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 `kwrite`. 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 (créer un nouveau fichier pour chaque schéma utilisé).

Pour commencer

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

   ```
   mkdir TP1
   cd TP1
   ```

3. Lancer éventuellement votre éditeur de texte préféré; ex :

   ```
   kwrite &
   ```

   Note : si vous n'en avez pas, utiliser l'éditeur `SciNotes` intégré à `scilab`.
4. Lancer `Scilab`

   ```
   scilab
   ```

---

1. Équation de convection

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

$$\begin{cases} \frac{\partial u}{\partial t} + V\frac{\partial u}{\partial x} = 0 & \text{pour tout } (x,t) \in [a,b]\times]0,T] \\ u(t=0,x) = u_0(x) & \text{pour tout } x \in \mathbb{R} \\ u(t,a) = u(t,b) & \text{pour tout } t \in \mathbb{R}^+ \end{cases} \tag{1}$$

Dans la suite, on prendra $a = 0$, $b = 1$, $V = 1$, et la donnée initiale $u_0(x) := \sin(2\pi x)$.

a. Quelle est la solution exacte de cette équation ?

b. Implémenter les schémas explicites décentrés aval et amont. On tracera les solutions approchées obtenues tous les 5 pas de temps par exemple. Etudier la stabilité de ces schémas suivant la valeur du paramètre

$$cfl := V\frac{\Delta t}{\Delta x}.$$

En pratique, on pourra télécharger le fichier `convection-explicite.sce` sur la page

`http://www.cmap.polytechnique.fr/~debouard/ens/MAP431`

et le compléter.

Conseils de programmation, à lire svp : Vos scripts devront commencer par la ligne

```
clear; close;
```

qui supprime les variables existantes et ferme les fenêtres graphiques.

`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. En particulier, pour le travail demandé ici, seule la boucle sur la variable en temps est nécessaire. Pour décaler circulairement un vecteur colonne de longueur N d'un cran vers le haut, on peut écrire

```
u=[v(2:n);v(1)];
```

Pour un vecteur ligne, on remplace le point virgule par une virgule. Cette méthode est très pratique quand on a des conditions aux bord périodiques comme c'est le cas ici.

Pour finir, on obtient une approximation de la constante π par la commande `%pi`.

c. On considère maintenant le schéma décentré amont. Que se passe t'il dans le cas $cfl = 1$? On impose $cfl = 1/2$. Vérifier visuellement la convergence lorsque le pas d'espace tend vers 0.

d. Etudier la convergence de la méthode en fonction de Δx en conservant $cfl = 1/2$. Pour cela, on pourra évaluer l'erreur en norme L^2 entre la solution numérique et la solution exacte au temps $T = 1$, et tracer le log de l'erreur en fonction de $\log(\Delta x)$ pour $\Delta x = 1/20,\ 1/40,\ 1/80,\ 1/160$.

e. Même questions (stabillité et convergence) pour le schéma de Lax-Wendroff :

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + V\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} - \left(\frac{V^2\Delta t}{2}\right)\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{(\Delta x)^2} = 0$$

Observer le comportement de ce schéma lorsque la condition initiale est un creneau ($u_0 = \mathbf{1}_{[0,1/2]}$). La solution approchée par le schéma de Lax-Wendroff reste t'elle positive ?

---

2. Équation de la chaleur - Schéma explicite

On se propose de résoudre numériquement l'équation de la chaleur posée sur une intervalle avec conditions au bord de type Neumann homogène

$$\begin{cases} \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0 & \text{pour tout } (x,t) \in]a,b[\times]0,T] \\ u(t=0,x) = u_0(x) & \text{pour tout } x \in]a,b[\\ \frac{\partial u}{\partial x}(t,a) = \frac{\partial u}{\partial x}(t,b) = 0 & \text{pour tout } t \in]0,T]. \end{cases} \tag{2}$$

On prendra comme plus haut $a = 0$, $b = 1$. Pour discrétiser l'opérateur Laplacien avec conditions de Neumann, on utilise la formule d'ordre 2

$$-\frac{\partial^2 v}{\partial x^2}(x_i) \approx \begin{cases} \dfrac{2v_0 - 2v_1}{(\Delta x)^2} & \text{si } i = 0, \\ \dfrac{-v_{i+1} + 2v_i - v_{i-1}}{(\Delta x)^2} & \text{si } 1 \leq i \leq N-1, \\ \dfrac{2v_N - 2v_{N-1}}{(\Delta x)^2} & \text{si } i = N \end{cases}$$

où $\Delta x := 1/N$ est le pas du maillage, les $x_i := i\Delta x$ sont les points de discrétisation et $v_i \approx v(x_i)$ pour $i = 1, \cdots, N$.

a. Ecrire un schéma explicite sous la forme

$$\frac{U^{n+1} - U^n}{\Delta t} + AU^n = 0$$

où $U^k = (u_0^k, \cdots, u_j^k \cdots, u_N^k)^t$.

b. Ecrire un script `scilab` qui calcule numériquement la solution de l'équation (2) à l'aide de ce schéma explicite centré. Tester la méthode avec $u_0(x) = 1 - 4(x - 1/2)^2$ et $T = 0.1$. On visualisera sur un même graphique la solution obtenue tous les dix pas de temps et la donnée initiale (fixer les axes grâce à la commande `plotframe`). Qu'observe t'on ?

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 de diminuer de manière drastique les besoins en mémoire et le temps de calcul. Pour la construction de la matrice, utiliser `speye`, `sparse`, ou la combinaison de commandes `diag(sparse(Vec,k))`.

c. Vérifier numériquement que la condition CFL

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

est nécessaire et suffisante pour assurer la stabilité du schéma.

d. On prend maintenant comme donnée initiale $u_0(x) = \cos(\pi x)$. Vérifier que la solution exacte est donnée par $u(t, x) = \exp(-\pi^2 t)u_0(x)$. Vérifier visuellement la convergence au temps $T = 0.1$.

---

2. Équation de la chaleur - Schéma implicite

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

a. Ecrire ce schéma sous a forme $MU^{n+1} = U^n$, pour une matrice M que l'on précisera.

Etudier l'ordre de convergence de ce schéma. On prendra pour donnée initiale $u_0(x) = \cos(\pi x)$ et on représentera le log de l'erreur L^2 à l'instant $T = 0.1$ en fonction de $\log(\delta x)$, pour $\Delta t = \Delta x = 1/100$, $1/200$, $1/400$, $1/800$, $1/1600$.

Conseils de programmation : Pour la résolution du système linéaire, utiliser la commande `A\b` qui renvoie (une valeur approchée de) la solution x du système linéaire $Ax = b$ quand A est inversible.

b. Même question pour le schéma de Crank-Nicolson (écrire le schéma sous la forme $BU^{n+1} = CU^n$).

Mémento Scilab

`Scilab` peut-être utilisé soit 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`

`exec('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

`a=[1,2,3;7 8 9]` - Création de la matrice $\begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix}$

`zeros(n,1)/zeros(1,n)` - Création d'un vecteur colonne/ligne de longueur n ayant tous ses coefficients nuls.

`ones(n,1)/ones(1,n)` - Création d'un vecteur colonne/ligne de longueur n ayant tous ses coefficients égaux à 1.

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

`:` - Opérateur d'extraction de sous-matrices et sous-vecteurs. exemple:
`u(1:10)=v(5:15)`
Copie le vecteur $(v_5, \ldots, v_{15})$ dans le vecteur $(u_1, \ldots, u_{15})$.

`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 $n \times n$.

`diag(V,k)` - Création d'une matrice pleine ayant les coefficients du vecteur V sur la diagonale k et des zeros ailleurs. Pour la diagonale principale on prend $k = 0$, pour la sous-diagonale, $k = -1$, pour la sur-diagonale, $k = 1$, etc..

`diag(sparse(V),k)` - Même résultat sous la forme d'une matrice creuse

Commandes graphiques

`plotframe` - Création d'un cadre dans la fenêtre graphique, exemple:

```
plotframe([0 0 1 1])
```

crée un système de coordonnées dans le carré unité.

`plot` - Pour afficher une ou plusieurs courbes. exemple: `plot(x,y, '-')` ou `plot(x,[y1, y2], '-')`

`clf()` - rafraichit la fenêtre graphique

`xpause(1000)` - provoque une courte pause

`xtitle` - définit le titre du graphique. exemple: `xtitle(''titre'')`