Projet Mathématiques Informatique

Rapport

Équation de la chaleur

Élèves :
Benjamin Demonsant
Emilie Ferreira
Loïs Gallaud
Nathan Grillet-Niess

Enseignant :
Sylvain Faisan

30 janvier 2023

Table des matières

1 Introduction

La simulation d'une propagation d'onde thermique est utilisée dans de nombreux domaines. Dans le domaine médical, par exemple, une simulation de la sorte peut être utilisée dans le cadre d'une ablation du cœur par radiofréquence. Elle permet de déterminer la puissance à délivrer dans le cathéter pour permettre de nécroser les tissus dans détruire les cellules qui ne sont pas malades. Une telle simulation peut également être utile dans le cas d'une construction ou dans le milieu industriel, il peut être intéressant de connaître la façon dont la température se propage dans certains matériaux.

L'équation étudiée est la suivante :

$$\frac{\partial T}{\partial t} = D.\Delta T \tag{1}$$

Le coefficient de diffusion (D) utilisé pour notre simulation est : $1,1909 \times 10^{-4}\, m^2.s^{-1}$. Ainsi, connaître la répartition de température sur une surface donnée grâce à une simulation est applicable dans de nombreux domaines et nous nous proposons de faire une resolution numérique de l'équation de la chaleur de différentes manières pour pouvoir les comparer.

2 Objectif du projet

Nous cherchons à résoudre numériquement l'équation de la chaleur en 2 dimensions de différentes façons. Pour cela nous considérons une plaque carrée d'un mètre de côté. On se servira de solutions analytiques pour vérifier la justesse et la précision de nos résultats et pour fixer des conditions aux limites qui vont donner, si la résolution est bien faite, une solution de l'équation de la chaleur. On suppose que les propriétés physiques de la plaque sont indépendantes du temps et de l'espace et que le matériau est isotrope.

La température en ses bords est donnée par un jeu de fonctions solution de l'équation de la chaleur. Nous avons ici 3 équations qui nous sont proposées pour nous permettre de vérifier nos simulations :

$$\begin{cases} T(x,y) = a.x + b.y + c.x.y + d \ (x,y \in [0,1]^2)(i) \\ T(x,y) = f.\sin g.x.\exp -g.y \ (x,y \in [0,1]^2)(ii) \\ T(x,y) = k.\sin l.y.\exp -l.x \ (x,y \in [0,1]^2)(iii) \end{cases} \tag{2}$$

Pour la matrice 1 on utilise les coefficients : a=5, b=5, c=20, d=5.

Pour la matrice 2 on utilise les coefficients : g=5 et h=8.

Pour la matrice 3 on utilise les coefficients : k=5 et l=8.

Nous ferons également les simulations du une fonction qui est une combinaison linéaire des 3 premières :

$T(x,y) = a.x + b.y + c.x.y + d + 3.(f.\sin g.x.\exp -g.y) + 2.(k.\sin l.y.\exp -l.x)$

Pour cette matrice 4 on utlilise les coefficients : a=1, b=1, c=10, d=1, g=20, h=10, k=30, l=8.

Nous cherchons par la suite à vérifier la validité de nos simulations, pour cela nous allons comparer les valeurs obtenues avec la simulation à celles théoriques en observant l'écart quadratique moyen entre ces deux matrices.

Langages utilisés
- Python
- Latex

3 Modélisation du problème

Dans un premier temps nous avons discrétisé l'espace et le temps en prenant : Une taille totale de la plaque de 1m de côté comme proposé dans le sujet. La taille de la plaque est donc fixe. On se propose alors d'effectuer un maillage régulier de la plaque pour pouvoir la modéliser par une matrice dont la dimension dépendra du pas de discrétisation. La taille de la matrice variera donc en fonction des simulations faites pour adapter au mieux la taille de la matrice aux simulations faites.

La taille de la matrice sera indiquée avec la lettre N.

3.1 Création de la matrice théorique

Nous avons commencé par la création de la matrice théorique qui va nous servir de référence pour la comparaison. On utilise les 3 équations fournies, décrites précédemment.

On fixe L=1m, une taille de matrice de $N = 10$. Nous choisissons des coefficients pour les fonctions données (a,b, c, d, f, g, k, l).

On affiche les matrices théoriques obtenues par le programme dans la section 8.1 de l'annexe.

Nous pouvons donc observer les différentes formes des matrices.

3.2 Création de la source de chaleur

La source de chaleur correspond à la température constante aux bords de la plaque suivant les fonctions définies précédemment.

La température constante au bord agit donc comme un thermostat sur la plaque et donc impose un profil de température.

Afin d'obtenir les matrices de source de la plaque, nous utilisons les matrices théoriques dont nous mettons à 0 toutes les valeurs n'étant pas sur les bords. On obtient donc les matrices suivantes selon le profil théorique utilisé, que l'on affiche dans la partie 8.2 de l'annexe.

3.3 Comparaison avec la matrice théorique : EQM

L'erreur quadratique moyenne (EQM) permet de vérifier la validité d'un modèle. Cet outil calcule l'écart entre la valeur théorique et la valeur obtenue à la manière d'une norme élevée au carré.

C'est une valeur toujours positive et plus l'EQM est proche de zéro, plus le modèle est fiable et donc précis.

L'EQM s'ecrit :

$$EQM = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\hat{T}(i,j) - T(i\Delta d, j\Delta d))^2 \tag{3}$$

L'analyse de l'EQM nous permettra de vérifier nos simulation et d'observer l'influence de certaines données sur la précision de la simulation.

4 Méthode de résolution en stationnaire

4.1 Définition

Le cas stationnaire consiste à rendre une équation indépendante du temps. On cherche donc à résoudre l'équation :

$$\Delta T = 0$$

4.2 Explication théorique

Pour obtenir un calcul numérisable de l'équation de la chaleur on commence par réaliser un développement limité à l'ordre 2 de la température en position. La démonstration complète est au point 8.3 de l'annexe.

En passant en forme matricielle notre système d'équations obtenues en Annexe partie : On résout Ax = y avec pour A,x et y :

$$A = \begin{pmatrix} D & -I & 0 & \cdots & 0 & 0 & 0 \\ -I & D & -I & \cdots & 0 & 0 & 0 \\ 0 & -I & D & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & D & -I & 0 \\ 0 & 0 & 0 & \cdots & -I & D & -I \\ 0 & 0 & 0 & \cdots & 0 & -I & D \end{pmatrix} \tag{4}$$

Avec I la matrice identitée et D la matrice tridiagonale ci-dessous :

$$D = \begin{pmatrix} d & c & 0 & \cdots & 0 & 0 & 0 \\ c & d & c & \ddots & 0 & 0 & 0 \\ 0 & c & d & \ddots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & d & c & 0 \\ 0 & 0 & 0 & \cdots & c & d & c \\ 0 & 0 & 0 & \cdots & 0 & c & d \end{pmatrix} \tag{5}$$

Avec $c = -1$ et $d = 4$.

Pour y on a :

$$y = \begin{pmatrix} r_{0,1} + r_{1,0} \\ r_{0,2} \\ r_{0,3} \\ \vdots \\ r_{m,m-3} \\ r_{m,m-2} \\ r_{m,m-1} + r_{m-1,m} \end{pmatrix} \tag{6}$$

et x :

$$x = \begin{pmatrix} T_{1,1} \\ T_{1,2} \\ r_{1,3} \\ \vdots \\ T_{m-1,m-3} \\ T_{m-1,m-2} \\ T_{m-1,m-1} \end{pmatrix} \tag{7}$$

On effectue alors la résolution du système linéaire avec un algorithme du type Gauss LU contenu dans la fonction resolve de la librairie Numpy.

4.3 Construction du code

Pour implanter la théorie précédente en informatique. On doit tout d'abord calculer la matrice A peut importe la taille de notre matrice initiale. Pour cela on a observé qu'elle était tridiagonale par blocs. Par ailleurs, le bloc nommé D est aussi une matrice tridiagonale.

Pour cela, on crée une fonction dans laquelle on construit les blocs composants de la matrice A à partir de matrices vides carrées de taille N-2 c'est-à-dire la taille de notre matrice initiale retranchée des bords où on connait la température quelque soit t. Pour se faire, on effectue des boucles for afin de remplir les matrices.

Il reste à construire notre matrice A de taille $(N-2)^2 \times (N-2)^2$ et la remplir à l'aide de nos blocs en jouant sur les indices des éléments de la matrice A.

Il faut aussi construire la matrice Y, Pour cela on crée une nouvelle fonction où on crée une matrice de taille $(N-2)^2 \times 1$ qu'on remplit avec les coefficients nécessaires, soit en récupérant les éléments dans la matrice source.

4.4 Résultats

On fixe la taille de la matrice à 10, pour limiter le temps de calcul. On obtient les résultats affichés en annexe (voir section 8.4).

On observe que les matrices obtenues sont identiques visuellement et numériquement avec cette méthode, nous le vérifierons par la suite de manière plus précise lors du calcul de l'EQM .

4.5 Utilisation de l'EQM, notion de précision et temps de calcul

Pour qantifier la précision obtenue avec le programme on va faire appel à l'écart quadratique moyen. De cette façon on obtient une comparaison chiffrée entre la matrice qu'on devrait obtenir et celle obtenue numériquement. Plus l'EQM entre les deux matrices est faible plus la matrice obtenue est précise. Pour la méthode stationnaire on peut faire varier la taille de la matrice et donc le pas de discrétisation. On obtient alors :

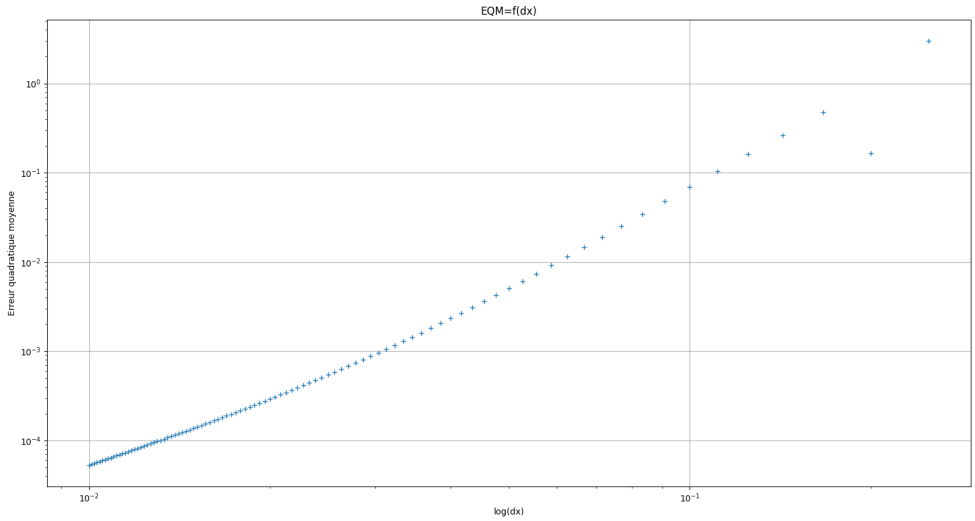

FIGURE 1 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Echelle "loglog"

On voit bien ici que plus dx est petit plus la précision obtenue est grande. On remarque cependant que la précicion peut varier de façon inattendue en certains points et que la précision obtenue ne dépend que des calculs effectués. Ainsi, par exemple, pour $dx = 0.01$ et donc une matrice de taille 100 on obtient une précision de $5e - 5$.

On observe pour chaques tailles de matrices les temps de calculs suivants :

En augmentant la taille de la matrice calculée (par extension : en diminuant dx), le résultat obtenu est de plus en plus précis, cependant, le temps de calcul augmente considérablement. On peut supposer que cela provient du nombre de calcul effectué : $(N-2)^2$ qui devient alors très grand (Ex : 9604 Equations à résoudre pour une matrice de taille 100).

4.6 Analyse de la méthode

La résolution numérique de l'équation stationnaire est donc très précise pour une petite discrétisation mais le temps de calcul augmente très rapidement et la précision obtenue dépend des conditions initiales. On le voit ici avec la courbe d'EQM réalisée de la même manière mais avec d'autres conditions initiales.

On voit bien que l'allure est différente. On ne peut donc que estimer l'erreur commise lors des calculs.

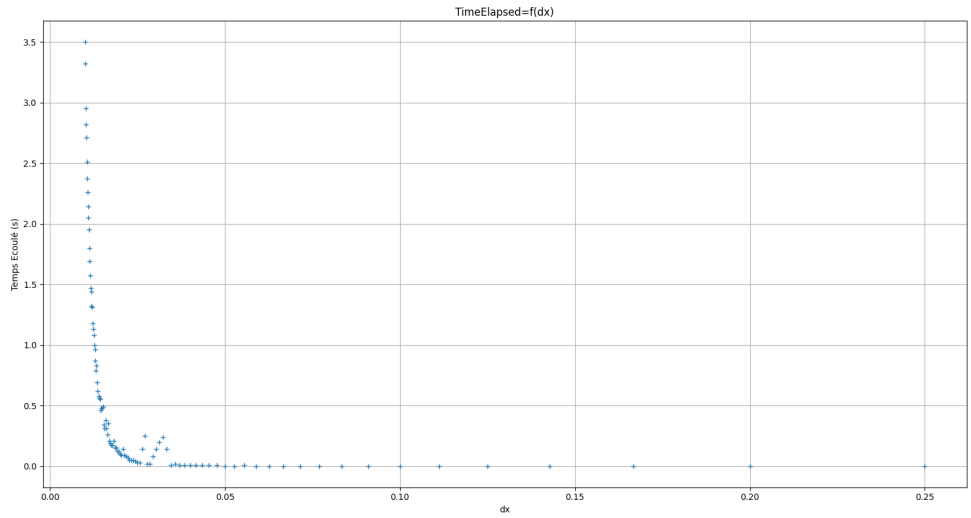

FIGURE 2 – Temps de calcul (s) en fonction du pas de discrétisation.

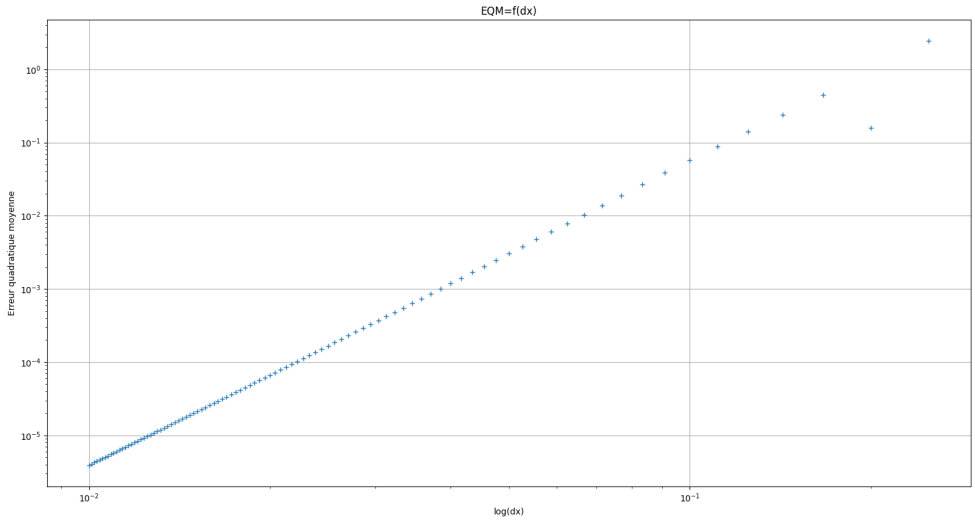

FIGURE 3 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Utilisation du jeu de donné numéro 2

5 Méthode temporelle : Schéma explicite

5.1 Explication du principe

Le schéma explicite dans une simulation numérique consiste à calculer des solutions d'une fonction à l'instant t+dt à partir des valeurs de la fonction à l'instant précédent, t.

5.2 Explication théorique

Pour la méthode explicite on reprend 1 en y injectant les discrétisations 19 et 18.
On obtient alors l'équation :

$$\frac{1}{\delta t}.(T_{i,j}(t+dt) - T_{i,j}(t)) = \frac{D}{\delta x^2}.(\sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} (T_{i+k,j+l}(t)) - 4.T_{i,j}(t))$$

et donc on retrouve après calculs en annexe partie 2 une équation de la forme :

$$Y_{m-2}^{n+1} = A_{m-2}.Y_{m-2}^{n} + \alpha.B$$

avec A :

$$A_{m-2} = \begin{pmatrix} D & I & 0 & \cdots & 0 & 0 & 0 \\ I & D & I & \cdots & 0 & 0 & 0 \\ 0 & I & D & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & D & I & 0 \\ 0 & 0 & 0 & \cdots & I & D & I \\ 0 & 0 & 0 & \cdots & 0 & I & D \end{pmatrix} \tag{8}$$

Avec $I = \alpha.Id$ et D :

$$D = \begin{pmatrix} d & c & 0 & \cdots & 0 & 0 & 0 \\ c & d & c & \ddots & 0 & 0 & 0 \\ 0 & c & d & \ddots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & d & c & 0 \\ 0 & 0 & 0 & \cdots & c & d & c \\ 0 & 0 & 0 & \cdots & 0 & c & d \end{pmatrix} \tag{9}$$

Pour $c = +\alpha$ et $d = 1 - 4\alpha$.

Et enfin avec

$$B = \begin{pmatrix} r_{0,1} + r_{1,0} \\ r_{0,2} \\ r_{0,3} \\ \vdots \\ r_{m,m-3} \\ r_{m,m-2} \\ r_{m,m-1} + r_{m-1,m} \end{pmatrix} \tag{10}$$

Y_{m-2}^{n+1} et Y_{m-2}^{n} sont les vecteurs composés des températures à l'instant t et t+dt pour les coordonnées à la manière de 7.

5.3 Création des matrices utilisées dans le système linéaire

Pour créer les matrices, de la même manière qu'en stationnaire, la matrice A est une matrice tridiagonale par blocs, il s'agit donc de construire dans une fonction les blocs composants de A et de construire A à partir de ces blocs. Pour cela on crée des matrices vides carrées de taille (N-2) et on remplit les matrices à l'aide de boucles for. On construit par la suite une matrice vide de taille $(N-2)^2 \times (N-2)^2$ et on remplit cette nouvelle matrice avec les blocs calculés précédemment à l'aide de boucles for.

Il faut aussi construire la matrice $\alpha.B$. Pour cela on crée une nouvelle fonction où on crée une matrice de taille $(N-2)^2 \times 1$ qu'on remplit avec les coefficients nécessaires, soit en récupérant les éléments dans la matrice source.

5.4 Code de la propagation

A partir des matrices créées précédemment, on effectue une boucle correspondant au nombre de points dans le temps, dans laquelle on calcule :

$$Y_{m-2}^{n+1} = A_{m-2}.Y_{m-2}^{n} + \alpha.B$$

Pour cela on utilise le module numpy pour effectuer le produit matriciel (numpy.dot) puis le programme nous permet de calculer le vecteur Y, où on utilise la commande numpy.reshape pour former une matrice carrée des solutions. A partir des valeurs inconnus, on remplit la matrice contenant les valeurs constantes aux bords avec ces valeurs afin d'obtenir une matrice complète de la simulation correspondant à la matrice après propagation du système.

5.5 Résultats

Nous avons simulé avec cette méthode explicite les 4 matrices, nous obtenons les matrices suivantes (voir annexe section 8.6) pour les paramètres dt=0.1, dx=0.1, $NT = 10^{-4}$ et la taille de la matrice est de 80.

On obtient donc une répartition de température sur l'ensemble de la plaque qui se stabilise et qui à vue d'œil ressemble à la matrice initiale. Nous allons vérifier avec l'EQM la précision de cette méthode.

5.6 Tentatives avec différents ALPHA

Avec plusieurs simulations, nous observons des résultats aberrants lorsque le facteur alpha est supérieur à $\frac{1}{4}$, en prenant pour valeur de ce coefficient 0,26 nous observons les matrices suivantes, avec les paramètres : TAILLE=80, dt=0.35, **ALPHA=0.26**, dx=1/80,

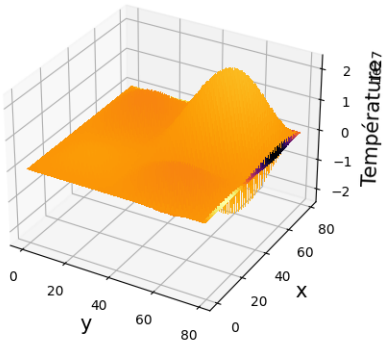

FIGURE 4 – Matrice avec la méthode explicite 1, Vue 3D

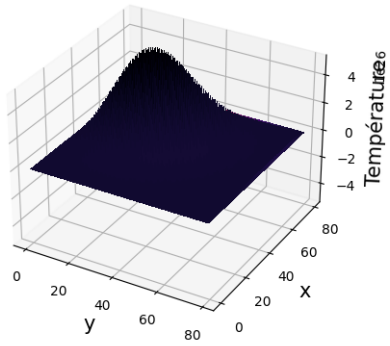

FIGURE 5 – Matrice avec la méthode explicite 2, Vue 3D

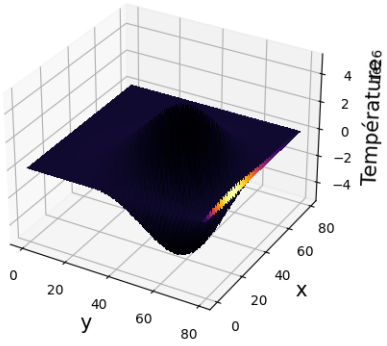

FIGURE 6 – Matrice avec la méthode explicite 3, Vue 3D

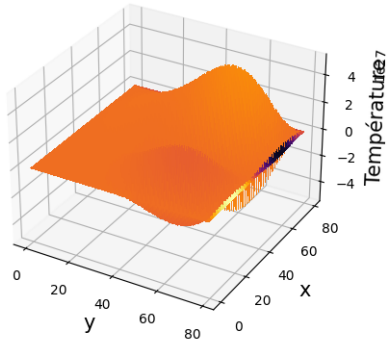

FIGURE 7 – Matrice avec la méthode explicite 4, Vue 3D

Alpha est donc un coefficient de stabilité, en effet si ce coefficient est trop élevé, les valeurs dans les matrices obtenues divergent et sont de l'ordre de 10^{26} ou 10^{27} ce qui est bien trop élevé pour les températures que nous espérions. Ainsi, cette méthode à une première limite : sa stabilité n'est pas garantie.

5.7 Précision obtenue avec l'EQM dans la zone de stabilité de la méthode

Ici on ne va vérifier que la précision obtenue avec l'EQM en fonction du nombre d'itération correspondantes au pas de temps. On obtient pour une taille fixée de matrice (N = 50) avec un pas de temps de $dx = 0.1$ soit $\alpha = \frac{D.\delta t}{\delta x^2} = 0.0286 << 0.25$:

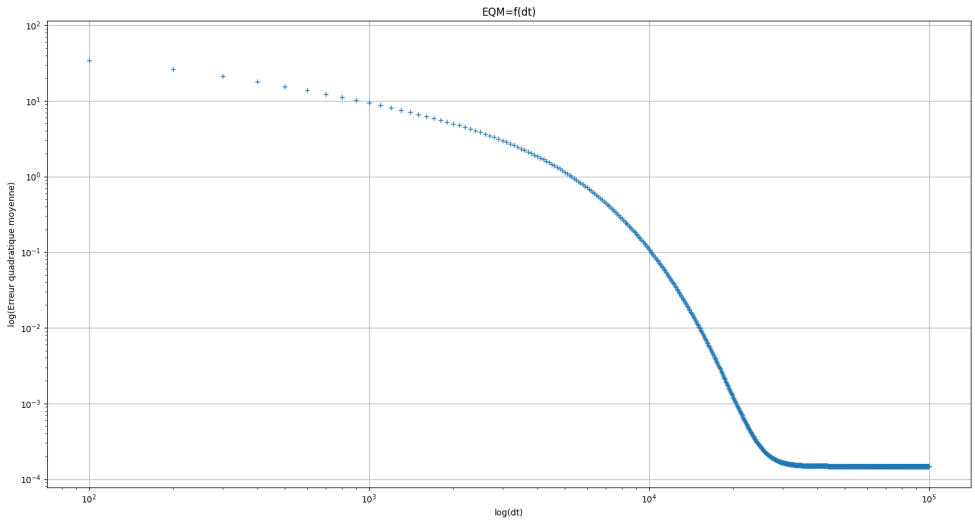

FIGURE 8 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Echelle "loglog"

$\Delta t = Nt.\delta t$ $5e4.0,1 = 5e3s$ est le temps écoulé avant convergence pour la matrice genérée par f5. On remarque, de plus, qu'une précision limite est atteinte à partir d'un certain rang quelque soient les conditions initiales. On peut alors se demander si on observe là les limites de notre méthode de discrétisation.

Pour ce qui est du temps de calcul, on peut visualiser à l'aide du programme qu'il est constant pour chaque étape de calcul et souvent assez faible. On peut donc contrôler d'une certaine façon le temps de calcul et selectionner la précision souhaitée sur la gamme de précision atteignable.

6 Méthode temporelle : Schéma implicite

6.1 Explication du principe

Le schéma implicite dans une simulation numérique consiste à calculer des solutions d'une fonction à l'instant t+dt à partir des valeurs de la fonction à l'instant précédent, t mais aussi à l'instant t+dt.

6.2 Explication théorique

Pour la méthode implicite, la théorie (voir détail partie 8.7 de l'annexe) reprend les idées de la théorie explicite mais on va calculer l'instant $T_{i,j}(t+dt)$ à partir des points adjacent au temps $t+dt$ lui aussi. Ce qui nous donne l'équation :

$$\frac{1}{\delta t}.(T_{i,j}(t+dt) - T_{i,j}(t)) = \frac{D}{\delta x^2}.(\sum_{\substack{k,l=-1 \\ \backslash\{(k=0,l=0)\}}}^{1} (T_{i+k,j+l}(t+dt)) - 4.T_{i,j}(t+dt))$$

avec la notation précédente, les calculs en annexe 3 et le rassemblement des inconnues on obtient une équation de la forme :

$$A_{m-2}.Y_{m-2}^{n+1} = Y_{m-2}^{n} + \alpha.B$$

avec A :

$$A_{m-2} = \begin{pmatrix} D & -I & 0 & \cdots & 0 & 0 & 0 \\ -I & D & -I & \cdots & 0 & 0 & 0 \\ 0 & -I & D & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & D & -I & 0 \\ 0 & 0 & 0 & \cdots & -I & D & -I \\ 0 & 0 & 0 & \cdots & 0 & -I & D \end{pmatrix} \quad (11)$$

Avec $I = \alpha.Id$ et D :

$$D = \begin{pmatrix} d & c & 0 & \cdots & 0 & 0 & 0 \\ c & d & c & \ddots & 0 & 0 & 0 \\ 0 & c & d & \ddots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & d & c & 0 \\ 0 & 0 & 0 & \cdots & c & d & c \\ 0 & 0 & 0 & \cdots & 0 & c & d \end{pmatrix} \quad (12)$$

Pour $c = -\alpha$ et $d = 1 + 4\alpha$.

Et enfin avec

$$B = \begin{pmatrix} r_{0,1} + r_{1,0} \\ r_{0,2} \\ r_{0,3} \\ \vdots \\ r_{m,m-3} \\ r_{m,m-2} \\ r_{m,m-1} + r_{m-1,m} \end{pmatrix} \quad (13)$$

Y_{m-2}^{n+1} et Y_{m-2}^{n} sont les vecteurs composés des températures à l'instant t et t+dt pour les coordonnées à la manière de 7.

6.3 Création des matrices utilisées dans le système linéaire

Pour créer les matrices, de la même manière que les autres méthodes, la matrice A est une matrice tridiagonale par blocs, il s'agit donc de construire dans une fonction les blocs composants de A et de construire A à partir de ces blocs. Pour cela on crée des matrices vides carrées de taille (N-2) et on remplit les matrices à l'aide de boucles for. On construit par la suite une matrice vide de taille $(N-2)^2 \times (N-2)^2$ et on remplit cette nouvelle matrice avec les blocs calculés précédemment à l'aide de boucles for.

Il faut aussi construire la matrice $\alpha.B$. Pour cela on crée une nouvelle fonction où on crée une matrice de taille $(N-2)^2 \times 1$ qu'on remplit avec les coefficients nécessaires, soit en récupérant les éléments dans la matrice source.

6.4 Code de la propagation

A partir des matrices créées précédemment, on effectue une boucle correspondant au nombre de points dans le temps, dans laquelle on calcule :

$$Y_{m-2}^{n+1} = A_{m-2}^{-1}.Y_{m-2}^{n} + A_{m-2}^{-1}.\alpha.B$$

Pour cela on utilise le module numpy pour effectuer les produits matriciels (np.dot) mais aussi l'inversion (np.linalg.inv) afin d'inverser la matrice A. Puis, comme on calcule un vecteur, on utilise np.reshape pour former une matrice carrée des solutions. On remplit finalement la matrice contenant les valeurs constantes aux bords avec ces valeurs afin d'obtenir une matrice complète de la simulation correspondant à la matrice après propagation du système.

6.5 Résultats

Nous avons simulé avec cette méthode implicite les 4 matrices, nous obtenons les matrices suivantes pour les paramètres dt=0,1, dx=0,1, $NT = 10^6$

Les matrices sont affichées en annexe (voir 8.8)

On obtient donc une répartition de température qui ressemble à celle trouvée pour la matrice théorique. Nous allons vérifier la précision de la méthode en traçant l'EQM pour cette méthode.

6.6 Tentatives avec différents ALPHA

Avec plusieurs tentatives, nous observons qu'avec la méthode implicite le modèle est très stable. En effet, nous observons des résultats cohérents même au-delà de $\alpha = 0,26$. Cela implique une liberté de choix notamment sur le choix du pas d'espace qui peut être augmenté sans altérer la stabilité de la simulation.

6.7 Précision obtenue avec l'EQM pour cette méthode

On se place dans les conditions suivantes : Taille de matrice N=50 et on regarde pour $dt = 0.1$ la courbe obtenue :

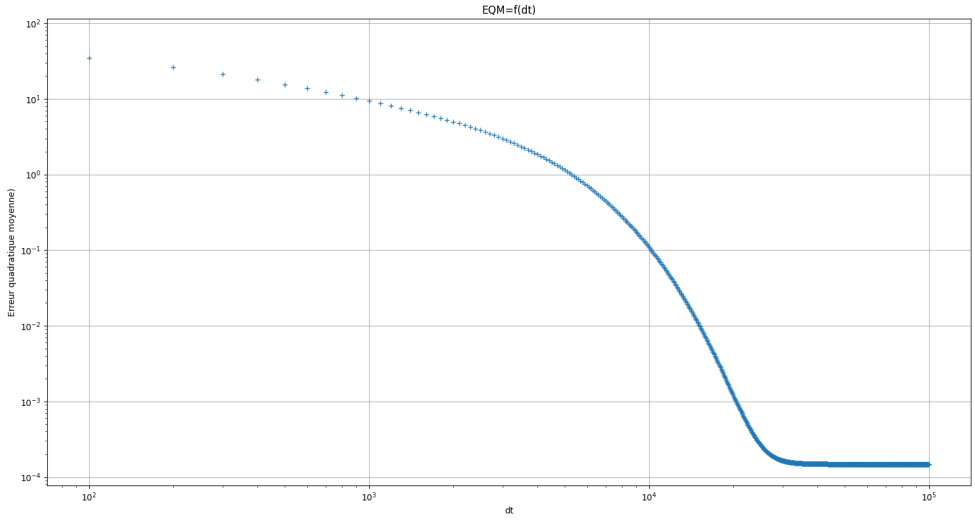

FIGURE 9 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Echelle "loglog"

On remarque alors que la courbe obtenue est identique à celle faite avec la méthode explicite on peut alors augmenter dt pour tester l'efficacité de la méthode :

On obtient bien la même courbe avec une convergence 100 fois plus rapide en Nt. Le temps de calcul étant sensiblement le même pour chaque Nt en explicite et en implicite. On peut donc se demander, sachant que le resultat (précision et convergence) est le même en implicite qu'en explicite si la précision dépend des conditions initiales. On teste alors pour N=50 et pour dt=10 (afin de converger rapidement) avec les conditions de la matrice 4 et de la matrice 2 les resultats :

On constate alors que la précision limite atteinte dépend des conditions initiales. On trace cette précision limite en fonction de la taille de la matrice.

Une précision maximale limite semble alors se déssiner.

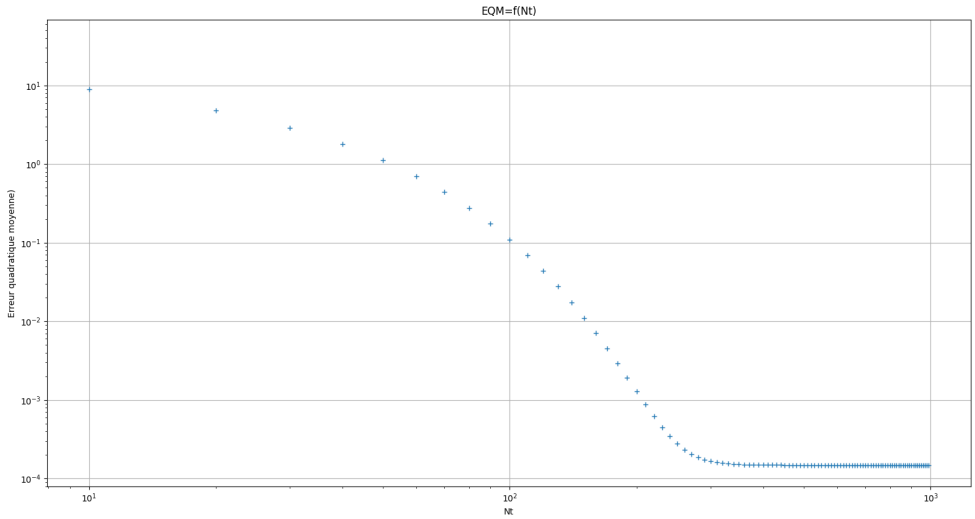

FIGURE 10 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Echelle "loglog"

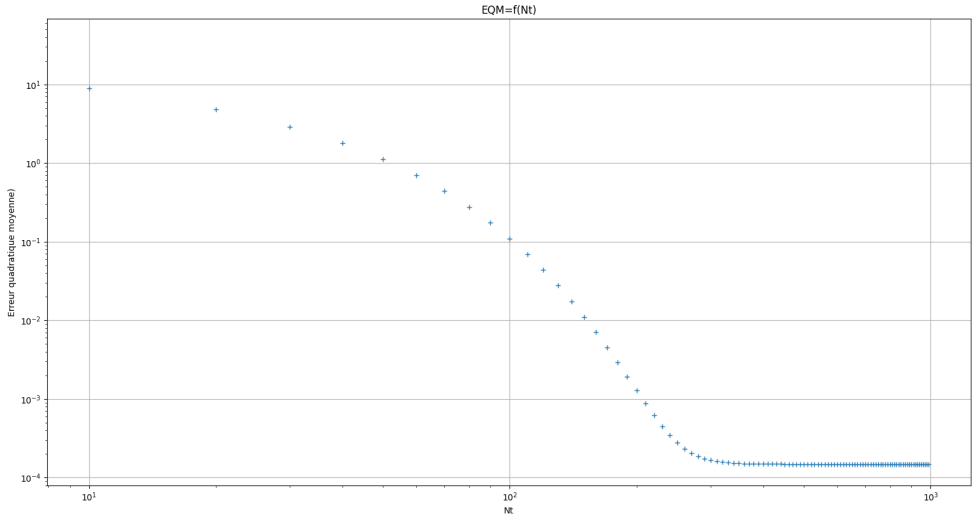

FIGURE 11 – Ecart Quadratique Moyen en fonction du pas de discrétisation. Echelle "loglog"

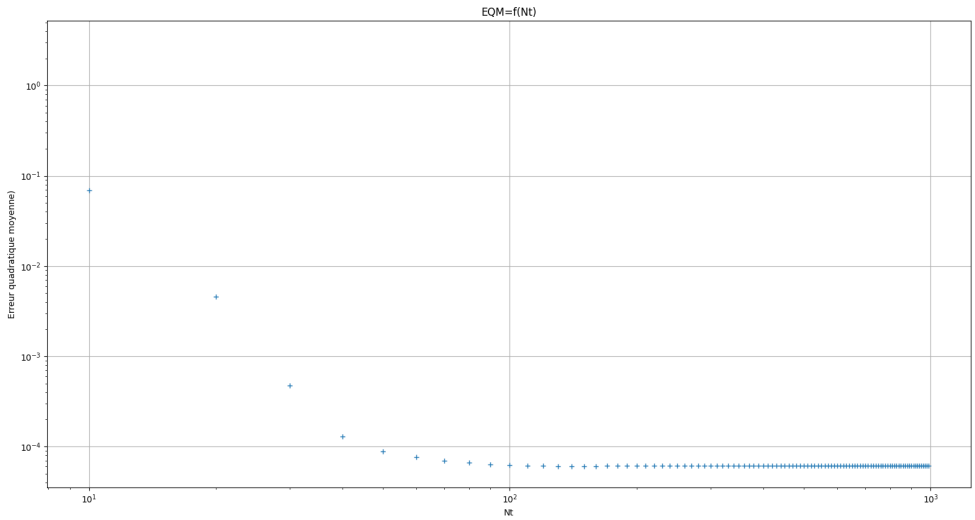

FIGURE 12 – Ecart Quadratique Moyen en fonction du nombre d'itérations temporelles. Echelle "loglog"

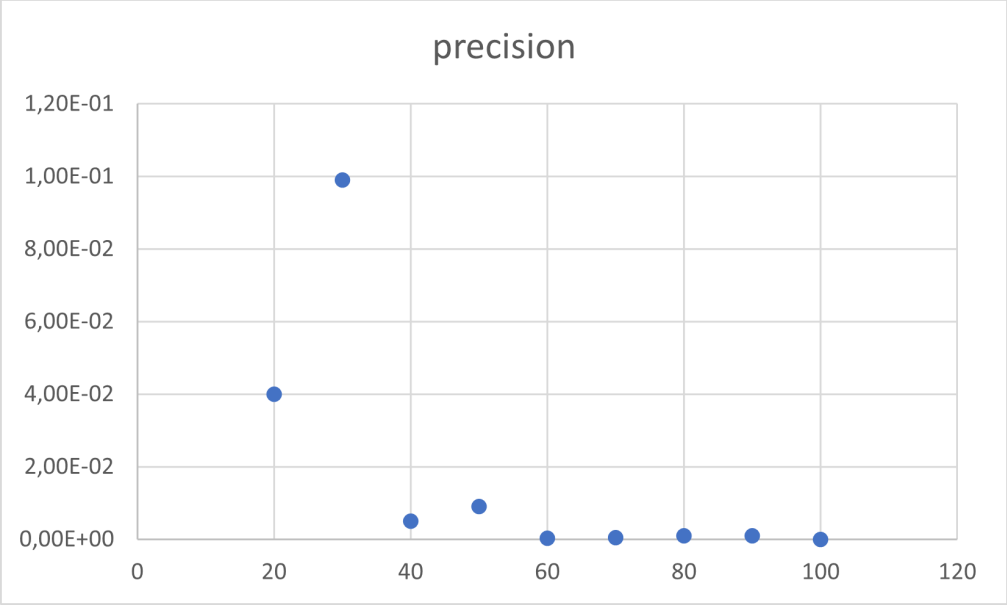

FIGURE 13 – Precision maximale obtenue en fonction de la taille de la matrice 4

7 Conclusion

Nous avons donc obervé trois méthodes différentes et les résultats obtenus nous ammène à la conclusion suivante : La méthode de résolution stationnaire possède un temps de calcul qui augmente exponentiellement avec la taille de la matrice mais on obtient une précision interréssante pour des grandes tailles de matrice (augmentation presque linéaire). Le grand problème de cette méthode est alors la précision qui ne peut être déterminée à l'avance car elle dépend des conditions initiales. La méthode explicite peut palier au problème de précision car on peut selectionner la précision désirée et on peut donc contrôler le temps de calcul par la même occasion. Un nouveau problème se pose alors, une précision limite semble se dessiner dans la méthode explicite et apparaît également dans la méthode implicite. On peut supposer que cette erreur systématique provient du modèle utilisé pour la discrétisation du problème. En faisant cette hypothèse on remarque alors que la précision obtenue en stationnaire n'est pas identique et est supérieure à la precision limite. On a donc un argument de plus en faveur des méthodes temporelles qui assurent une précision maximale en dépit d'un grand temps de calcul.

Pour ce qui est de la stabilité on remarque donc que la précision en stationnaire est plutôt bonne mais liée a des paramêtres indéterminés. La méthode explicite possède un critère de convergence assez contraignant : pour $\alpha < \frac{1}{4}$ la matrice diverge. On ne peut donc pas avoir un dx petit et un dt grand à la fois. Ce qui n'est pas le cas pour la méthode implicite qui permet des α très divers. On choisira donc préférentiellement la méthode stationnaire pour avoir un résultat rapide dans le cas où la précision n'est pas un soucis, on prendra la méthode explicite pour des faibles valeurs de α car la précision est contrôlable (mais le temps de calcul croit avec la précision) et on prendra la méthode implicite pour résoudre des systèmes avec des α quelconques rapidement en sachant que la précision sera plus faible pour des α plus faible.

8 Annexe

Toutes les simulations avec les différents coefficients peuvent être refaites avec le programme fourni. Un README écrit en markdown contenant le tutoriel d'utilisation de l'interface graphique tkinter est mis à la disposition de l'utilisateur.

8.1 Matrices Théoriques

Matrice numéro 1 :

```
[[ 5.   5.5  6.   6.5  7.   7.5  8.   8.5  9.   9
 [ 5.5  6.2  6.9  7.6  8.3  9.   9.7 10.4 11.1 11
 [ 6.   6.9  7.8  8.7  9.6 10.5 11.4 12.3 13.2 14
 [ 6.5  7.6  8.7  9.8 10.9 12.  13.1 14.2 15.3 16
 [ 7.   8.3  9.6 10.9 12.2 13.5 14.8 16.1 17.4 18
 [ 7.5  9.  10.5 12.  13.5 15.  16.5 18.  19.5 21
 [ 8.   9.7 11.4 13.1 14.8 16.5 18.2 19.9 21.6 23
 [ 8.5 10.4 12.3 14.2 16.1 18.  19.9 21.8 23.7 25
 [ 9.  11.1 13.2 15.3 17.4 19.5 21.6 23.7 25.8 27
 [ 9.5 11.8 14.1 16.4 18.7 21.  23.3 25.6 27.9 30
```

FIGURE 14 – Matrice numérique théorique 1

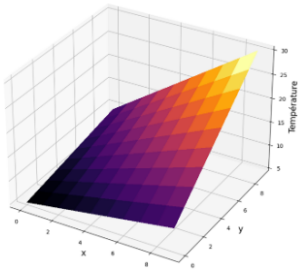

FIGURE 15 – Matrice théorique 1, Vue 3D 1

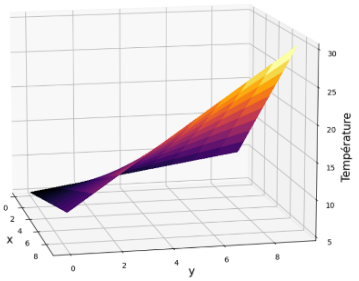

FIGURE 16 – Matrice théorique 1, Vue 3D 2

Matrice numéro 2 :

```
[[ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 3.59  1.61  0.72  0.33  0.15  0.07  0.03  0.01  0.01  0.  ]
 [ 5.    2.25  1.01  0.45  0.2   0.09  0.04  0.02  0.01  0.  ]
 [ 3.38  1.52  0.68  0.31  0.14  0.06  0.03  0.01  0.01  0.  ]
 [-0.29 -0.13 -0.06 -0.03 -0.01 -0.01 -0.   -0.   -0.   -0.  ]
 [-3.78 -1.7  -0.76 -0.34 -0.15 -0.07 -0.03 -0.01 -0.01 -0.  ]
 [-4.98 -2.24 -1.01 -0.45 -0.2  -0.09 -0.04 -0.02 -0.01 -0.  ]
 [-3.16 -1.42 -0.64 -0.29 -0.13 -0.06 -0.03 -0.01 -0.01 -0.  ]
 [ 0.58  0.26  0.12  0.05  0.02  0.01  0.    0.    0.    0.  ]
 [ 3.97  1.78  0.8   0.36  0.16  0.07  0.03  0.01  0.01  0.  ]]
```

FIGURE 17 – Matrice munérique théorique 2

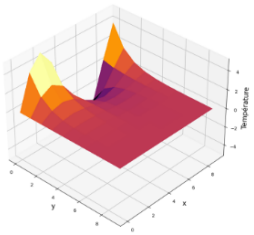

FIGURE 18 – Vue 3D

Matrice numéro 3 :

```
[[ 0.    3.59  5.    3.38 -0.29 -3.78 -4.98 -3.16  0.58  3.97]
 [ 0.    1.61  2.25  1.52 -0.13 -1.7  -2.24 -1.42  0.26  1.78]
 [ 0.    0.72  1.01  0.68 -0.06 -0.76 -1.01 -0.64  0.12  0.8 ]
 [ 0.    0.33  0.45  0.31 -0.03 -0.34 -0.45 -0.29  0.05  0.36]
 [ 0.    0.15  0.2   0.14 -0.01 -0.15 -0.2  -0.13  0.02  0.16]
 [ 0.    0.07  0.09  0.06 -0.01 -0.07 -0.09 -0.06  0.01  0.07]
 [ 0.    0.03  0.04  0.03 -0.   -0.03 -0.04 -0.03  0.    0.03]
 [ 0.    0.01  0.02  0.01 -0.   -0.01 -0.02 -0.01  0.    0.01]
 [ 0.    0.01  0.01  0.01 -0.   -0.01 -0.01 -0.01  0.    0.01]
 [ 0.    0.    0.    0.   -0.   -0.   -0.   -0.    0.    0.  ]]
```

FIGURE 19 – Matrice numérique théorique 3

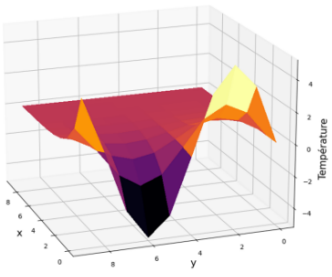

FIGURE 20 – Vue 3D

Matrice numéro 4

```
[[  1.   44.1  61.2  41.8  -2.1 -43.9 -58.2 -36.2   8.8  49.5]
 [ 51.6  39.6  36.1  23.6   2.9 -16.  -22.  -11.7   9.1  27.9]
 [ 55.8  31.1  22.9  15.4   5.9  -2.1  -4.1   1.3  11.4  20.7]
 [  9.8   9.9  11.1  10.2   7.5   5.2   5.5   9.1  14.7  20. ]
 [-44.  -11.5   1.9   7.1   8.8   9.7  11.5  14.5  18.5  22.2]
 [-56.  -16.3   0.    7.2  10.8  13.3  15.9  19.   22.4  25.8]
 [-15.2  -1.1   6.   10.4  13.7  16.6  19.7  23.   26.5  29.9]
 [ 41.1  20.   14.5  14.6  16.8  19.8  23.2  26.8  30.5  34.3]
 [ 61.2  27.8  18.1  17.1  19.3  22.6  26.4  30.5  34.6  38.8]
 [ 26.6  15.6  14.5  17.   20.8  25.   29.5  34.1  38.7  43.3]]
```

FIGURE 21 – matrice théorique numérique 4

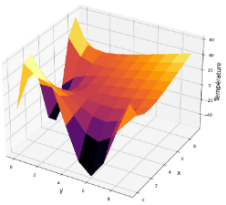

FIGURE 22 – Matrice théorique 4, Vue 3D 1

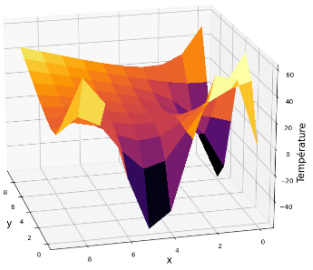

FIGURE 23 – Matrice théorique 4, Vue 3D 2

8.2 Matrices source

```
[[ 5.   5.5  6.   6.5  7.   7.5  8.   8.5  9.   9.5]
 [ 5.5  0.   0.   0.   0.   0.   0.   0.   0.  11.8]
 [ 6.   0.   0.   0.   0.   0.   0.   0.   0.  14.1]
 [ 6.5  0.   0.   0.   0.   0.   0.   0.   0.  16.4]
 [ 7.   0.   0.   0.   0.   0.   0.   0.   0.  18.7]
 [ 7.5  0.   0.   0.   0.   0.   0.   0.   0.  21. ]
 [ 8.   0.   0.   0.   0.   0.   0.   0.   0.  23.3]
 [ 8.5  0.   0.   0.   0.   0.   0.   0.   0.  25.6]
 [ 9.   0.   0.   0.   0.   0.   0.   0.   0.  27.9]
 [ 9.5 11.8 14.1 16.4 18.7 21.  23.3 25.6 27.9 30.2]]
```

FIGURE 24 – Matrice source numérique 1

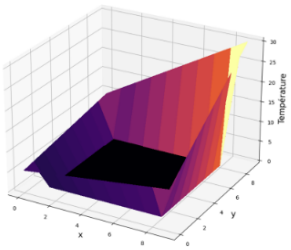

FIGURE 25 – Matrice source 1, Vue 3D 1

```
[[ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 3.59  0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 5.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 3.38  0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [-0.29  0.    0.    0.    0.    0.    0.    0.    0.   -0.  ]
 [-3.78  0.    0.    0.    0.    0.    0.    0.    0.   -0.  ]
 [-4.98  0.    0.    0.    0.    0.    0.    0.    0.   -0.  ]
 [-3.16  0.    0.    0.    0.    0.    0.    0.    0.   -0.  ]
 [ 0.58  0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 3.97  1.78  0.8   0.36  0.16  0.07  0.03  0.01  0.01  0.  ]]
```

FIGURE 26 – Matrice munérique source 2

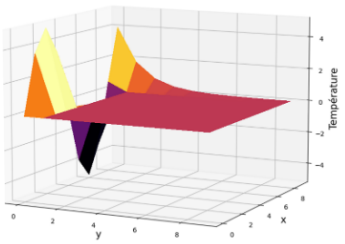

FIGURE 27 – Vue 3D

```
[[ 0.    3.59  5.    3.38 -0.29 -3.78 -4.98 -3.16  0.58  3.97]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    1.78]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.8 ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.36]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.16]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.07]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.03]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.01]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.01]
 [ 0.    0.    0.    0.   -0.   -0.   -0.   -0.    0.    0.  ]]
```

FIGURE 28 – Matrice numérique source 3

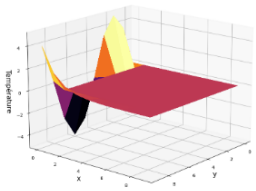

FIGURE 29 – Vue 3D

```
[[  1.   44.1  61.2  41.8  -2.1 -43.9 -58.2 -36.2   8.8  49.5]
 [ 51.6   0.    0.    0.    0.    0.    0.    0.    0.   27.9]
 [ 55.8   0.    0.    0.    0.    0.    0.    0.    0.   20.7]
 [  9.8   0.    0.    0.    0.    0.    0.    0.    0.   20. ]
 [-44.    0.    0.    0.    0.    0.    0.    0.    0.   22.2]
 [-56.    0.    0.    0.    0.    0.    0.    0.    0.   25.8]
 [-15.2   0.    0.    0.    0.    0.    0.    0.    0.   29.9]
 [ 41.1   0.    0.    0.    0.    0.    0.    0.    0.   34.3]
 [ 61.2   0.    0.    0.    0.    0.    0.    0.    0.   38.8]
 [ 26.6  15.6  14.5  17.   20.8  25.   29.5  34.1  38.7  43.3]]
```

FIGURE 30 – matrice source numérique 4

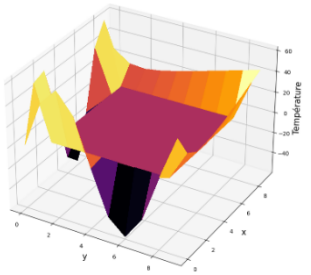

FIGURE 31 – Matrice source 4, Vue 3D 1

8.3 Mise en équation de la méthode stationaire

On notera par la suite la température en un point du plan de coordonnée x,y au temps t : $T(x, y, t)$ En suivant un modèle dans lequel seul les voisins les plus proches du point considéré on un effet sur celui-ci on peut se limiter au calcul de développement limités aux bords du point seulement.

Par exemple dans cette matrice 2×2, le coefficient $a_{1,1}$ (en bleu) ne sera calculé qu'à partir des éléments en rouge :

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

On obtient donc après développement :

$$\begin{cases} T(x+dx,y,t) = T(x,y,t) + \delta x.\dfrac{\partial T}{\partial x}(T(x,y,t)) + \dfrac{\delta x^2}{2}\dfrac{\partial^2 T}{\partial x^2}(T(x,y,t)) + o(dx^2) & (i) \\ T(x-dx,y,t) = T(x,y,t) - \delta x.\dfrac{\partial T}{\partial x}(T(x,y,t)) + \dfrac{\delta x^2}{2}\dfrac{\partial^2 T}{\partial x^2}(T(x,y,t)) + o(dx^2) & (ii) \\ T(x,y-dy,t) = T(x,y,t) - \delta y.\dfrac{\partial T}{\partial y}(T(x,y,t)) + \dfrac{\delta y^2}{2}\dfrac{\partial^2 T}{\partial y^2}(T(x,y,t)) + o(dy^2) & (iii) \\ T(x,y+dy,t) = T(x,y,t) + \delta y.\dfrac{\partial T}{\partial y}(T(x,y,t)) + \dfrac{dy^2}{2}\dfrac{\partial^2 T}{\partial y^2}(T(x,y,t)) + o(dy^2) & (iv) \end{cases} \tag{14}$$

On soustrait alors 14 (ii) à 14 (i) et 14 (iv) à 14 (iii). Ce qui donne :

$$\begin{cases} \dfrac{\partial^2 T}{\partial x^2}(T(x,y,t)) &= \dfrac{T(x+dx,y,t) + T(x-dx,y,t) - 2.T(x,y,t)}{\delta x^2} \\ \dfrac{\partial^2 T}{\partial y^2}(T(x,y,t)) &= \dfrac{T(x,y+dy,t) + T(x,y-dy,t) - 2.T(x,y,t)}{\delta y^2} \end{cases} \tag{15}$$

On obtient donc :

$$\begin{aligned} \Delta T = &\frac{T(x+dx,y,t) + T(x-dx,y,t) - 2.T(x,y,t)}{\delta x^2} \\ &+ \frac{T(x,y+dy,t) + T(x,y-dy,t) - 2.T(x,y,t)}{\delta y^2} \end{aligned} \tag{16}$$

Or dans notre situation $dx = dy$ au vu de la discrétisation carré ce qui donne :

$$\Delta T = \frac{T(x+dx,y,t) + T(x-dx,y,t) + T(x,y+dy,t) + T(x,y-dy,t) - 4.T(x,y,t)}{\delta x^2} \tag{17}$$

Pour simplifier l'ecriture on introduit une notation qui rend explicite la discrétisation :

$$\begin{aligned} x + dx &\Leftrightarrow j+1 \\ y + dy &\Leftrightarrow i+1 \\ \Rightarrow T(x,y,t) &\Leftrightarrow T_{i,j}(t) \end{aligned}$$

Néanmoins, en fesant cela, il faut garder en tête que : $y = j.dy$ et $x = i.dx$. On peut alors écrire :

$$\Delta T_{i,j}(t) = \frac{1}{\delta x^2}. \sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} \left(T_{i+k,j+l}(t) \right) - 4.T_{i,j}(t) \tag{18}$$

De même avec un DL à l'ordre 1 en temps, il vient :

$$\frac{\partial T}{\partial t}(i,j,t) = \frac{1}{\delta t}.(T_{i,j}(t+dt) - T_{i,j}(t)) \tag{19}$$

En stationnaire, on a donc que $\Delta T = 0$ ce qui amène :

$$\begin{aligned} & & \Delta T_{i,j} &= 0 \\ \Rightarrow \quad & \frac{1}{\delta x^2}. \sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} \left(T_{i+k,j+l} \right) - 4.T_{i,j} &= 0 \\ \Rightarrow \quad & & 4.T_{i,j} &= \sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} \left(T_{i+k,j+l} \right) \end{aligned} \tag{20}$$

En sachant que notre matrice de taille m à calculer est de la forme :

$$S_m = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,m-1} & r_{0,m} \\ r_{1,0} & T_{1,1} & \cdots & T_{1,m-1} & r_{1,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{m-1,0} & T_{m-1,1} & \cdots & T_{m-1,m-1} & r_{m-1,m} \\ r_{m,0} & r_{m,1} & \cdots & r_{m,m-1} & r_{m,m} \end{pmatrix}$$

avec $r_{i,j}$ les coéfficients fixé du bord.

On Obtient un système d'équations de la forme :

$$\begin{cases} 4.T_{1,1} & -T_{1,2} & -T_{2,1} & & = r_{0,1} & +r_{1,0} \\ 4.T_{1,2} & -T_{1,1} & -T_{2,2} & -T_{1,3} & = r_{0,2} & \\ & & & & \vdots & \\ 4.T_{m-1,m-2} & -T_{m-1,m-1} & -T_{m-1,m-3} & -T_{m-2,m-2} & = r_{m,m-2} & \\ 4.T_{m-1,m-1} & -T_{m-1,m-2} & -T_{m-2,m-1} & & = r_{m-1,m} & +r_{m,m-1} \end{cases}$$

8.4 Matrices avec la méthode stationnaire, vue 3D

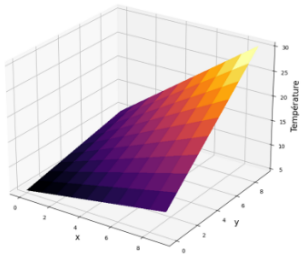

FIGURE 32 – Matrice en stationnaire 1, Vue 3D

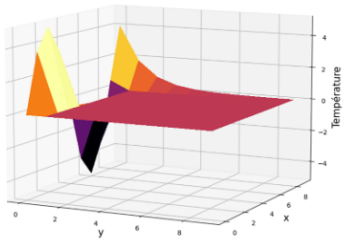

FIGURE 33 – Matrice stationnaire 2, Vue 3D

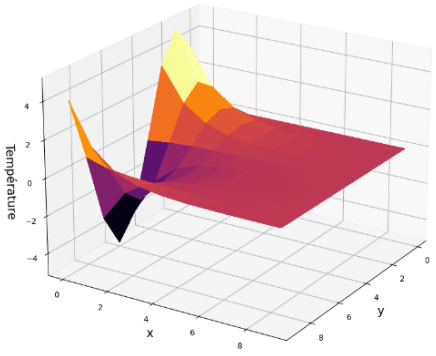

FIGURE 34 – Matrice stationnaire 3, Vue 3D

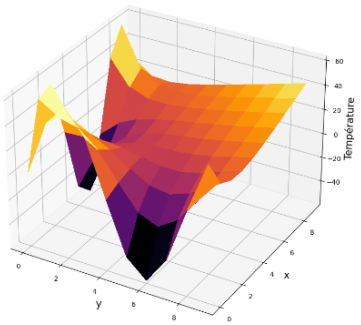

FIGURE 35 – Matrice stationnaire 4, Vue 3D

8.5 Mise en équation de la méthode explicite

À la suite de l'équation discrétisée en 5.2 il vient :

$$T_{i,j}(t+dt) = \frac{D.\delta t}{\delta x^2}.(\sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} (T_{i+k,j+l}(t)) - 4.T_{i,j}(t)) + T_{i,j}(t)$$

en posant $\alpha = \frac{D.\delta t}{\delta x^2}$ il vient :

$$T_{i,j}(t+dt) = \sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} (\alpha.T_{i+k,j+l}(t)) + (1-4\alpha).T_{i,j}(t) \tag{21}$$

En se rappelant que la matrice est initialement de la forme suivante :

$$S_m = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,m-1} & r_{0,m} \\ r_{1,0} & T_{1,1} & \cdots & T_{1,m-1} & r_{1,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{m-1,0} & T_{m-1,1} & \cdots & T_{m-1,m-1} & r_{m-1,m} \\ r_{m,0} & r_{m,1} & \cdots & r_{m,m-1} & r_{m,m} \end{pmatrix}$$

On obtient le système :

$$\begin{cases} T_{1,1}(t+dt) = \alpha.(r_{0,1} + r_{1,0} + T_{1,2}(t) + T_{2,1}(t)) + (1-4\alpha).T_{1,1}(t) \\ T_{1,2}(t+dt) = \alpha.(r_{0,2} + r_{3,1} + T_{2,2}(t) + T_{1,1}(t)) + (1-4\alpha).T_{1,2}(t) \\ \vdots \\ T_{m-1,m-2}(t+dt) = \alpha.(r_{m,m-2} + T_{m-1,m-1}(t) + T_{m-1,m-3}(t) + T_{m-2,m-2}(t)) + (1-4\alpha).T_{m-1,m-2}(t) \\ T_{m-1,m-1}(t+dt) = \alpha.(r_{m-1,m} + r_{m,m-1} + T_{m-1,m-2}(t) + T_{m-2,m-1}(t)) + (1-4\alpha).T_{m-1,m-1}(t) \end{cases}$$

Qui est mis sous forme matricielle au point 5.2.

8.6 Matrices avec la méthode explicite, vue 3D

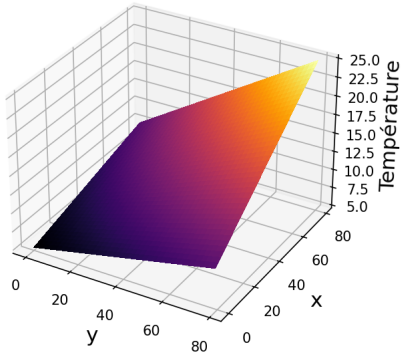

FIGURE 36 – Matrice avec la méthode explicite 1, Vue 3D

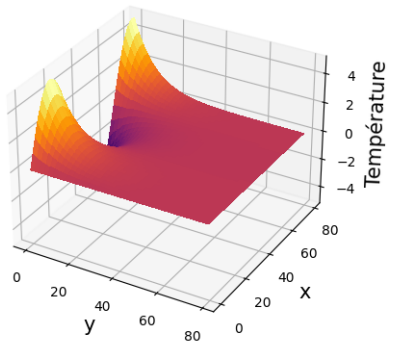

FIGURE 37 – Matrice avec la méthode explicite 2, Vue 3D

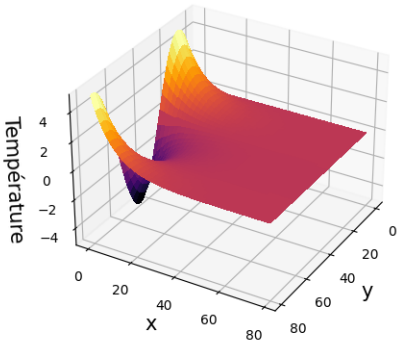

FIGURE 38 – Matrice avec la méthode explicite 3, Vue 3D

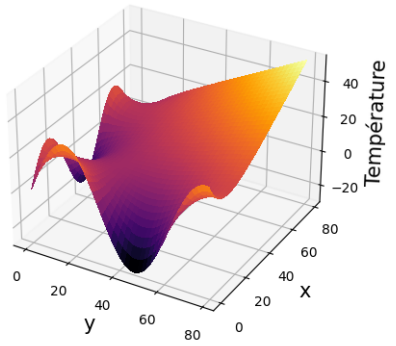

FIGURE 39 – Matrice avec la méthode explicite 4, Vue 3D

8.7 Mise en équation de la méthode imlicite

À la suite de l'équation discrétisée en 6.2 il vient :

$$(T_{i,j}(t+dt) - T_{i,j}(t)) = \alpha.(\sum_{\substack{k,l=-1 \\ \backslash\{(k=0,l=0)\}}}^{1} (T_{i+k,j+l}(t+dt)) - 4.T_{i,j}(t+dt))$$

$$\Rightarrow T_{i,j}(t+dt) - \alpha.(\sum_{\substack{k,l=-1 \\ \backslash\{(k=0,l=0)\}}}^{1} (T_{i+k,j+l}(t+dt)) + 4.T_{i,j}(t+dt)) = T_{i,j}(t)$$

$$\Rightarrow - \sum_{\substack{k,l=-1 \\ \setminus\{(k=0,l=0)\}}}^{1} \left(\alpha.T_{i+k,j+l}(t+dt) \right) + (1+4\alpha).T_{i,j}(t+dt) = T_{i,j}(t)$$

En se rappelant que la matrice est initialement de la forme suivante :

$$S_m = \begin{pmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,m-1} & r_{0,m} \\ r_{1,0} & T_{1,1} & \cdots & T_{1,m-1} & r_{1,m} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{m-1,0} & T_{m-1,1} & \cdots & T_{m-1,m-1} & r_{m-1,m} \\ r_{m,0} & r_{m,1} & \cdots & r_{m,m-1} & r_{m,m} \end{pmatrix}$$

On obtient le système :

$$\begin{cases} (1+4\alpha).T_{1,1}(t+dt) - \alpha.T_{1,2}(t+dt) - \alpha.T_{2,1}(t+dt) = \alpha.(r_{0,1}+r_{1,0}) + T_{1,1}(t) \\ (1+4\alpha).T_{1,2}(t+dt) - \alpha.T_{1,1}(t+dt) - \alpha.T_{2,2}(t+dt) - \alpha.T_{1,3}(t+dt) = \alpha.r_{0,2} + T_{1,2}(t) \\ \vdots \\ (1+4\alpha).T_{m-1,m-2}(t+dt) - \alpha.(T_{m-1,m-1}(t+dt) + T_{m-1,m-3}(t+dt) + T_{m-2,m-2}(t+dt)) = \\ \alpha.r_{m,m-2} + T_{m-1,m-2}(t) \\ (1+4\alpha).T_{m-1,m-1}(t+dt) - \alpha.(T_{m-1,m-2}(t+dt) + T_{m-2,m-1}(t+dt)) = \\ \alpha.(r_{m-1,m} + r_{m,m-1}) + T_{m-1,m-1}(t) \end{cases}$$

Qui est mis sous forme matricielle au point 6.2.

8.8 Matrices avec la méthode implicite

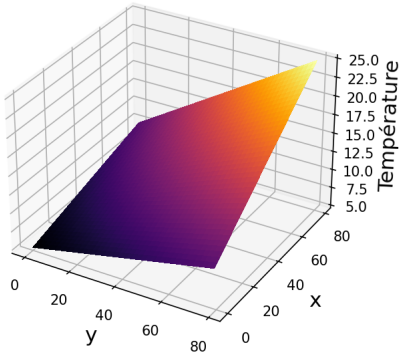

FIGURE 40 – Matrice avec la méthode implicite 1, Vue 3D

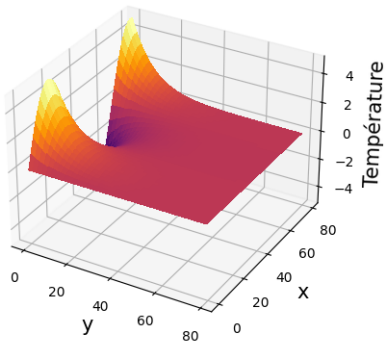

FIGURE 41 – Matrice avec la méthode implicite 2, Vue 3D

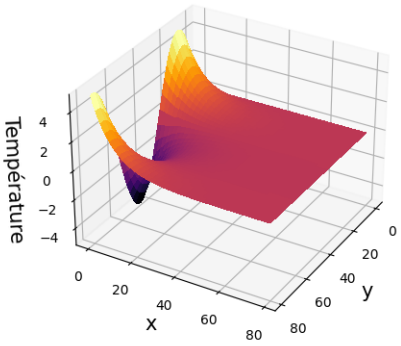

FIGURE 42 – Matrice avec la méthode implicite 3, Vue 3D

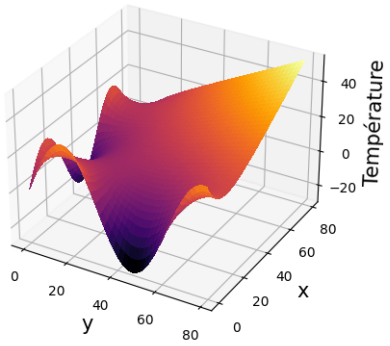

FIGURE 43 – Matrice avec la méthode implicite 4, Vue 3D