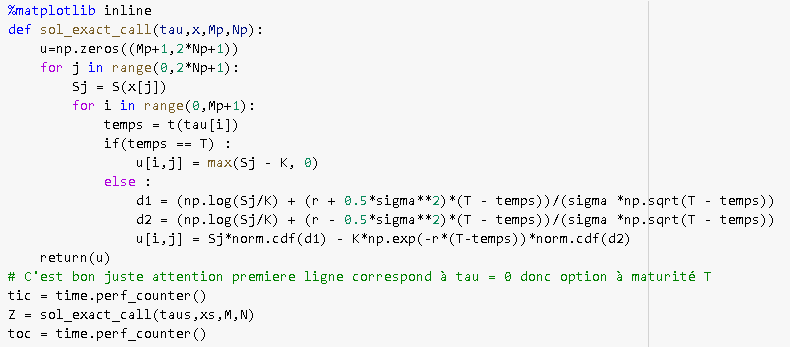


On réalise ici une boucle en espace dans une boucle en temps pour remplir notre matrice u (qui nous permet toujours de trouver notre pay-off grâce à la fonction V\_Call) par le biais d’un algorithme de descente et de montée.

Le remplissage se fait ici élément par élément.

1. **Analyse des résultats obtenus :**

Tout d’abord, afin de pouvoir comparer les résultats de nos schémas numériques, nous avons implémenté le calcul d’une solution exacte sur Python avec le code suivant :



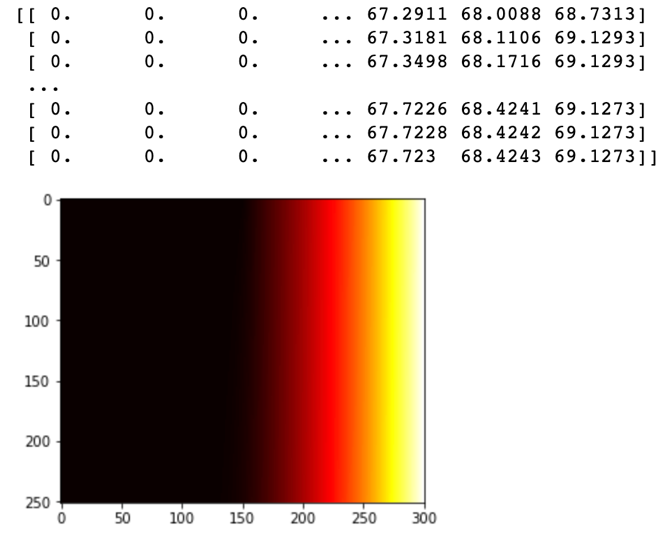
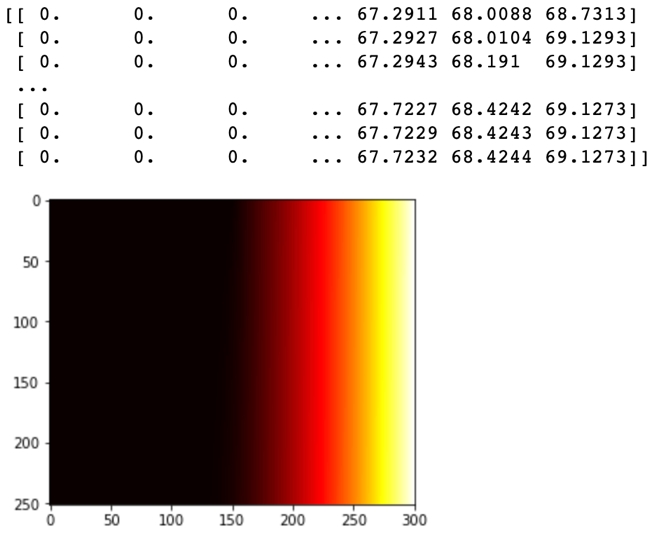
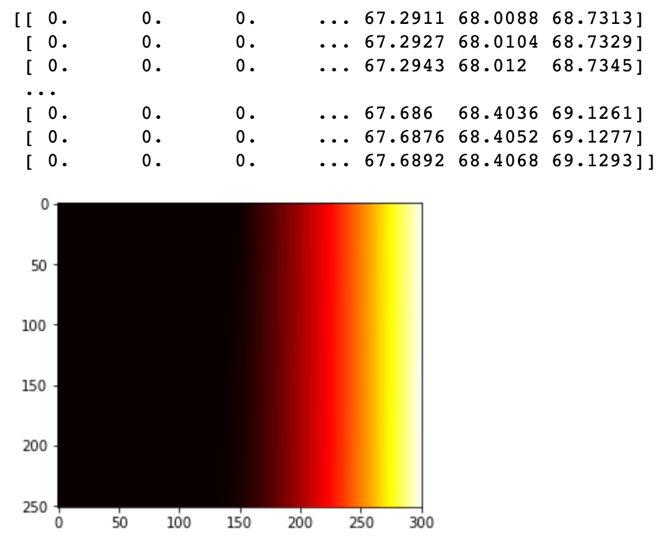
**Vérification et temps d’exécution :**

Maintenant que nous avons développé les schémas d’Euler explicite et implicite pour le calcul d’un Call, nous allons les analyser grâce à des représentations graphiques et des interprétations.

Pour ces résultats, nous utiliserons comme jeu de données (sauf indication contraire) :



Tout d’abord, nous avons voulu savoir si nos résultats générés par Euler implicite et Euler explicite étaient justes. Pour cela, nous avons vérifié que les matrices générées étaient semblables grâce à une représentation graphique utilisant une coloration *cmap.*



(1)

(3)

(2)

Ces vérifications nous permettent d’être confiants sur les résultats obtenus. En effet, les algorithmes d’Euler explicite (2) et d’Euler implicite (3) donne des matrices semblables à la solution exacte (1).

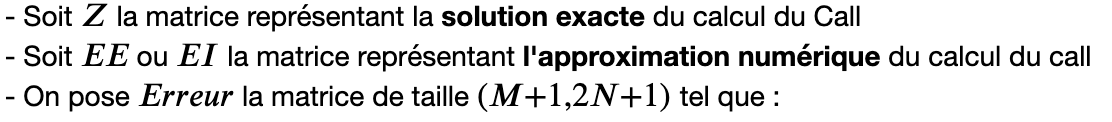
De plus, lors de la génération de ces résultats, nous avons trouvé utile de chronométrer les temps d’exécution. Ceux-ci sont sans-appel et montrent bien l’efficacité des méthodes numériques qui permettent d’avoir une approximation du résultat de manière bien plus rapide :



En effet, les méthodes numériques obtiennent un résultat 60 fois plus rapidement. On remarque également que les 2 méthodes de calculs ont à peu près le même temps d’exécution soit le même coût en calcul. La légère différence vient sûrement de la double boucle dans l’algorithme d’Euler implicite qui avait été évité dans Euler explicite dû à sa simplicité d’implémentation.

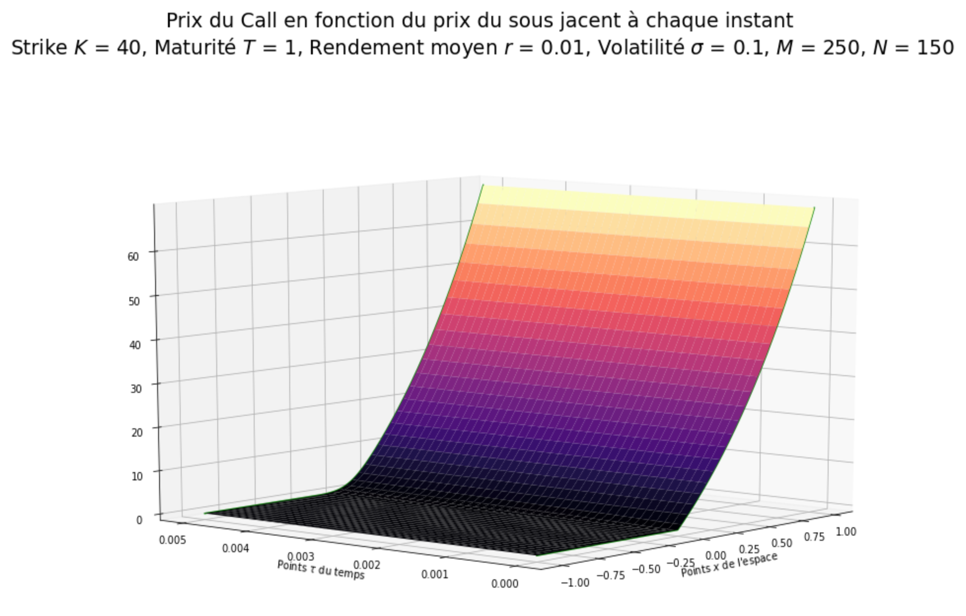
**Erreur sur les schémas d’Euler explicite et implicite :**

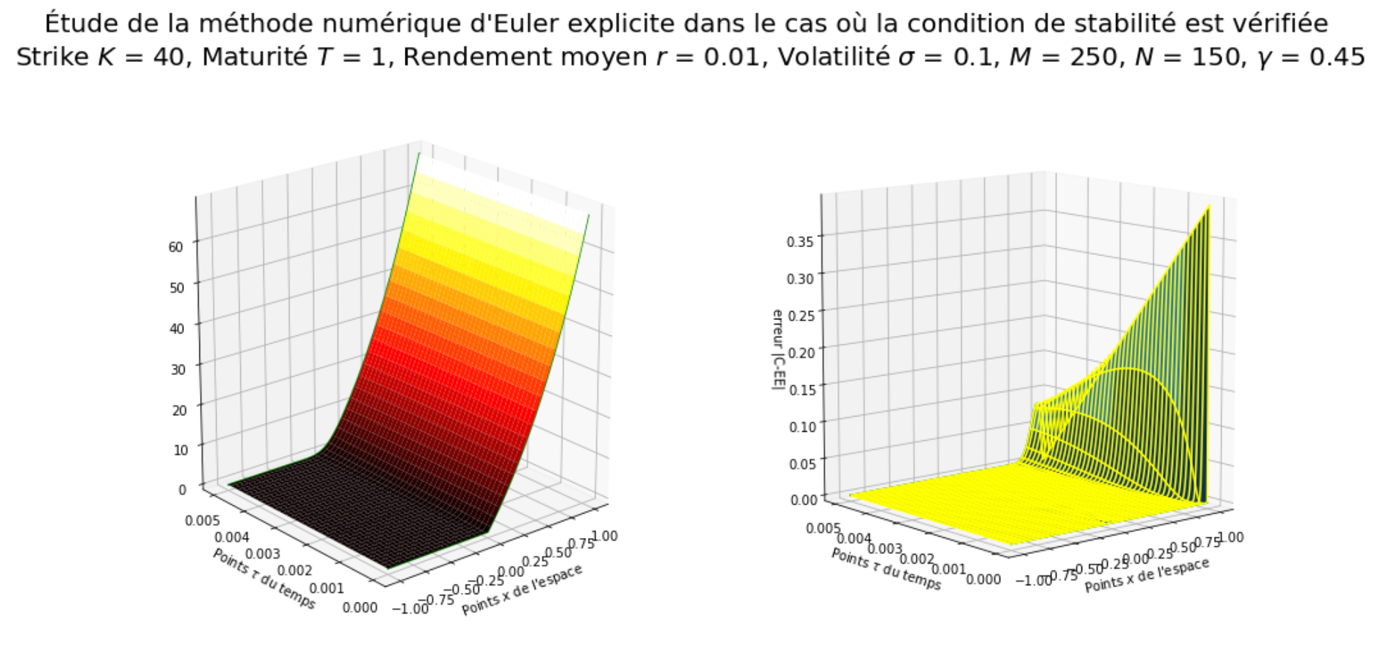
On va maintenant définir l'erreur pour étudier l'évolution de l'écart entre la solution réelle et la solution numérique en fonction du pas de temps et du pas d'espace.

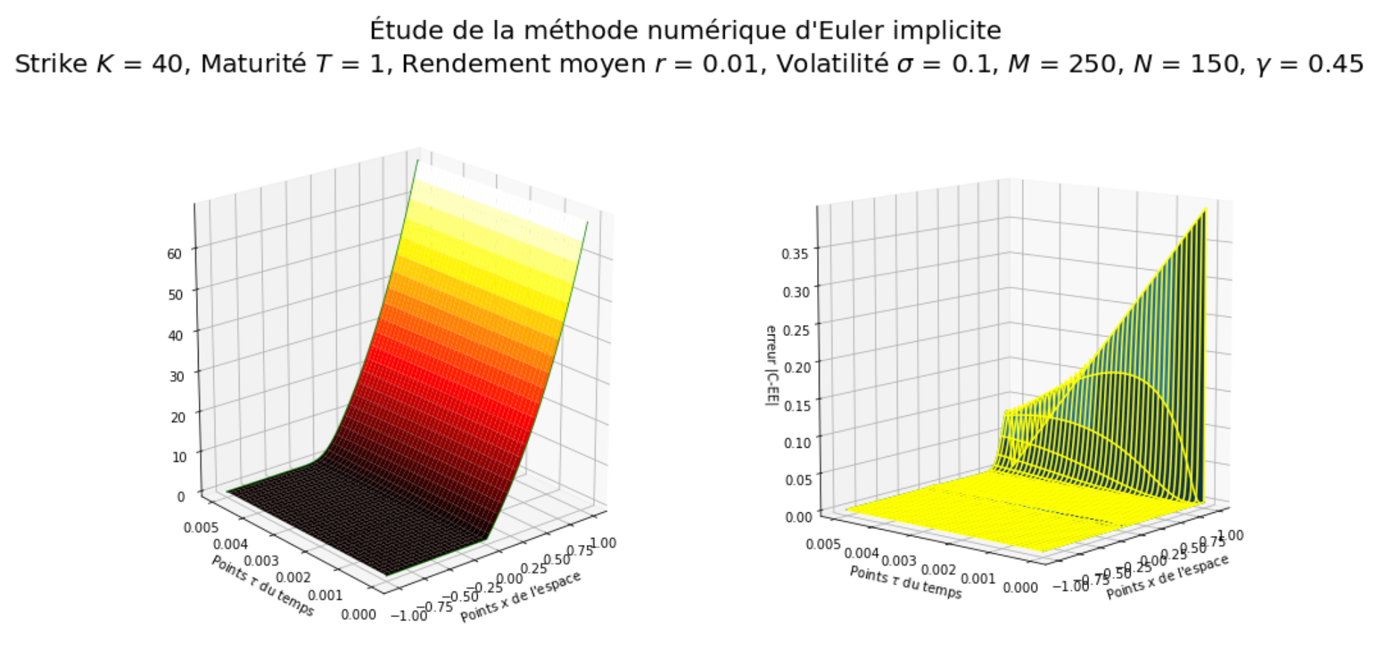




Nous allons utiliser cette matrice d’erreur pour mettre en évidence la distance entre les approximations et la solution exacte, en chaque point du domaine d’étude. Les résultats sont les suivants, sous forme de représentation en 3D :







Sur ces représentations, à gauche on peut voir la modélisation de la solution exacte et à droite la modélisation de notre solution donnée par nos schémas respectifs.

On voit que l’allure en 3D des schémas numériques est la même que celle de la représentation 3D du Call pour la solution exacte.

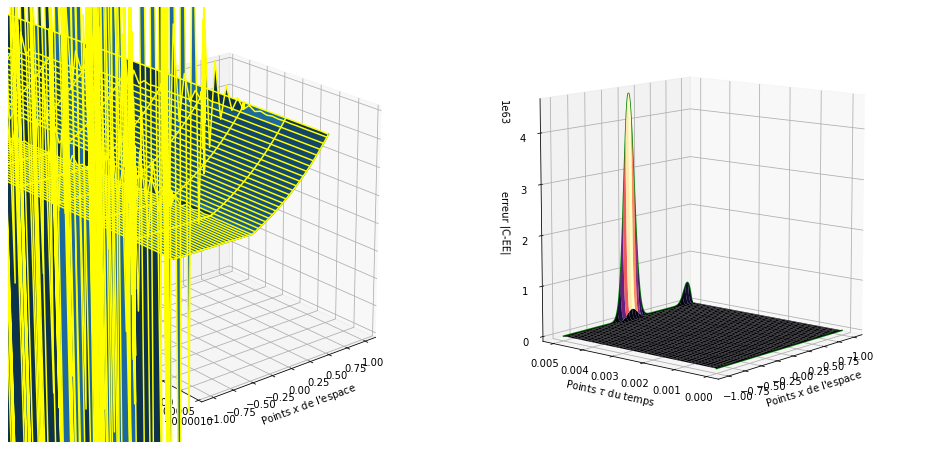
Pour Euler explicite, on note que là où il semble y avoir une erreur plus grande c’est vers la fin des points de l’espace modélisés. Nous pensons que cela est dû à la condition au bord lorsque qui n’est pas proche de la réalité. On a alors une modélisation quelque peu erronée autour de ce bord et plus proche de la solution exacte lorsqu’on s’en éloigne.

Pour Euler implicite, on remarque un pic d’erreur au point x=0 de l’espace.

Les erreurs sont en tout cas similaires pour les deux schémas.

En comparaison, voici une modélisation du rendu d’Euler explicite lorsque la condition de stabilité n’est pas respectée avec l’erreur que cela engendre :





On voit que dans ce cas précis, on est très loin du résultat donné par la solution exacte et l’erreur est immense. Il faut donc faire très attention à la condition de stabilité lorsque l’on utilise le schéma d’Euler explicite.

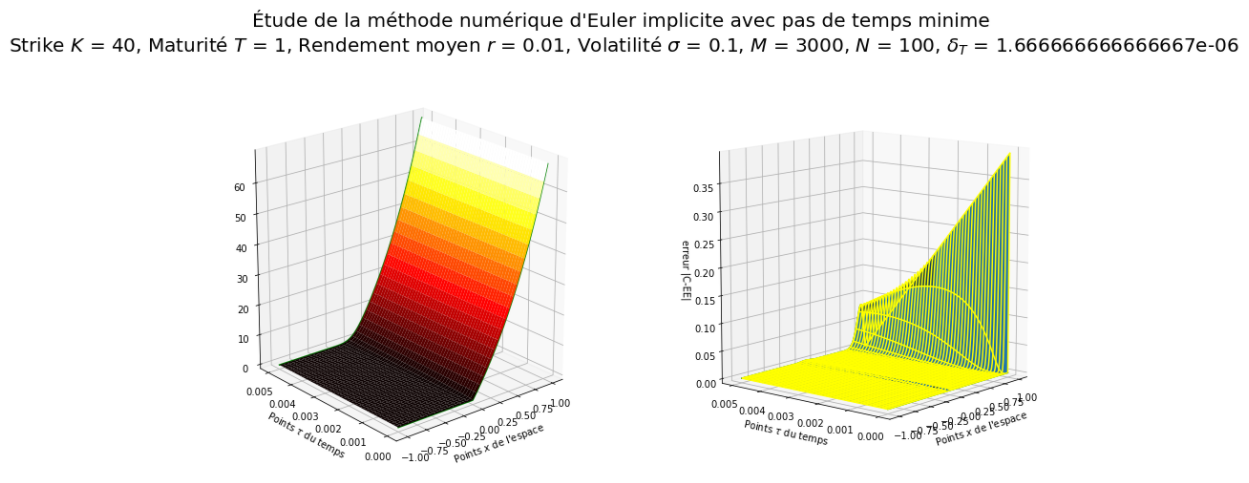
A contrario, le schéma d’Euler implicite étant stable par nature ne nécessite pas de vigilance sur ce point.

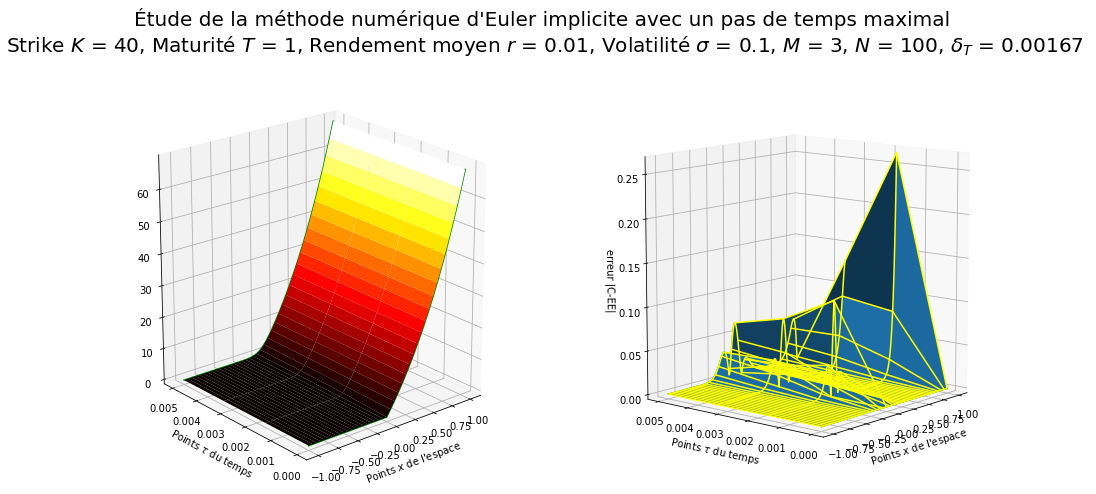
**Evolution de l’erreur en fonction du pas de temps et du pas d’espace :**

Dans cette partie, nous avons concentré nos essais et calculs sur le schéma d’Euler pour ne pas avoir à nous soucier de la condition de stabilité.

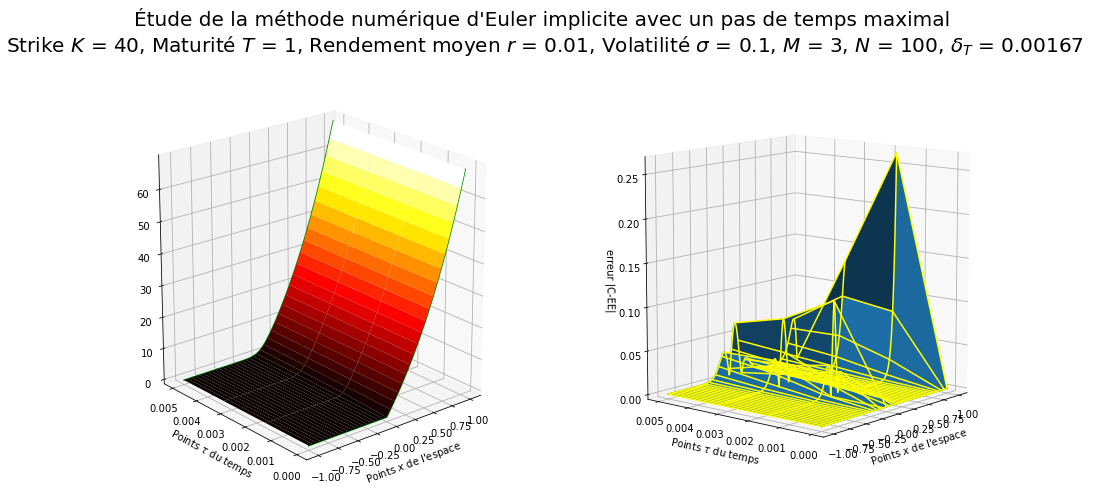
Nous avons ainsi fait varier les pas d’espace et de temps pour pouvoir analyser quels en seraient les impacts.

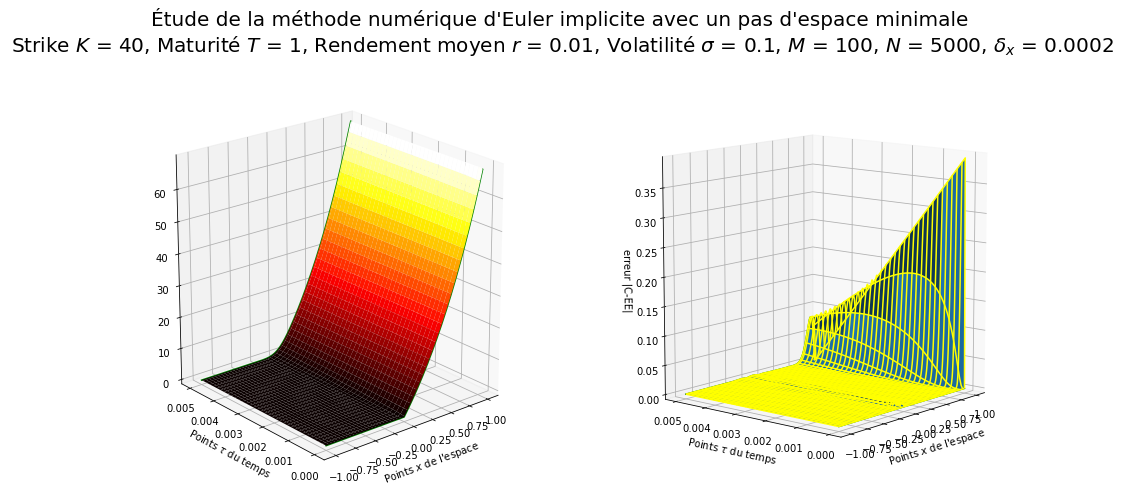
Variation du pas de temps :





Variation du pas d’espace :





**Conclusion :**

Dans ce projet, nous avons utilisé des méthodes numériques pour modéliser le pricing d’une option européenne. Tout d’abord nous avons procédé à une étude théorique de l’équation de Black-scholes, que nous avons ramené à une équation de la chaleur, sans négliger les conditions aux bords, par des habiles changements de variable. Cette transformation nous a permit d’utiliser des schémas numériques afin d’approximer numériquement une solution de notre EDP. Nous avons fait le choix d’employer les schémas d’Euler implicite et explicite, que nous avons implémenté dans python et dont nous avons fait l’analyse des résultats. Il en est ressorti que ces schémas donnent une solution approximée de très bonne qualité (avec une faible erreur) pour un temps de calcul intéressant (10 fois supérieur pour Euler implicite mais ne nécessite en revanche pas de condition de stabilité). Les variations de pas de temps et d’espace ont montré