TP MÉTHODES NUMÉRIQUES

Table des matières

Première partie 1

1. **Retour rapide sur Matlab**

(1) Etudier rapidement (descriptif et fonctionnement) les commandes Matlab suivantes : *cputime, tic, toc, etime, flops, whos.*

(2) Construire la matrice A de taille $N \times N$ avec $N = 256$, symétrique avec pour diagonales non nulles : la principale d'indice 0, dont la valeurs est uniformément 4, et celles d'indice 1 et $N/2$ de valeur 1. (On pourra utiliser *zeros, ones, eye, diag*).

(3) Ecrire un programme Matlab qui mesure la taille mémoire de A, le temps de calcul et le nombre d'opérations nécessaires pour effectuer 10 fois :

(a) $inv(A) * f$

(b) $A \backslash f$.

(4) Appliquer *spy* à la matrice A. Reprendre la construction de A sous forme creuse (voir *speye, spdiags*).

(5) Reprendre les calculs précédents avec différentes valeurs de N et présenter les résultats sous forme de graphique ; conclure.

Deuxième partie 2

2. **Equation de transport linéaire en dimension** 1

On considère le problème de transport pur unidimensionnel :

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in [a, b],\ t \in [0, T] \tag{1}$$

Avec $u(0, x) = u_0(x)$ définie sur $[a, b]$ suffisamment régulière , où β est la vitesse de transport.

Question préliminaire : Pourquoi équation de transport ? :

Montrer que $u(t, x) = u_0(x - \beta t)$ est l'unique solution de Eq. 1 (on vérifiera (*existence*) que la formule donnée est bien solution de l'équation Eq. 1, puis (*unicité*) que pour tout u solution de Eq. 1, on a, pour tout $x_0 : t \mapsto \varphi(t) = u(t, x_0 + \beta t)$ est constante). On "transporte" donc bien avec une vitesse β.

Notations Dans toute la suite on notera M le nombre de points en espace intérieurs au segment $]a, b[$, k le pas de temps, $N = T/k$ le nombre d'itérations en temps, $h = (b-a)/(M+1)$ le pas de discrétisation en espace, et enfin u_m^n la valeur approchée de u au point $t = nk$ et $x = a + mh$. Et on prendra

$$u_0(x) = e^{-5(5x-1)^2}$$

Enfin on supposera que $u(t, a) = 0,\ \forall t$.

2.1. **Schéma de courant.** On considère le schéma explicite suivant :

$$\frac{1}{k}(u_m^{n+1} - u_m^n) + \frac{\beta}{h}(u_m^n - u_{m-1}^n) = 0, \quad \forall m \in \{1, \cdots, M+1\},\ \forall n \in \{1, \cdots, N-1\} \tag{2}$$

La condition initiale est donnée par $u_m^0 = u_0(x_m)\ \forall m \in \{1, \cdots, M+1\}$.

(1) On pose $U_{NEW} = (u_1^{n+1}, \cdots, u_{M+1}^{n+1})^t$ et $U_{OLD} = (u_1^n, \cdots, u_{M+1}^n)^t$. Ecrire la formule donnant chacune des composantes de U_{NEW} en fonction des composantes de U_{OLD} (on pourra utiliser une formulation matricielle).

(2) On se donne a, b, T, M et k. Ecrire un programme matlab **cvt1d.m** qui effectue la résolution numérique de Eq. 1 avec le schéma Eq. 2 et qui trace
— soit sur un même graphe la solution à $t = T$ et la solution exacte.

— soit sur un même graphe la solution calculée tous les ftr (donné) pas de temps.

Dans tout la suite, on considèrera $a = 0,\ b = 2,\ \beta = 1$.

(3) Pour les valeurs $M = 99$ et $k = 0.01$,
   — comparer la solution approchée et la solution exacte au temps $t = 1.0$.
   — représenter sur un même graphique un "historique" de la solution entre $t = 0.0$ et $t = 1.0$.
   — que se passe-t-il pour $t = 3.0$?

(4) Pour $M = 99$ et $T = 0.5$, comparer les solutions obtenues avec $k = 0.01$, $k = 0.02$, $k = 0.021$ et $k = 0.05$. Conclure. Que se passe-t-il pour $k = 0.02$? Montrer que ce résultat était prévisible en regardant la formule du schéma.

(5) On fixe $\dfrac{k}{h} = 0.5$. Représenter sur un même graphique la solution exacte, et les solutions approchées à $T = 1.0$ pour $M = 49$, $M = 99$, $M = 199$, $M = 399$ et $M = 799$. Déterminer pour chacune de ces valeurs de M, le maximum de la solution ainsi que sa position et l'erreur absolue en norme L_2.

(6) Que se passe-t-il si on prend $\beta = -1$?

2.2. **Schéma de Lax-Wendroff.** On considère le schéma explicite suivant :

$$\text{(3)} \qquad \begin{cases} \dfrac{1}{k}(u_m^{n+1} - u_m^n) + \dfrac{\beta}{2h}(u_{m+1}^n - u_{m-1}^n) - \dfrac{\beta^2 k}{2h^2}(u_{m+1}^n - 2u_m^n + u_{m-1}^n) = 0, \\ \\ \forall m \in \{1, \cdots, M\},\ \forall n \in \{1, \cdots, N-1\} \end{cases}$$

On prendra pour u_{M+1}^{n+1} la valeur approchée par le schéma de Courant. On admettra que le schéma Eq. 3 est inconditionnellement stable (il n'y a pas de contrainte entre h et k).

(1) On pose $U_{NEW} = (u_1^{n+1}, \cdots, u_{M+1}^{n+1})^t$ et $U_{OLD} = (u_1^n, \cdots, u_{M+1}^n)^t$. Ecrire la formule donnant chacune des composantes de U_{NEW} en fonction des composantes de U_{OLD} (on pourra utiliser une formulation matricielle). Ecrire un programme matlab **cvt2d.m** qui effectue la résolution numérique de Eq. 1 avec le schéma Eq. 3

(2) Pour $M = 99$ et $k = 0.01$, comparer la solution aprochée obtenue à $T = 1.0$ avec la solution exacte au même temp.

(3) On fixe $\dfrac{k}{h} = 0.5$. Représenter sur un même graphique la solution exacte, et la solutions approchées à $T = 1.0$ pour $M = 49$, $M = 99$, $M = 199$, $M = 399$ et $M = 799$. Déterminer pour chacune de ces valeurs de M, le maximum de la solution ainsi que sa position et l'erreur absolue en norme L_2.

2.3. **Schéma de Lax-Friedrich.** On considère le schéma explicite suivant :

$$\text{(4)} \qquad \begin{cases} \dfrac{1}{k}\left(u_m^{n+1} - \dfrac{1}{2}(u_{m-1}^n + u_{m+1}^n)\right) + \dfrac{\beta}{2h}(u_{m+1}^n - u_{m-1}^n) = 0, \\ \\ \forall m \in \{1, \cdots, M\},\ \forall n \in \{1, \cdots, N-1\} \end{cases}$$

On prendra pour u_{M+1}^{n+1} la valeur approchée par le schéma de Courant. On admettra que le schéma Eq. 4 est inconditionnellement stable (il n'y a pas de contrainte entre h et k). Reprendre les questions de la section 15.

2.4. **Schéma de Leap-Frog.** On considère le schéma explicite suivant :

$$\begin{cases} \dfrac{1}{2k}(u_m^{n+1} - u_m^{n-1}) + \dfrac{\beta}{2h}(u_{m+1}^n - u_{m-1}^n) = 0, \\ \\ \forall m \in \{1, \cdots, M\},\ \forall n \in \{1, \cdots, N-1\} \end{cases} \tag{5}$$

On prendra pour u_{M+1}^{n+1} la valeur approchée par le schéma de Courant. u_m^1 est donnée par le schéma de Courant.

Reprendre les questions de la section 15.

2.5. **Schéma de Crank-Nicholson.** On considère le schéma semi-implicite suivant :

$$\begin{cases} \dfrac{1}{k}(u_m^{n+1} - u_m^n) + \dfrac{\beta}{2}\left(\dfrac{u_{m+1}^{n+1} - u_{m-1}^{n+1}}{2h} + \dfrac{u_{m+1}^n - u_{m-1}^n}{2h}\right) = 0, \\ \\ \forall m \in \{1, \cdots, M\},\ \forall n \in \{1, \cdots, N-1\} \end{cases} \tag{6}$$

On prendra pour u_{M+1}^{n+1} la valeur approchée par le schéma de Courant.

Reprendre les questions de la section 15.

Comparer les solutions issues des cinq schémas (Eq. 2, Eq. 3, Eq. 4, Eq. 5 et Eq. 6).

3. **Equation de transport non-linéaire en dimension** 1

On considère le problème de transport pur unidimensionnel :

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \quad x \in [0, L],\ t \in [0, T] \tag{7}$$

Avec $u(0, x) = u_0(x)$ définie sur $[0, L]$ suffisamment régulière , où $f : \mathbb{R} \to \mathbb{R}$ est $\mathcal{C}^2$ et on note $a(u) = f'(u)$. dans tout la suite du TP, on choisira

$$f(u) = \frac{u^2}{2} \tag{8}$$

On considère les schéma numériques issus d'une formalisation volumes finis, dont certains sont une généralisation des schémas vus dans le cas linéaire précédent afin d'approcher la solution de (7). Leur forme générale est

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}^n - F_{i-1/2}^n\right) \tag{9}$$

avec

$$F_{i+1/2}^n = \mathcal{F}(u_i^n, u_{i+1}^n)$$

où $\mathcal{F}$ est le flux satisfaisant la condition de consistance $\mathcal{F}(u, u) = f(u)$.

3.1. **Schémas conservatifs.**

3.1.1. *Schéma d'upwind conservatif :*

$$\mathcal{F}(u_i^n, u_{i+1}^n) = \begin{cases} f(u_i^n) & \text{si} \quad a\left(\frac{u_i^n + u_{i+1}^n}{2}\right) \geq 0 \\ f(u_{i+1}^n) & \text{si} \quad a\left(\frac{u_i^n + u_{i+1}^n}{2}\right) < 0 \end{cases} \tag{10}$$

3.1.2. *Schéma de Roe :*

(11)

$$\mathcal{F}(u_i^n, u_{i+1}^n) = \begin{cases} f(u_i^n) & \text{si} \quad A\left(u_i^n, u_{i+1}^n\right) > 0 \\ f(u_{i+1}^n) & \text{si} \quad A\left(u_i^n, u_{i+1}^n\right) < 0 \end{cases} \quad \text{où } A(u, v) = \begin{cases} \frac{f(u) - f(v)}{u - v} & \text{si} \quad u \neq v \\ f'(u) = a(u) & \text{si} \quad u = v \end{cases}$$

3.1.3. *Schéma de Engcquist-Osher :*

(12)

$$\mathcal{F}(u_i^n, u_{i+1}^n) = f^+(u_i^n) + f^-(u_{i+1}^n) \text{ où } f^+(u) = \int_0^u \max(a(s), 0)ds,\ f^-(u) = \int_0^u \min(a(s), 0)ds$$

3.1.4. *Schéma de Lax-Friedrichs :*

$$\mathcal{F}(u_i^n, u_{i+1}^n) = \frac{1}{2}\left(f(u_i^n) + f(u_{i+1}^n)\right) - \frac{\Delta t}{2\Delta x}(u_{i+1}^n - u_i^n) \tag{13}$$

3.1.5. *Schéma de Rusanov (ou Local Lax-Friedrichs).*

$$\mathcal{F}(u_i^n, u_{i+1}^n) = \frac{1}{2}\left(f(u_i^n) + f(u_{i+1}^n)\right) - \frac{a_{i+1/2}^n}{2}(u_{i+1}^n - u_i^n) \text{ où } a_{i+1/2}^n = \max_{u\in[u_i^n), u_{i+1}^n]} |a(u)| \tag{14}$$

3.1.6. *Schéma de Lax-Wendroff.*

$$\mathcal{F}(u_i^n, u_{i+1}^n) = \frac{1}{2}\left(f(u_i^n) + f(u_{i+1}^n)\right) - \frac{\Delta t}{2\Delta x} a\left(\frac{u_{i+1}^n + u_i^n}{2}\right)\left(f(u_{i+1}^n) - f(u_i^n)\right) \tag{15}$$

3.2. **Schéma non conservatif.** Jusque là, on a considéré une approximation du flux issu des volumes finis, sur l'équations mise sous forme conservative ($\partial_t u + \partial_x f(u) = 0$), mais on développer le terme $\partial_x f(u)$ sous la forme $f'(u)\partial_x u$ ($= a(u)u'(x)$, on dit que l'équation est alors mise sous forme non conservative.

3.2.1. *Schéma upwind non conservatif.*

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a(u_i^n)\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0 \quad \text{si} \quad a(u_i^n) > 0 \tag{16}$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a(u_i^n)\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0 \quad \text{si} \quad a(u_i^n) < 0 \tag{17}$$

(18)

qui peut s'écrire

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \left(a(u_i^n) + |a(u_i^n)|\right)\frac{u_i^n - u_{i-1}^n}{2\Delta x} + \left(a(u_i^n) - |a(u_i^n|\right)\frac{u_{i+1}^n - u_i^n}{2\Delta x} = 0 \tag{19}$$

3.3. **Expériences numériques.** On considèrera plusieurs conditions initiales

$$u_0(x) = e^{-(x-2)^2/0.1} \tag{20a}$$

$$u_0(x) = \begin{cases} 1 - |x-2| & \text{si } 1 \le x \le 3 \\ 0 & \text{sinon} \end{cases} \tag{20b}$$

$$u_0(x) = \begin{cases} 1 & \text{si } 1 \le x \le 2 \\ 0 & \text{sinon} \end{cases} \tag{20c}$$

$$u_0(x) = \begin{cases} -1 & \text{si } 0 \le x \le 1 \\ 1 & \text{si } 1 \le x \le 2 \\ -1 & \text{si } 2 \le x \le 5 \end{cases} \tag{20d}$$

Pour tous le schémas, la condition de stabilité (dite condition CFL ou Courant Freidrichs Levy) est donnée par

$$\Delta t \le \frac{\Delta x}{\max_u |f'(u)|}$$

(1) Calculer les fonctions f^+ et f^- du flux d'Engquist-Osher dans le cas de l'équation (Eq. 8).

(2) Implémenter la résolution de l'équation (Eq. 7) en utilisant les sept schémas décrits plus haut de $t = 0$ jusqu'au temps $T = 1$. On considèrera l'intervalle en x $[0, 5]$, avec u pas d'espace $\Delta x = 0.01$ et un pas de temps tel que $\Delta t = 0.95\Delta x$ en considérant la condition initiale (Eq.20c), et en utilisant des conditions au bord périodiques ($u(0) = u(5)$).

(3) Comparer les sept schémas dans le cas des conditions initiales (Eq. 20a) et (Eq. 20b). Qu'en concluez vous ? choisir un schéma et tracer l'évolution de la solution approchée en fonction du temps.

(4) Montrer l'effet de condition CFL sur la stabilité des différents schémas.

(5) Comparer les schémas upwind conservatif (Eq. 10) et upwind conservatif (Eq. 19) pour la condition initiale (Eq. 20c). Que remarquez vous ?

(6) Que donne le schéma de Roe pour la condition initiale (Eq. 20d) ? Comment interpétez vous le résultat ? Est ce que les autre schémas ont le même défaut ?

Troisième partie 3

4. **Equation de Laplace en dimension** 1 **en différence finies**

On considère le problème d'équations aux dérivées partielles de dimension 1 suivant, pour $a < b$:

$$-u''(x) = f(x) \quad \forall x \in]a, b[, \quad u(a) = u(b) = 0 \quad \text{avec } f \text{ donnée.} \tag{21}$$

(1) Résoudre explicitement ce problème pour les cas suivants :

(a) $a = 0$, $b = \pi$ et $f(x) = \sin(x)$.

(b) $a = -1$, $b = 2$ et $f(x) = (x - a)(x - b)$.

(c) $a = -\sqrt{2}$, $b = \sqrt{2}$ et $f(x) = -(x^2 + 4x)\exp(x)$.

(2) On va utiliser une méthode de différences finies pour résoudre de façon approchée le problème (Eq. 21). Soit M entier, on pose $h = (b - a)/(M + 1)$ et $x_i = a + i * h$, pour $i = 1, \cdots, M$. On veut alors calculer une solution approchée de (Eq. 21) aux points x_i, notée u_i.

(a) En utilisant la formule de Taylor à l'ordre 4, montrer qu'on peut approcher l'équation (21) par le systèmes d'équations

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f(x_i) \qquad i = 1, \cdots, M \tag{22}$$

Et écrire les M équations sous la forme d'un système linéaire $A_M U_M = F_M$ avec $U_M = (u_1, \cdots, u_M)^T$.

(b) Créer un fichier d'instruction **lap1d.m** qui permet de résoudre puis de visualiser les solutions exactes et approchées du problème (Eq 21).

(c) Calculer pour $M = 2^K$, $K = 2, \cdots, 9$, l'erreur entre la solution approchée calculée et la solution exacte

$$e_M = \max_{1 \le i \| eM} |u_i - u(x_i)|$$

Afficher M, e_m, e_M/h et e_M/h^2.

5. Equation de la chaleur en dimension 1 en différences finies

On considère maintenant l'équation instationnaire de la chaleur en dimension 1 :

$$\frac{\partial u}{\partial t}(t,x) = \alpha \frac{\partial^2 u}{\partial x^2}(t,x), \quad x \in]a,b[,\ t \in [0,T]. \tag{23}$$

avec $u(0,x) = u_0(x)$, pour tout $x \in]a,b[$, et $u(t,a) = u(t,b) = 0$ pour tout $t \in [0,T]$. u_0 étant donnée et vérifiant les conditions aux limites $u_0(a) = u_0(b) = 0$.

Notations Dans toute la suite on notera M le nombre de points en espace intérieurs au segment $]a,b[$, k le pas de temps, $N = T/k$ le nombre d'itérations en temps, $h = (b-a)/(M+1)$ le pas de discrétisation en espace, et enfin u_m^n la valeur approchée de u au point $t = nk$ et $x = a + mh$.

Schéma explicite de discrétisation en temps. :

$$\frac{1}{k}\left(u_m^{n+1} - u_m^n\right) - \frac{\alpha}{h^2}\left(u_{m+1}^n - 2u_m^n + u_{m-1}^n\right) = 0, \quad \forall m \in \{1,\cdots,M\},\ \forall n \in \{1,\cdots,N-1\} \tag{24}$$

La condition initiale est donnée par $u_m^0 = u_0(m)$.

(1) On pose $U_{NEW} = (u_1^{n+1},\cdots,u_M^{n+1})^t$ et $U_{OLD} = (u_1^n,\cdots,u_M^n)^t$. Ecrire la formule donnant chacune des composantes de U_{NEW} en fonction des composantes de U_{OLD}.

(2) On considère la condition initiale définie par $u_0(x) = 2x$ si $x \in]a,(b+a)/2]$ et par $u_0(x) = 2(a+b-x)$ si $x \in [(b+a)/2,b[$. Ecrire un programme matlab **chal1d.m** qui en fonction de a,b,T,M et k, effectue la résolution numérique de (Eq.23) grâce au schéma (Eq. 24) et trace sur un même graphique la solution à $t = 0$, , tous les ftr (donné) pas de temps et en $t = T$.

Dans tout la suite, on considèrera $a = 0,\ b = 1,\ \alpha = 1$.

(3) Exécuter le programme pour $M = 49$, $k = 0.0001$, $T = 0.1$ en affichant les solutions tous les 100 pas de temps.

(4) Exécuter le programme pour $M = 49$,, $T = 0.01$ pour les valeurs de k suivante $0,0001$, $0,0002$,$0,00021$,$0,00022$, en affichant les solutions tous les 10 pas de temps. Pour chacune des valeurs de k, on calculera le terme $\alpha k/h^2$. Que peut on en conclure ?

(5) reprendre la question (3) avec $u_m^0 = \mu_m(x_m - a)(x_m - b)$ pour des valeurs aléatoires de μ_m comprises entre 0 et 1.

(6) Pour la condition initiale décrite dans la question (2), la solution est connue et est donnée par :

$$u(t,x) = \sum_{j=0}^{\infty} \frac{8}{(2j+1)^2\pi^2}(-1)^j \sin((2j+1)\pi x)\exp(-(2j+1)^2\pi^2\alpha t)$$

en remarquant qu'à $t \neq 0$ fixé, $\exp(-(2j+1)^2\pi^2\alpha t)$ tends vers 0 quand $j \to \infty$, programmer le calcul de cette solution exacte.

(7) Modifier le programme de façon à calculer l'erreur entre la solution calculée et la solution exacte en norme L_∞ à $T = 0.5$. On prendra successivement $h = 0.1$, 0.05, 0.025, 0.0125, 0.00625 avec $\alpha k/h^2 = 0.25$. Tracer l'erreur en fonction de h avec une échelle log-log.

Schéma implicite de discrétisation en temps. :

$$(25)\quad \begin{cases} \dfrac{1}{k}\left(u_m^{n+1}-u_m^n\right) - \dfrac{\alpha}{h^2}\left((1-\theta)(u_{m+1}^n-2u_m^n+u_{m-1}^n)+\theta(u_{m+1}^{n+1}-2u_m^{n+1}+u_{m-1}^{n+1})\right)=0, \\ \\ \forall m\in\{1,\cdots,M\},\ \forall n\in\{1,\cdots,N-1\} \end{cases}$$

avec $\theta\in]0,1]$. Si $\theta=1/2$ ce schéma s'appelle le schéma de Crank-Nicholson, pour $\theta=1$ on parlera de schéma totalement implicite. Dans toute la suite on considèrera $\theta=1/2$.

On reprendra les questions suivantes du schéma explicite avec les modifications suivantes :

Question (1) : la relation entre U_{NEW} et U_{OLD} sera faite à l'aide d'une matrice que l'on précisera.

Question (2) : le programe matlab s'appellera **chal1d-imp.m**.

Question (3) : pas de modification.

Question (4) : on comparera les résultats avec les résultats de la question (4) du schéma explicite.

Question (7) : pas de modification.

6. Equation de Laplace en dimension 1 en éléments finis

On considère le même problème que ci dessus, soient $a<b$ deux réels, soit $\Omega=]a,b[$:

$$(26)\quad \begin{cases} -\Delta u &= f \quad \forall x\in\Omega \\ u &= 0 \quad \forall x\in\Gamma=\partial\Omega \end{cases}$$

avec $f:\Omega\to\mathbb{R}$ une fonction de classe $\mathcal{C}^2$ donnée. On considère l'espace

$$H_0^1(\Omega)=\{v\in L^2(\Omega),\ v'\in L^2(\Omega),\ v(a)=v(b)=0\}$$

6.1. **Rappels.** Il a été vu en cours que résoudre (Eq.26) au sens des distributions est équivalent à résoudre le problème variationnel suivant : trouver $u\in H_0^1(\Omega)$ tel que

$$(27)\quad \int_\Omega u'(x)v'(x)dx=\int_\Omega f(x)v(x)dx \quad \forall v\in H_0^1(\Omega)$$

Résoudre (Eq. 27) tel quel n'est pas réaliste, l'espace $H_0^1(\Omega)$ étant de dimension infinie. L'idée des éléments finis est de trouver une approximation $V_n\subset H_0^1(\Omega)$, de dimension finie, et dont on peut exhiber une base explicite et qui rend le problème facilement calculable. Dans cet environnement, on calcule donc une approximation $u_n\in V_h$ de $u\in H_0^1(\Omega)$ avec $\lim\limits_{n\to\infty}u_n=u$: trouver $u_n\in V_n$ tel que :

$$(28)\quad \int_\Omega u_n'(x)v'(x)dx=\int_\Omega f(x)v(x)dx \quad \forall v\in V_n$$

Question Soit $(\Phi_1,\cdots,\ \Phi_n)$ une base de V_n. Montrer que $u_n=\sum\limits_{i=1}^n\alpha_k\Phi_k$ est solution de (Eq. 28) si et seulement si le vecteur $A=(\alpha_1,\cdots,\alpha_n)^t$ est solution d'un système linéaire $MA=B$ où $M\in\mathcal{M}_{n\times n}$ et $B\in\mathbb{R}^n$ l'on précisera l'expression des M_{ij} et des B_j.

On construit V_n de façon à ce que la matrice M soit la plus creuse possible (le moins de termes non nuls possible) pour cela on va considèrer l'espace P_1 des fonctions affines par morceaux. Plus précisément, on discrétise $\Omega=]a,b[$ en n intervales $[x_i,x_{i+1}]$, $i\in\{0,\cdots,n+1\}$ où $x_j=a+jh$ avec $h=\dfrac{b-a}{n+1}$ et on pose

$$V_n:=V^h=\{v_h\in\mathcal{C}^0([a,b]),\ v_h \text{ est affine sur } [x_j,x_{j+1}],\ 0\le j\le n\},\ v_h(a)=v_h(b)=0$$

Question Vérifier que V^h est de dimension finie n dont une base est formée des fonctions ψ_i, $i = 1, \cdots, n$ définies par $\psi_i(x_j) = \delta_{ij}$, soit

$$\psi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h} & \text{si } x_{i-1} \leq x \leq x_i \\ \dfrac{x_{i+1} - x}{h} & \text{si } x_i \leq x \leq x_{i+1} \\ 0 & \text{sinon} \end{cases}$$

Question Dessiner les ψ_i, appelées communément les "fonctions chapeau".

Question Calculer explicitement en fonction de h les coefficients de la matrice M.

Question Calculer une approximation des coefficients B_i du second membre B en utilisant par exemple la formule des trapèzes.

Question Ecrire un programme Matlab *LaplaceEF1D(a,b,n,f)* permettant de calculer la solution u_h du problème approché.

Question On considère $f(x) = \pi^2 \sin(\pi x)$. Calculer la solution analytique du problème (Eq.26).

Question Calculer la solution approchée de la solution et comparer à la solution exacte calculée à la question précédente.

Question Calculer pour différents h, la norme L^2 de l'erreur $P_{V^h}(u) - u_h$ où $P_{V^h}(u)$ est la projection de la solution exacte sur V^h (on pourra remarquer que cette intégrale peut être calculée grâce à la matrice de masse

$$W_{i,j)} = \int_\Omega \psi_i(x)\psi_j(x)dx$$

En déduire l'ordre de la méthode en norme L^2.

7. **Equation de la chaleur en dimension 1 en éléments finis**

On considère l'équation de la chaleur :

$$\text{(29)} \qquad \begin{cases} \partial_t u(t,x) - \Delta u(t,x) = f(t,x) & \forall x \in \Omega,\ \forall t > 0 \\ u(t,x) = 0 & \forall x \in \Gamma = \partial\Omega,\ \forall t > 0, \\ u(0,x) = u^0(x) & \forall x \in \Omega \end{cases}$$

On note u^m une approximation de $u(m\Delta t, \cdot)$.

0n considère un approximation de type implicite de (Eq. 29) :

$$\frac{1}{\Delta t}(u^{m+1} - u^m) - \Delta u^{m+1} = f(t^{m+1}, x)$$

Question Ecrire la formulation variationnelle de cette formulation implicite, écrire un programme Matlab *Laplace-impEF1D(a,b,n,f)*

0n considère un approximation de type explicite de (Eq. 29) :

$$\frac{1}{\Delta t}(u^{m+1} - u^m) - \Delta u^m = f(t^m, x)$$

Question Ecrire la formulation variationnelle de cette formulation explicite, écrire un programme Matlab *Laplace-expEF1D(a,b,n,f)*

Question En reprenant l'exemple de l'exercice en différences fini, comparer les résultats numériques et la solution explicite en norme $L^\infty(0,T) \times L^2(\Omega)$ pour les deux schémas (en vérifiant la numériquement la condition de stabilité pour le schéma explicite).

8. Résolution d'EDP via FreeFem++

L'objectif de ce TP consiste à résoudre numériquement, par la méthode des éléments finis, divers problèmes variationnels en dimension 2 d'espace. A cet effet, on va utiliser le logiciel libre `FreeFem++` développé au Laboratoire Jacques-Louis Lions de Paris 6. Ce dernier permet de résoudre très simplement de nombreux problèmes variationnels.

Une documentation de `FreeFem++` est accessible sur `www.freefem.org`, à l'adresse suivante

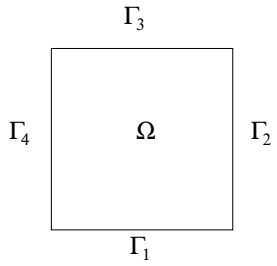
`http://www.freefem.org/ff++/ftp/freefem++doc.pdf`

8.1. Le Laplacien.

8.1.1. *Prise en main.* On se propose de résoudre numériquement le problème consistant à déterminer u tel que

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ u = 0 & \text{sur } \partial\Omega \end{cases} \tag{30}$$

dans le carré $\Omega =]0,1[^2$: Le script `FreeFem++` ci-dessous résout précisément ce problème en dix



lignes seulement (sans comptabiliser les commentaires) ! On utilise le rouge pour les commandes `FreeFem++`.

```
//Nombre de mailles suivant x et y
int Nbnoeuds=10;
```

Le texte suivi de `//` constitue des commentaires ignorés par `FreeFem++`. Chaque nouvelle variable introduite doit être précédée de son type (ici `int`, c'est à dire un entier).

```
//Definition du maillage
mesh Th=square(Nbnoeuds,Nbnoeuds,[x,y]);
```

La fonction `square` retourne un maillage structuré. Les deux premiers arguments fixent le nombre de noeuds suivant x et y respectivement. Le troisième argument est une paramétrisation de Ω pour x et y variant entre 0 et 1 (dans notre cas, c'est l'identité). Les cotés du carré sont numérotés de 1 à 4, dans le sens trigonométrique, le coté inférieur portant le label 1 (voir la figure).

```
//Fonction de x et de y
func f=x*y;

//Definition de l'espace des elements finis P1 associe
//au maillage Th
fespace Vh(Th,P1);

//uh et vh sont des elements de Vh
Vh uh,vh;
```

Les fonctions u_h et v_h appartiennent à l'espace V_h des fonctions P_1. Notons que si l'on souhaite utiliser des éléments finis P_2 et non P_1, il suffit de remplacer `P1` par `P2` dans la définition de `Vh` .

```
//Definition du probleme variationnel
problem chaleur(uh,vh,solver=LU)=
        int2d(Th)(dx(uh)*dx(vh)+dy(uh)*dy(vh))
        -int2d(Th)(f*vh)
        +on(1,2,3,4,uh=0)
;
```

La fonction `problem` permet de définir un problème variationnel, que nous dénommons ici `chaleur` . Ici, on définit le problème consistant à déterminer

$$u_h \in V_h^0 = \{w_h \in V_h \ : \ w_h = 0, \ \text{sur } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4\}$$

tel que

$$\int_\Omega \nabla u_h . \nabla v_h dx - \int_\Omega f v_h dx = 0,$$

pour tout $v_h \in V_h^0$. Notons que cette commande définit le problème mais ne le résout pas. Le problème (30) est résolu numériquement par la commande

```
//Resolution du problème
chaleur;

//On affiche le résultat
plot(uh,wait=1);
```

Remarque- La méthode de résolution utilisée pour résoudre le système est la factorisation LU.

Recopier ce script (c'est à dire les parties rouges) à l'aide de votre éditeur de texte favori et sauver le dans un fichier (sous le nom `chaleur.edp` par exemple). **Remarque-** `acroread` est munis d'une fonction permettant de faire des copier-coller.

Pour exécuter le script sous `FreeFem++`, il suffit de taper la commande shell

```
FreeFem++ chaleur.edp
```

Le résultat du calcul s'affiche dans une fenêtre graphique. Pour reprendre la main dans le shell, cliquer dans cette dernière.

8.1.2. *Est-ce que ça converge ?* On souhaite vérifier que la solution obtenue à l'aide de `FreeFem++` converge bien vers la solution exacte lorsque le pas du maillage tend vers zéro.

A cet effet, on choisit un second membre f de sorte que la solution du Laplacien u soit connue (en fait, on choisit plutôt u et on en déduit f), puis on calcule la solution discrète u_h pour des maillages de plus en plus fins. Une fois le choix de f et de u effectués, vérifier la décroissance de $\|u - u_h\|_{L^2}$ et de $\|\nabla u - \nabla u_h\|_{L^2}$.

Quelques indications pour répondre à la question :

- Le type réel est `real`. Attention, lors de la manipulation de quantités réelles, les entiers doivent être suivit d'un point. Dans le cas contraire, `FreeFem++` les interprète comme des variables de type `int` , ce qui peut être source d'erreurs.
- Les intégrales sur le maillage `Th` se calculent à l'aide de
  `int2d(Th)(` expression à intégrer `)`
- On peut faire des boucles sous `FreeFem++`. La syntaxe est identique à celle du `c++`

```
int Nbiter=10;
for(int i=0;i<Nbiter;i++){
```

On fait ceci-cela ...
```
};
```
— On peut effectuer des sorties textes soit par la sortie texte standard, soit sur la fenêtre graphique FreeFem++
```
real erreur=0.;
cout<<''erreur =''<<erreur<<endl;
```
et pour une sortie sur la fenêtre graphique en même temps que u_h,
```
real erreur=0.;
string legende=''erreur =''+erreur;
plot(uh,cmm=legende,wait=1);
```

8.1.3. *Quelques variantes.* On peut aisément modifier le script initial pour résoudre des problèmes variationnels du même type, avec conditions de Neumann, ou de Fourier par exemple.

Après avoir déterminer leur formulation variationnelle, résoudre numériquement à l'aide de FreeFem++ les deux problèmes suivant

$$\begin{cases} -\Delta u + u = f & \text{dans } \Omega \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \partial\Omega \end{cases} \tag{31}$$

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ \alpha u + \frac{\partial u}{\partial n} = g & \text{sur } \partial\Omega, \end{cases} \tag{32}$$

où f et g sont des fonctions quelconques que vous choisirez (non toutes deux nulles tout de même), et α est un réel strictement positif.

Indication : Dans la formulation variationnelle du problème (32) apparaît une intégrale sur le bord de Ω. Une intégrale sur bord $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$ se définit sous FreeFem++ par la commande
```
int1d(Th,1,2,3,4)(expression à intégrer );
```

8.2. **Résolution de l'élasticité linéarisée.** Le logiciel FreeFem++ permet également de résoudre des problèmes vectoriels. On va chercher à résoudre le problème de l'élasticité linéaire dans une poutre de longueur L et de largeur unité, fixé à son extrémité gauche, soumise à des forces volumiques f.

8.2.1. *Position du problème.* Soit $\Omega =]0, L[\times]0, 1[$, on note Γ_i, $i = 1, \ldots, 4$ les cotés du rectangle Ω en utilisant la même numérotation que précédemment. On introduit un maillage régulier $\mathcal{T}_h$ et W_h l'espace des fonctions P_1 à valeurs vectorielles (dans $\mathbb{R}^2$) associé. L'approximation de Galerkin du problème de l'élasticité linéaire consiste à déterminer

$$u_h = [u_h^1, u_h^2] \in X_h = \left\{ [w_h^1, w_h^2] \in W_h \ : \ w_h^1 = 0 \text{ et } w_h^2 = 0 \text{ sur } \Gamma_4 \right\}$$

tel que

$$\int_\Omega 2\mu e(u_h) : e(v_h) + \lambda(\operatorname{div} u_h)(\operatorname{div} v_h) dx - \int_\Omega f.v_h dx = 0, \tag{33}$$

pour tout $v_h \in X_h$, où $e(v_h)$ est le tenseur métrique linéarisé,

$$e(v_h) = (\nabla v_h + (\nabla v_h)^T)/2,$$

et μ, λ sont les coefficients de Lamé du matériau.

Résoudre ce problème variationnel à l'aide de FreeFem++. On choisira λ, μ et f à sa convenance.

Indications :

La définition de l'espace d'éléments finis P_1 à valeurs dans $\mathbb{R}^2$, s'effectue sous FreeFem++ par la commande

```
fespace Wh(Th,[P1,P1]);
```

Un élément de W_h est désigné sous FreeFem++ par ces deux composantes. L'initialisation d'un élément $u_h \in W_h$ s'effectue donc par la commande

```
Wh [uh1,uh2];
```

L'expression développée de la formulation variationnelle (33) est

$$\int_\Omega 2\mu \left(\partial_x u_h^1 \partial_x v_h^1 + \partial_y u_h^2 \partial_y v_h^2 + (\partial_x u_h^2 + \partial_y u_h^1)(\partial_x v_h^2 + \partial_y v_h^1)/2\right) + \lambda(\partial_x u_h^1 + \partial_y u_h^2)(\partial_x v_h^1 + \partial_y v_h^2)dx - \int_\Omega f^1 v_h^1 + f^2 v_h^2 dx = 0$$

8.2.2. *Affichage.* L'affichage de données vectorielles à l'aide de la fonction plot n'est pas idéal. Par conte, FreeFem++ offre la possibilité de visualiser la déformation d'un maillage $\mathcal{T}_h$. Dans un premier temps, on définit le maillage déformé

```
real exa=0.1;   //coefficient d'exageration
mesh Sh=movemesh(Th,[x+exa*uh1,y+exa*uh2]);
```

Il suffit alors d'afficher le maillage déformé $\mathcal{S}_h$

```
plot(Sh);
```

Enfin, une donnée intéressante consiste à visualiser la norme du tenseur des contraintes. Observez-vous l'apparition de singularités ?

8.2.3. *Raffinement de maillage.* On peut améliorer le calcul précédent en travaillant sur un maillage de plus en plus fin. Cependant, utiliser un maillage uniforme n'est pas optimal. Il s'avère plus intéressant de raffiner dans les régions où il se passe réellement quelque chose, c'est à dire où le gradient de la solution u_h varie rapidement. A nouveau, FreeFem++ propose une solution clé en main. La fonction adaptmesh permet de raffiner le maillage en l'adaptant à une fonction spécifiée :

```
real erreur=0.001;
Th=adaptmesh(Th,uh1,uh2,err=erreur);
```

permet d'adapter le maillage avec une précision inversement proportionnelle à erreur en fonction de $u_h = [u_h^1, u_h^2]$.