© *J.-B. A. K. <jean-baptiste.apoung@math.u-psud.fr>*

Devoir surveillé : 26 mars 2015
(Durée 1h15)

IMPORTANT- Consignes

- Les notes de cours et les scripts de TP sont autorisés. La consultation des pages Internet, en particulier de votre message électronique, est interdite. Le non respect de cette consigne entraînera l'annulation de votre note.
- Commencez par créer un répertoire **`M325_DS1_###`** où **`###`** est votre NOM. Travailler dans ce répertoire. Il devra contenir tous les fichiers dont vous aurez eu besoin.
- Lorsque la réponse à une question nécessite des commentaires rédigés, vous devez mettre ces commentaires dans vos fichiers en utilisant les commentaires du langage C.
- A la fin de l'examen, vous devez envoyer, selon votre groupe de TP, votre répertoire **`M325_DS1_###`** zippé, par mail à l'une des adresses suivantes :
  - **`clementine.courtes@math.u-psud.fr`**
  - **`irene.kaltenmark@cmla.ens-cachan.fr`**

  En cas de doute ou difficulté dans la réalisation de cette procédure, n'hésitez pas à demander de l'aide à l'enseignant en salle.

Thème - 1 *Problème*

On considère l'équation aux dérivées partielles suivante :

$$\left\{\begin{array}{rcll} \dfrac{\partial u(t,x)}{\partial t} - \dfrac{\partial^2 u(t,x)}{\partial x^2} & = f(t,x) & \forall (t,x) \in]0,T[\times]0,1[\\ u(t,0) & = 0 & \forall t \in [0,T] \\ u(t,1) & = 0 & \forall t \in [0,T] \\ u(0,x) & = u_0(x) & \text{sur } [0,1] \end{array}\right. \tag{1}$$

où, f est une fonction continue de ses arguments et u_0 une fonction de classe C^2 avec $u_0(0) = 0$ et $u_0(1) = 0$.

Nous effectuons une discrétisation de ce problème en suivant la démarche vue en cours (reportez-vous y en cas de doute).

Pour la semi-discrétisation spatiale, on se donne un entier $N \in \mathbb{N}^*$ et on définit une subdivision uniforme de $[0,1]$ de pas $h = \frac{1}{N+1}$. Ceci génère les points $0 = x_0 < \ldots < x_j < \ldots < x_{N+1} = 1$ définis par $x_j = j \times h, j = 0 \ldots, N+1$.

On désigne par $U(t)$ le vecteur de $\mathbb{R}^n$ dont la j-ème composante est une approximation de $u(t,x_j), \quad \forall t \in [0,T], j = 0 \ldots, N$. On obtient le système d'équations différentielles ordinaires suivant :

$$\left\{\begin{array}{rcll} \dfrac{dU(t)}{dt} + A_h U(t) & = F(t) & \text{sur }]0,T[\\ U(0) & = U_0 \end{array}\right. \tag{2}$$

où $A_h = \text{tridiag}(-\frac{1}{h^2}, \frac{2}{h^2}, -\frac{1}{h^2})$ est la matrice du laplacien en dimension 1, de taille N. On a posé $F(t) = [f(t,x_1), \ldots, f(t,x_N)]^T \ \forall t \in]0,T[$ et $U_0 = [u_0(x_1), \ldots, u_0(x_N)]^T$.

Pour la semi-discrétisation en temps, on se donne un entier $K \in \mathbb{N}^*$ et on considère une subdivision uniforme de $[0,T]$ de pas $\Delta t = \frac{T}{K}$, qui génère les instants $t_k = k\Delta t, k = 0, \ldots, K$. On désigne pour tout $k = 0 \ldots, K$, par U^k une valeur approchée de la solution $U(t_k)$ de (2) fournie par le θ-schéma suivant ($\theta \in [0,1]$) :

$$\left\{\begin{array}{rcll} \dfrac{U^{k+1} - U^k}{\Delta t} + (1-\theta) A_h U^k + \theta A_h U^{k+1} & = (1-\theta) F(t_k) + \theta F(t_{k+1}), & k = 0, \ldots, K-1 \\ U^0 & = U_0 \end{array}\right. \tag{3}$$

qui se met encore sous la forme

$$\begin{cases} B_{\Delta t,h,\theta}\, U^{k+1} = B_{\Delta t,h,\theta-1}\, U^k + R^k_{\theta,h}, & k = 0,\dots,K-1 \\ U^0 = U_0 \end{cases} \tag{4}$$

où on a posé $R^k_{\theta,h} = (1-\theta)F(t_k) + \theta F(t_{k+1}) \quad \forall k = 0,\dots,K-1$. $B_{\Delta t,h,\beta} = \left(\frac{1}{\Delta t} I_N + \beta A_h\right)\ \beta \in \mathbb{R}$

Q-1 : Construction des matrices et vecteurs.

Q-1-1 : Ecrire une une fonction **`gsl_matrix * gen_matrice_B (double dt, int N, double beta)`**, qui génère la matrice $B_{\Delta t,h,\beta}$ définie ci-dessus. (*On rappelle la relation* $h = \frac{1}{N+1}$).

Q-1-2 : Ecrire une fonction **`void gen_vecteur_R (gsl_vector* R, double dt, int N, double theta, int k, const gsl_vector* x)`** qui calcule et stocke dans R (déjà alloué) le vecteur $R^k_{\theta,h}$ défini ci-dessus. Ici, **`dt`** désigne le pas du temps Δt, **`N`** la taille du maillage, **`k`** l'indice de l'instant courant t_k, **`x`** le vecteur de taille $N+2$ définissant la grille (maillage).

Q-2 : Constructibilité du schéma (4)

Q-2-1 : Rappeler les conditions assurant la constructibilité du schéma (4).

Q-2-2 : Ecrire une fonction **`double est_sym_def_pos (const gsl_matrix* A)`** qui pour une matrice carrée donnée, retourne 1 si la matrice est symétrique définie positive et 0 sinon. (*On pourra calculer la différence* $B = A^T - A$ *et utiliser la fonction* ***`gsl_matrix_isnull (const gsl_matrix* A)`*** *qui retourne 1 si la matrice est nulle et 0 sinon.* On fournit aussi une fonction **`void valeurs_propres_extremes(gsl_matrix * A, double lambda[2])`** qui pour une matrice diagonalisable dans $\mathbb{R}$, retourne dans $\lambda[0]$ respectivement $\lambda[1]$, sa plus petite respectivement plus grande valeur propre.)

Q-2-3 : Vérifications : On pose $N = 100, \Delta t = 10^{-1}$.

- Vérifier numériquement que la matrice $B_{\Delta t,h,\theta}$ est symétrique pour tout $\theta \in [0,1]$: *on pourra tirer au hasard un valeur* $\theta \in [0,1]$ *et vérifier que* $B_{\Delta t,h,\theta}$ *est symétrique définie positive.*
- On peut répéter cette expérience aléatoire un certain nombres de fois et déterminer la moyenne empirique et conclure.

Le schéma peut alors s'écrire sous la forme :

$$\begin{cases} U^{k+1} = B^{-1}_{\Delta t,h,\theta} B_{\Delta t,h,\theta-1}\, U^k + B^{-1}_{\Delta t,h,\theta} R^k_{\theta,h}, & k = 0,\dots,K-1 \\ U^0 = U_0 \end{cases} \tag{5}$$

Q-3 : Stabilité du schéma

Q-3-1 : Rappeler les conditions assurant la stabilité de ce schéma pour la norme $\|\ \|_h^2 = h\|\ \|_2^2$ de $\mathbb{R}^N$. où h est le pas de discrétisation en espace et $\|\ \|_2$ la norme euclidienne.

Q-3-2 : Ecrire une fonction **`double est_normale (const gsl_matrix* A)`** qui pour une matrice carrée donnée retourne 1 si la matrice est normale et 0. sinon. (*On pourra calculer la différence* $B = A^T\ A - A *\ A^T$ *et vérifier qu'elle est bien nulle.*

Q-3-3 : Vérifications : On pose $N = 100, \Delta t = 10^{-2}$.

- Vérifier numériquement que la matrice $B^{-1}_{\Delta t,h,\theta}\ B_{\Delta t,h,\theta-1}$ est normale pour tout $\theta \in [0,1]$: *on pourra tirer au hasard un valeur* $\theta \in [0,1]$ *et vérifier que* $B^{-1}_{\Delta t,h,\theta}\ B_{\Delta t,h,\theta-1}$ *est normale.*
- On peut répéter cette expérience aléatoire un certain nombre de fois, déterminer la moyenne empirique et conclure.

Q-3-4 : Fournir une fonction **`double schema_est_stable (double dt, int N, double theta)`** qui retourne 1 si le schéma est stable en norme $\|\ \|_h$ et 0 sinon. *On pourra générer et calculer les valeurs propres extrêmes de la matrice* $B^{-1}_{\Delta t,h,\theta}\ B_{\Delta t,h,\theta-1}$.

Q-3-5 : Tester la fonction pour les valeurs suivantes :

- $N = 100$, $\Delta t = 10^{-3}$, $\theta = 0.75$
- $N = 100$, $\Delta t = 10^{-3}$, $\theta = 0.4$
- $N = 100$, $\Delta t = 2\ 10^{-}4$, $\theta = 0.4$

Conclure en évoquant un résultat du cours.

Q-4 : Calcul de solutions et comparaisons

On souhaite calculer la valeur approchée de la solution à l'instant final, U^K, en partant de la donnée initiale $u_0(x)$ et du terme source $f(t,x)$ choisis tels que la solution exacte soit $u(t,x) = \sin(\pi x)e^{-\pi^2 t}$.

Q-4-1 : Est-il raisonnable d'utiliser la formule (5) dans laquelle on a calculé et stocké explicitement $B^{-1}_{\Delta t,N,\theta}$?

En utilisant si possible les outils développer en Tps **`decompLU, descenteRemonteeLU etc.`** Ecrire une fonction qui calcule le dernier terme U^K fourni par le schéma.

Q-4-2 : Pour $N = 100$, $\Delta t = 10^{-1}$, $\theta = 1.$, $T = 1$ écrire dans un fichier la solution finale obtenue.

Q-4-3 : Pour $N = 100$, $\Delta t = 10^{-1}$, $\theta = 0.5$, $T = 1$ écrire dans un fichier la solution finale obtenue.

Q-4-4 : Laquelle des solutions obtenues ci-dessus vous semble plus précise ? Justifier.

Thème - 2 *Utilitaires*

Les quelques recommandations suivantes peuvent être utiles. Dans toutes les fonctions qui suivent, les matrices sont supposées déjà allouées à la bonne taille. On sera aussi amené à inclure le fichier entête **`gsl_blas.h`**. s et r sont des scalaires.

- $C = s\ A \times B + r\ C$, faire **`gsl_blas_dgemm (CblasNoTrans,CblasNoTrans, s, A, B, r, C)`** .
- $C = s\ A^T \times B + r\ C$, faire **`gsl_blas_dgemm (CblasTrans,CblasNoTrans, s,A, B, r, C)`** .
- $C = s\ A \times B^T + r\ C$, faire **`gsl_blas_dgemm (CblasNoTrans,CblasTrans, s,A, B, r, C)`** .
- $A = B$, faire **`gsl_matrix_memcpy (A, B)`**.
- $A = A^T$, faire **`gsl_matrix_transpose (A)`**.
- $A = B^T$, faire **`gsl_matrix_transpose_memcpy (A, B)`**.
- $A = A + B$, faire **`gsl_matrix_add (A,B)`**.
- $A = A - B$, faire **`gsl_matrix_sub (A,B)`**.
- $A = s * A$, faire **`gsl_matrix_scale (A,s)`** où s est un scalaire.
- $A = I$, faire **`gsl_matrix_set_identity (A)`**. I matrice identité.

On fournit aussi les fonctions suivantes :

Listing 1 – Quelques fonctions fournies

```
/* Il faut ajouter :
#include <gsl/gsl_eigen.h>
#include <gsl/gsl_blas.h>
*/
void valeurs_propres_extremes(gsl_matrix * A, double lambda[2])
{
  int n = A->size1;  int m = A->size2;
  assert(n == m);
  gsl_vector *eval = gsl_vector_alloc (n);
  gsl_eigen_symm_workspace * w = gsl_eigen_symm_alloc (n);
  gsl_eigen_symm (A, eval, w);
  gsl_vector_minmax (eval, &lambda[0], &lambda[1]);
  gsl_eigen_symm_free(w);
  gsl_vector_free(eval);
}
//Calcul l'inverse d'une matrice inversible et le retourne
gsl_matrix* gen_matrix_inverse(gsl_matrix*  A)
 {
  int n = A->size1;   int m = A->size2;  assert(n == m);
  int s;
  gsl_permutation * p = gsl_permutation_alloc (A->size1);
  gsl_matrix* inverse = gsl_matrix_alloc (A->size1, A->size2);
  gsl_matrix* lu = gsl_matrix_alloc (A->size1, A->size2);
  gsl_matrix_memcpy (lu,A);
  gsl_linalg_LU_decomp (lu, p, &s);
  gsl_linalg_LU_invert (lu, p, inverse);
  gsl_permutation_free (p);
  gsl_matrix_free (lu);
  return inverse;
  }
```