Mise en œuvre informatique à l'aide de Matlab

1. *Points à retenir* :
   - Notions et commandes Matlab des TP1 et TP2.
   - Avantage du profil pour calculer la factorisation de Cholesky d'une matrice symétrique définie-positive.
   - Efficacité de la résolution d'une système linéaire à l'aide de la factorisation de Cholesky.
2. *Fonctionnalités Matlab à maîtriser* :
   - Commande de résolution d'un système linéaire `x=A\f`.
   - Commandes affichant le temps calcul `tic ;... ;toc`, `cputime`.
   - Commande `find` pour déterminer rapidement les indices des éléments non nuls d'un tableau.
   - Commande `loglog` pour affichage en échelle $\log - \log$.

MA261. Introduction au calcul scientifique
Corrigé 4 : Discrétisation et résolution de la corde vibrante 1d

`Nous considérons l'évolution d'une corde vibrante de longueur 4, de l'instant initial zéro à l'instant final` $T = 2$.

Le problème instationnaire 1d à résoudre est (pour $c = 10$) : *trouver u tel que*

$$\frac{\partial^2 u}{\partial t^2}(x,t) - c^2 \frac{\partial^2 u}{\partial x^2}(x,t) = 0 \text{ pour } (x,t) \in]0,4[\times]0,2[, \tag{4}$$

$$u(x,0) = 2\sin(\frac{3\pi x}{2})\cos(\frac{\pi x}{2}) \text{ pour } x \in [0,4], \tag{5}$$

$$\frac{\partial u}{\partial t}(x,0) = 0 \text{ pour } x \in [0,4], \tag{6}$$

$$u(0,t) = u(4,t) = 0 \text{ pour } t \in [0,2]. \tag{7}$$

Exercice 12 Discrétisation par différences finies.
On note h_x le *pas de discrétisation en espace*, et h_t le *pas de discrétisation en temps* :

$$h_x = \frac{4}{n_x + 1}, \quad h_t = \frac{2}{n_t + 1}.$$

On note de plus $(x_j, t_m)_{j,m}$ les *points de discrétisation* dans $[0,4] \times [0,2]$, et $(u_j^m)_{j,m}$ les *valeurs approchées* de $u(x_j, t_m)_{j,m}$.

1. Que valent (x_j, t_m) ? (Sans oublier de préciser les bornes de variations de j et m.)
2. Comment prendre en compte les conditions aux limites ?
   Et les conditions initiales (5) et (6) ?
3. Construire un schéma de discrétisation de $\dfrac{\partial^2 u}{\partial x^2}(x_j, t_m)$.
4. Construire un schéma de discrétisation de $\dfrac{\partial^2 u}{\partial t^2}(x_j, t_m)$.
5. En regroupant les informations précédentes, vérifier que l'on aboutit au *schéma numérique* ci-dessous :

$$u_0^m = u_{n_x+1}^m = 0, \text{ pour } 0 \le m \le n_t + 1 \ ; \tag{8}$$

$$u_j^1 = u_j^0 = 2\sin(\frac{3j\pi h_x}{2})\cos(\frac{j\pi h_x}{2}), \text{ pour } 0 \le j \le n_x + 1 \ ; \tag{9}$$

$$u_j^{m+1} = \frac{c^2 h_t^2}{h_x^2}(u_{j+1}^m - 2u_j^m + u_{j-1}^m) + 2u_j^m - u_j^{m-1}, \text{ pour } 1 \le j \le n_x, \ 1 \le m \le n_t. \tag{10}$$

Corrigé 12 1. $(x_j, t_m) = (jh_x, mh_t)$, $0 \le j \le n_x + 1$, $0 \le m \le n_t + 1$.

2. Conditions aux limites : $u_0^m = u_{n_x+1}^m = 0$, $0 \le m \le n_t + 1$.
   Condition initiale (5) : $u_j^0 = 2\sin(\dfrac{3j\pi h_x}{2})\cos(\dfrac{j\pi h_x}{2})$, $0 \le j \le n_x + 1$.
   Condition initiale (6) : $\dfrac{u_j^1 - u_j^0}{h_t} = 0$, $0 \le j \le n_x + 1$.

3. Schéma à trois points selon x :

$$\frac{\partial^2 u}{\partial x^2}(x_j, t_m) \approx \frac{u_{j+1}^m - 2u_j^m + u_{j-1}^m}{h_x^2},\ 1 \leq j \leq n_x,\ 1 \leq m \leq n_t.$$

4. Schéma à trois points selon t :

$$\frac{\partial^2 u}{\partial t^2}(x_j, t_m) \approx \frac{u_j^{m+1} - 2u_j^m + u_j^{m-1}}{h_t^2},\ 1 \leq j \leq n_x,\ 1 \leq m \leq n_t.$$

5. Facile !

Exercice 13 Stockage et calcul de $(u_j^m)_{0\leq j\leq n_x+1, 0\leq m\leq n_t+1}$.
On considère dans cette partie la mise en œuvre informatique (en `Matlab`) du schéma numérique (8-10).

1. A un instant t_m, pourquoi peut-on se contenter de calculer le vecteur $\vec{v}^m \in \mathbb{R}^{n_x}$ de composantes $(u_j^m)_{1\leq j\leq n_x}$?
2. Vérifier que pour $1 \leq m \leq n_t$, $\vec{v}^{m+1}$ est solution de

$$\vec{v}^{m+1} = (2\mathbb{I}_{n_x} + \alpha_0 \mathbb{A}_1')\vec{v}^m - \vec{v}^{m-1},\ \text{avec } \mathbb{A}_1' = \frac{1}{(h_x)^2}\begin{pmatrix} 2 & -1 & \dots & \dots & 0 \\ -1 & 2 & -1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \dots & \dots & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n_x \times n_x}.$$

   Déterminer α_0.
3. Ecrire une fonction `Matlab` qui retourne $(u_j^{m+1})_{1\leq j\leq n_x}$, avec comme arguments d'entrée n_x, n_t et m.
4. On fixe $h_x = 1/40$ ($n_x = 159$), et on laisse h_t varier :
   (a) Quelle est la valeur de $(u_{20}^{n_t+1})$ pour

$$h_t = \frac{2}{10000}, \frac{2}{4000}, \frac{2}{2000}, \frac{2}{800}, \frac{2}{799}\ ?$$

   (b) En visualisant la solution au temps final $T = 2$ pour ces différents pas de temps, qu'en conclure au sujet de la stabilité numérique de la méthode ?

Corrigé 13 1. *A priori*, on doit déterminer $(u_j^m)_{0\leq j\leq n_x+1}$ pour m donné, or on sait déjà que $u_0^m = u_{n_x+1}^m = 0$.

2. On fixe m dans (10), et on laisse varier j dans $\{1, \cdots, n_x\}$: on arrive à

$$\vec{v}^{m+1} = -c^2 h_t^2 \mathbb{A}_1' \vec{v}^m + 2\vec{v}^m - \vec{v}^{m-1},$$

   et en particulier $\alpha_0 = -(ch_t)^2$.

3. Dans le fichier `corde_exp.m`, créer la fonction :

```
function u = corde_exp(nx,nt,c,m)
% u = corde_exp(nx,nt,c,m)
% ------------------------
% x varie de 0 a 4
% t varie de 0 a 2
```

```
% Entrees :
% (nx+2) nombre de points de discretisation en espace, de 0 a nx+1
% (nt+2) nombre de points de discretisation en temps, de 0 a nt+1
% c vitesse de propagation
% (m+1)ht dernier instant de la simulation
%
% Schema :
% Explicite
%
% Sortie :
% valeurs de la solution approchee a l'instant (m+1)ht : u(1:nx,m+1)
% (u(0,(m+1)ht) et u(4,(m+1)ht) sont connues exactement)%

% Pas de discretisation
hx = 4/(nx+1);
ht = 2/(nt+1);
coeff = -(c*ht)^2;
% Conditions initiales
v0 = zeros(nx,1);
v1 = zeros(nx,1);
for j = 1:nx
  v0(j) = 2*sin(3*j*pi*hx/2)*cos(j*pi*hx/2);
  v1(j) = v0(j);
end
% Schema explicite
% ATTENTION : A1 est la matrice du Laplacien sur ]0,4[ !!
% D'ou A1 = 1/(4^2)*Laplacien1d(nx) !
A1 = 1/(4^2)*Laplacien1d(nx);
B1 = (2*speye(nx)+coeff*A1);
for inc = 1:m
   u = B1*v1 - v0;
  v0 = v1;
  v1 = u;
end
```

4. Résultats et interprétation
   (a) Valeurs de $(u_{20}^{n_t+1})$:

$$h_t = \ 0.9998,\ 0.9998,\ 0.9997,\ 1.0000,\ -2.1912\,10^{18}.$$

   (b) La discrétisation semble fonctionner si, et seulement si,

$$h_t \leq \frac{2}{800} = \frac{1}{400} = \frac{h_x}{c} \Longleftrightarrow ch_t \leq h_x.$$

Exercice 14 Stabilité numérique.
Pour résoudre la difficulté mise en évidence à la question 4 de l'exercice précédent, on propose un second schéma numérique, en remplaçant le terme $(u_{j+1}^m - 2u_j^m + u_{j-1}^m)$ dans (10) par la moyenne

$$\frac{1}{2}(u_{j+1}^{m+1} - 2u_j^{m+1} + u_{j-1}^{m+1}) + \frac{1}{2}(u_{j+1}^{m-1} - 2u_j^{m-1} + u_{j-1}^{m-1}).$$

1. Pourquoi parle-t-on de moyenne ?
2. Ecrire ce second schéma, en conservant la notation $(u_j^m)_{j,m}$ pour la solution approchée. Si on utilise à nouveau $\vec{v}^m \in \mathbb{R}^{n_x}$, vérifier que pour $1 \leq m \leq n_t$, $\vec{v}^{m+1}$ est cette fois solution du système linéaire

$$(\alpha_1 \mathbb{A}_1' + \mathbb{I}_{n_x})\vec{v}^{m+1} = 2\vec{v}^m - (\alpha_1 \mathbb{A}_1' + \mathbb{I}_{n_x})\vec{v}^{m-1}.$$

   Déterminer α_1. Qu'en déduire sur la nature de la matrice $\alpha_1 \mathbb{A}_1' + \mathbb{I}_{n_x}$?
3. Ecrire une fonction `Matlab` qui retourne $(u_j^{m+1})_{1 \leq j \leq n_x}$, avec comme arguments d'entrée n_x, n_t et m.
4. On fixe $h_x = 1/40$ ($n_x = 159$), et on laisse à nouveau h_t varier... En visualisant la solution au temps final $T = 2$ pour différents pas de temps, qu'en conclure au sujet de la stabilité numérique du second schéma ?
5. Comparer rapidement les avantages et inconvénients des deux schémas.

Corrigé 14

1. On approche $\dfrac{\partial^2 u}{\partial x^2}(x_j, t_m)$ selon la moyenne des schémas à trois points en (x_j, t_{m+1}) et (x_j, t_{m-1}).
2. On trouve facilement $\alpha_1 = (ch_t)^2/2 = -\alpha_0/2$.
   La matrice $(\alpha_1 \mathbb{A}_1' + \mathbb{I}_{n_x})$ est donc SDP.
3. Dans le fichier `corde_imp.m`, créer la fonction :

```
 function u = corde_imp(nx,nt,c,theta,m)
% u = corde_imp(nx,nt,c,theta,m)
% -----------------------------
% Entrees :
% (nx+2) nombre de points de discretisation en espace, de 0 a nx+1
% (nt+2) nombre de points de discretisation en temps, de 0 a nt+1
% c vitesse de propagation
% theta degre d'implicitation (theta dans [0,.5])
% (m+1)ht dernier instant de la simulation
%
% Schema :
% Implicite (theta-schema)
%
% Sortie :
% valeurs de la solution approchee a l'instant (m+1)ht : u(1:nx,m+1)
% (u(0,(m+1)ht) et u(4,(m+1)ht) sont connues exactement)
%

% Pas de discretisation
% x varie de 0 a 4
% t varie de 0 a 2
hx = 4/(nx+1);
ht = 2/(nt+1);
coeff = (c*ht)^2;
% Conditions initiales
v0 = zeros(nx,1);
v1 = zeros(nx,1);
```

```
for j = 1:nx
  v0(j) = 2*sin(3*j*pi*hx/2)*cos(j*pi*hx/2);
  v1(j) = v0(j);
end
% Schema implicite
A1 = 1/(4^2)*Laplacien1d(nx);
B1 = theta*coeff*A1 + speye(nx);
B2 = 2*speye(nx)-(1-2*theta)*coeff*A1;
C1 = inv(B1);
for inc = 1:m
   u = C1*(B2*v1 - B1*v0);
  v0 = v1;
  v1 = u;
end
```

4. On a une *stabilité inconditionnelle* : les variations de h_t ne sont plus limitées. Par contre, on constate (pour h_t croissant) un amortissement et/ou un déphasage de la solution numérique, lorsque l'on superpose les solutions à l'instant final $T = 2$.
5. Premier schéma (*explicite*) :
   plus rapide, mais pas toujours stable (condition à respecter : $ch_t \leq h_x$).
   Second schéma (*implicite*) :
   moins rapide, mais toujours stable.

Mise en œuvre informatique à l'aide de Matlab

Points à retenir :

– Notions et commandes Matlab des TP1, TP2 et TP3.