TP de Calcul Numérique

Nicolas BOUTON

2020

1 Exercice 1

Developpement limité :

$$T(x_i + h) = T(x_i) + h\left(\frac{\delta T}{\delta x}\right)_i + h^2\left(\frac{\delta^2 T}{\delta x^2}\right)_i + O(h^2)$$
$$T(x_i - h) = T(x_i) - h\left(\frac{\delta T}{\delta x}\right)_i + h^2\left(\frac{\delta^2 T}{\delta x^2}\right)_i + O(h^2)$$

On somme et on inverse le signe :

$$-T(x_i + h) + 2T(x_i) - T(x_i - h) = -h^2\left(\frac{\delta^2 T}{\delta x^2}\right)_i + O(h^2)$$
$$\frac{-T(x_i + h) + 2T(x_i) - T(x_i - h)}{h^2} = -\left(\frac{\delta^2 T}{\delta x^2}\right)_i$$

Or on a :

$$-k\left(\frac{\delta^2 T}{\delta x^2}\right)_i = g_i, k > 0$$

On se permet de négligé k car c'est une constante dans nos prochain calcul :

$$-T(x_i + h) + 2T(x_i) - T(x_i - h) = h^2 g_i$$

On écrit le système d'équation :

$$\begin{array}{ll} u_0 = T_0 & i = 0 \\ -u_0 + 2u_1 - u_2 = h^2 g_1 & i = 1 \\ ... & ... \\ -u_{k-1} + 2u_k - u_{k+1} = h^2 g_k & i = k \\ ... & ... \\ -u_{n-1} + 2u_n - u_{n+1} = h^2 g_n & i = n \\ u_n = T_n & i = n+1 \end{array}$$

Avec les conditions aux bords on obtient :

$$\begin{array}{l} 2u_1 - u_2 = h^2 g_1 + T_0 \\ -u_{n-1} + 2u_n = h^2 g_n + T_n \end{array}$$

Donc on explicite le système linéaire $Au = g$:

$$A = \begin{bmatrix} 2 & -1 & 0 & - & - & - & 0 \\ -1 & 2 & -1 & . & & & | \\ 0 & -1 & . & . & . & & | \\ | & . & . & . & . & . & | \\ | & & . & . & . & -1 & 0 \\ | & & & . & -1 & 2 & -1 \\ 0 & - & - & - & 0 & -1 & 2 \end{bmatrix}$$

$$u = \begin{bmatrix} T_1 \\ | \\ T_n \end{bmatrix}$$

$$g = \begin{bmatrix} h^2 T_1 + T_0 \\ h^2 T_2 \\ | \\ h^2 T_{n-1} \\ h^2 T_n + T_1 \end{bmatrix}$$

Comme il n'y a pas de source de chaleur, on a $\forall i \in [1, n] : h^2 g_i = 0$

D'où $g = \begin{bmatrix} T_0 \\ 0 \\ | \\ 0 \\ T_1 \end{bmatrix}$

Et la solution analytique qui se déguage est :

$$T(x) = T_0 + x(T_1 - T_0)$$

2 Exercice 2

2.1 Arch

2.1.1 Bibliothèque

Pour l'intallation des bibliothèque **cblas** et **lapacke** :
$ sudo pacman -S cblas lapacke

2.1.2 Makefile

Il faut modifier la ligne qui link les librairie en linkant la bibliothèque **cblas**:
#
# – librairies
LIBS=-llapacke -lcblas -lm

3 Exercice 3

3.1 Question 1

Les matrices pour utiliser **BLAS** et **LAPACK** en **C** sont allouées et déclarées de la même manière que les tableaux en **C**. Mais elles doivent être stockées dans l'un des formats suivant :

- stockage conventionnel en 2 dimension (ex: int tab[10][10])
- stockage compact pour les matrices symétrique, hermitienne et triangulaire (stockage dans un tableau à 1 dimension des éléments de la matrice supérieur ou inférieur)
- stockage bandes pour les matrices à bandes (cad que les diagonales autour de la diagonale principale contiennent la plupart des NNZ) (GB et GE)
- utilisation de 2 ou 3 tableaux à 1 dimension pour stocker les matrices bidiagonale et tridiagonale respectivement

source : `http://performance.netlib.org/lapack/lug/node121`

3.2 Question 2

- Les constantes LAPACK_ROW_MAJOR et LAPACK_COL_MAJOR signifie la priorité ligne ou colonne respectivement de la représentation de la matrice.
- Effectivement, cet argument sert si on utilise un stockage par priorité ligne ou colonne car il faut préciser si on a utilisé une priorité ligne ou colonne pour stocker la matrice pour pouvoir faire les bons calculs.

3.3 Question 3

La **leading dimension** permet de savoir qu'elle élément correspond à la prochaine colonne ou la prochaine ligne suivant le stockage colonne ou ligne respectivement.

- Si on choisis un stockage priorité ligne, alors la **leading dimension** correspond au nombre d'élément d'une ligne pour pouvoir accéder à la ligne suivante.
- Si on choisis un stockage priorité colonne, alors la **leading dimension** correspond au nombre d'élément d'une colonne pour pouvoir accéder à la colonne suivante.

3.4 Question 4

3.4.1 Résumé

La fonction LAPACKE_dgbsv permet de calculer le résultat d'un système linéaire du type $A * X = B$, avec **X** l'inconnu, **A** une matrice et **B** le second membre, où **X** et **B** peuvent être des vecteurs ou des matrices.

3.4.2 Argument

Elle prend en argument la dimension de la matrice, le nombre du sous-diagonnale ainsi que de sur-diagonnale, la leading dimension de **A** et de **B**, le nombre de colonne de **B** ainsi que tableau d'entier pour stoker les indices de permutation.

3.4.3 Implémentation

Cette fonction implémente une décomposition **LU** à pivot partiel et la méthode de dessente et de remonté.

3.4.4 Note importante

Pour la factorisation **LU**, la fonction a besoin d'un vecteur de travail ou il stockera les pivots. Suivant le stockage choisis on rajoutera une ligne ou une colonne avant de stocker notre matrice car le vecteur doit apparaître en premier.

3.4.5 Sources

`http://www.math.utah.edu/software/lapack/lapack-d/dgbsv.html`

3.5 Question 5

A titre comparatif nous prendrons une matrice de taille 10 x 10.

3.5.1 Stockage priorité colonne

0.000000	0.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000
0.000000	−1.000000	2.000000	−1.000000

3.5.2 Stockage priorité ligne

0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000	2.000
−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	0.000

Pour que le tableau soit affichable nous avons dû enlever les 3 derniers 0 de la sortie, mais cela ne change pas le résultat.

3.5.3 Remarques

Nous pouvons apercevoir que nous avons le bon résultat, étant donné que la première ligne qui réservé à **BlAS** pour son vecteur de travail est con-

stitué de zéro ainsi que le premier élément de la diagonale supérieur et le dernier élément de le diagonale inférieur.
De plus pour vérifier le résultat il suffit de calculer la transposé de l'une des matrices et la comparé à la deuxième car elles doivent être égales. Ici la transposé d'une des 2 matrices est égales à la deuxième.

4 Exercice 5

4.1 Question 1

4.1.1 Equation

La fonction **cblas_dgbmv** permet de calculer l'équation suivante :

$$y = \alpha * A * x + \beta * y$$

Dans notre cas on a l'équation suivante :

$$A * u = b$$

Qu'on peut écrire avec les notations de l'équation juste au dessus :

$$A * y = x$$

Avec la question précédente on a calculer y. On peut donc tester la fonction sur l'équation suivante :

$$x = 1 * A * y + 0 * x$$

4.1.2 Argument

La fonction **cblas_dgbmv** prends à peut près les mêmes paramètres que **LAPACKE_dgbsv** plus les coefficient α et *beta*, le nombre de ligne et de colonne de la matrice en format classic ainsi qu'un paramètre qui indique si la matrice est une transposé ou non.
La fonction prend comme précondition que y et x soit des vecteurs.

4.1.3 Note importante

Contrairement à la fonction **LAPACKE_dgbsv**, la fonction **cblas_dgbmv** n'attend pas à ce qu'on lui laisse un vecteur au début de la matrice.

4.2 Question 2

Comme méthode de validation, nous proposons de calculé l'erreur relative suivante.
Etant donné que nous calculons l'équation suivante :

$$b = A * u$$

Nous allons calculer l'erreur relative suivante :

$$relres = \frac{\|b - A * u\|}{\|b\|}$$

Le résultat exacte de **B** est stocké dans le fichier **B.dat**.

4.2.1 Priorité Ligne

Le résultat est stocké dans le fichier **Y_row.dat**.
Cela ne marche pas encore.

```
---------- Poisson 1D ----------

 INFO DGBSV = 0

The relative residual error is relres = 1.764638e-16

 DGBSV :

The relative residual error is relres = 1.121773e+00

---------- End ----------
```

4.2.2 Priorité Colonne

Le résultat est stocké dans le fichier **Y_col.dat**.

```
---------- Poisson 1D ----------

 INFO DGBSV = 0
```

```
The relative residual error is relres = 1.764638e-16
```

DGBSV :

```
The relative residual error is relres = 5.102197e-16
```

———— End ————

4.2.3 Explication

L'erreur relative de la fonction **cblas_dgbmv** est un peut plus élevé que l'erreur relative de la fonction **LAPACKE_dgbsv** qui peut être expliqué par le fait que le bit de signe ne change pas et nous nous retrouvons avec des zéro négatif au lieu de positif mais l'erreur reste acceptable car il est de l'ordre de la précision machine.

5 Exercice 5

5.1 Question 1

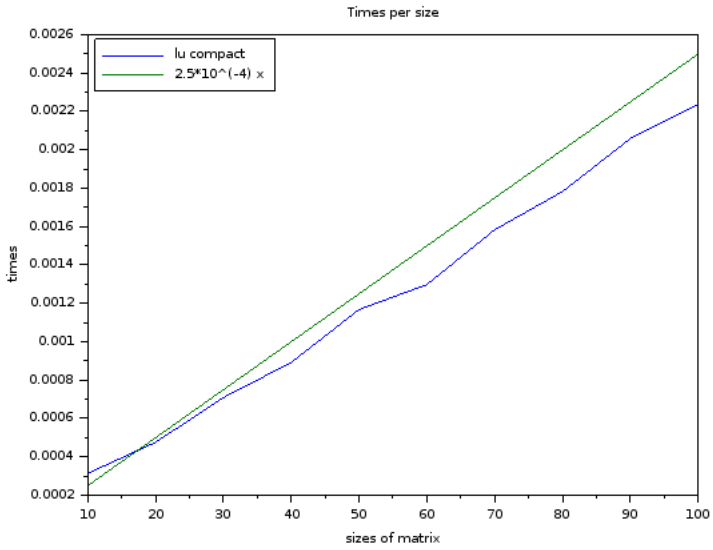
Le code scilab de la fonction se trouve dans le fichier **src/facto_lu.sci**.

5.2 Question 2

Le code scilab du script de test se trouve dans le fichier **src/script_facto_lu.sci**.

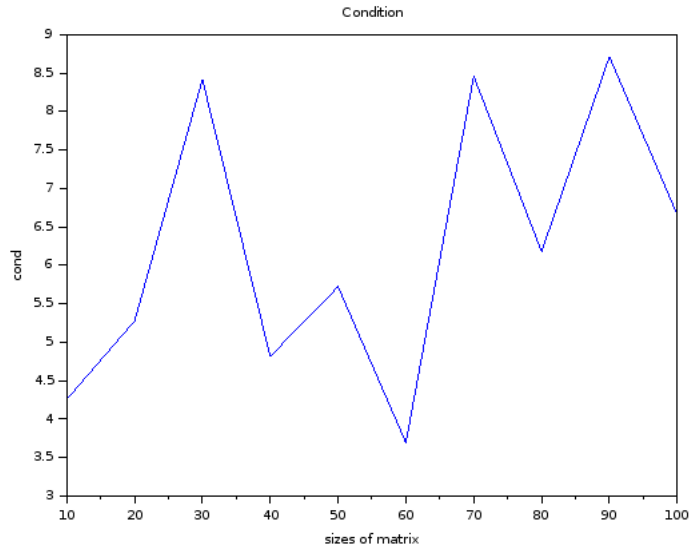
5.2.1 Graphiques

Temps :



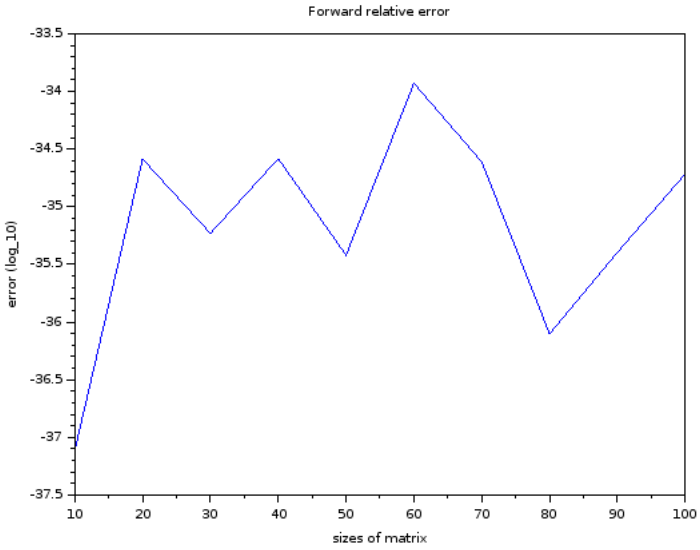
- Nous pouvons apercevoir que le temps est plutôt linéaire.

<u>Conditionnement :</u>



Le conditionnement reste bon pour des tailles de matrice entre 10 et 100 étant donné qu'il ne dépasse pas 10^{10}, ce qui rendrait l'erreur pour un

double une erreur pour un float.
Erreur relative avant :



L'erreur relative arrière est respectable étant donné qu'elle est de l'ordre de 10^{-34} pour des matrices de taille **100 x 100** ce qui donne des bornes pour l'erreur vers 10^{-24} (étant donné que le conditionnement est de $\approx 10^{10}$). Et cette borne est également respectable comparé à la précision machine de 10^{-16}.

5.2.2 Conclusion

Donc nous pouvons dire que l'algorithme et son implémentation est validé.

6 Exercice 6

6.1 Question 1

6.2 Question 2

6.3 Question 3

6.4 Question 4

6.5 Question 5

7 Annexe

Dépôt github : `https://github.com/Sholde/CN/tree/master/partie_2/poisson`