Projet 1 : Résolution de l'équation instationnaire de la chaleur 1D

Serge Bays

Février 2010

Nous allons étudier le problème physique de la propagation de la chaleur dans un mur.

Ce problème sera modélisé, puis analysé mathématiquement afin d'en déterminer les solutions exactes dépendantes de la condition initiale et des conditions aux limites. Ceci se fera en une dimension d'espace, à travers deux cas particuliers : une température initiale uniforme dans le mur avec une brusque variation de la température extérieure et une température initiale non uniforme dans le mur avec une température extérieure fixe égale à la température intérieure.

Une étude numérique sera menée sur ces deux cas. Le problème sera discrétisé en temps et en espace avec un maillage de l'espace à pas fixe ou variable et nous utiliserons un schéma aux différences finies en temps et en espace, soit explicite centré, soit implicite. Le problème se traduira alors par un calcul direct itéré ou par la résolution itérée d'un système linéaire.

Le calcul des solutions approchées et la comparaison avec les solutions exactes seront effectués à l'aide de programmes écrits en C++ ; les résultats seront ensuite présentés sous forme graphique.

Les deux cas particuliers seront résolus dans les deux premières études avec un maillage en espace à pas fixe et un schéma aux différences finies explicite. Dans une troisième étude, nous reprendrons ces deux cas avec un schéma implicite qui nous montrera son avantage. Nous présenterons enfin dans une quatrième étude l'intérêt éventuel d'un maillage en espace à pas variable.

Les programmes C++ suivront cette progression ; nous commencerons par un programme très simple (version 1) que nous complèterons en avançant les études (version 2, 3, 4), afin d'obtenir un programme final (version 5) capable de proposer les différents choix : premier ou deuxième cas, schéma explicite ou implicite, pas fixe ou variable.

Table des matières

1 Modélisation du problème

Considérons un mur d'épaisseur ℓ, qui se trouve initialement à une température uniforme θ_0 (température de la chambre). A l'instant $t = 0$, la température extérieure (en $x = 0$) monte brusquement à $\theta_s > \theta_0$, valeur maintenue constante par une source de chaleur. On suppose que la température à $x = \ell$ est gardée à sa valeur initiale θ_0.

Nous sommes donc dans le cadre d'un domaine à une dimension, modélisé par un segment de droite $[0; \ell]$.

La propagation de la chaleur dans le mur de diffusivité thermique κ, constante strictement positive, est décrite par l'équation de la chaleur :

$$\frac{\partial u}{\partial t} - \kappa \frac{\partial^2 u}{\partial x^2} = 0$$

avec l'inconnue $u(x,t) = \theta(x,t) - \theta_0$.

Ce sera l'objet de la première étude où nous choisirons donc la condition initiale $u_0(x) = 0$ avec des conditions aux limites de type Dirichlet :

$$u(0,t) = \theta_s - \theta_0 = u_s \text{ et } u(\ell,t) = 0, \quad \forall t > 0$$

Nous examinerons ensuite dans la deuxième étude le cas où la tempérarure initiale du mur n'est pas uniforme et la température extérieure reste fixée à θ_0 ; il suffira de modifier la condition aux limites $u(0,t) = \theta_s - \theta_0 = u_s = 0$ et la condition initiale, en prenant par exemple :

$$u_0(x) = u(x,0) = \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x)$$

2 Analyse mathématique

2.1 Solution générale

Nous allons utiliser la décomposition en ondes simples de la solution :

$$u(x,t) = \sum_{k\in\mathbb{N}^*} \hat{u}_k(t)\phi_k(x), \quad \phi_k(x) = \sin(\frac{k\pi}{\ell}x)$$

Nous pouvons ici dériver "sous le signe somme" et calculer :

$$\frac{\partial u}{\partial t} = \sum_{k\in\mathbb{N}^*} \hat{u}'_k(t)\phi_k(x)$$

$$\frac{\partial^2 u}{\partial x^2} = -\sum_{k\in\mathbb{N}^*} \hat{u}_k(t)\frac{k^2\pi^2}{\ell^2}\phi_k(x)$$

Alors si $u(x,t)$ est solution de l'équation de la chaleur $\dfrac{\partial u}{\partial t} - \kappa \dfrac{\partial^2 u}{\partial x^2} = 0$ nous obtenons :

$$\sum_{k\in\mathbb{N}^*} \left(\hat{u}'_k(t) + \kappa\hat{u}_k(t)\frac{k^2\pi^2}{\ell^2}\right)\phi_k(x) = 0$$

d'où nous déduisons l'équation différentielle vérifiée par chaque fonction $\hat{u}_k$:

$$\hat{u}_k'(t) + \kappa \hat{u}_k(t) \frac{k^2\pi^2}{\ell^2} = 0$$

la forme de chaque fonction $\hat{u}_k$ est donc :

$$\hat{u}_k(t) = A_k \exp\left(-\Big(\frac{k\pi}{\ell}\Big)^2 \kappa t\right)$$

et les fonctions solutions u s'écrivent :

$$u(x,t) = Ax + B + \sum_{k\in\mathbb{N}^*} A_k \exp\left(-\Big(\frac{k\pi}{\ell}\Big)^2 \kappa t\right) \phi_k(x)$$

2.2 Solutions particulières

2.2.1 Premier cas

Pour la première étude, les conditions $u(0,t) = u_s$ et $u(\ell,t) = 0$ imposent $B = u_s$ et $A\ell + B = 0$, soit $A = -\frac{u_s}{\ell}$.

Les solutions $u(x,t)$ s'écrivent donc :

$$u(x,t) = (1 - \frac{x}{\ell})u_s + \sum_{k\in\mathbb{N}^*} A_k \exp\left(-\Big(\frac{k\pi}{\ell}\Big)^2 \kappa t\right) \phi_k(x)$$

La condition initiale $u(x,0) = 0$ se traduit par :

$$(\frac{x}{\ell} - 1)u_s = \sum_{k\in\mathbb{N}^*} A_k \phi_k(x)$$

Les coefficient A_k sont donc les coefficients du développement en série de Fourier de la fonction impaire et périodique de période 2ℓ définie sur $[0;\ell]$ par $P(x) = (\frac{x}{\ell} - 1)u_s$.

soit :

$$A_k = \frac{2}{\ell} \int_0^\ell (\frac{x}{\ell} - 1)u_s \sin(\frac{k\pi x}{\ell}) dx$$

le calcul de l'intégrale donne :

$$A_k = \frac{2}{\ell}\left[(\frac{x}{\ell} - 1)u_s(-\frac{\ell}{k\pi})\cos(\frac{k\pi x}{\ell})\right]_0^\ell = A_k = -\frac{2}{\ell}u_s\frac{\ell}{k\pi} = -\frac{2u_s}{k\pi}$$

et la solution exacte est donc :

$$u(x,t) = (1 - \frac{x}{\ell})u_s + \sum_{k\in\mathbb{N}^*} -\frac{2u_s}{k\pi} \exp\left(-\Big(\frac{k\pi}{\ell}\Big)^2 \kappa t\right) \sin(\frac{k\pi}{\ell}x)$$

2.2.2 Deuxième cas

Pour la deuxième étude, la condition $u_s = 0$ et $u(\ell, t) = 0$ imposent $A = B = 0$ et la solution cherchée est de la forme :

$$u(x,t) = \sum_{k\in\mathbb{N}^*} A_k \exp\left(-\Big(\frac{k\pi}{\ell}\Big)^2 \kappa t\right) \phi_k(x)$$

La condition initiale $u_0(x) = u(x,0) = \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x)$ se traduit par :

$$\sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x) = \sum_{k\in\mathbb{N}^*} A_k \phi_k(x)$$

Les coefficient A_k sont donc tous nuls, exceptés $A_1 = 1$ et $A_{10} = \frac{1}{4}$.

La solution exacte s'écrit alors :

$$u(x,t) = \exp\left(-\Big(\frac{\pi}{\ell}\Big)^2 \kappa t\right) \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\exp\left(-\Big(\frac{10\pi}{\ell}\Big)^2 \kappa t\right) \sin(\frac{10\pi}{\ell}x)$$

3 Première étude - schéma explicite - premier cas

3.1 Discrétisation du problème

Nous allons utiliser un maillage régulier du temps, en posant $t_n = n\delta t$, pour $n = 0, 1, \ldots, N$, avec une limite maximale $N\delta t = t_{max}$.

En espace, nous utiliserons de même un maillage régulier du segment $[0; \ell]$.
Le domaine sera discrétisé de la façon suivante :

$$[0;\ell] = \bigcup_{i=0}^{M-2} [x_m; x_{m+1}]$$

en posant $x_m = m\delta x$, pour $m = 0, 1, \ldots, M-1$, avec $\delta x = \ell/(M-1)$.

Au temps t_n, la fonction solution $u(x, t_n)$ sera approchée par la fonction U^n définie aux points du maillage x_m.

$U^n(x_m)$ sera noté U^n_m, pour $m = 0, \ldots, M-1$.

3.2 Schéma de résolution

Nous allons utiliser un schéma explicite centré pour calculer par différences finies en temps et en espace la solution de l'équation

$$\partial_t u - \kappa \partial_{xx} u = 0$$

avec les conditions données précédement.

Nous avons $\delta x = \dfrac{\ell}{M-1}$ et $\delta t = \dfrac{t_{max}}{N}$ où M et N sont les paramètres de discrétisation,
$x_m = m\delta x$, $t_n = n\delta t$ et on cherche à approcher au temps t_n la solution u en (x_m, t_n) par $U^n_m = U(x_m, t_n)$.

On discrétise chacun des opérateurs de dérivation en utilisant des développements de Taylor de la solution cherchée, soit

$$\partial_t u(x_m, t_n) \simeq \frac{u_m^{n+1} - u_m^n}{\delta t}$$

et

$$\partial_{xx} u(x_m, t_j) \simeq \frac{u_{m+1}^j - 2u_m^j + u_{m-1}^j}{\delta x^2}$$

Nous obtenons alors le problème à résoudre :

$$\frac{U_m^{n+1} - U_m^n}{\delta t} - \kappa \frac{U_{m+1}^j - 2U_m^j + U_{m-1}^j}{\delta x^2} = 0$$

Le schéma est explicite si $j = n$, soit :

$$\frac{U_m^{n+1} - U_m^n}{\delta t} - \kappa \frac{U_{m+1}^n - 2U_m^n + U_{m-1}^n}{\delta x^2} = 0$$

pour $1 \leq m \leq M-2$ avec U_m^0 donné pour tout m par la condition initiale, U_0^n et U_{M-1}^n donnés pour tout n par les conditions aux limites.

3.3 Résolution numérique

On donne la valeur de la diffusivité thermique $\kappa = 1$, les conditions aux limites $U_0^n = u_s = 1$ et $U_{M-1}^n = 0$ pour tout n et la condition initiale $U_m^0 = 0$ pour tout m.

Si nous posons $\beta = \kappa \dfrac{\delta t}{\delta x^2}$, nous obtenons pour $1 \leq m \leq M-2$, le schéma :

$$U_m^{n+1} = \beta U_{m+1}^n + (1 - 2\beta) U_m^n + \beta U_{m-1}^n$$

avec

$$U_0^{n+1} = U_0^n = 1$$

et

$$U_{M-1}^{n+1} = U_{M-1}^n = 0$$

La condition de stabilité de ce schéma est $\kappa \dfrac{\delta t}{\delta x^2} \leq \dfrac{1}{2}$ et le choix du pas en temps δt dépend donc du pas en espace δx. Avec $\kappa = 1$, nous pouvons choisir ici $\delta t = \dfrac{\delta x^2}{2}$.

Remarquons que dans ce cas particulier, $\beta = \kappa \dfrac{\delta t}{\delta x^2} = \dfrac{1}{2}$, et le schéma se simplifie en

$$U_m^{n+1} = \frac{1}{2}(U_{m+1}^n + U_{m-1}^n) \qquad \text{pour} \qquad 1 \leq m \leq M-2$$

L'avantage de ce schéma est le calcul relativement simple qu'il entraîne. Le désavantage est que le pas en temps étant limité, cela implique un grand nombre de calculs intermédiaires (à chaque pas) pour arriver à un temps long.

3.4 Programmation en C++

3.4.1 Programme version 1 et résultats

Nous pouvons créer un premier programme simple, nommé `Chaleur1d_v1.cpp`, pour résoudre le problème posé. Les besoins en variables sont :

- les données du problème M, ℓ, u_s, κ,
- les pas de discrétisation dx et dt,
- deux vecteurs U et $Uold$ représentant les valeurs de la solution approchée aux points du maillage, au temps ndt et $(n-1)dt$, et un vecteur Uex représentant les valeurs de la solution exacte au temps ndt.

Les vecteurs U et $Uold$ sont initialisés en traduisant la condition initiale et les conditions aux limites.

Ensuite, à chaque pas de temps, nous calculons à partir du vecteur $Uold$ le vecteur U au pas suivant à l'aide du schéma numérique.

Puis nous écrivons dans un fichier les valeurs du vecteur U à différents instants auxquels nous calculons également le vecteur Uex, la solution exacte, afin de pouvoir les comparer. Le calcul de Uex s'effectue en prenant les vingt premiers termes de la somme, ce qui donne une assez bonne approximation. Ce calcul est fait seulement aux instants où nous souhaitons écrire les résultats, ceci permettant une certaine économie.

Enfin, nous calculons la norme sup de la différence entre la solution numérique et la solution exacte.

La figure 1 présente le résultat obtenu dans les premiers instants. C'est ce que nous pouvions attendre. En effet, la différence entre la solution numérique et la solution exacte est importante, particulièrement près du point $x = 0$, (environ 0.023), qui subit la brusque variation de température.

Cette différence qui diminue rapidement avec le temps s'explique en grande partie par l'approximation faite sur le calcul de la solution exacte, approximation qui est bonne à partir de l'instant $5dt$, et par le fait que la fonction température n'est pas continue à l'instant $t = 0$, puisqu'elle est nulle sauf en $x = 0$ où elle vaut 1.

Si nous voulons nous intéresser aux instants précédents, une meilleure approximation de la solution exacte, plus coûteuse en calcul, est de prendre les 100 premiers termes de la somme. Le résultat obtenu est présenté sur la figure 2.

Toutes les figures de cette première étude sont réalisées avec $dt = \dfrac{dx^2}{2} = 0.0002$.

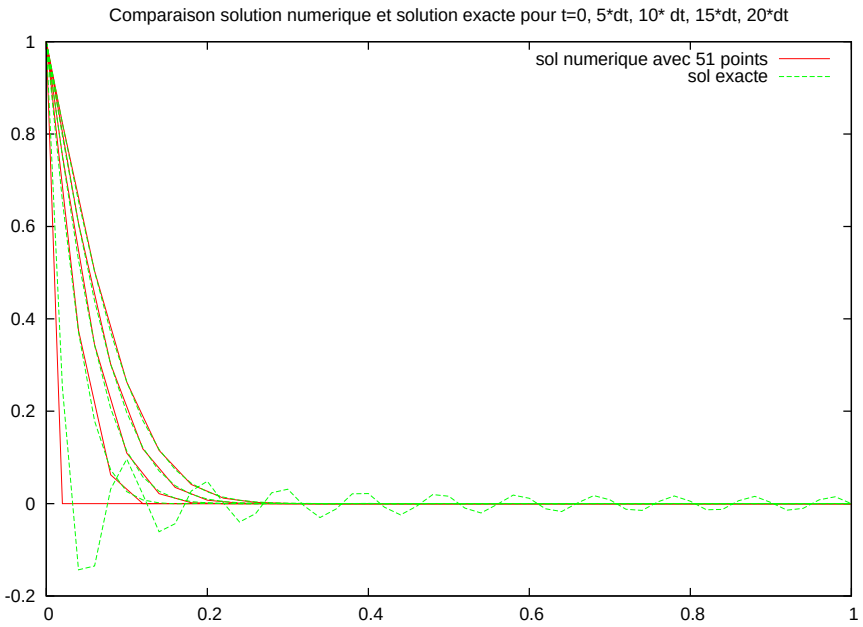

FIG. 1 – *Résultat avec le programme* `Chaleur1d_v1.cpp`

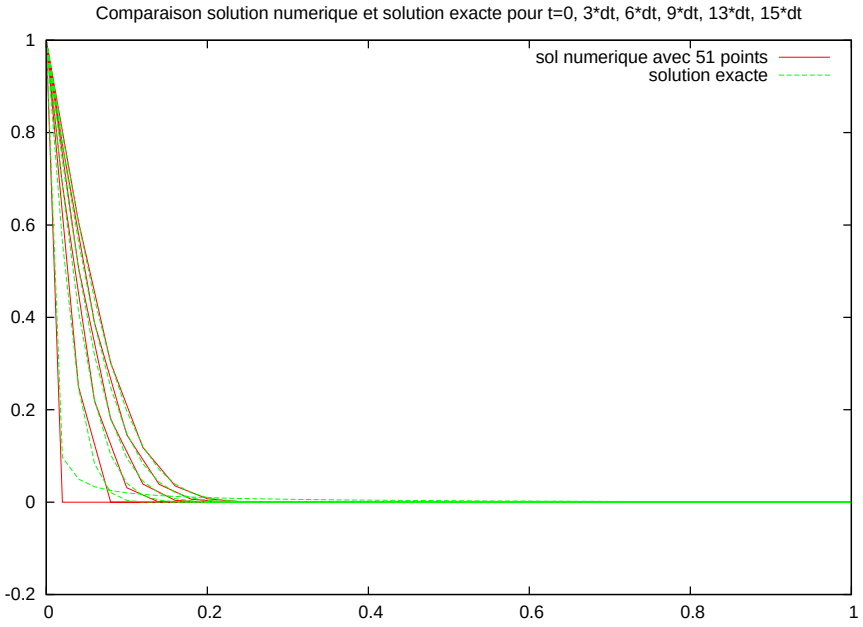

FIG. 2 – *Résultat avec le programme* `Chaleur1d_v1.cpp`

3.4.2 Programme version 2 et résultats

Nous souhaitons maintenant que le programme puisse résoudre d'autres problèmes similaires, (changement des données, du maillage, du calcul des solutions), ce qui va entraîner une augmenation importante de sa taille et de sa complexité. Il est donc intéressant d'utiliser une programmation modulaire. Nous aurons donc des fichiers séparés avec une classe pour le maillage, une classe pour les vecteurs solutions, le calcul, l'affichage et un fichier principal contenant les données et les différents choix possibles.

Nous allons donc créer une première classe pour le maillage du temps et de l'espace, la classe `Mesh1d`.

Le fichier `Mesh1d_v2.hpp` contient la déclaration de la classe avec les attributs nécessaires au maillage de l'espace et du temps, soit : M, L, dx, dt ainsi que le vecteur du maillage x.

Il contient également le prototype des méthodes qui sont le constructeur de la classe et les méthodes permettant de récupérer la valeur des attributs car tous les attributs ont un droit d'accès privé.

Les différentes méthodes sont implémentées dans le fichier `Mesh1d_v2.cpp`.

La deuxième classe `VectSol` est crée avec les fichiers `VectSol_v2.hpp` et `VectSol_v2.cpp`.

Cette classe, déclarée dans le fichier `VectSol_v2.hpp`, a besoin du maillage donc elle a pour attributs la classe `Mesh1d`. Les autres attributs sont la constante de diffusivité thermique κ nécessaire au schéma de résolution et les trois vecteurs U, $Uold$ et Uex, solutions à deux pas de temps successifs et solution exacte du problème. Ici encore les attributs ont un droit d'accès privé.

Les méthodes sont : le constructeur de la classe, une méthode qui calcule la solution exacte, une méthode qui calcule la solution au pas suivant, une méthode qui permet d'écrire les résultats dans un fichier et une méthode qui calcule l'erreur en norme sup entre la solution approchée et la solution exacte. Toutes ces méthodes sont implémentées dans le fichier `VectSol_v2.cpp`.

Le programme principal `Chaleur1d_v2.cpp` contient les données M, L, κ. Il fait appel aux constructeurs des classes `Mesh1d` et `VectSol` afin de construire le maillage, puis initialiser les vecteurs.

Nous pouvons ici entrer suivant nos besoin le nombre de pas en temps maximal et le nombre de pas entre le calcul de la solution exacte et l'écriture dans un fichier des résultats.

L'appel aux différentes méthodes permettant de calculer la solution numérique et la solution exacte, de les comparer et d'écrire les résultats dans un fichier se fait alors de manière itétée suivant le pas de temps.

Lorsque nous lançons ce programme, à l'invite nous pouvons entrer un nombre de pas de temps maximal égal à 20 et le nombre de pas entre deux affichages égal à 5. Nous obtenons le résultat graphique de la figure 3.

Nous choisissons ensuite un nombre de pas de temps maximal égal à 800 et le nombre de pas entre deux affichages égal à 40. Nous obtenons le résultat graphique de la figure 4.

Sur les figures 5 et 6, nous voyons une représentation en 3D de l'évolution des solutions en fonction du temps.

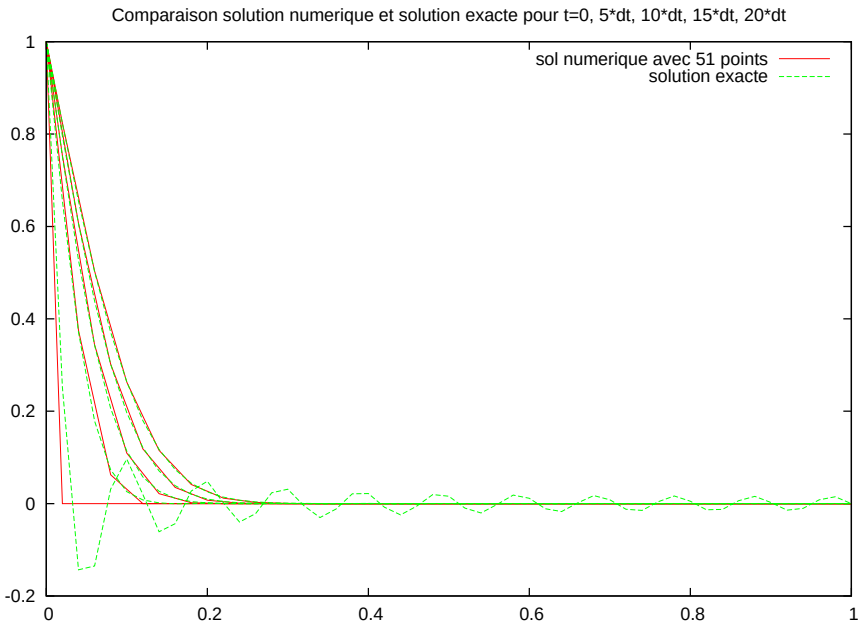

FIG. 3 – *Résultat avec le programme* `Chaleur1d_v2.cpp` *pour* t *petit*

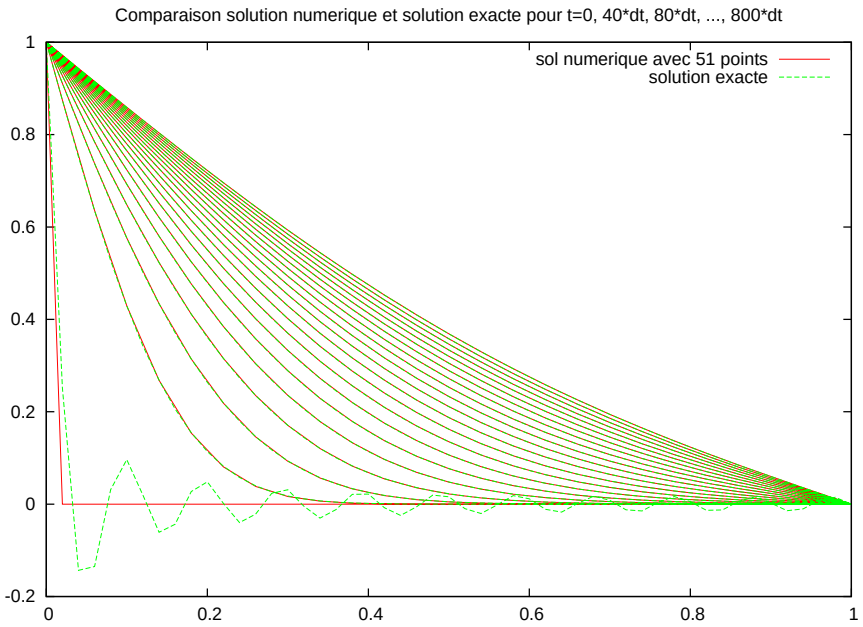

FIG. 4 – *Résultat avec le programme* `Chaleur1d_v2.cpp` *pour* t *grand*

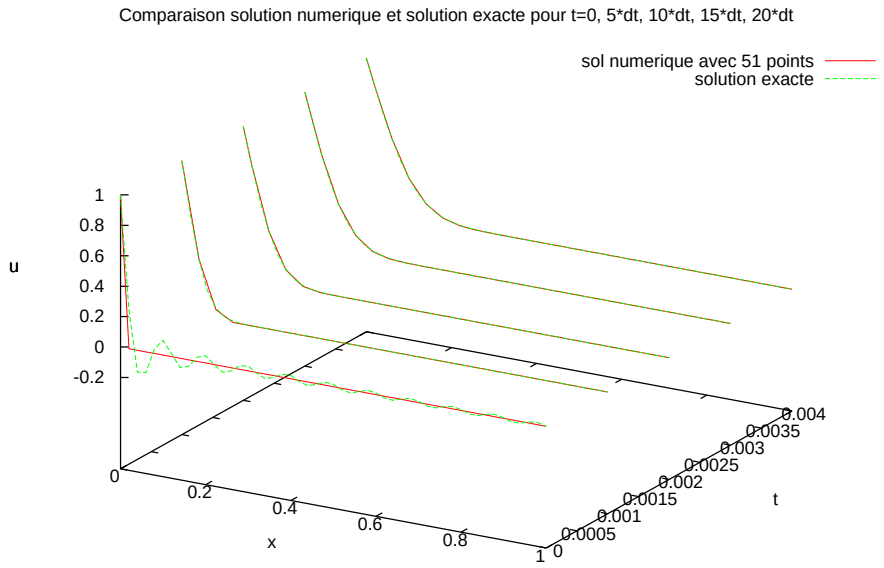

FIG. 5 – *Résultat en 3D avec le programme* `Chaleur1d_v2.cpp` *pour* t *petit*

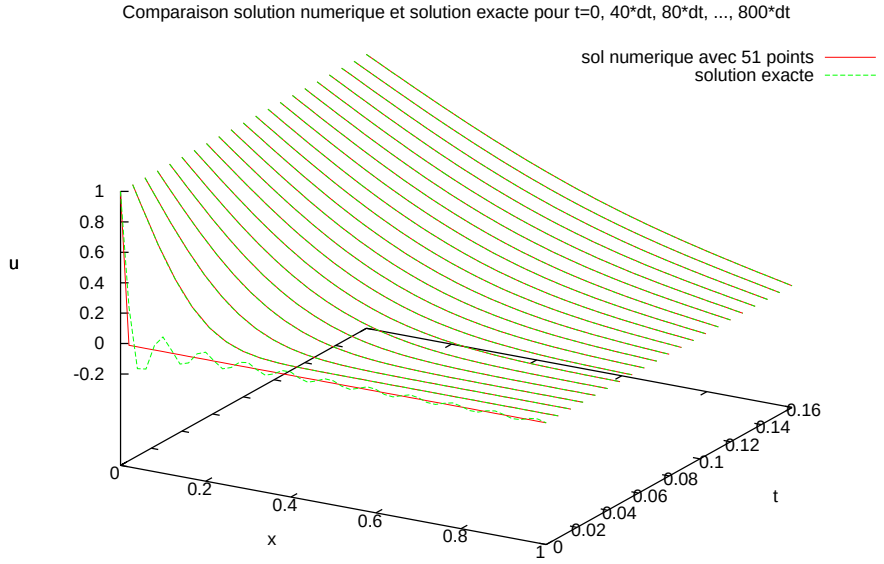

FIG. 6 – *Résultat en 3D avec le programme* `Chaleur1d_v2.cpp` *pour* t *grand*

Logiquement, la température évolue très rapidement dans les premiers instants près du point $x = 0$ qui subit la brusque variation. Ensuite son évolution est plus régulière et la température tend lentement vers son état d'équilibre donné par la fonction $u(x) = 1 - x$.

L'écart maximal entre la solution numérique et la solution exacte est de l'ordre de 2.10^{-2} après 10 itérations, de l'ordre de 2.10^{-3} après 100 itérations et de l'ordre de 2.10^{-4} après 1000 itérations.

Si e est l'écart et k le nombre de pas en temps, nous avons l'approximation $e \simeq \dfrac{1}{5k}$.

La figure 7 nous montre la variation de cet écart en fonction du nombre de pas en temps, en échelle logarithmique.

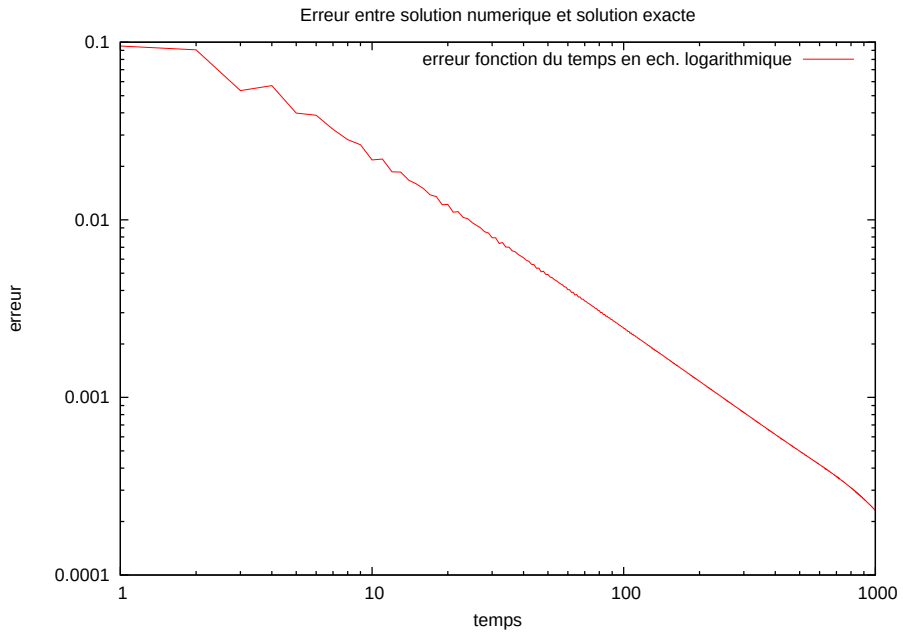

FIG. 7 – *Erreur en échelle logarithmique de dt à 1000dt*

4 Deuxième étude - schéma explicite - deuxième cas

Nous reprenons l'étude précédente en modifiant la condition aux limites $u_s = 0$ et la condition initiale : $u_0(x) = u(x,0) = \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x)$

La température extérieure reste constante, égale à la température intérieure et c'est la température initiale du mur qui n'est pas uniforme.

4.1 Programmation en C++

Le programme précédent doit être modifié au niveau de la condition initiale et des conditions aux limites et pour le calcul de la solution exacte.

Les nouvelles versions des fichiers sont `Mesh1d_v3.hpp`, `Mesh1d_v3.cpp`, `VectSol_v3.hpp`, `VectSol_v3.cpp` et `Chaleur_v3.hpp`.

Les fichiers du maillage ne sont pas modifiés. Nous ajoutons dans les fichiers `VectSol_v3.hpp` et `VectSol_v3.cpp` le calcul de la nouvelle solution exacte et nous modifions le constructeur afin de prendre en compte la possibilité de résoudre le nouveau problème ou l'ancien.

Dans le fichier `Chaleur_v3.hpp`, nous donnons maintenant la possibilité de choisir le problème à résoudre.

4.2 Résultats

Les figures 8, 9, 10 et 11 sont réalisées avec $dt = 0.0002$

Nous voyons clairement sur la figure 8 l'amortissement des ondes présentes dans la condition initiale. Ceci se passe dans un temps relativement court. La partie correspondant à la période $\frac{\ell}{5}$ devient négligeable devant la partie principale correspondant à la période 2ℓ.

Cette partie principale a elle-même une amplitude tendant vers 0 puisque la solution d'équilibre est la fonction nulle sur le segment, ce que l'on voit sur la figure 9.

Les figures 10 et 11 représentent les mêmes évolutions en 3D.

L'erreur entre la solution numérique et la solution exacte reste de l'ordre de 10^{-3} à 10^{-4}, lorsqu'on effectue entre 50 et 1000 itérations. Elle reste donc de l'ordre de dt ou dx^2 ce qui était prévisible.

Si nous voulons gagner une décimale en précision, nous devons jouer sur dt et dx en même temps puisque le schéma lie le pas dt au pas dx. Et ceci augmente rapidement le nombre de calculs.

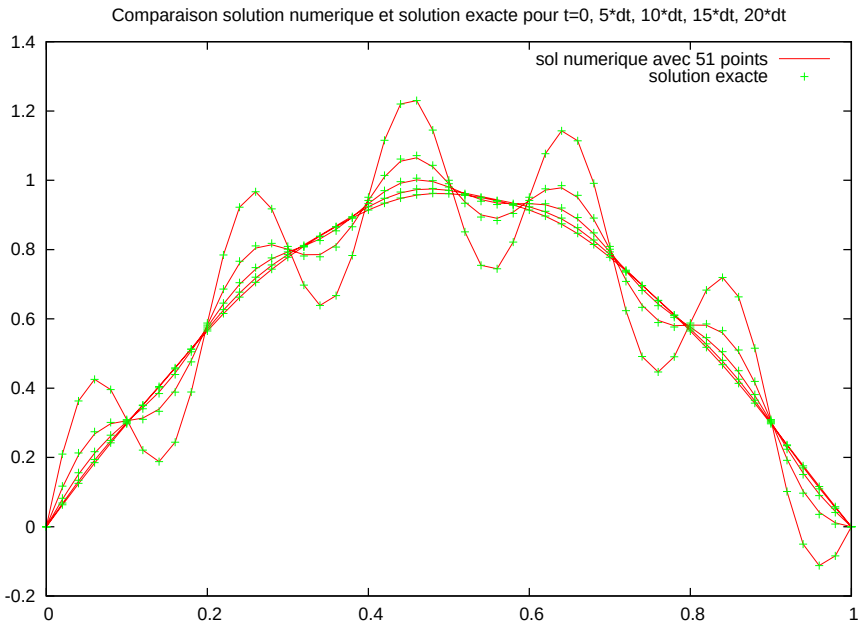

FIG. 8 – *Résultat avec le programme* `chaleur1d_v3.cpp` *pour* t *petit*

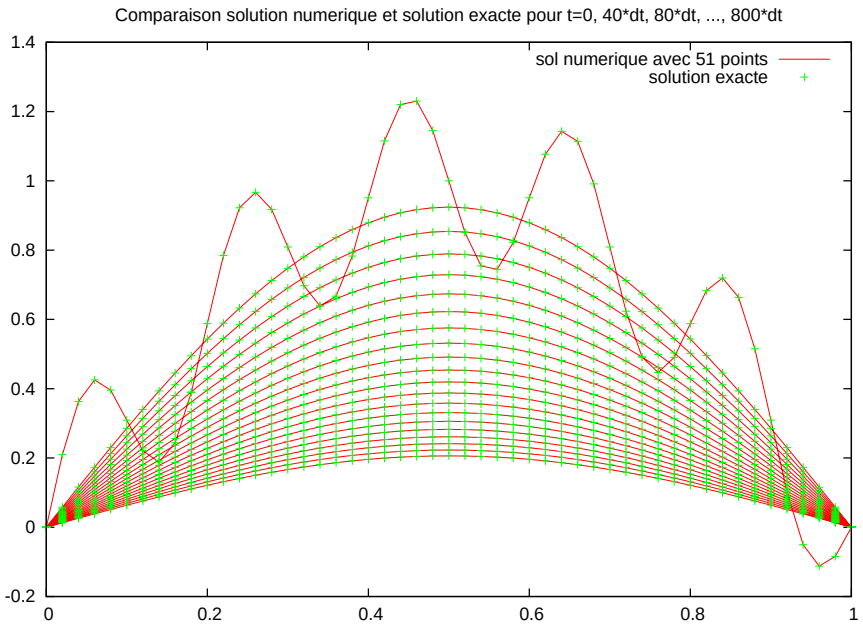

FIG. 9 – *Résultat avec le programme* `chaleur1d_v3.cpp` *pour* t *grand*

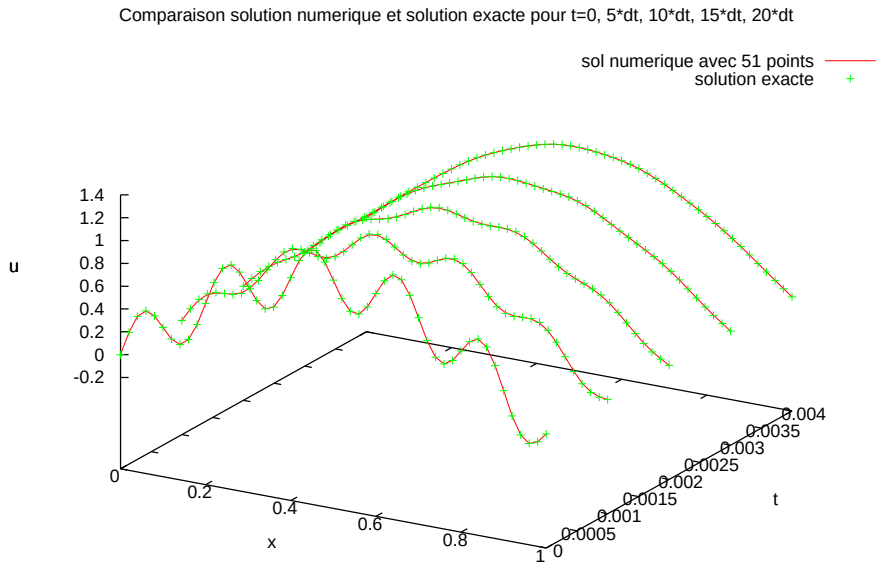

FIG. 10 – *Résultat en 3D avec le programme* `chaleur1d_v3.cpp` *pour* t *petit*

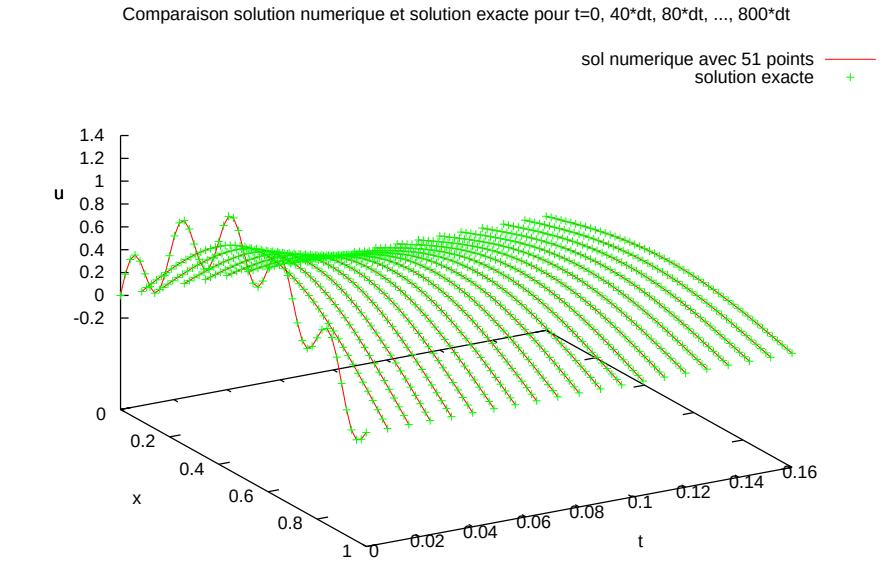

FIG. 11 – *Résultat en 3D avec le programme* `chaleur1d_v2.cpp` *pour* t *grand*

5 Troisième étude - schéma implicite

Nous allons traiter maintenant les questions précédentes avec un schéma implicite. Nous en dégagerons plus loin l'intérêt.

5.1 Schéma implicite

Le schéma obtenu dans le première étude, partie 3.2, était le suivant :

$$\frac{U_m^{n+1} - U_m^n}{\delta t} - \kappa \frac{U_{m+1}^j - 2U_m^j + U_{m-1}^j}{\delta x^2} = 0$$

Le schéma est implicite si $j = n + 1$, soit :

$$\frac{U_m^{n+1} - U_m^n}{\delta t} - \kappa \frac{U_{m+1}^{n+1} - 2U_m^{n+1} + U_{m-1}^{n+1}}{\delta x^2} = 0$$

pour $1 \leq m \leq M - 2$ avec U_m^0 donné pour tout m par la condition initiale, U_0^n et U_{M-1}^n donnés pour tout n par les conditions aux limites.

Ce schéma peut encore s'écrire :

$$-\kappa \frac{\delta t}{\delta x^2} U_{m+1}^{n+1} + (1 + 2\kappa \frac{\delta t}{\delta x^2}) U_m^{n+1} - \kappa \frac{\delta t}{\delta x^2} U_{m-1}^{n+1} = U_m^n$$

Le calcul direct pour déterminer le vecteur U au temps $n + 1$ en fonction du vecteur U au temps n est ici impossible. Il nous faut résoudre un système linéaire.

5.2 Systèmes linéaires

Nous posons $\beta = \kappa \frac{\delta t}{\delta x^2}$.

Les équation s'écrivent alors pour $1 \leq m \leq M - 2$:

$$-\beta U_{m+1}^{n+1} + (1 + 2\beta) U_m^{n+1} - \beta U_{m-1}^{n+1} = U_m^n$$

Pour le premier problème étudié, la condition initiale et les conditions aux limites se traduisent par : $U_m^0 = 0$ pour $m \neq 0$ et $U_0^{n+1} = U_0^n = 1, \quad U_{M-1}^{n+1} = U_{M-1}^n = 0$ pour tout n.

Le système à résoudre peut donc s'écrire :

$$\begin{pmatrix} 1 & 0 & 0 & ... & 0 & 0 \\ -\beta & 1+2\beta & -\beta & 0 & ... & 0 \\ 0 & -\beta & 1+2\beta & -\beta & 0 & . \\ . & & & & & . \\ . & & & & & . \\ . & ... & -\beta & 1+2\beta & -\beta & 0 \\ 0 & ... & 0 & -\beta & 1+2\beta & -\beta \\ 0 & 0 & ... & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ U_1^{n+1} \\ . \\ . \\ . \\ . \\ U_{M-2}^{n+1} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ U_1^n \\ . \\ . \\ . \\ . \\ U_{M-2}^n \\ 0 \end{pmatrix}$$

avec un vecteur initial $\begin{pmatrix} 1 \\ 0 \\ . \\ . \\ 0 \\ . \\ 0 \\ 0 \end{pmatrix}$

Pour le second problème étudié, la condition initiale et les conditions aux limites se traduisent par : $U_m^0 = \sin(\frac{\pi}{\ell} mdx) + \frac{1}{4}\sin(10\frac{\pi}{\ell} mdx)$ pour tout m et $U_0^{n+1} = U_0^n = 0, \quad U_{M-1}^{n+1} = U_{M-1}^n = 0.$

Le système à résoudre peut donc s'écrire :

$$\begin{pmatrix} 1 & 0 & 0 & ... & 0 & 0 \\ 0 & 1+2\beta & -\beta & 0 & ... & 0 \\ 0 & -\beta & 1+2\beta & -\beta & 0 & . \\ . & & & & & . \\ . & & & & & . \\ . & ... & -\beta & 1+2\beta & -\beta & 0 \\ 0 & ... & 0 & -\beta & 1+2\beta & 0 \\ 0 & 0 & ... & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ U_1^{n+1} \\ . \\ . \\ . \\ . \\ U_{M-2}^{n+1} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ U_1^n \\ . \\ . \\ . \\ . \\ U_{M-2}^n \\ 0 \end{pmatrix}$$

avec un vecteur initial $\begin{pmatrix} 0 \\ sin(\pi\frac{dx}{\ell}) \\ . \\ . \\ sin(\pi\frac{mdx}{\ell}) \\ . \\ sin(\pi\frac{(M-2)dx}{\ell}) \\ 0 \end{pmatrix}$

Dans les deux cas, la matrice du système étant tridiagonale, nous pouvons envisager la factorisation LU qui permet de résoudre ensuite simplement le système.

5.3 Programmation en C++

Les programmes précédents ne sont plus suffisants.

Il est nécessaire maintenant de pouvoir construire un système linéaire et le résoudre. Nous allons donc ajouter à ces programmes une nouvelle classe, la classe `MatTriaDiag`.

Cette classe est déclarée dans le fichier `MatTriaDiag.hpp`. Les membres sont la classe `Mesh1d` pour le maillage, la constante β, trois vecteurs a, b et c pour modéliser la matrice tridiagonale et les trois vecteurs U, $Uold$ et Uex habituels. Ces membres ont des droits d'accès privés.

Outre le constructeur de la classe, les méthodes sont la factorisation LU, la résolution du système, le calcul de la solution exacte, l'écriture dans un fichier des résultats et le calcul d'erreur. Ces méthodes sont implémentées dans le fichier `MatTriaDiag.cpp`. Ceci est fait pour pouvoir résoudre nos deux problèmes

Le programme version 4 contient donc les fichiers de la version précédente auxquels nous ajoutons les deux fichiers nommés `MatTriaDiag_v4.hpp` et `MatTriaDiag_v4.hpp` (pour la cohérence des noms). C'est à partir du fichier principal `Chaleur1d_v4.cpp` que nous appellerons le constructeur et les méthodes de la classe `MatTriaDiag` afin de résoudre notre problème.

Si nous voulons profiter des avantages du schéma implicite, il est aussi nécessaire de modifier le constructeur du maillage afin de pouvoir choisir le pas en temps dans le cas d'une utilisation de ce schéma.

Avec ce programme version 4, nous pourrons, non seulement répondre à toutes les questions précédentes avec le choix du problème et le nombre de pas en temps, mais aussi choisir le type de schéma utilisé, soit explicite, soit implicite.

5.4 Résultats

Avec le schéma explicite nous étions limités pour le choix de dt. Dans les études précédentes nous avions $dt = \frac{dx^2}{2}$; pour $dx = 0.02$ cela donne $dt = 0.0002$. Cela assurait la stabilité du schéma. Mais pour faire varier t de 0 à 0.1, cela nécessitait 500 itérations.

Avec le schéma implicite, nous n'avons plus cette contrainte. Pour faire varier t de 0 à 0.1, nous pouvons choisir un pas $dt = 0.002$. Soit 10 fois moins d'itérations. La figure 12 illustre ceci sur le problème 2.

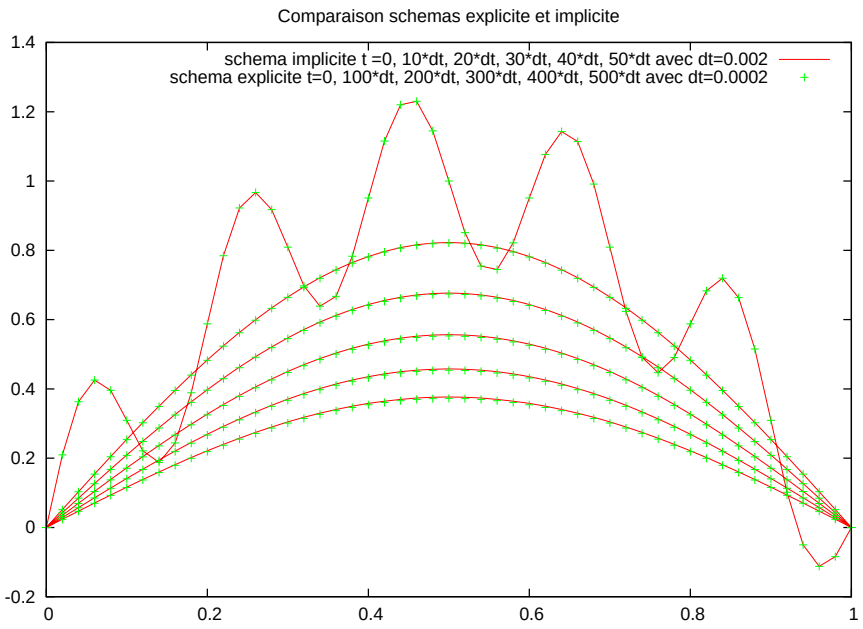

FIG. 12 – *Résolution numérique avec le programme* `Chaleur1d_v4.cpp` *et 51 points*

Le même avantage se voit sur la figure 13 avec le problème 1.

Le pas en temps est quatre fois plus grand pour le schéma implicite est nécessite donc moins d'itérations pour le même résultat.

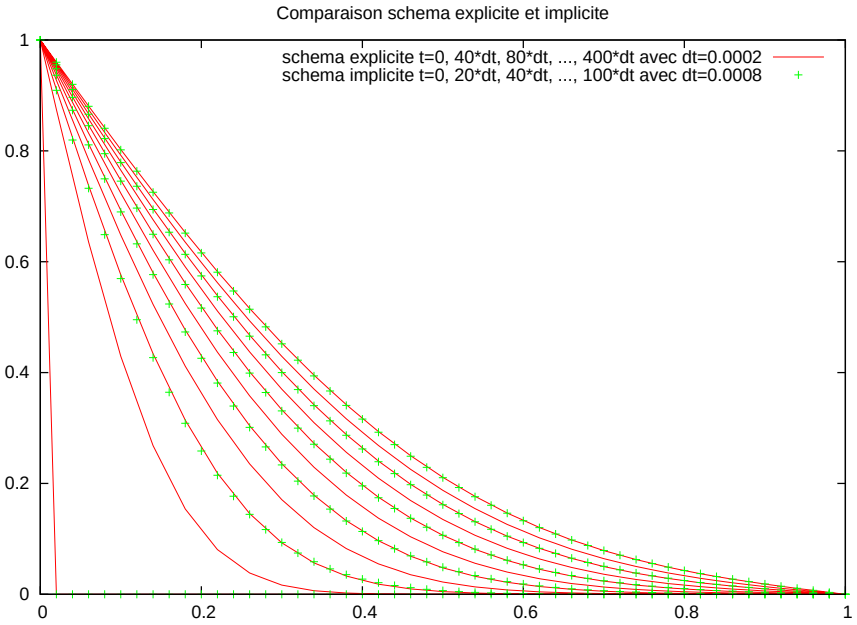

FIG. 13 – *Résolution numérique avec le programme* `Chaleur1d_v4.cpp` *et 51 points*

Cet avantage n'est bien sûr pas illimité. Plus le pas dt est grand et plus nous perdons en précision.

La figure 14 illustre cette remarque. Nous avons choisi $dt = 0.01$ pour le schéma explicite, ce qui représente 50 fois moins d'itérations par rapport au schéma implicite. Mais nous pouvons constater la perte en précision.

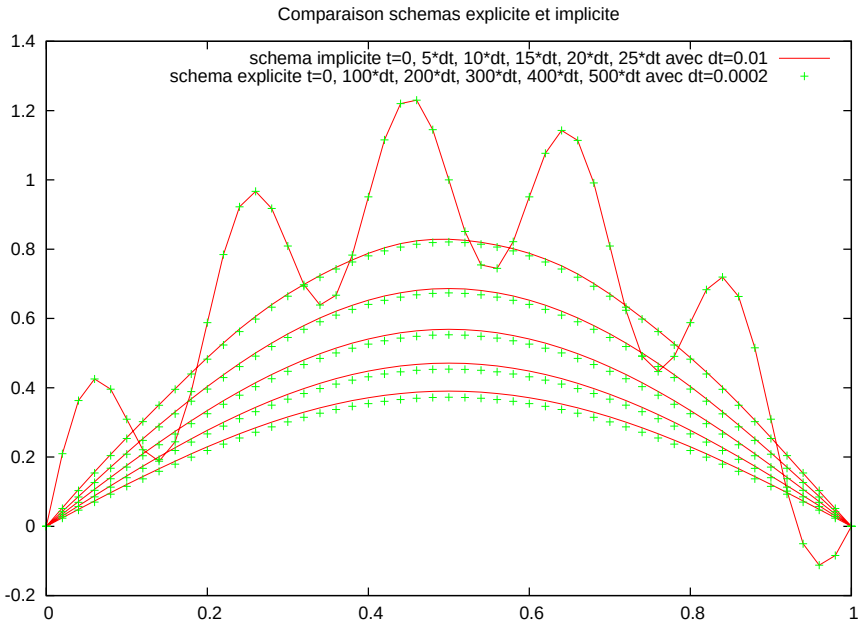

FIG. 14 – *Résolution numérique avec le programme* `Chaleur1d_v4.cpp` *et 51 points*

Il est donc important de bien choisir dt en fonction de la précision désirée.

Mais avec le schéma implicite nous avons aussi la possiblité de diminuer le pas en espace tout en gardant un pas en temps fixe. Cependant l'erreur reste gouvernée par dt tant que dx^2 est petit ou de l'ordre de dt.

Le tableau suivant, obtenu sur le problème 1, nous le confirme :

dt	dx	erreur à $t = 0.02$
0.002	0.02	0.0138501
0.002	0.002	0.0137756
0.0002	0.02	0.00146453
0.0002	0.002	0.00137767
0.00002	0.02	0.000296638
0.00002	0.002	0.000138495

6 Quatrième étude - maillage à pas variable

Nous allons maintenant construire un maillage à pas variable en espace et l'utiliser en reprenant les études précédentes. La discrétisation de l'opérateur différentiel doit en tenir compte. Il nous faudra donc modifier la classe du maillage et celle qui contient le schéma de résolution.

6.1 Discrétisation de l'opérateur différentiel

En notant $u_i = u(x_i)$, nous écrivons les développements de u aux point x_{m+1} et x_{m-1} :

$$u_{m+1} = u_m + u'_m(x_{m+1} - x_m) + u^{(2)}_m \frac{(x_{m+1} - x_m)^2}{2} + u^{(3)}_m \frac{(x_{m+1} - x_m)^3}{6} + \dots$$
$$u_{m-1} = u_m + u'_m(x_{m-1} - x_m) + u^{(2)}_m \frac{(x_{m-1} - x_m)^2}{2} + u^{(3)}_m \frac{(x_{m-1} - x_m)^3}{6} + \dots$$

On élimine u'_m en multipliant la première égalité par $(x_{m-1} - x_m)$, la deuxième égalité par $(x_{m+1} - x_m)$ puis en faisant la différence. Nous obtenons alors :

$$u^{(2)}_m = \frac{2}{x_{m+1} - x_{m-1}} \left[\frac{u_{m+1}}{x_{m+1} - x_m} + \frac{u_{m-1}}{x_m - x_{m-1}} - u_m \left(\frac{1}{x_{m+1} - x_m} + \frac{1}{x_m - x_{m-1}} \right) \right] + O(\Delta x)$$

avec

$$\Delta x = \sup_{1 \le m \le M-1} |x_m - x_{m-1}|$$

Pour le schéma explicite, cela donne :

$$U^{n+1}_m = \frac{2\kappa\delta t}{(x_{m+1} - x_{m-1})(x_{m+1} - x_m)} U^n_{m+1} + \left(1 - \frac{2\kappa\delta t}{(x_{m+1} - x_m)(x_m - x_{m-1})} \right) U^n_m + \frac{2\kappa\delta t}{(x_{m+1} - x_{m-1})(x_m - x_{m-1})} U^n_{m-1}$$

Si $x_{m+1} - x_m = x_m - x_{m-1} = \delta x$, alors $x_{m+1} - x_{m-1} = 2\delta x$ et nous retrouvons le schéma explicite à pas fixe :

$$U^{n+1}_m = \frac{\kappa\delta t}{\delta x^2} U^n_{m+1} + \left(1 - \frac{2\kappa\delta t}{\delta x^2} \right) U^n_m + \frac{\kappa\delta t}{\delta x^2} U^n_{m-1}$$

Pour le schéma implicite, nous obtenons :

$$-\frac{2\kappa\delta t}{(x_{m+1} - x_{m-1})(x_{m+1} - x_m)} U^{n+1}_{m+1} + \left(1 + \frac{2\kappa\delta t}{(x_{m+1} - x_m)(x_m - x_{m-1})} \right) U^{n+1}_m - \frac{2\kappa\delta t}{(x_{m+1} - x_{m-1})(x_m - x_{m-1})} U^{n+1}_{m-1} = U^n_m$$

Si $x_{m+1} - x_m = x_m - x_{m-1} = \delta x$, alors $x_{m+1} - x_{m-1} = 2\delta x$ et le schéma s'écrit :

$$-\frac{2\kappa\delta t}{2\delta x \delta x} U^{n+1}_{m+1} + \left(1 + \frac{2\kappa\delta t}{\delta x \delta x} \right) U^{n+1}_m - \frac{2\kappa\delta t}{2\delta x \delta x} U^{n+1}_{m-1} = U^n_m$$

Nous retrouvons le schéma implicite à pas fixe :

$$-\frac{\kappa\delta t}{\delta x^2} U^{n+1}_{m+1} + \left(1 + \frac{2\kappa\delta t}{\delta x^2} \right) U^{n+1}_m - \frac{\kappa\delta t}{\delta x^2} U^{n+1}_{m-1} = U^n_m$$

6.2 Programmation en C++

Si nous voulons un maillage utilisable pour tous les cas étudiés auparavant, il nous faut modifier le constructeur de la classe maillage qui va utiliser une fonction de maillage $fmesh$ définie dans le fichier principal et ajouter parmi les membres de la classe un pointeur sur la fonction maillage. Le vecteur représentant les points du maillage ayant des droits d'accès privés comme tous les attributs de la classe, il nous faut une méthode pour le récupérer.

Pour utiliser un maillage fixe il suffira de choisir une fonction de maillage $fmesh(i) = \dfrac{i}{M-1}\ell$ avec $i \in \{0, 1, \ldots, M-1\}]$.

Nous choisissons ici une fonction de maillage variable définie par :

$$fmesh(x) = \ell \sin\left(\epsilon \frac{\pi}{2} \frac{x}{N-1}\right) / \sin\left(\epsilon \frac{\pi}{2}\right)$$

avec par exemple $\epsilon = 3/4$. Le maillage sera donc plus fin près du point $x = 1$.

Si nous souhaitons un maillage plus fin près du point $x = 0$, nous pouvons utiliser la fonction de maillage définie par :

$$fmesh(x) = \ell \frac{\exp(x/(M-1)) - 1}{\exp(1) - 1}$$

Sur la figure 15 nous pouvons voir les différents maillages du segment $[0; 1]$.

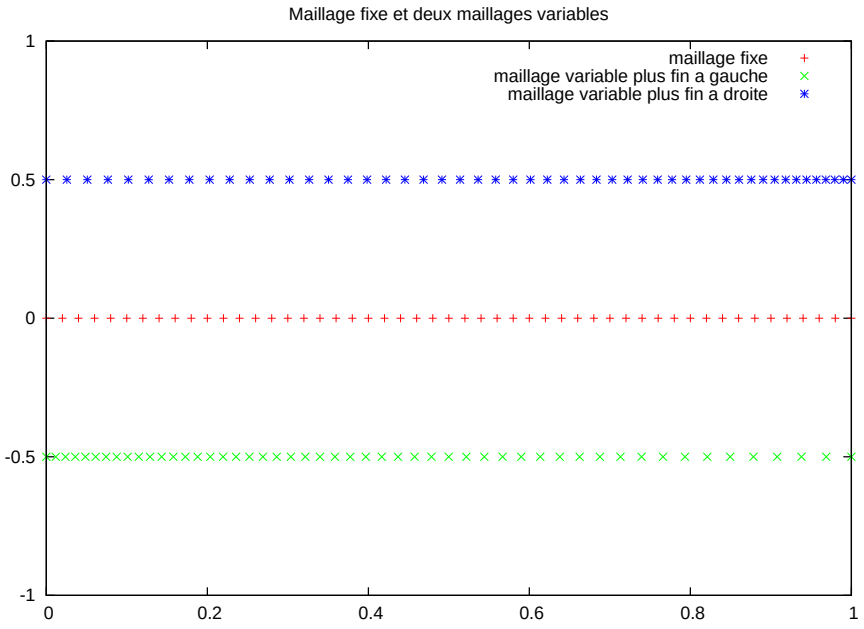

FIG. 15 – *Différents maillages avec 51 points*

Nous devons maintenant modifier les classes `VectSol` pour le schéma explicite et `MatTriaDiag` pour le schéma implicite puisque la discrétisation de l'opérateur différentiel dépend du maillage. Cette modification apparait dans le constructeur de la classe.

Nous n'utilisons plus le pas dx du maillage mais le vecteur x représentant les points du maillage. Il faut donc faire des modifications dans les méthodes de calcul des solutions exactes et d'écriture des résultats dans un fichier. Dans le cas du schéma explicite, puisque la stabilité dépend du lien entre dt et dx, il est aussi nécessaire de calculer le pas minimal du maillage variable afin d'en déduire les possibilités pour dt.

Le programme version 5 permet donc maintenant de répondre à toutes les questions posées avec le choix du problème, du type de maillage, du schéma, du pas en temps dans le cas d'un schéma implicite et enfin le nombre de pas en temps maximal ainsi que le nombre de pas entre deux affichages.

6.3 Résultats

Nous utiliserons les maillages présentés sur la figure 15.

Si nous utilisons le même nombre de points, il y a un gain en précision là où le maillage est plus fin mais une perte là où il est moins fin. Donc pour une courbe comme celle du problème 2 qui a des variations symétriques, il y a un intérêt si nous étudions une partie seulement de la solution.

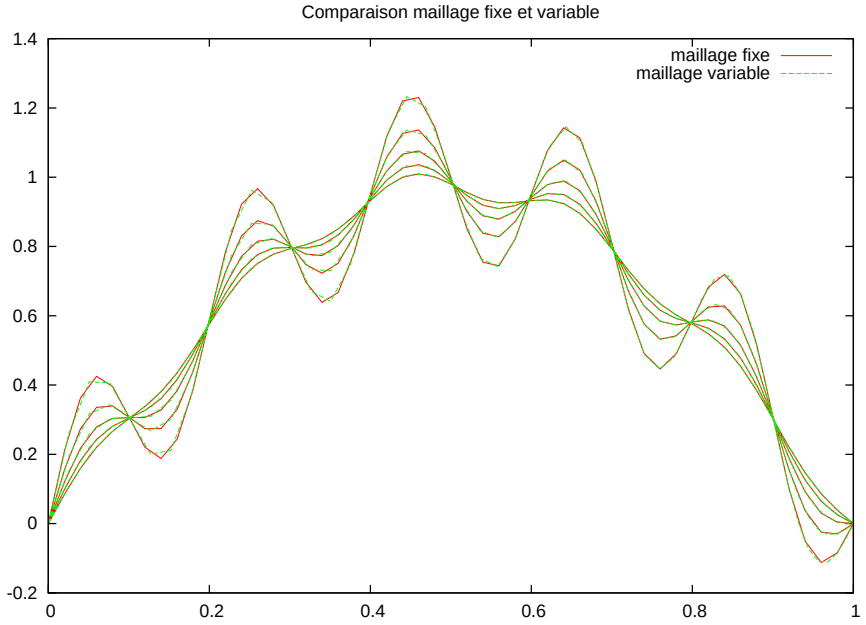

FIG. 16 – *t varie de 0 à 0.002, avec $dt = 0.00005$*

Sur les figures 17 et 18 nous avons agrandit les extémités

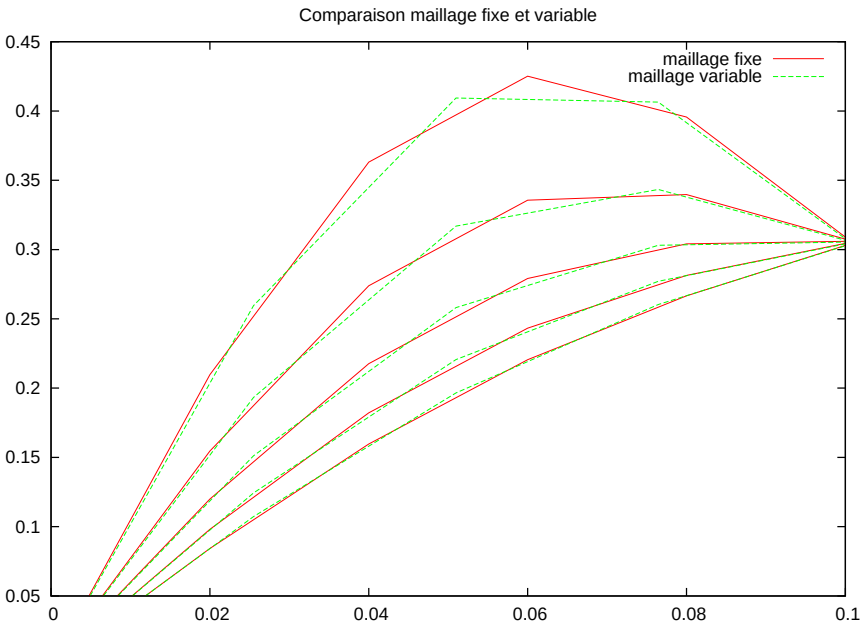

FIG. 17 – *Agrandissemnt partie gauche*

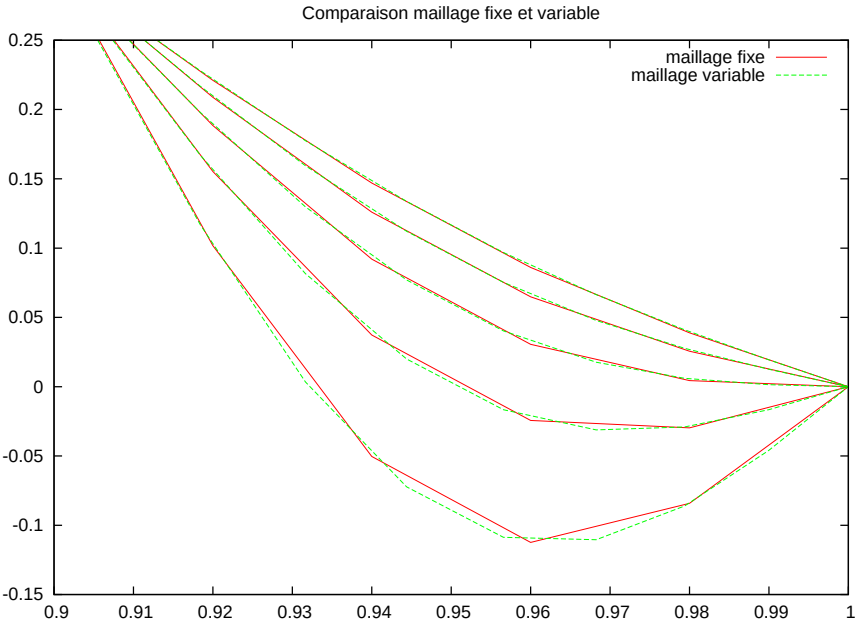

FIG. 18 – *Agrandissement partie droite*

Avec un pas dt petit, l'erreur dépend en priorité de dx et si nous mesurons l'écart en norme sup avec la solution exacte, nous perdons globalement en précision comme le montre le tableau suivant réalisé sur le problème 2 :

nombre de pas avec $dt = 0.00005$	erreur maillage fixe	erreur maillage variable
10	0.00398591	0.0056472
20	0.00493566	0.00701853
30	0.00458526	0.00654493
40	0.00378823	0.00542766

Sur le problème 1, la solution numérique est moins précise près de $x = 0$; nous avons mesuré l'écart en norme sup entre les solutions numériques et la solution exacte, avec un maillage fixe, un maillage plus fin à droite et un maillage plus fin à gauche.

Le pas en temps est $dt = 0.00005$ et nous utilisons le schéma implicite.

Les courbes des solutions numériques avec un maillage variable et de la solution exacte avec un maillage fixe sont présentées dans les figures 19 et 20.

Le tableau suivant présentent les écarts entre la solution exacte et les solutions numériques obtenues pour les trois maillages.

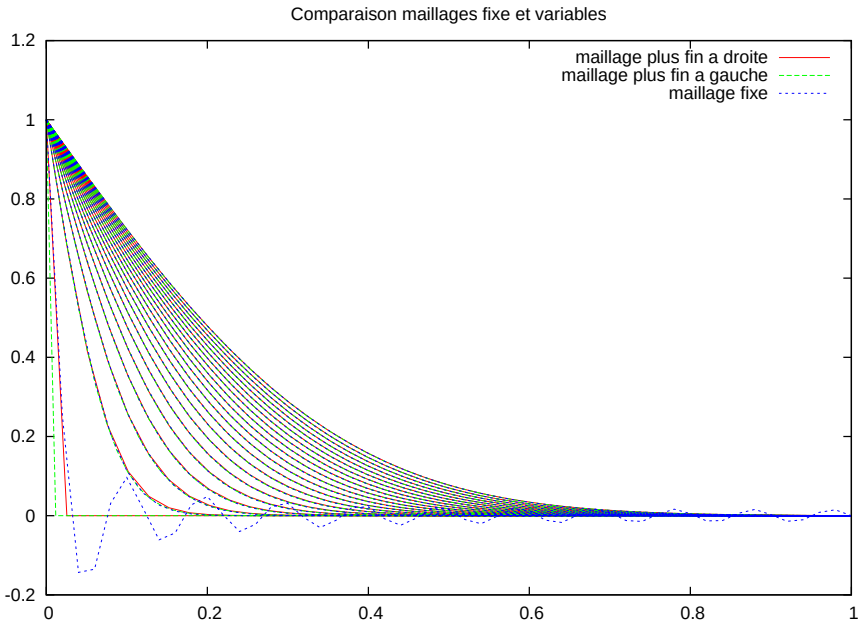

FIG. 19 – *Deux maillages variables*

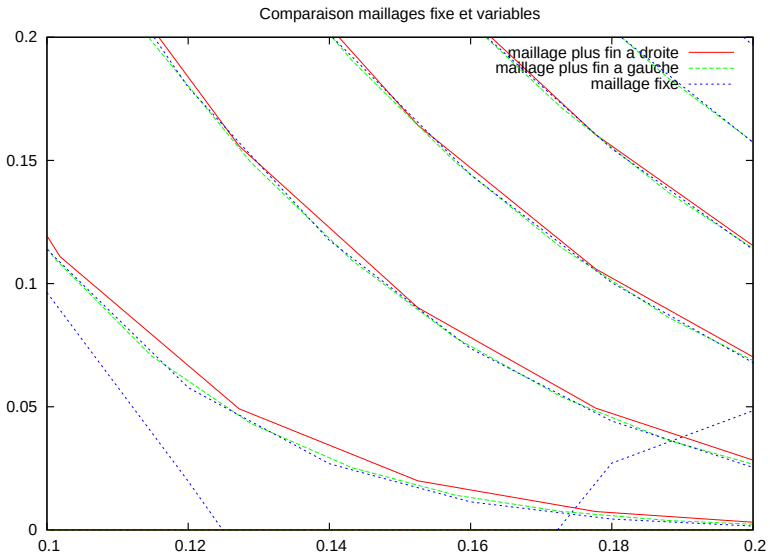

FIG. 20 – *Agrandissement partie gauche*

nombre de pas avec $dt = 0.00005$	erreur maillage fixe	erreur maillage plus fin à droite	erreur maillage plus fin à gauche
40	0.00438962	0.00491951	0.00388982
80	0.00218834	0.00251292	0.00200363
120	0.00143059	0.00168931	0.00135602
160	0.00109783	0.00127558	0.00103457
200	0.000868304	0.00101889	0.000841487
240	0.000732063	0.000853394	0.000711361
280	0.000623685	0.000735664	0.000616638
320	0.000547969	0.00064595	0.000545377
360	0.000487846	0.000576777	0.000491263
400	0.000436754	0.000522306	0.0004461
440	0.000398746	0.000477187	0.000410435
480	0.000365995	0.000438926	0.000379748
520	0.000336943	0.000405877	0.000354213
560	0.000312914	0.000377923	0.000331856
600	0.000292777	0.000354617	0.000312847
640	0.000274392	0.00033342	0.000295638
680	0.000257628	0.000314227	0.000281071
720	0.000243591	0.000298397	0.000267379
760	0.000231144	0.000283466	0.000255814
800	0.000219564	0.000269923	0.00024497

Nous remarquons donc bien l'intérêt d'avoir un maillage plus fin là où la solution exacte est la plus irrégulière, et la perte de cet intérêt lorsque, suivant l'évolution du temps, elle devient plus régulière

7 Annexe : les programmes

les différentes versions montrent l'évolution des programmes avec les questions posées. La version 5, version définitive, répond à toutes les questions.

1. Version 1
   - `Chaleur1d_v1.cpp` : schéma explicite.

2. Version 2
   - `Chaleur1d_v2.cpp` : premier problème
   - `Mesh1d_v2.hpp` et `Mesh1d_v2.cpp` : classe maillage fixe.
   - `VectSol_v2.hpp` et `VectSol_v2.cpp` : classe résolution schéma explicite.

3. Version 3
   - `Chaleur1d_v3.cpp` : les deux problèmes
   - `Mesh1d_v3.hpp` et `Mesh1d_v3.cpp` : classe maillage fixe.
   - `VectSol_v3.hpp` et `VectSol_v3.cpp` : classe résolution schéma explicite.

4. Version 4
   - `Chaleur1d_v4.cpp` : les deux problèmes
   - `Mesh1d_v4.hpp` et `Mesh1d_v4.cpp` : classe maillage fixe.
   - `VectSol_v4.hpp` et `VectSol_v4.cpp` : classe résolution schéma explicite.
   - `MatTriaDiag_v4.hpp` et `MatTriaDiag_v4.cpp` : classe résolution schéma implicite.

5. Version 5
   - `Chaleur1d_v5.cpp` : les deux problèmes
   - `Mesh1d_v5.hpp` et `Mesh1d_v5.cpp` : classe maillage fixe et variable.
   - `VectSol_v5.hpp` et `VectSol_v5.cpp` : classe résolution schéma explicite.
   - `MatTriaDiag_v5.hpp` et `MatTriaDiag_v5.cpp` : classe résolution schéma implicite.