

Pour lancer **scilab**, il suffit de taper la commande du même nom dans une fenêtre **xterm**. Un petit mémento, récapitulant quelques fonctions utiles, est disponible à la fin de l'énoncé de ce TP. Enfin, vous aurez besoin d'un éditeur de texte. Vous pouvez par exemple utiliser l'éditeur **xemacs**. Il est également conseillé de créer un répertoire afin d'y enregistrer les divers fichiers que vous serez amené à créer au cours de ce TP (en créer au moins un nouveau pour chaque partie du TP).

---

Pour commencer

1. Ouvrir une fenêtre **xterm**.
2. Créer un répertoire

```
mkdir TP1  
cd TP1
```

3. Lancer un éditeur de texte

```
xemacs &
```

4. Lancer **Scilab**

```
scilab
```

---

### 1 Équation de la chaleur - Schéma explicite

On se propose de résoudre numériquement l'équation de la chaleur

$$\begin{cases} \frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = 0 & \text{pour tout } (x, t) \in \mathbb{R} \times ]0, T] \\ u(t = 0, x) = u_0(x) & \text{pour tout } x \in \mathbb{R}. \end{cases} \quad (1)$$

On rappelle que la solution exacte de cette équation est connue et que

$$u(x, t) = \frac{1}{\sqrt{4\pi\nu t}} \int_{-\infty}^{\infty} u_0(y) \exp\left(-\frac{(x-y)^2}{4\nu t}\right) dy. \quad (2)$$

**a.** Écrire un script **scilab** qui calcul numériquement la solution de l'équation (1) à l'aide d'un schéma explicite centré d'ordre 2. Afin de tester la méthode, on pourra choisir

$$u_0(x) = \max(0, 1 - x^2),$$

$\nu = 1$  et  $T = 1$ . Bien entendu, il est impossible d'utiliser la méthode des différences finies sur  $\mathbb{R}$  tout entier. On se restreindra donc à un "gros" domaine, par exemple  $] -10, 10[$  et on imposera les conditions  $u(10, t) = u(-10, t) = 0$  pour tout temps  $t$ . A chaque itération en temps (ou toutes les  $p$  itérations), on

visualisera la solution obtenue.

Conseils de programmation: **Scilab** est optimisé pour effectuer rapidement des opérations sur les matrices et vecteurs. Il est donc fortement conseillé d'effectuer vos opérations directement sur les vecteurs et non sur les éléments de ces vecteurs. Ainsi, il est beaucoup plus rapide (et clair) d'effectuer sous **Scilab**

```
u=v+w;
```

pour additionner les deux vecteurs  $v$  et  $w$ , que d'effectuer une boucle explicite

```
for i=1:size(v,'c')
u(i)=v(i)+w(i);
end
```

De même, il vaut mieux effectuer l'opération

```
u(2:10)=v(1:9);
```

que

```
for i=1:9
u(i+1)=v(i);
end
```

pour décaler un vecteur sur la droite. En particulier, pour le travail demandé ici, seule la boucle sur la variable en temps est nécessaire.

**b.** Vérifier numériquement que la condition CFL

$$2\nu\Delta t \leq (\Delta x)^2$$

est nécessaire et suffisante pour assurer la stabilité du schéma.

**c.** Calculer numériquement la solution exacte donnée par (2) au temps final  $t = T$ . Vérifier visuellement la convergence de la solution calculée par la méthode des différences finies vers la solution exacte en superposant sur un même graphique les deux solutions obtenue pour  $t = T$ .

---

## 2 Équation de la chaleur - Schéma Implicite

Le schéma explicite précédent à le désavantage de nécessiter l'usage de pas de temps d'autant plus petits que la discrétisation en espace est fine. Afin de circonvvenir à ce problème, une solution consiste à utiliser le schéma implicite centré.

Reprendre toutes les questions précédentes avec le schéma implicite centré. Vérifier en particulier la stabilité du schéma même lorsque la condition CFL du schéma explicite n'est pas vérifiée.

Conseils de programmation: Les remarques précédentes sont toujours valables. De plus, la matrice du système introduit est creuse. Cette particularité doit absolument être exploitée. Elle permet d'économiser de manière drastique capacités en temps de calcul et mémoire. Pour la construction de la matrice, utiliser **spzeros**, **speye** et l'opérateur **colon**. Pour la résolution du système linéaire, utiliser la factorisation de Cholesky **chfact(A)** et **chsolve**

---

## 3 Équation de convection

On se propose de résoudre numériquement l'équation de convection avec conditions de périodicité au bord

$$\begin{cases} \frac{\partial u}{\partial t} + V \frac{\partial u}{\partial x} = 0 & \text{pour tout } (x, t) \in [a, b] \times ]0, T] \\ u(t = 0, x) = u_0(x) & \text{pour tout } x \in \mathbb{R} \\ u(t, a) = u(t, b) & \text{pour tout } t \in \mathbb{R}^+ \end{cases} \quad (3)$$

On pourra choisir  $a = -10$ ,  $b = 10$  de sorte à travailler sur le même domaine que précédemment.

- a. Quelle est la solution exacte de cette équation ?
- b. Implémenter le schéma explicite centré de l'équation d'advection. Vérifier qu'il est inconditionnellement instable.
- c. Rendre le schéma précédent  $L^2$  stable en utilisant une version implicite du schéma (utiliser les fonctions `lufact` et `lusolve(splu, b)` pour résoudre le système linéaire). On comparera la solution obtenue avec la solution exacte pour tout temps.
- d. Implémenter les schémas explicites décentrés amont et aval. Étudier stabilité et convergence.
- e. Même questions pour les schémas de Lax-Friedrichs et Lax-Wendroff. Comparer le comportement de ces deux derniers schémas lorsque la condition initiale est un créneau.

## Mémento Scilab

**Scilab** peut-être utilisé directement en tapant les commandes en ligne, soit par l'intermédiaire de fichier de fonctions (`.sci`) et de scripts (`.sce`), c'est à dire de listes de commandes exécutables. La syntaxe de **Scilab** est très proche de **Matlab**. Pour connaître la fonction d'une commande **Scilab**, taper `help` suivi du nom de la commande. Par exemple

```
help help;
```

La commande `apropos` permet de rechercher les occurrences d'un mot dans l'aide de **Scilab**. Enfin, rappelons qu'une commande suivie d'un point virgule désactive sa sortie texte. Quelques commandes utiles (consulter l'aide de `scilab` pour plus de détails)

### Commandes diverses

`exec('toto.sce')` - Exécute le script `toto.sce`

`getf('fonctions.sci')` - Charge dans l'environnement **Scilab** les fonctions définies par le fichier `fonctions.sci`

`halt()` - Effectue une pause jusqu'à l'activation du clavier

`modulo(r, s)` - Calcul le  $r$  modulo  $s$

`for ... end` - Déclaration d'une boucle. exemple:

```
for i=1:nx
    u[i]=i;
end
```

`clear` - Supprime les variables précédemment définies

`exit` - Pour quitter **Scilab**

### Opérations sur les vecteurs et matrices

`linspace(x1, x2, n)` - Création d'un vecteur colonne aux valeurs équiréparties

`:` - Opérateur d'extraction de sous-matrices et sous-vecteurs. Exemple:

```

u(1:10)=v(5:15)
    Copie le vecteur  $(v_5, \dots, v_{15})$  dans le vecteur  $(u_1, \dots, u_{15})$ .
' - Opérateur transposé. Exemple:
    u=[1.  1.];
    v=u';
    Définit  $v$  comme étant la transposée de  $u$ .
spzeros(n,p) - Création d'une matrice creuse nulle de taille  $n \times p$ .
speye(n,n) - Création d'une matrice identité creuse de taille  $n \times n$ .
spcho=chfact(A) - Factorisation de Cholesky de la matrice creuse  $A$ 
chsolve(spcho,b) - Calcul  $A^{-1}b$ , où  $\text{spcho}=\text{chfact}(A)$ 
splu = lufact(A) - Factorisation LU de la matrice creuse  $A$ 
lusolve(splu,b) - Calcul  $A^{-1}b$ , où  $\text{splu}=\text{lufact}(A)$ 

```

### Commandes graphiques

```

xinit() - Création d'une nouvelle fenêtre graphique
xdel() - Suppression d'un fenêtre graphique
xbasc() - Rafraîchit la fenêtre graphique
plotframe - Création d'un cadre dans la fenêtre graphique, exemple:
    plotframe([0,1,0,1],[4,2,4,2],"titre")
    crée un système de coordonnées dans le carré unité muni du titre
    titre.
plot2d - Pour affiche une ou plusieurs courbes. Exemple:
    x=linspace(0,2*pi,50);
    y1=sin(x);
    y2=cos(x);
    plotframe([0,-1,2*pi,1],[4,2,4,2],"sinus et cosinus");
    plot2d(x,[y1' y2'],leg="sin(x)@cos(x)");
xtitle - Définit le titre du graphique. Exemple: xtitle("titre")

```