TP n°1
Équation de diffusion

Le but de ce TP numérique est d'implémenter, d'étudier et de comparer différents schémas numériques de résolution de l'équation de diffusion. Ce sujet, les notes de cours associées ainsi que les ébauches de programmes nécessaires sont disponibles à l'adresse suivante : http ://userpages.irap.omp.eu/ rbelmont/.

Le langage de programmation choisi pour ce TP est *Python*. Pour ceux qui ne connaissent pas ce langage, une courte note sur les points principaux à connaître est donnée en fin de sujet.

1 Équation de diffusion

On se propose de résoudre numériquement, sur le domaine $x \in [0, L]$, l'équation de la chaleur 1D pour un champ $u(t, x)$, un coefficient de diffusion D avec un jeu de conditions initiales et de conditions aux bords de Dirichlet (type thermostat pour la temperature) :

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$$
$$u(0, x) = \sin\left(\frac{\pi x}{L}\right)$$
$$u(t, 0) = u(t, L) = 0$$

On rappelle que le paramètre CFL pour une équation de diffusion discrétisée en temps et en espace avec des pas δt et δx respectivement, est :

$$\lambda = \frac{D\delta t}{\delta x^2}$$

Les solutions numériques pourront être comparées à la solution analytique de cette équation :

$$T(t, x) = \sin\left(\frac{\pi x}{L}\right) e^{-\pi^2 \frac{Dt}{L^2}}$$

Sans perte de généralité, on pourra prendre $L = 1\text{m}$ et $D = 1\ \text{m}^2\text{s}^{-1}$ dans la suite. Pour démarrer on pourra utiliser $n_x = 64$ points en x, une condition CFL conservative $\lambda = 0.25$ et intégrer jusque $t_{\max} = 0.1\text{s}$. Ces paramètres seront ensuite à faire varier.

2 Schéma explicite FTCS

Une ébauche de programme python vous est proposée dans le fichier `diff_expl.py`.

Q01– S'en inspirer et le compléter avec un schéma d'ordre 2 FTCS (*Forward in Time, Centered in Space*).
Q02– Vérifier expérimentalement le critère de stabilité CFL. Dépend-il du nombre de points en x ?
Q03– Lorsque le schéma est stable, comment se comparent les solutions théorique et numérique à grands temps ?
Q04– Quel est l'effet du nombre de points en x ?
Q05– L'effet est-il toujours le même pour toutes les valeurs stables de la conditions CFL ? Que ce passe-t-il pour $\lambda \approx 0.17$? Expliquer ces résultats dans le cadre de l'analyse d'erreur de consistence.

3 Schéma implicite de Crank-Nicolson

Une ébauche de programme python vous est proposée dans le fichier `diff_impl.py`. Il suppose que le schéma implicite peut se mettre sous forme matricielle impliquant une matrice tri-diagonale. Dans ce cas, l'inversion de matrice peut se faire avec des algorithmes beaucoup plus rapides qu'une inversion complète et générale. En python, cette inversion se fait alors avec la fonction `tridag` à qui il faut fournir les 3 diagonales $\{a_i\}, \{b_i\}$ et $\{c_i\}$ et le membre de droite $\{s_i\}$.

$$\begin{pmatrix} b_0 & c_0 & 0 & ... & 0 & 0 & 0 \\ a_0 & b_1 & c_1 & ... & 0 & 0 & 0 \\ 0 & a_1 & b_2 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & b_{n-3} & c_{n-3} & 0 \\ 0 & 0 & 0 & ... & a_{n-3} & b_{n-2} & c_{n-2} \\ 0 & 0 & 0 & ... & 0 & a_{n-2} & b_{n-1} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ ... \\ u_{n-3} \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ ... \\ s_{n-3} \\ s_{n-2} \\ s_{n-1} \end{pmatrix}$$

Q06– Écrire le schéma numérique de Crank-Nicolson pour tous les points strictement intérieurs au domaine.
Q07– En déduire que le schéma peut se mettre sous une forme matricielle : $M_{ij}u_j = s_i$ où M_{ij} est tridiagonale. Expliciter les termes a_i de la diagonale inférieure, b_i de la grande diagonale, et c_i de la diagonale supérieure. Calculer finalement les composantes s_i.
Q08– Expliciter les valeurs aux bords a_{n_x-2}, c_0, b_0, b_{n_x-1}, s_0 et s_{n_x-1} qui permettent de reproduire les conditions aux bords.
Q09– S'inspirer du programme fourni, et le compléter avec le schéma implicite de Crank-Nicholson.
Q10– Vérifier expérimentalement l'effet du paramètre CFL sur la stabilité.
Q11– Comment se comparent les solutions théorique et numériques à grand temps ?
Q12– Quel est l'effet du nombre de points en x ?
Q13– Quel est l'effet du paramètre CFL ?

4 Pour aller plus loin

Q014– Adapter votre schéma préférer pour y imposer des conditions au bord de type Von Neumann $\partial u/\partial x = 0$, c'est à dire un flux de chaleur nul aux bord.
Q015– Le schéma de Richardson est un schéma d'ordre 2 centré en temps et en espace. Implémenter ce schéma et tester sa stabilité pour différentes valeurs du coefficient CFL

5 Memo Python

•Un programme python *myprog.py* s'execute depuis un shell avec la commande `>python myprog.py` (ou bien depuis un mode interactif (`ipython`) avec la commande `>run myprog`).

•En Python, les tabulations sont interprétées pour séparer les fonctions, les boucles, les conditions etc... Y faire très attention.

•Les commentaires commencent par le symbol `#`.

•Python est optimisé pour travailler efficacement sur les tableaux et les vecteurs. Essayer au maximum de procéder ainsi plutôt que de travailler composante par composante avec des boucles.

•Avec le package *pylab*, les opérations, fonctions et constantes usuelles sont connues (** pour les puissances, sin(), exp(), abs(), int(), modulo : n%2., π...).

•La syntaxe d'une boucle sur i allant de 0 à 9 est `for i in range(0,10):`

•On accède aux éléments d'un tableau avec `tab[0]`, `tab[6]`. Attention, le premier indice de tableau est 0.

•La syntaxe `tab[2:5]` permet de sélectionner le sous tableau constitué des éléments allant de 2 à 4 inclus (et pas 5 !).

•`linspace(x1,x2,n)` crée un tableau de n valeurs régulièrement espacées entre x1 et x2 inclus.

•`zeros(n)` et `ones(n)` produisent des vecteurs constitués de n zéros et n uns respectivement

•`roll(u,-3)` décale les éléments d'un tableau de 3 indices vers la gauche.

•Avec le package *pylab*, la syntaxe de plot est très similaire à celle de matlab.

•Ainsi, `plot(x,u,'+r')` trace u en fonction de x, avec des croix rouges.

•Des appels successifs à plusieurs instructions `plot()` se superposent sur un même graphique. L'affichage final ne se fait qu'avec l'instruction `show()`.