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

Fiche de TP 1 : Production de quelques matrices

Le présent TP porte sur la production de quelques systèmes linéaires. Ce sera aussi l'occasion de renouer avec la programmation en C et de se familiariser avec certains utilitaires dont nous aurons besoins dans la suite.

Afin de mener à bien la résolution des exercices proposés, on fera usage des utilitaires :

- *Csparse ou GSL au choix, pour le stockage des matrices et vecteurs et pour la résolution des systèmes linéaires.*
- *Gnuplot pour la représentation graphique des solutions.*

Une présentation succincte de ces utilitaires est accessible sur Dokeos dans le dossier dédié au présent cours.

Thème - 1 *Génération de matrices particulières*

Note 1.

Dans cette partie on travaillera avec les matrices pleines. On fera donc usage de l'utilitaire GSL. Le but de cet exercice est de générer aléatoirement des matrices particulières. Ces matrices seront utiles pour tester différents algorithmes. (Les constructions proposées devront être justifiées). Il est particulièrement conseiller de créer un fichier entête contenant les déclarations des fonctions demandé et un fichier source contenant les définitions de ces fonctions.

On fournit ici une fonction qui génère une matrice de taille $n \times m$, dont les éléments sont choisis aléatoirement entre les valeurs 0 et 1.

Listing 1 – Fonction générant une matrice aléatoire de taille n x m d'éléments situés entre 0 et 1

```
gsl_matrix *
genMatAleatoire (int n, int m)
{
  gsl_rng *r;
  gsl_matrix *A;
  int i, j;
  gsl_rng_env_setup ();
  r = gsl_rng_alloc (gsl_rng_mt19937);
  A = gsl_matrix_alloc (n, m);
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      gsl_matrix_set (A, i, j, gsl_rng_uniform (r));
  gsl_rng_free (r);
  return A;
 }
```

Q-1 : Écrire une fonction de prototype **`gsl_matrix* genMatSymetrique(int n)`**, renvoyant une matrice réelle symétrique de taille $n \times n$.

Q-2 : Écrire une fonction de prototype **`gsl_matrix* genMatReguliere(int n)`**, renvoyant une matrice réelle inversible de taille $n \times n$.

Q-3 : Écrire une fonction de prototype **`gsl_matrix* genMatReguliereI(int n)`**, renvoyant une matrice réelle triangulaire inférieure inversible de taille $n \times n$.

Q-4 : Écrire une fonction de prototype **`gsl_matrix* genMatRegulieresS(int n)`**, renvoyant une matrice réelle triangulaire supérieure inversible de taille $n \times n$.

Q-5 : Écrire une fonction de prototype **`gsl_matrix* genMatHasard(int m, int n,int p)`**, renvoyant une matrice réelle de dimension $m \times n$, dont les éléments sont choisis aléatoirement entre les valeurs $-p$ et p.

Q-6 : Écrire une fonction de prototype **`gsl_matrix* genMatHasardBin(int m,int n)`**, renvoyant une matrice réelle de dimension $m \times n$, dont les éléments sont choisis aléatoirement parmi les deux valeurs 0 et 1.

Q-7 : Écrire une fonction de prototype **`gsl_matrix* genMatSdp(int n)`**, renvoyant une matrice réelle symétrique définie positive de taille $n \times n$.

Q-8 : Écrire une fonction de prototype **`gsl_matrix* genMatDiagDom(int n)`**, renvoyant une matrice réelle à diagonale dominante de taille $n \times n$.

Q-9 : Ecrire une fonction de prototype **`gsl_matrix* genMatRang(int m,int n, int r)`**, renvoyant une matrice réelle de taille $m \times n$ de rang r fixé.

Thème - 2 *Discrétisation du Laplacien*

Note 2.

On se propose ici de construire le système linéaire résultant de la discrétisation par différences finies du Laplacien sur une grille (maillage) cartésien et d'inverser le système linéaire obtenu.
On pourra au choix recourir aux librairies Csparse et GSL pour la gestion des matrices et vecteurs. On fera usage de l'utilitaire Gnuplot pour la représentation graphique des solutions. Consulter les notes sur le présentation de ces utilitaires. Des exemples qui y figurent peuvent vous servir de guide pour la mise en oeuvre.

On souhaite donc résoudre numérique par différences finies l'équation de Laplace sur un carré, avec les conditions aux limites de Dirichlet sur le bord. Ce problème s'écrit :

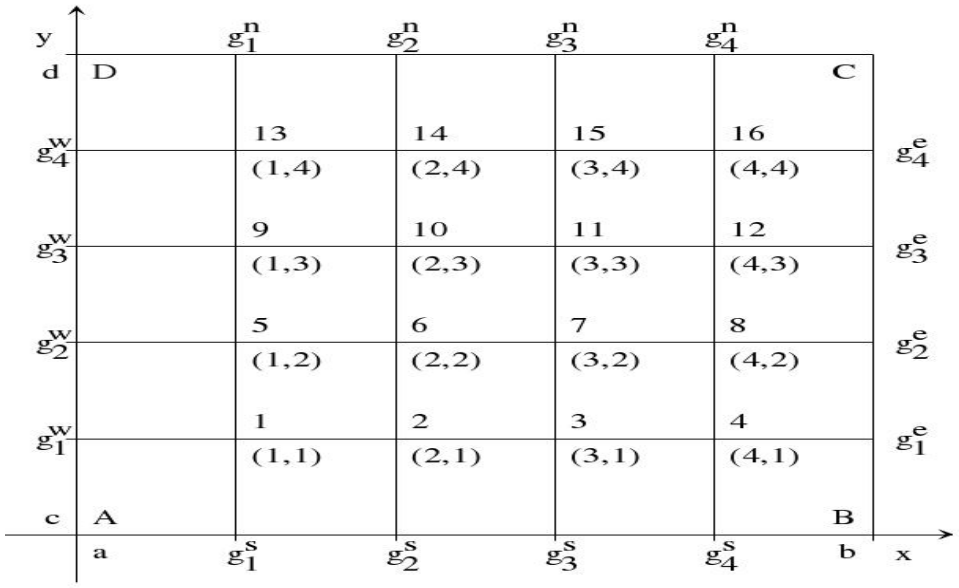
$$\left\{ \begin{array}{rcll} -\Delta u & = & f, & x \in \Omega \\ u & = & g, & x \in \Gamma := \partial\Omega \end{array} \right. \tag{1}$$

avec $\Omega =]a, b[\times]c, d[\in \mathbb{R}^2$ et f et g données. Pour des raisons de simplicité, on peut supposer $d - c = b - a$ et on effectue la discrétisation suivante de l'intérieur du carré : on fixe $N \geq 1$ et on considère le pas de discrétisation $h = (b-a)/(N+1)$ à la fois selon la direction x et y. On pose alors $x_j = a + jh$ $(1 \leq j \leq N)$ et $y_k = c + kh$ $(1 \leq k \leq N)$ et on cherchera à calculer les $N \times N$ valeurs de la solution du problème (1) en les points (x_j, y_k) $(1 \leq j, k \leq N)$ ainsi définis (voir figure).

Q-10 : On se place au point $M(x_j, y_k)$ du maillage $(1 \leq j, k \leq N)$. A l'aide de la formule de Taylor appliquée selon les directions x et y, montrer que pour toute fonction $u \in C^2(\overline{\Omega})$, on a, en notant $u_j, k = u(x_j, y_k)$:

$$\Delta u(x_j, y_k) = \frac{1}{h^2}(u_{j+1,k} - 2u_{j,k} + u_{j-1,k}) + \frac{1}{h^2}(u_{j,k+1} - 2u_{j,k} + u_{j,k-1}) + \mathcal{O}(h^2). \tag{2}$$

Q-11 : On souhaite à présent résoudre numériquement (1). Pour cela, on va écrire que l'équation de Laplace est vérifiée en tous les points et on va négliger le terme de Taylor d'ordre 2 dans (2). On se ramène alors comme d'habitude à un système linéaire à résoudre. On convient donc de repérer le point considéré par un seul indice en espace en utilisant la convention : $M_1(x_1, y_1), M_2(x_2, y_1), \ldots, M_N(x_N, y_1), M_{N+1}(x_1, y_2), \ldots, M_{N^2}(x_N, y_N)$ (voir figure où l'on a choisi $N = 4$). On note alors v_i la valeur approchée de la solution au point $M_i (i \leq i \leq N^2)$.



Q-11-1 : Ecrire l'équation vérifiée pour un point ne bordant pas la frontière de Ω, en ne faisant intervenir que l'indice i.

Q-11-2 : Pour quelles valeurs de i le point M_i correspondant borde-t-il la frontière ? Ecrire soigneusement les équations obtenues pour ces valeurs en tenant compte des contributions du bord. On définira les valeurs de g en les points des bords par $(g_1^s, \ldots, g_N^s)^T$ sur AB, $(g_1^n, \ldots, g_N^n)^T$ sur DC, $(g_1^e, \ldots, g_N^e)^T$ sur BC, $(g_1^w, \ldots, g_N^w)^T$ sur AD.

Q-12 : En déduire l'expression du système approché sous la forme d'un système matriciel de la forme $Ax = b$, avec $A \in \mathcal{M}_{N^2}(\mathbb{R})$, $\quad x = (v_1, \ldots, v_N^2)^T \in \mathbb{R}^{N^2}$ et $b \in \mathbb{R}^{N^2}$. Ecrire alors un programme qui calcule la solution approchée de (1) par différences finies.

Q-13 : Utiliser le programme construit pour effectuer la résolution numérique de (1) avec les paramètres suivants : $a = c = 0, b = d = \pi$, $f(x, y) = \frac{4}{5} \sin\left(\frac{x}{2}\right) \sin(y), g \equiv 0$ sur AB, CD et DA , $g = \sin(y)$ sur BC. On vérifiera que $u(x, y) = \sin\left(\frac{x}{2}\right) \sin(y)$ est la solution exacte de (1) et on utilisera les différentes possibilités de visualisation $3d$ pour représenter la solution approchée sur le domaine $\overline{\Omega}$.

Q-14 : Calculer la norme infinie de l'erreur commise sur la solution avec cette méthode, pour des valeurs de N de la forme $N = 2^j, j = 2, 3, 4$ etc. Proposer alors une méthode pour trouver numériquement l'entier p tel que l'on ait l'estimation $\max\limits_{i=1,\ldots,N^2} |u(M_i) - v_i| \leq Ch^p$.

Q-15 : Calculer la plus grande et la plus petite et la plus grande valeur propre de la matrice A ainsi que les vecteurs propres associées et représenter graphique ces vecteurs propres. On prendra les mêmes paramètres que dans la question précédente avec $N = 2^4$. (*Cette question vise simplement à se familiariser davantage aux utilitaires. Utiliser GSL.)*