Matmeca 1ère année - TP de Calcul Scientifique en Fortran
année 2018-2019

TP 3 : Approximation par différences finies de l'équation de la chaleur

Le but de ce TP est de programmer quelques schémas numériques vus en cours pour résoudre l'équation de la chaleur avec conditions aux limites de Dirichlet posée dans $[0, 1]$:

$$\begin{aligned}
&\partial_t T(t,x) = D\partial_{xx} T(t,x),\\
&T(t=0,x) = T_{init}(x),\\
&T(t,x=0) = 0, \quad T(t,x=1) = 0,
\end{aligned}$$

ainsi que d'étudier le comportement de la solution exacte et des solutions numériques.

On rappelle qu'il est *indispensable* d'utiliser des options de debuggage pendant la phase de conception de vos programmes (au moins `-fcheck=all`).

1 Solution exacte

On se donne la condition initiale $T_{init}(x) = x(1-x)$, et le coefficient de diffusion $D = 1$. Il est possible de calculer la solution exacte par la méthode vue en cours. La méthode de séparation des variables donne $T(t,x) = e^{-Dk^2\pi^2 t} B \sin k\pi x$, à la condition que la donnée initiale soit de la forme $B \sin k\pi x$. Comme ce n'est pas le cas, il faut décomposer la donnée initiale en série de sinus (une variante de la série de Fourier qui permet d'éliminer les cosinus). On trouve :

$$T_{init}(x) = \sum_{k=0}^{+\infty} \frac{8}{((2k+1)\pi)^3} \sin(2k+1)\pi x.$$

Enfin, le principe de superposition permet de conclure que la solution cherchée est

$$T(t,x) = \sum_{k=0}^{+\infty} e^{-D(2k+1)^2\pi^2 t} \frac{8}{((2k+1)\pi)^3} \sin(2k+1)\pi x.$$

Question 1. Dans un programme nommé `chaleur_exacte`, stocké dans le fichier `chaleur_exacte.f90`, programmez le calcul de cette solution (en prenant 5 modes seulement) et visualisez le résultat à l'écran (avec gnuplot, à différents instants de 0 à $t_{max} = 0.1$) pour vérifier que cela correspond bien au comportement qualitatif vu en cours.

Pour cela, n'hésitez pas à séparer le programme en modules : en particulier un module contenant la fonction générant la solution exacte sera utile pour la suite, de même qu'un module contenant certains variables comme D ou π.

Ensuite, effectuez `nbplot=20` sorties fichiers de la solution, où chaque sortie est une liste de `imax+2` points de coordonnées $(x_i, T(t,x_i))$ avec t un temps entre 0 et `tmax`. Prenez

`imax=200`. Le fichier correspondant sera appelé `sol_exacte_XXX.dat` où `XXX` est une chaîne de caractères représentant le numéro n de la sortie (entre 0 et `nbplot`). On rappelle l'astuce qui permet cela : déclarer la chaîne de caractères `nc` avec

```
character(len=20) :: nc
```

puis utiliser l'instruction

```
write(n c,*) n
```

et enfin

```
open(unit=10,file='sol_exacte_'//trim(adjustl(nc))//'.dat')
```

Chaque fichier peut ensuite être visualisé avec gnuplot (en superposant les courbes ou pas).

2 Résolution par le schéma d'Euler explicite à 3 points

Question 2. Copiez le fichier précédent dans le fichier `chaleur_explicite.f90`, et modifiez le programme, nommé à présent `chaleur_explicite`, pour utiliser le schéma suivant :

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = D\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2},$$

pour $i = 1$ à i_{max}, avec la donnée initiale $T_i^0 = T_{init}(x_i)$ et les conditions aux limites $T_0^n = 0$ et $T_{i_{max}+1}^n = 0$. Le maillage est uniforme de pas $\Delta x = \frac{1}{i_{max}+1}$, avec $i_{max} + 2$ noeuds définis par $x_i = i\Delta x$, pour $i = 0$ à $i_{max} + 1$. Ce schéma permet de calculer une approximation T_i^n de $T(t_n, x_i)$ aux points du maillage, pour tout temps de la forme $t_n = n\Delta t$, où Δt est le pas de temps du schéma.

Le programme commence par la définition des paramètres `imax, nplot, tmax`, le calcul de Δx et Δt. On rappelle que le schéma est L^∞ stable sous la condition CFL $\Delta t < \frac{\Delta x^2}{2D}$, ce qui conduit en général à poser $\Delta t = cfl\frac{\Delta x^2}{2D}$, où $cfl = 0.9$.

Contrairement à ce qui se passe avec une EDO, on évite de stocker la solution numérique d'une EDP à chaque instant (la place mémoire devient vite très grande). La solution à l'instant n est en fait stockée dans un tableau à une dimension, de taille `imax`, nommé par exemple `T`. La solution à l'instant $n+1$ est stockée dans le tableau auxiliaire `Tnpun`. À la fin du pas de temps, on met à jour la solution en copiant `Tnpun` dans `T`.

Pour la prise en compte des conditions aux limites, deux approches sont possibles. La première consiste à traiter différemment les cas `i=1` et `i=imax`. La deuxième évite ce traitement particulier en élargissant le tableau `T` avec la taille `T(0:imax+1)`, et à imposant en permanence `T(0)=0` et `T(imax=1)=0` (mais alors attention à la mise à jour de `T` par `Tnpun`).

Comparez alors la solution numérique à la solution exacte précédente et commentez vos résultats. Prenez `imax=50`. Pour éviter de générer trop de courbes, il est conseillé de ne créer que deux fichiers de sortie `sol_exacte_XXX.dat` et `sol_EE_XXX.dat`, où `XXX` est une chaîne de caractères représentant le temps final atteint à la fin de la simulation.

Question 3. Solution basse et haute fréquence

On modifie la donnée initiale en posant

$$T_{init}(x) = \sin(2\pi x) - 0.2\sin(20\pi x),$$

pour laquelle la solution exacte est maintenant

$$T(t,x) = e^{-D4\pi^2 t}\sin(2\pi x) - e^{-D20^2\pi^2 t}0.2\sin(20\pi x).$$

Adaptez votre programme à ce nouveau problème et créez une fonction qui génère la donnée initiale dans le même module que celui contenant la fonction générant la solution exacte.

Vérifiez que le schéma d'Euler explicite capture bien l'atténuation rapide des oscillations hautes fréquences (comparaison solution exacte et solution numérique), et que les basses fréquences sont atténuées en temps plus grand (prendre par exemple les temps `tmax=1E-04, 2E-04, 1E-3, 5E-3, 0.01, 0.05`). Il vaut mieux prendre ici `imax=100`. Comme les oscillations hautes fréquences sont atténuées très vite, il est intéressant ici d'afficher aussi la donnée initiale pour se rendre compte de cette atténuation.

Question 4. Fichier de données.

Pour la suite, vous gagnerez du temps en modifiant votre programme pour lire dans un fichier de données les paramètres `imax, tmax, cfl` et un code qui permet de choisir une des trois données initiales précédentes. Cela évite de recompiler le code à chaque nouveau jeu de données.

Après ces modifications, testez à nouveau votre programme sur l'exemple précédent.

Question 5. Créneau.

Même question que précédemment avec une donnée initiale créneau :

$$T_{init}(x) = \begin{cases} 1 & \text{si} \quad \dfrac{1}{4} < x < \dfrac{3}{4}, \\ 0 & \text{sinon.} \end{cases}$$

Observez que le créneau est progressivement atténué et lissé.

Question 6. Instabilités.

À présent, vous allez étudier le comportement des solutions numériques lorsque la condition CFL n'est pas respectée. Prenez `cfl = 1.1`, et observez le comportement des solutions numériques correspondants aux trois conditions initiales vues précédemment.

Précisez à partir de quel temps les oscillations apparaissent, et tracez quelques courbes significatives. Précisez dans quelle partie de l'intervalle $[0,1]$ les oscillations apparaissent en premier et tentez d'expliquer ce phénomène.

NB : la solution numérique étant discrète, il est en général préférable de la tracer sous forme de points (ce que fait la commande `plot` sans option). Cependant ici, pour que les oscillations soient faciles à interpréter, il vaut mieux joindre ces points par des segments, ce que permet l'option `with linespoints`.

3 Euler implicite

Copiez le fichier précédent dans le fichier `chaleur_implicite.f90`, et renommez le programme `chaleur_implicite`. Le but de cette section est de modifier le programme pour

utiliser le schéma d'Euler implicite

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = D\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2}.$$

Le maillage est le même que précédemment, et les conditions aux limites s'écrivent maintenant $T_0^{n+1} = 0$ et $T_{imax+1}^{n+1} = 0$.

Ce schéma s'écrit sous forme matricielle

$$(I + \Delta t A)T^{n+1} = T^n,$$

où

$$A = -\frac{D}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{pmatrix} \qquad T^n = \begin{pmatrix} T_1^n \\ \vdots \\ T_i^n \\ \vdots \\ T_{imax}^n \end{pmatrix}.$$

Notez que la matrice A est la même (au facteur D près) que celle construite dans le TP précédent sur l'équation de Poisson. Cette matrice étant symétrique définie positive, il en est de même pour la matrice $I + \Delta t A$, et le système peut être résolu la méthode de Cholesky. Vous pouvez ainsi ré-utiliser le solveur que vous avez programmé lors des TP précédents (une version creuse est préférable à la version pleine). Attention cependant : pour économiser du temps calcul, il faut effectuer la factorisation une seule fois, avant la boucle en temps.

Ce schéma d'Euler implicite présente l'avantage d'être stable sans condition CFL : le pas de temps peut donc être aussi grand que l'on veut.

Question 7. Dans un premier temps, testez ce schéma avec le même pas de temps qu'Euler explicite, sur la condition initiale créneau, et vérifiez que vous obtenez à peu près la même solution.

Question 8. Vérifiez ensuite que la solution reste stable, et de même qualité, avec `cfl=1.1`, sur la condition initiale créneau.

Question 9. Enfin, constatez l'intérêt de ce schéma en prenant `imax=300` et `tmax=0.5`, et en comparant le temps calcul et la qualité des solutions obtenues avec Euler explicite et Euler implicite (ce dernier étant utilisé avec `cfl=100`).

La différence en temps calcul devrait être facile à percevoir. Si nécessaire, vous pouvez utiliser la commande fortran `cpu_time`, de la façon suivante :

```
real(pr) :: t1,t2

call cpu_time(t1)
! . . .
! Code dont on veut mesurer la durée
! . . .
call cpu_time(t2)
```

4 Crank-Nicolson

Terminez ce TP avec la programmation du schéma de Crank-Nicolson, qui s'écrit

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{1}{2}\left(D\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} + D\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{\Delta x^2} \right),$$

soit sous forme matricielle

$$(I + \frac{\Delta t}{2}A)T^{n+1} = (I - \frac{\Delta t}{2}A)T^n.$$

Ce schéma doit être programmé dans le même programme qu'Euler implicite, à la suite du schéma précédent.

Pour cela, notez que la matrice du système linéaire à résoudre est la même que pour Euler implicite (avec $\frac{\Delta t}{2}$ au lieu de Δt), alors que le second membre doit être calculé, comme pour Euler explicite (avec $\frac{\Delta t}{2}$ au lieu de Δt). Attention aux conditions aux limites.

Question 10. Comparez les résultats donnés par ce schéma (avec la condition initiale en sinus, pour un temps final `tmax=0.2`, et une `cfl` de 150) aux résultats donnés par les schémas d'Euler implicite, d'Euler explicite, et la solution exacte. Commentez les résultats (des schémas sont-ils plus précis que d'autres et pourquoi ?).