`Un memento Scilab vous est fourni en fin d'énoncé.`

On cherche à déterminer une fonction $u(x,y)$ solution de l'équation aux dérivées partielles (équation de Laplace)

$$-\triangle u(x,y) = f(x,y) \quad \text{pour } (x,y) \in \Omega =]0,1[\times]0,1[\tag{1}$$

et vérifiant les conditions aux bords

$$u(x,y) = 0 \quad \text{pour } (x,y) \in \partial\Omega. \tag{2}$$

Soit $h = 1/(N+1)$ un pas d'espace que l'on choisit commun aux deux directions x et y. On définit les points de discrétisation (x_i, y_j) par

$$x_i = ih,\ i = 0 \dots N+1, \quad y_j = jh,\ j = 0 \dots N+1,$$

et on cherche à calculer une approximation de u aux points internes (x_i, y_j), $1 \le i, j \le N$.

Pour cela, on utilise le schéma aux différences finies suivant

$$\frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h^2} = f_{i,j}, \quad 1 \le i, j \le N \tag{3}$$

où $u_{i,j}$ désigne l'approximation recherchée de $u(x_i, y_j)$ et $f_{i,j} = f(x_i, y_j)$.

Soit $\bar{u}_j$ le vecteur dont les composantes sont les n inconnues de la ligne j, $\bar{u}_j = (u_{1,j}, u_{2,j}, \dots, u_{N,j})^t$ et $\bar{f}_j = (f_{1,j}, f_{2,j}, \dots, f_{N,j})^t$. En prenant en compte les conditions aux bords (2), le schéma (3) peut alors se réécrire

$$\frac{-\bar{u}_{j-1} + B\bar{u}_j - \bar{u}_{j+1}}{h^2} = \bar{f}_j$$

où

$$B = \begin{pmatrix} 4 & -1 & 0 & \dots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \dots & 0 & -1 & 4 \end{pmatrix}.$$

Finalement, en utilisant à nouveau les conditions aux bords, on peut réécrire le système complet sous la forme

$$A\bar{u} = \bar{f} \tag{4}$$

où l'inconnue est $\bar{u} = (\bar{u}_1, \dots, \bar{u}_N)^t$, le second membre $\bar{f} = (\bar{f}_1, \dots, \bar{f}_N)^t$ et la matrice A :

$$A = \begin{pmatrix} B & -I_N & 0 & \dots & 0 \\ -I_N & B & -I_N & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -I_N & B & -I_N \\ 0 & \dots & 0 & -I_N & B \end{pmatrix}.$$

1. Etude numérique du schéma

1. Ecrire une fonction `A=Laplace2d(n)` renvoyant la matrice A de taille $n^2 \times n^2$. Utiliser la fonction `Mataffiche` fournie pour visualiser la matrice ainsi construite.
   Indications : On ne demande pas à cette étape d'utiliser les commandes `sparse` de `Scilab`. On utilisera les commandes `eye` et `diag`.

2. Ecrire une fonction `b=Smlaplace2d(n,f)` renvoyant le second membre $\bar{f}$ de l'équation (4), étant donnés n et la fonction f.

3. Validation. On pose $f(x,y) = 2x(1-x) + 2y(1-y)$, de sorte que la solution du problème soit $u(x,y) = x(1-x)y(1-y)$. Pour $N = 10$, calculer la solution obtenue grâce à la commande `A\ b`. En utilisant la commande `matrix`, construire la matrice U avec $U(i,j) = u_{i,j}$ et représenter graphiquement les résultats à l'aide de la fonction `plot3d`. Comparer la solution obtenue avec la solution exacte en calculant par exemple la norme infinie de l'erreur.
   Indications : Pour construire la matrice U_{exacte} avec $U_{exacte}(i,j) = u(x_i, y_j)$, on pourra utiliser la méthode suivante :
   ```
   xx=x'*ones(1,n); //chaque colonne de xx = x
   yy=ones(n,1)*y; //chaque ligne de yy=y
   Uexacte=uexacte(xx,yy);
   ```
   où `uexacte(x,y)` est une fonction `scilab` que vous aurez programmée.

4. On choisit cette fois f de sorte que la solution u ne soit pas polynomiale : $u(x,y) = (x-1)(y-1)\sin(\pi x)\sin(\pi y)$ est solution de (1) pour $f(x,y) = 2\pi^2 u(x,y) - 2\pi[(y-1)\sin(\pi y)\cos(\pi x) + (x-1)\sin(\pi x)\cos(\pi y)]$. Quelle est la valeur maximale de N pour laquelle `Scilab` peut faire les calculs ?

5. La question précédente montre qu'il va falloir utiliser le caractère creux de la matrice A pour parvenir à résoudre précisément le problème. Utiliser la fonction `lu` de `Scilab` pour calculer la décomposition LU de A et visualiser L et U grâce à la fontion `Mataffiche`. Faire de même avec une matrice creuse aléatoire (utiliser `sprand`).

6. Convergence. Pour étudier la convergence du schéma, on va utiliser la commande `A\ b` pour résoudre le système avec une matrice A creuse. Ecrire une fonction `A=Laplace2dSparse(n)` renvoyant la matrice A de type `sparse` (voir memento). Pour chacune des valeurs $N = 5k$, $k = 1 \dots 10$, calculer l'erreur entre la solution numérique et la solution approchée. Tracer la courbe représentant pour de grandes valeurs de N le logarithme de l'erreur en fonction de logarithme de N. On considérera la norme infinie ponctuelle.

2. Résolution par méthodes de gradient

Rappel des algorithmes.

Algorithme du gradient à pas fixe :

$$\text{(GPF)} \left| \begin{array}{l} \text{On choisit } x^0 \text{ un vecteur initial et } \alpha > 0 \text{ un pas fixe} \\ \text{Iterer pour } n \geq 0,\ x^{n+1} = x^n + \alpha r^n \text{ avec } r^n = b - Ax^n \end{array} \right.$$

Algorithme du gradient à pas optimal :

$$\text{(GPO)} \left| \begin{array}{l} \text{On choisit } x^0 \text{ un vecteur initial} \\ \text{Iterer pour } n \geq 0, \\ \qquad x^{n+1} = x^n + \alpha^n r^n \text{ avec } \alpha^n = \dfrac{(r^n, r^n)}{(Ar^n, r^n)} \text{ et } r^n = b - Ax^n \end{array} \right.$$

Algorithme du gradient conjugué :

$$
\text{(GC)} \left| \begin{array}{l} \text{On choisit } x^0 \text{ un vecteur initial et on pose } r^0 = b - Ax^0,\ p^0 = r^0 \\ \text{Iterer pour } n \geq 0, \\ \quad x^{n+1} = x^n + \alpha^n p^n \text{ avec } \alpha^n = \dfrac{(r^n, r^n)}{(Ap^n, p^n)} \\ \quad r^{n+1} = r^n - \alpha^n A p^n \\ \quad p^{n+1} = r^{n+1} + \beta^n p^n \text{ avec } \beta^n = \dfrac{(r^{n+1}, r^{n+1})}{(r^n, r^n)} \end{array} \right.
$$

1. Ecrire les fonction `Scilab`
   `[x,iter]=GPF(A,b,tol,alpha,itermax,x0)`
   `[x,iter]=GPO(A,b,tol,itermax,x0)`
   `[x,iter]=GC(A,b,tol,itermax,x0)`
   qui renvoient `x` la solution calculée et `iter` le nombre d'itérations effectuées pour chacun des algorithmes de gradient. Valider ces fonctions sur des matrices simples. On pourra vérifier le nombre d'itérations effectuées par (GC). Pour (GPF), tracer le nombre d'itérations en fonction de α et retrouver la valeur optimale théorique que doit prendre α pour minimiser le nombre d'itérations. Comparer les trois méthodes (choisir le meilleur α dans le cas du gradient à pas fixe) en traçant pour chacune d'elle le logarithme du résidu en fonction du nombre d'itérations.

2. Dans le cas d'une matrice de taille 2×2, tracer la suite des itérés x^n calculés pour chacune des méthodes. Que se passe-t-il quand le conditionnement de la matrice augmente ?

3. Utiliser (GC) pour résoudre le problème de Laplace. Vérifier le résultat obtenu.

3. Préconditionnement

1. Fabriquer une matrice diagonale D très mal conditionnée de taille 500×500. Choisir un second membre y et résoudre $Dx = y$ en utilisant (GC). Quel est le nombre d'itérations effectuées ?

2. On souhaite résoudre le système $\bar{D}x = y$ où $\bar{D}$ est une perturbation symétrique de D : $\bar{D} = D + \Delta$ avec Δ symétrique et creuse (on pourra utiliser la fonction `sprand` pour construire un tel Δ). Quel est le nombre d'itérations effectuées par (GC) pour résoudre le problème ? Pour accélérer la convergence, on souhaite utiliser l'algorithme du gradient conjugué préconditionné :

   Algorithme du gradient conjugué préconditionné:

$$
\text{(GCP)} \left| \begin{array}{l} \text{On choisit } x^0 \text{ un vecteur initial et on pose } r^0 = b - Ax^0,\ p^0 = M^{-1}r^0 \\ \text{Iterer pour } n \geq 0, \\ \quad x^{n+1} = x^n + \alpha^n p^n \text{ avec } \alpha^n = \dfrac{(M^{-1}r^n, r^n)}{(Ap^n, p^n)} \\ \quad r^{n+1} = r^n - \alpha^n A p^n \\ \quad p^{n+1} = M^{-1}r^{n+1} + \beta^n p^n \text{ avec } \beta^n = \dfrac{(M^{-1}r^{n+1}, r^{n+1})}{(M^{-1}r^n, r^n)} \end{array} \right.
$$

   Quel est le nombre d'itérations effectuées par (GCP) si on préconditionne le système par $M = D$?
   Indication : Lors de la programmation de (GCP), on prendra garde à ne calculer M^{-1} qu'une seule fois par itération.

Mémento Scilab

`Scilab` peut-être utilisé 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. Enfin, rappelons qu'une commande suivie d'un point virgule désactive sa sortie texte.

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
`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 $\begin{pmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{pmatrix}$.
`linspace(x1,x2,n)` - Création d'un vecteur de taille n de valeurs équiréparties entre $x1$ et $x2$.
`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.
`[p:q]` - Vecteur colonne contenant la liste des entiers de p à q.
`:` - 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 $(u_1, \ldots, u_{15})$.
`'` - Opérateur transposé. Exemple : `u=[1. 1.]; v=u';` définit v comme étant la transposée de u.
`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 zéros 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.
`M=matrix(V,m,n)` - Crée une matrice M de taille $m \times n$ à partir d'un vecteur V de taille mn en le découpant en n morceaux de taille m.

Commandes graphiques

`xinit()` - Création d'une nouvelle fenêtre graphique
`xdel()` - Suppression d'un fenêtre graphique
`xbasc()` - Rafraîchit la fenêtre graphique
`plotframe([xmin ymin xmax ymax])` - Création d'un cadre $[xmin, xmax] \times [ymin, ymax]$ dans la fenêtre graphique
`plot(x,y,'.');` - Trace les couples de points de coordonnées $(x(i), y(i))$ dans la fenêtre graphique.
On peut relier les points en remplaçant `'.'` par `'-'`.
On peut aussi tracer deux courbes simultanément par : `plot(x,[y, z],'-');` où `x`,`y` et `z` sont des vecteurs colonne
`xpause(1000);` - provoque une courte pause (ajuster l'entier en paramètre pour obtenir une pause plus ou moins longue).
`xtitle` - Définit le titre du graphique. Exemple: `xtitle("titre")`
`plot3d(x,y,M,36,89,' ',[0,2,4]);` - Trace la surface définie par $S(x_i, y_j) = M_{i,j}$. Voir l'aide de `Scilab` pour le détail des options.