Matmeca 1ère année - TP de Calcul Scientifique en Fortran
année 2019-2020

TP 2 : résolution de l'équation de Poisson

Consignes pour l'évaluation du TP :

- à l'issue de cette séance de TP, vous devrez rédiger un rapport ;
- ce rapport ainsi que les programmes réalisés devront être déposés sur moodle au plus tard le 9 mars à 8h ;
- aucun remise de fichier (rapport ou programme) ne sera possible après cette limite de temps, et la note de 0 sera attribuée s'il manque rapport ou programme ;
- le format du rapport imposé est le format pdf ;
- votre rapport devra comporter :
  - des graphiques commentés (solution numérique, solution exacte, courbes de convergence) ;
  - les réponses aux questions posées ;
  - toute autre information ou commentaire que vous jugerez pertinents
- en ce qui concerne les programmes : l'enseignant qui évaluera votre travail devra pouvoir compiler et exécuter vos programmes pour vérifier qu'ils fonctionnent correctement. Par conséquent :
  - tous les fichiers nécessaires à la compilation de vos programmes devront être déposés sur moodle (fichier fortran du programme principal, fichiers fortran des modules, makefile) ;
  - ne déposez pas de fichier exécutable, fichier .mod, fichiers de résultats, etc.

Aide gnuplot pour générer une image à inclure dans le rapport : pour générer un fichier image (par exemple le fichier `toto.png`, de format png) à partir d'un graphique gnuplot, il suffit de lancer, dans gnuplot, les commandes suivantes :

```
set term png
set output "toto.png"
replot
set term x11
```

1 Conditions aux limites de Dirichlet

Le but de cette partie est de programmer le schéma vu en cours pour résoudre l'équation de Poisson

$$\begin{aligned} -u''(x) &= f(x), \qquad x \in [0,1], \\ u(0) &= u(1) = \alpha, \end{aligned}$$

où $f(x) = \sin(20\pi x)$ et α est une constante donnée. La solution exacte de l'équation est $u_{ex}(x) = \frac{\sin(20\pi x)}{(20\pi)^2} + \alpha$.

Tout ce qui suit doit être programmé dans le programme `poisson`. Il est impératif d'utiliser l'option de compilation `-fcheck=all` qui permettra de détecter les erreurs de tableaux à l'exécution du programme. Cette option ralentit cependant l'exécution, et doit être désactivée quand le programme fonctionne correctement.

Question 1. Maillage, schéma, système, résolution.

L'intervalle $[0,1]$ est découpé en $n+1$ intervalles, et les noeuds du maillage sont notés x_i pour $i = 0$ à $n+1$. En utilisant la formule de différences finies centrées d'ordre 2, le schéma s'écrit

$$-\frac{1}{\Delta x^2}(u_{i+1} - 2u_i + u_{i-1}) = f(x_i),$$

pour $i = 1$ à n, avec les données au bord $u_0 = \alpha$ et $u_{n+1} = \alpha$. Ce schéma s'écrit sous forme matricielle $Au = b$, avec

$$A = -\frac{1}{\Delta x^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{pmatrix} \qquad u = \begin{pmatrix} u_1 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{pmatrix} \qquad b = \begin{pmatrix} f(x_1) + \frac{\alpha}{\Delta x^2} \\ \vdots \\ f(x_i) \\ \vdots \\ f(x_{imax}) + \frac{\alpha}{\Delta x^2} \end{pmatrix}.$$

Programmez la résolution de ce système par la méthode de Cholesky : vous pouvez ré-utiliser le module `mod_algebre` du semestre précédent, avec en particulier les fonctions `reschol` et `chol`. Il est aussi conseillé de programmer le remplissage de la matrice A dans ce module.

Si besoin, vérifiez que votre solveur fonctionne correctement : affichez la matrice A pour $n = 4$, et faites afficher la différence relative entre A et LL^T. Enfin, pour $n = 40$ générez un second membre aléatoire b (avec la subroutine `random_number`), calculez u et vérifiez que la différence relative entre Au et b est bien de l'ordre de l'erreur machine.

Question 2. Calcul de la solution

On fixe maintenant les données au bord avec $\alpha = \frac{1}{(20\pi)^2}$. Pour chaque maillage correspondant à $n = 20$, 40, 80, 160, calculez la solution donnée par le schéma et stockez-la dans un fichier dont le nom est `sol_num_xxx.dat` où `xxx` est une chaîne de caractères correspondant à n. On rappelle l'astuce qui permet cela : déclarer la chaîne de caractères `nc` avec

```
character(len=20) :: nc
```

puis utiliser l'instruction

`write(nc,*) n`
et enfin
`open(unit=10,file='sol_num_'//trim(adjustl(nc))//'.dat')`

Pour la comparaison graphique, calculez la solution exacte avec 1000 points et stockez-la dans le fichier `sol_exacte.dat`.

Affichez ensuite avec gnuplot les différentes solutions et la solution exacte, et vérifiez que la solution numérique converge bien vers la solution exacte.

Gnuplot pouvant tracer n'importe quelle fonction donnée par une formule, on peut aussi directement tracer la solution exacte avec la commande
`replot (1+sin(2*pi*10*x))/(2*pi*10)**2`

Question 3. Calcul de l'erreur, courbe de convergence, ordre.

Pour chaque maillage, calculez l'erreur quadratique $E = (\sum_{i=1}^{n}(u(x_i) - u_i)^2 \Delta x)^{1/2}$. Pour cela, la solution exacte doit évidemment être calculée uniquement aux points du maillage. Faites afficher le pas Δx et l'erreur à l'écran.

Créez ensuite (à la main) le fichier `courbe_erreur.dat` dont chaque ligne contient le pas Δx et l'erreur E correspondante.

Affichez cette courbe, puis avec la technique vue dans le TP précédent (régression linéaire), vérifiez que l'ordre du schéma est bien 2.

2 Conditions aux limites Dirichlet/Neumann

On modifie la condition aux limites à gauche pour obtenir le problème suivant :

$$\begin{aligned} -u''(x) &= f(x), \qquad x \in [0,1], \\ u'(0) &= 0, \qquad u(1) = 0, \end{aligned}$$

où $f(x) = e^x$. La solution exacte de l'équation est $u_{ex}(x) = x - 1 + e - e^x$.

Ce problème peut être discrétisé à l'aide du schéma précédent, mais il nécessite en plus une discrétisation de la condition de Neumann $u'(0) = 0$. Dans cette partie, nous étudions donc deux discrétisation différentes : une d'ordre 1, et l'autre d'ordre 2.

Question 4. Approximation par une formule d'ordre 1.

Comme $u(0)$ n'est plus une donnée du problème, il convient à présent de rajouter u_0 au vecteur des inconnues. Il faut aussi se donner une équation supplémentaire. Celle-ci est obtenue à partir de la conditions de Neumann $u'(0) = 0$, en utilisant la formule décentrée à droite

$$u'(0) = \frac{u(\Delta x) - u(0)}{\Delta x} + O(\Delta x),$$

qui donne la relation $\frac{u_1 - u_0}{\Delta x} = 0$. Pour garder une matrice symétrique, cette relation est

divisée par $-\Delta x$ de sorte que le schéma s'écrive :

$$
\begin{aligned}
&-\frac{1}{\Delta x^2}(u_1 - u_0) = 0 \\
&-\frac{1}{\Delta x^2}(u_{i+1} - 2u_i + u_{i-1}) = f(x_i), \qquad i = 1 \text{ à } i_{max} \\
&-\frac{1}{\Delta x^2}(-2u_{i_{max}} + u_{i_{max}-1}) = f(x_{i_{max}}).
\end{aligned}
$$

Ce schéma s'écrit sous forme matricielle $Au = b$, avec

$$
A = -\frac{1}{\Delta x^2}\begin{pmatrix}
-1 & 1 & & & & \\
1 & -2 & 1 & & & \\
 & 1 & -2 & 1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & \ddots & \ddots & 1 \\
 & & & & 1 & -2
\end{pmatrix}
\qquad
u = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{pmatrix}
\qquad
b = \begin{pmatrix} 0 \\ f(x_1) + \frac{\alpha}{\Delta x^2} \\ \vdots \\ f(x_i) \\ \vdots \\ f(x_{imax}) + \frac{\alpha}{\Delta x^2} \end{pmatrix}.
$$

Cette matrice est donc bien symétrique, et on peut montrer qu'elle est définie positive.
À présent, effectuez les tâches suivantes :

1. modifiez votre programme pour qu'il résolve le nouveau problème à l'aide du schéma ci-dessus ; pour cela, deux approches sont possibles :
   — soit vous souhaitez conserver le fait que tous vos tableaux ont des indices qui commencent à 1, et vous devez alors changer la numérotation utilisée dans le schéma ci-dessus ;
   — soit vous voulez garder la numérotation du schéma. Dans ce cas, il faut allouer vos tableaux dans le programme principal avec des indices commençant à 0. Si votre subroutine de solveur linéaire est programmée avec des tableaux à *profil implicite*, vous n'avez pas besoin de la modifier : le solveur est sans doute programmé avec un premier indice égal à 1, mais le complilateur va automatiquement décaler les indices du tableau d'entrée, même si ses indices commencent à 0.
2. superposez sur un graphique la solution exacte et la solution numérique obtenue pour quelques maillages différents ($i_{max} = 10$ à 80), puis commentez ces résultats ;
3. tracez ensuite la courbe d'erreur et donnez l'ordre du convergence du schéma que vous observez expérimentalement.

Attention : le programme doit encore fonctionner pour les problèmes de la première partie. Le cas échéant, travaillez sur des copies des subroutines à modifier.

Question 5. Approximation par une formule d'ordre 2.

Remarquons que la formule différence finie précédente est basée sur le développement limité de $u(\Delta x)$ autour de 0 à l'ordre un :

$$u(\Delta x) = u(0) + u'(0)\Delta x + O(\Delta x^2).$$

Pour améliorer la précision du schéma, on pousse alors ce développement limité à l'ordre deux :

$$u(\Delta x) = u(0) + u'(0)\Delta x + u''(0)\frac{\Delta x^2}{2} + O(\Delta x^3).$$

En utilisant la condition de Neumann $u'(0) = 0$ et l'équation de Poisson évaluée en $x = 0$, ce développement s'écrit

$$u(\Delta x) = u(0) - f(0)\frac{\Delta x^2}{2} + O(\Delta x^3).$$

Cela donne la relation

$$u_1 = u_0 - f(0)\frac{\Delta x^2}{2},$$

que l'on peut ré-écrire sous la forme

$$-\frac{1}{\Delta x^2}(u_1 - u_0) = f(0)/2.$$

La matrice du sysème est donc inchangée, et seule la première composante du second membre est modifiée : on remplace $b_0 = 0$ par $b_0 = f(0)/2$.

Programmez ce schéma à la suite du précédent dans votre programme, et effectuez le même travail (courbes, commentaires, courbe d'erreur, ordre de convergence), et concluez sur l'intérêt de ce nouveau schéma.

3 Questions hors barème

Ces questions rapporteront des points en plus. Il est vivement conseillé d'essayer de les traiter, mais cela n'est pas obligatoire.

Question 6. Un solveur creux.

Le but est ici d'économiser du temps calcul et de la place mémoire en utilisant un solveur de Cholesky adapté au creux de la matrice A. On peut démontrer que la matrice L a la même structure tridiagonale que A : les éléments extradiagonaux nuls de A restent nuls après factorisation. Il est alors possible d'économiser du temps calcul en ne faisant pas les opérations dont on sait à l'avance qu'elles donneront 0, et on peut économiser de la place mémoire en ne stockant que les éléments tridiagonaux. Cela se démontre facilement par simple identification.

En adoptant une numérotation locale de 1 à n, on note alors d_i ($i = 1$ à n) les coefficients diagonaux de L et l_i ($i = 1$ à $n-1$) les coefficients de sa première sous-diagonale. On note aussi $\gamma = 1/\Delta x^2$, $\beta = 2/\Delta x^2$ et $\alpha = -1/\Delta x^2$, qui permettent de définir les coefficients de A. L'algorithme de factorisation s'écrit alors :

```
d(1) = sqrt(gamma)
pour i=1 à n-1 faire
   l(i) = alpha / d(i)
   d(i+1) = sqrt(beta - l(i)^2)
fin pour
```

Les phases de descente et de remontée s'écrivent :

```
% descente
x(1) = b(1) / d(1)
pour i=2 à n
   x(i) = ( b(i) - l(i-1) * x(i-1) ) / d(i)
fin pour

% remontee
x(n) = x(n) / d(n)
pour i=n-1 à 1 par pas de -1 faire
   x(i) = ( x(i) - l(i) * x(i+1) ) / d(i)
fin pour
```

Programmez ce solveur de Cholesky dans votre module, puis utilisez-le dans votre programme, et constatez la forte accélération du temps calcul, en prenant par exemple $n = 1000$.

Question 7. Calcul de l'ordre sans solution exacte.

Dans les problèmes réels, on ne dispose évidemment pas de la solution exacte. Comment alors vérifier numériquement l'ordre d'un schéma ? Il existe une astuce basée sur l'utilisation simultanée de deux solutions numériques : c'est le procédé d'extrapolation de Richardson, utilisé pour accélérer la convergence d'une série d'approximations (c'est par exemple la base de la méthode de Romberg en intégration numérique).

Dans notre cas, son utilisation est la suivante. On calcule d'abord la solution u^G pour un maillage de pas Δx, appelé maillage *grossier*, dont les points sont nommés x_i^G. Si le schéma est d'ordre p, et que Δx est assez petit, on peut supposer que l'erreur de convergence s'écrit

$$u(x_i^G) - u_i^G = C(x_i^G)\Delta x^p, \tag{1}$$

où la fonction C est inconnue, et supposée ne dépendre que de x (et pas du maillage).

On calcule ensuite la solution u^F sur un maillage deux fois plus fin, appelé maillage *fin*, donc de pas $\Delta x/2$ dont les points sont nommés x_j^F. Il faut que ce nouveau maillage contienne tous les points de l'ancien maillage. Il suffit pour cela de rajouter un point au milieu de chaque intervalle de l'ancien maillage. L'erreur correspondante s'écrit alors

$$u(x_j^F) - u_j^F = C(x_j^F)\left(\frac{\Delta x}{2}\right)^p = C(x_j^F)\frac{\Delta x^p}{2^p}. \tag{2}$$

On fait maintenant la différence entre les deux erreurs, ce qui va nous permettre de calculer l'erreur et l'ordre de convergence. Attention, il faut pour cela n'utiliser que les points en commun entre les deux maillages, c'est-à-dire ceux du maillage grossier. Supposons donc que x_i^G soit un point du maillage grossier : il est facile de voir que dans le maillage fin, ce point est x_{2i}^F. En faisant la différence entre (1) et (2), on a pour le membre de gauche :

$$(u(x_{2i}^F) - u_{2i}^F) - (u(x_i^G) - u_i^G) = u_i^G - u_{2i}^F,$$

et pour le membre de droite :

$$C(x_{2i}^F)\frac{\Delta x^p}{2^p} - C(x_i^G)\Delta x^p = C(x_i^G)\Delta x^p(2^{-p} - 1).$$

Finalement, on a obtenu la relation

$$u_{2i}^F - u_i^G = (1 - 2^{-p})C(x_i^G)\Delta x^p = A_i \Delta x^p.$$

La quantité de gauche est connue : c'est la différence entre les solutions numériques calculées, aux mêmes points, sur un maillage grossier et un maillage deux fois plus fin. Elle permet de calculer l'exposant p du membre de droite, qui est l'ordre du schéma.

En pratique, on procédera donc de la façon suivante :

1. calculez les solutions numériques pour les maillages avec 20, 40, 80, 160, et 320 points. Attention : chaque maillage doit avoir 2 fois plus de points que le précédent (et non pas un n deux fois plus grand) ;
2. calculez l'erreur quadratique $E = \sqrt{\sum_{i=1}^{n_G}(u_{2i}^F - u_i^G)^2 \Delta x}$ et écrire dans un fichier le couple $\Delta x, E$ (attention : Δx est le pas du maillage grossier) ;
3. affichez la courbe, et calculez l'ordre par régression linéaire, comme dans les questions précédentes.

Si vous avez programmé le tout correctement, vous devriez obtenir à nouveau un ordre à peu près égal à 2.