Introduction à Scilab pour le Calcul Scientifique

Alexandre Ern

October 10, 2017

Contents

1 Introduction

On considère le problème modèle

$$\begin{cases} \dfrac{\partial u}{\partial t}(x,t) + a\dfrac{\partial u}{\partial x}(x,t) = 0, \quad x \in \mathbb{R}, t \geq 0, \\ u(x,0) = u_0(x), \quad x \in \mathbb{R}. \end{cases} \tag{1}$$

L'inconnue est la fonction u qui dépend de la coordonnée spatiale x et du temps t. Les données du problème sont la fonction u_0 appelée *condition initiale* et le paramètre réel a appelé *vitesse d'advection*. On vérifie aisément que la solution de (1) est

$$u(x,t) = u_0(x - at). \tag{2}$$

Une interprétation physique de l'équation (1) est la suivante : on considère un fleuve rectiligne représenté par la droite réelle et qui s'écoule à la vitesse a. La quantité $u(x,t)$ représente la concentration au point x et à l'instant t d'un produit polluant qui a été déversé accidentellement dans la fleuve à $t = 0$ selon le profil u_0. Le polluant est transporté par l'écoulement et à un temps t fixé, la fonction $x \mapsto u(x,t)$ modélise la répartition de polluant le long du fleuve.

Pour simplifier, nous allons supposer que la donnée initiale u_0 est périodique (en espace) de période L si bien que nous pouvons nous restreindre à l'intervalle $x \in [0, L]$ et chercher la solution u satisfaisant

$$\begin{cases} \frac{\partial u}{\partial t}(x,t) + a\frac{\partial u}{\partial x}(x,t) = 0, \quad x \in [0,L], t \geq 0, \\ u(x,0) = u_0(x), \quad x \in [0,L], \\ u(L,t) = u(0,t), \quad t \geq 0. \end{cases} \tag{3}$$

La dernière équation est une *condition limite de périodicité*.

Afin d'approcher numériquement la solution u, on se donne

- un entier N permettant de définir un pas d'espace $\delta x = \frac{L}{N-1}$ et un maillage uniforme de l'intervalle $[0, L]$ constitué des N points $x_i = (i-1)\delta x$, $1 \leq i \leq N$;
- un pas de temps δt permettant de construire la suite d'instants discrets $t^n = n\delta t$ pour $n \geq 0$.

Notre objectif est d'évaluer des quantités U_i^n, $1 \leq i \leq N$ et $n \geq 0$, telles que

$$U_i^n \simeq u(x_i, t^n).$$

L'idée est d'approcher les dérivées partielles dans (3) à l'aide de développements de Taylor et d'écrire par exemple

$$\begin{cases} U_i^{n+1} & \simeq & U_i^n + \delta t\, \frac{\partial u}{\partial t}(x_i, t^n), \\ U_{i+1}^n & \simeq & U_i^n + \delta x\, \frac{\partial u}{\partial x}(x_i, t^n). \end{cases}$$

On obtient

$$\frac{U_i^{n+1} - U_i^n}{\delta t} + a\frac{U_i^n - U_{i-1}^n}{\delta x} = 0. \tag{4}$$

En réarrangeant (4), il vient

$$U_i^{n+1} = U_i^n - \nu(U_i^n - U_{i-1}^n), \tag{5}$$

où nous avons posé

$$\nu = \frac{a\delta t}{\delta x}. \tag{6}$$

Ce paramètre, sans dimension physique et connu sous le nom de *nombre de Courant*, jouera un rôle déterminant par la suite.

L'équation (5) définit un schéma numérique qu'on appelera *schéma upwind* (la terminologie, inspirée de l'anglais, rappelle le fait que la dérivée en espace a été décalée dans la direction d'où provient le vent (ou le courant)). Notons que (5) est un schéma *explicite* : étant données les valeurs de U_i^n pour $0 \leq i \leq N$, (5) permet de calculer explicitement les valeurs de U_i^{n+1} pour $0 \leq i \leq N$. La simulation numérique s'arrête lorsque le temps courant t^n dépasse un temps physique de simulation choisi à l'avance.

Il nous reste à approcher la condition initiale et la condition limite en écrivant

$$\begin{cases} U_i^0 = u_0(x_i), \quad 0 \leq i \leq N, \\ U_1^n = U_N^n, \quad n \geq 0. \end{cases}$$

2 Premier script Scilab : programmer un schéma

On écrira un script Scilab qu'on appelera `script1.sce` et qui comprendra 4 parties :

1. initialisation des paramètres physiques : vitesse de propagation, longueur du domaine de calcul, temps maximum de simulation et condition initiale. Pour cette dernière, on prendra une fonction en créneau

$$u_0(x) = \begin{cases} 1 & \text{si } 1 < x < 1.5, \\ 0 & \text{sinon.} \end{cases} \tag{7}$$

2. initialisation des paramètres numériques; on choisira $N = 201$ et on en déduira δx puis on choisira le nombre de Courant $\nu = 0.8$ et on en déduira le pas de temps δt.

3. boucle en temps; initialiser $(U_i^0)_{0\leq i\leq N}$ puis évaluer $(U_i^{n+1})_{0\leq i\leq N}$ en fonction de $(U_i^n)_{0\leq i\leq N}$ tant que $t^n \leq T$.

4. visualisation graphique des résultats; afficher sur une même fenêtre la solution exacte et la solution approchée.

Conseils Scilab.

- initialiser le script par la commande `clear;` qui nettoye l'espace mémoire occupé avant le lancement du script.

- **paramètres physiques** :

```
a=0.1;// vitesse d'advection
Tmax=10;// temps maximum de simulation
L=5;// longueur du domaine
```

  Pour la condition initiale, on utilisera la fonction `bool2s` dont on consultera l'aide en ligne (help → Scilab Programming → `bool2s`). La condition initiale sera programmée sous forme de fonction Scilab, ce qui devrait donner quelque chose comme

```
function v=condinit(x)
  v=bool2s((x > 1. ) & (x < 1.5));
endfunction
```

  On notera que `v` est une variable locale à la fonction `condinit`.

- **paramètres numériques** :

```
N=201;// nb de points de discrétisation en espace
dx=L/(N-1);// pas d'espace
nu=0.8;// nombre de Courant-Friedrichs-Levy
dt=nu*dx/a;// pas de temps
```

- **initialisation du maillage et de la donnée initiale** :

```
x=linspace(0,L,N)';
u=condinit(x);
```

(voir l'aide en ligne; la transposition permet de manipuler des vecteurs colonne ce qui est plus pratique pour la sortie graphique).

- **boucle en temps** : à chaque temps discret t^n, on effectue le calcul (5) de façon vectorielle en tenant compte de la condition de périodicité

```
tps=dt:dt:Tmax;
for t=tps do
  uold=u;
  u(2:N)=uold(2:N)-nu*(uold(2:N)-uold(1:N-1));
  u(1)=u(N);
end
t=tps($);// £ pointe sur le dernier élément du vecteur
```

- **sortie graphique** : voir l'aide en ligne Help → Graphic Library. On pourra par exemple utiliser les commandes suivantes :

```
xselect()
xbasc()
xtitle('temps='+string(t)+', dt='+string(dt)+', NU='+string(nu));
plot2d([x,x],[u,uexact],style = [5,2],strf = "111",leg = "sol. numerique@sol. exact
       rect = [0,-1.5,L,1.5]);
```

- pour exécuter le script, taper dans la fenêtre Scilab la commande `exec script1.sce`.

- afin de mesurer le temps d'exécution du programme, on insérera la commande

```
timer();
```

avant le démarrage de la boucle en temps et la commande

```
timer()
```

en dernière ligne du script Scilab. Le temps d'exécution écoulé entre les 2 appels de la fonction `timer` sera affiché dans la fenêtre Scilab.

Question 1 *Augmenter progressivement le paramètre ν et observer les résultats. Quelle est la valeur critique?*

3 Deuxième script Scilab : variations sur la condition initiale et le schéma numérique

Outre la condition initiale (7), on souhaite également pouvoir considérer la condition initiale

$$u_0(x) = \sin(8\pi x/L). \tag{8}$$

De plus, on souhaite avoir le choix entre 2 schémas numériques : le schéma upwind (5) et le schéma de Lax-Wendroff

$$U_i^{n+1} = U_i^n - \frac{\nu}{2}(U_{i+1}^n - U_{i-1}^n) + \frac{\nu^2}{2}(U_{i+1}^n - 2U_i^n + U_{i-1}^n), \tag{9}$$

dont la justification sera étudiée en cours de Calcul Scientifique.

Ecrire un script Scilab qu'on appelera `script2.sce` et qui offre ces 2 options.

Conseils Scilab.

- on introduira deux nouvelles variables qu'on pourra par exemple appeler `choixCI` et `choixSCH` et qui prendront la valeur 1 ou 2 en fonction de l'option retenue. On utilisera la commande `select` de Scilab dont la syntaxe est

```
function v=condinit(x)
  select choixCI
    case 1 then
     v=bool2s((x > 1. ) & (x < 1.5));
    case 2 then
     v=sin(8*%pi*x/L);
  end
endfunction
```

- à chaque temps discret t^n, on effectue le calcul (9) de façon vectorielle en tenant compte de la condition de périodicité (il y a maintenant deux cas particuliers à prendre en compte, car le schéma peut "déborder" sur la gauche ou sur la droite) :

```
u(2:N-1)=uold(2:N-1)-nu/2*(uold(3:N)-uold(1:N-2))+ ...
         nu^2/2*(uold(3:N)-2*uold(2:N-1)+uold(1:N-2));
u(N)=uold(N)-nu/2*(uold(2)-uold(N-1))+nu^2/2*(uold(2)-2*uold(N)+uold(N-1));
u(1)=u(N);
```

Question 2 *Repartir de $\nu = 0.8$ et essayer les 4 possibilités du couple condition initiale / schéma numérique. Quelles conclusions tirez-vous ? Augmenter progressivement ν pour le schéma de Lax-Wendroff et observer.*

4 Troisième script Scilab : animation graphique

L'objectif de cette nouvelle étape est de pouvoir visualer la solution numérique au fil des itérations au lieu de visualiser uniquement sa valeur finale.

Partant du script `script2.sce`, on réalisera un nouveau script qu'on appelera `script3.sce`.

Conseils Scilab.

- avant la boucle en temps, il convient pour réaliser une animation d'initialiser la fenêtre graphique de la façon suivante :

  ```
  driver("X11")
  xselect()
  xbasc()
  xset("pixmap",1)
  ```

  La commande `driver("X11")` permet d'accélérer l'affichage en ne stockant pas en mémoire toutes les données relatives au tracé de la courbe mais seulement les plus importantes (perte de la possibilité du zoom par exemple). La commande `xset("pixmap",1)` évitera le clignotement de la fenêtre lors de la boucle en temps.

- au sein de la boucle en temps, la fenêtre graphique est rafraichie de la façon suivante :

  ```
  xset("wwpc")
  uexact=a_ecrire;
  xtitle(a_ecrire);
  plot2d(a_ecrire);
  xset("wshow")
  ```

  La commande `xset("wwpc")` efface le contenu courant de la fenêtre graphique en mémoire. La commande `xset("wshow")` affiche dans la fenêtre graphique le contenu courant de la mémoire.

- à l'issue de la boucle en temps, on revient à la configuration initiale de la fenêtre graphique

  ```
  xset("pixmap",0)
  driver("Rec")
  ```

Question 3 *Reprendre les essais précédents puis augmenter progressivement le paramètre ν et observer (pour le schéma de Lax-Wendroff, prendre $\nu = 1.2$, $\nu = 1.5$ et $\nu = 1.8$). Quelles conclusions en tirer?*

5 Quatrième script Scilab : un schéma avec inversion matricielle

Nous allons modifier le schéma de Lax-Wendroff (9) en "implicitant le terme de diffusion" de la façon suivante

$$U_i^{n+1} = U_i^n - \frac{\nu}{2}(U_{i+1}^n - U_{i-1}^n) + \frac{\nu^2}{2}(U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}). \tag{10}$$

En introduisant les vecteurs $W^n = (U_i^n - \frac{\nu}{2}(U_{i+1}^n - U_{i-1}^n))_{0\leq i\leq N-1}$ et $U^{n+1} = (U_i^{n+1})_{0\leq i\leq N-1}$ ainsi que la matrice A d'ordre $N-1$ donnée par

$$A = \begin{pmatrix} 1+\nu^2 & -\frac{\nu^2}{2} & 0 & \dots & 0 & -\frac{\nu^2}{2} \\ -\frac{\nu^2}{2} & \ddots & \ddots & & & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & & & \ddots & \ddots & -\frac{\nu^2}{2} \\ -\frac{\nu^2}{2} & 0 & \dots & 0 & -\frac{\nu^2}{2} & 1+\nu^2 \end{pmatrix}$$

le schéma (10) s'écrit sous la forme

$$AU^{n+1} = W^n.$$

La condition de périodicité est utilisée directement dans le schéma numérique afin d'éliminer U_N^n. La taille des vecteurs est donc $N-1$ et non plus N comme dans les sections précédentes. Ce choix a été fait pour simplifier la structure de la matrice A.

On constate que le schéma (10) n'est plus explicite mais *implicite* : si U^n est connu, l'évaluation de U^{n+1} nécessite l'inversion d'un système linéaire. On s'attend donc à ce que le coût d'une itération dans (10) soit plus élevé que dans (9). Cependant, comme nous le verrons dans les expériences numériques, le schéma implicite (10) n'est pas limité par la valeur du nombre de Courant ν ce qui permet de considérer des pas de temps plus grands et donc de réaliser moins d'itérations en temps. Il n'est donc pas évident de déterminer *a priori* qui du schéma explicite ou implicite sera le plus performant.

Afin de réaliser les expériences numériques, on partira du script `script3.sce` dont on enlevera les parties relatives au schéma numérique. Le nouveau script sera appelé `script4.sce`.

Conseils Scilab.

- initialisation de la matrice `A` : on utilisera la fonction `diag` de Scilab :

```
aa=nu^2/2;
A=diag((1+2*aa)*ones(1,dimA))-diag(aa*ones(1,dimA-1),-1)-diag(aa*ones(1,dimA-1),1);
A(1,dimA)=-aa;
A(dimA,1)=-aa;
```

- initialisation de `Wn` :

```
Wn(2:dimA-1)=u(2:dimA-1)-nu/2*(u(3:dimA)-u(1:dimA-2));
Wn(1)=u(1)-nu/2*(u(2)-u(dimA));
Wn(dimA)=u(dimA)-nu/2*(u(1)-u(dimA-1));
```

- deux options sont possibles pour la boucle en temps : soit résoudre le système linéaire à chaque pas de temps

```
u=A\Wn;//  (dans la boucle en temps)
```

soit inverser la matrice A avant les itérations en temps puis utiliser son inverse

```
B=inv(A);
tps=dt:dt:Tmax;
for t=tps do
  // a_ecrire
  u=B*Wn;
end
```

Question 4 *Comparer les 2 options pour l'inversion du système linéaire. Réaliser des expériences numériques en faisant varier le nombre de Courant. Discuter des performances relatives des schémas explicite et implicite.*