Projet résolution de grands systèmes linéaires creux

Grands systèmes linéaires creux : projet d'étude du Laplacien 2D

Auteurs :
CHARRUEY Adrien
POURTIER Grégoire

Superviseurs :
NOUSSAIR Ahmed
MICHEL Sixtine

Collège Sciences et technologies

19 Décembre 2019

1 Le problème approché

1. On a $\Omega =]0,a[\times]0,b[$.
On définit un maillage $\Omega_{h,k}$ de Ω : on introduit h = $\frac{a}{n}$ (avec n$\in \mathbb{N}^*$) le pas de discrétisation de $]0,a[$, et k = $\frac{b}{m}$ (avec m$\in \mathbb{N}^*$) le pas de discrétisation de $]0,b[$.

$$\Omega_{h,k} = \big\{(x_i,y_j) 0 \leq i \leq n \ et \ 0 \leq j \leq m\big\}$$

avec $\forall (i,j) \in [\![0,n]\!] \times [\![0,m]\!]$, et x_i=0+ih et y_j=0+jk .

2. On remplace maintenant l'opérateur Δ par sa discrétisation par différences finies aux points (x_i,y_j). — Taylor

On a $\Delta u(x,y) = \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y)$.

$u(x+h,y) = u(x,y) + \frac{\partial u}{\partial x}(x,y)h + \frac{1}{2}[\frac{\partial^2 u}{\partial x^2}(x,y)h^2] + o(h^2)$
$u(x-h,y) = u(x,y) - \frac{\partial u}{\partial x}(x,y)h + \frac{1}{2}[\frac{\partial^2 u}{\partial x^2}(x,y)h^2] + o(h^2)$
$u(x,y+k) = u(x,y) + \frac{\partial u}{\partial y}(x,y)k + \frac{1}{2}[\frac{\partial^2 u}{\partial y^2}(x,y)k^2] + o(k^2)$
$u(x,y-k) = u(x,y) - \frac{\partial u}{\partial y}(x,y)k + \frac{1}{2}[\frac{\partial^2 u}{\partial y^2}(x,y)k^2] + o(k^2)$

Donc on obtient :

$\frac{\partial^2 u}{\partial x^2}(x,y) \simeq \frac{u(x+h,y)-2u(x,y)+u(x-h,y)}{h^2}$ et $\frac{\partial^2 u}{\partial y^2}(x,y) \simeq \frac{u(x,y+k)-2u(x,y)+u(x,y-k)}{k^2}$

On pose $u_{ij} = u(x_i,y_j)$
$\frac{\partial^2 u}{\partial x^2}(x_i,y_j) \simeq \frac{u_{i+1j}-2u_{ij}+u_{i-1}j}{h^2}$ et $\frac{\partial^2 u}{\partial y^2}(x_i,y_j) \simeq \frac{u_{ij+1}-2u_{ij}+u_ij_{-1}}{k^2}$

Donc $\forall (i,j) \in [\![1,n-1]\!] \times [\![1,m-1]\!]$,

$\Delta u(x_i,y_j) = \frac{u_{i+1j}-2u_{ij}+u_{i-1}j}{h^2} + \frac{u_{ij+1}-2u_{ij}+u_ij_{-1}}{k^2}$

3. L'équation (1) de l'énoncé donne $\forall (i,j) \in [\![1,n-1]\!] \times [\![1,m-1]\!]$,
$-\Delta u(x_i,y_j) + \alpha u_{ij} = f(x_i,y_j) \qquad$ avec $\alpha \geq 0$
$\iff -\frac{u_{i+1j}}{h^2} - \frac{u_{ij+1}}{k^2} + (2\frac{h^2+k^2}{h^2k^2} + \alpha)u_{ij} - \frac{u_{i-1j}}{h^2} - \frac{u_{ij-1}}{k^2} = f(x_i,y_j)$
avec $\forall (i,j) \in [\![1,n-1]\!] \times [\![1,m-1]\!]$,
$f(x_i,y_j) = (x_i^2 + y_j^2 + \alpha)\sin(x_iy_j) = (i^2h^2 + j^2k^2 + \alpha)\sin(ihjk)$

Conditions aux bords :
$\forall i \in [\![0,n]\!]$, $si\ j = 0 \quad (y_0 = 0) \qquad u_{i0} = g(x_i,0)$

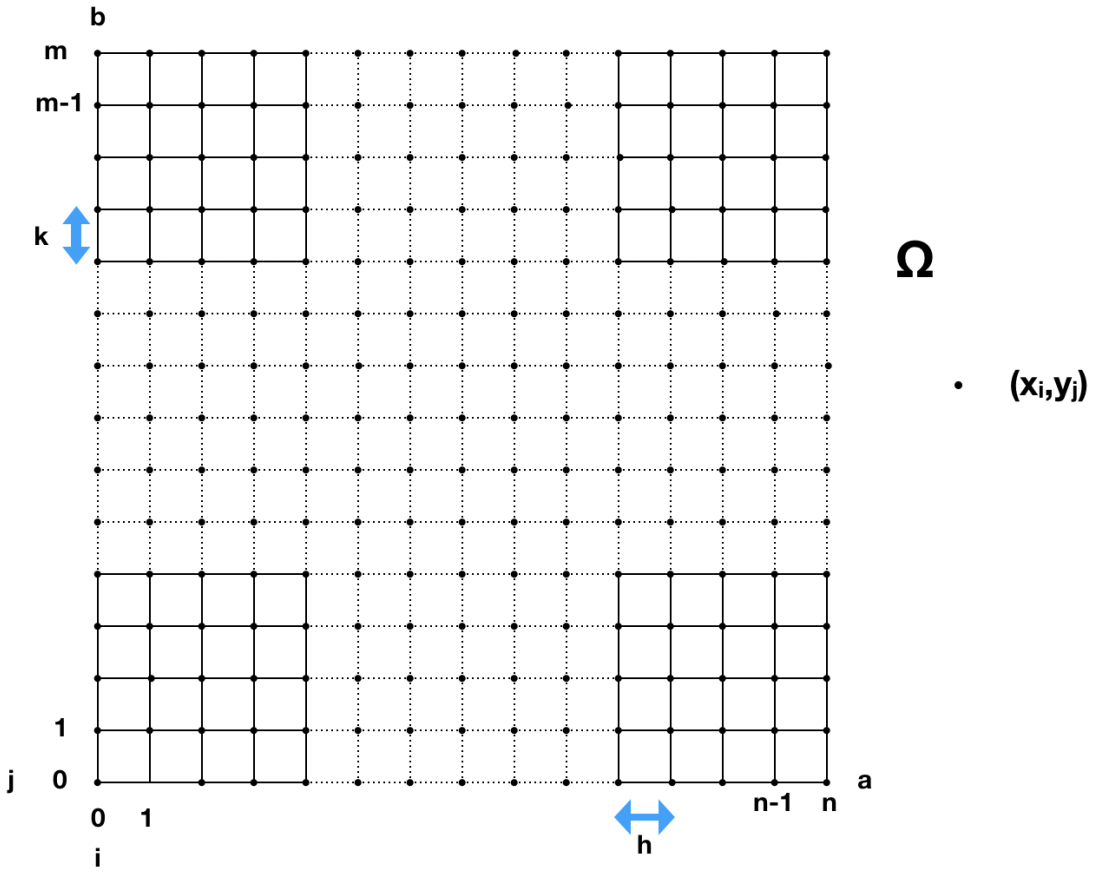
$$\begin{aligned} si\ j = m \quad (y_m = b) \quad & u_{im} = g(x_i, b) \\ \forall j \in [\![0, m]\!],\ si\ i = 0 \quad (x_0 = 0) \quad & u_{0j} = g(0, y_j) \\ si\ i = n \quad (x_n = a) \quad & u_{nj} = g(a, y_j) \\ avec\ \forall x, y \in \Omega, g(x, y) = \sin(xy) & \end{aligned}$$

Si on considère les équations pour
$\forall j \in [\![1, m-1]\!]$ et i = 1 : on a dans le membre de gauche un
$u_{i-1j} = u_{0j} = g(0, y_j)$, on le fait passer dans le second membre.
On fait donc passer dans le second membre $\frac{g(0,jk)}{h^2}$.

De même :
$\forall j \in [\![1, m-1]\!]$ et i = $n-1$: on fait passer dans le second membre $\frac{g(a,jk)}{h^2}$.
$\forall i \in [\![1, n-1]\!]$ et j = 1 : on fait passer dans le second membre $\frac{g(ih,0)}{k^2}$.
$\forall i \in [\![1, n-1]\!]$ et j = $m-1$: on fait passer dans le second membre $\frac{g(ih,b)}{k^2}$.

Dans les autres cas, c'est-à-dire $\forall (i, j) \in [\![2, n-2]\!] \times [\![2, m-2]\!]$, on a :
$-\frac{u_{i+1j}}{h^2} - \frac{u_{ij+1}}{k^2} + (2\frac{h^2+k^2}{h^2k^2} + \alpha)u_{ij} - \frac{u_{i-1j}}{h^2} - \frac{u_{ij-1}}{k^2} = f(x_i, y_j)$

4.



On numérote les x_i de gauche à droite, comme étant chaque point de la discrétisation suivant l'axe des x, avec i allant de 0 à n et les y_j de bas en haut, comme étant les points de la discrétisation suivant l'axe des y, avec j allant de 0 à m.
Le numéro global du point (x_i, y_j) est N(i,j) = i + (j-1)n avec $(i,j) \in [\![0, n]\!] \times [\![0, m]\!]$.

5. Le problème s'écrit matriciellement sous la forme AU=B, avec

$$\mathrm{U} = \begin{vmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{1m-1} \\ u_{21} \\ u_{21} \\ u_{22} \\ \vdots \\ u_{2m-1} \\ \vdots \\ u_{n-11} \\ \vdots \\ u_{n-1m-1} \end{vmatrix}, \qquad \text{et}$$

$$\mathrm{A} = \begin{pmatrix} \frac{2(h^2+k^2)}{h^2k^2}+\alpha & -\frac{1}{k^2} & 0 \cdots 0 & -\frac{1}{h^2} & 0 & 0 \cdots 0 \\ -\frac{1}{k^2} & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ 0 & -\frac{1}{k^2} & \ddots & \ddots & & \\ -\frac{1}{h^2} & \ddots & \ddots & & & \\ 0 & \ddots & & & & \\ \vdots & \ddots & & & & \\ 0 \cdots 0 & & & & & \end{pmatrix}, \qquad \text{et}$$

le second membre B est :

$$B = \begin{pmatrix} f(x_1,y_1) + \frac{g(0,h)}{h^2} + \frac{g(h,0)}{k^2} \\ f(x_1,y_2) + \frac{g(0,2k)}{h^2} \\ \vdots \\ f(x_1,y_{m-1}) + \frac{g(0,(m-1)k)}{h^2} + \frac{g(h,b)}{k^2} \\ f(x_2,y_1) + \frac{g(2h,0)}{k^2} \\ f(x_2,y_2) \\ \vdots \\ f(x_2,y_{m-1}) + \frac{g(2h,b)}{k^2} \\ \vdots \\ f(x_{n-1},y_1) + \frac{g(a,k)}{h^2} + \frac{g((n-1)h,0)}{k^2} \\ f(x_{n-1},y_2) + \frac{g(a,2k)}{h^2} \\ \vdots \\ f(x_{n-1},y_{m-1}) + \frac{g(a,(m-1)k)}{h^2} + \frac{g((n-1)h,b)}{k^2} \end{pmatrix}$$

6. On a $\forall i = 1, ..., (n-1)(m-1)$ $|a_{ii}| = |\frac{2(h^2+k^2)}{h^2k^2} + \alpha| = \frac{2(h^2+k^2)}{h^2k^2} + \alpha > 0$ ($\alpha \geq 0$).
Nous avons également $h > 0$ et $k > 0$ et :

$$\sum_{\substack{j=1 \\ j\neq i}}^{(n-1)(m-1)} |a_{ij}| \leq |-\frac{1}{h^2}| + |-\frac{1}{h^2}| + |-\frac{1}{k^2}| + |-\frac{1}{k^2}|$$

$$\leq \frac{2(h^2+k^2)}{h^2k^2} \leq \frac{2(h^2+k^2)}{h^2k^2} + \alpha$$

Donc $\forall$ i = 1,...,(n-1)(m-1) on a :

$$\sum_{\substack{j=1 \\ j\neq i}}^{(n-1)(m-1)} |a_{ij}| \leq |a_{ii}|$$

et pour i = 1, on a $|a_{11}| = \frac{2(h^2+k^2)}{h^2k^2} + \alpha > |-\frac{1}{k^2}| + |-\frac{1}{h^2}| = \sum_{j=2}^{(n-1)(m-1)} |a_{1j}|$

Donc A est à diagonale fortement dominante par construction de A on voit qu'aucune ligne n'est combinaison linéaire de l'autre donc la matrice A est bien irréductible.
Donc on en déduit que A est inversible.

A est une matrice bande A[m-1,m-1]. On peut donc utiliser un stockage bande pour stocker A.
A contient un grand nombre de 0, on pourrait donc aussi utiliser un stockage creux.
A est symétrique, on peut donc ne stocker que la partie supérieure ou inférieure de A.
A est nulle en dehors de ses 5 diagonales, on peut donc utiliser un stockage diagonal.
Enfin ses propriétés qui permettent de choisir les méthodes de résolution adaptées au problème sont que A est inversible, et que A est symétrique avec les $a_{ii} > 0$ pour tout i et à diagonale fortement dominante et irréductible, donc A est définie positive.

Les différentes méthodes directes que l'on pourrait mettre en oeuvre pour résoudre le problème sont :

— le pivot de Gauss ;
— la décomposition A = LU (sans permutations), car A est inversible ;
— la décomposition PA = LU, avec méthode du pivot partiel (on choisit le plus grand pivot en valeur absolue), ou en choisissant le premier pivot non-nul, car A est inversible ;

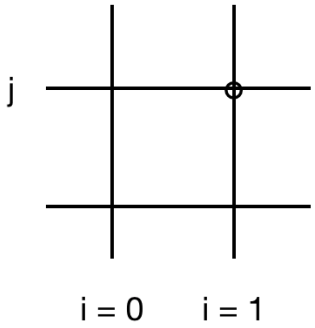
— la décomposition de Cholesky, A = SS^T, car A est symétrique et définie positive.

A est symétrique et définie positive, donc la méthode de relaxation sera convergente si $0 < \omega < 2$.
A est à diagonale fortement dominante et irréductible, alors la méthode de Jacobi convergera.

2 Autres aspects à éventuellement considérer

1. Problème initial $(P1) = \begin{cases} -\Delta u + \alpha u = f & \text{sur } \Omega \\ u = g & \text{sur } \partial\Omega \end{cases}$

   Autre problème (autres conditions aux bords) $(P2) = \begin{cases} -\Delta u + \alpha u = f & \text{sur } \Omega \\ u' = g & \text{sur } \partial\Omega \end{cases}$

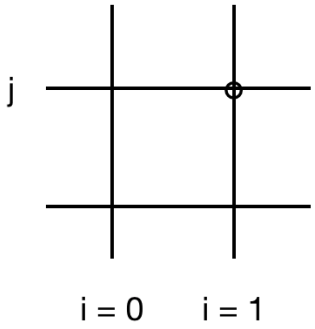


$u_{1j} = u(x_0 + h, y_j)$
$u_{1j} = u_{0j} + hu'_{0j} \Rightarrow u_{0j} = u_{1j} - hu'_{0j}$
$\underline{u_{0j} = u_{1j} - hg_{0j}}$

$\Delta u_{ij} = \frac{u_{i-1j} - 2u_{ij} + u_{i+1}j}{h^2} + \frac{u_{ij-1} - 2u_{ij} + u_ij_{+1}}{k^2}$

si j = 0, $y_0 = 0$ alors $\underline{u_{i0} = u_{i1} - kg_{i0}}$ (car $u_{i1} = u(x_i, y_0 + k) = u_{i0} + ku'_{i0} = u_{i0} + kg_{i0}$)
si j = m, $y_m =$ b alors $\underline{u_{im} = u_{im-1} - kg_{im}}$
si i = 0, $x_0 = 0$ alors $\underline{u_{0j} = u_{1j} - hg_{0j}}$
si i = n, $x_n =$ a alors $\underline{u_{nj} = u_{n-1j} - hg_{nj}}$

2. $(P3) = \begin{cases} -\Delta u + \alpha u = f & \text{sur } \Omega \\ u' = g + \lambda u & \text{sur } \partial\Omega \end{cases}$



$u_{1j} = u(x_0 + h, y_j) \Rightarrow u_{1j} = u_{0j} + hu'_{0j}$
$u_{1j} = u_{0j} + hg_{0j} + h\lambda u_{0j} \Rightarrow u_{1j} = u_{0j}(1 + h\lambda) + hg_{0j} \Rightarrow \underline{u_{0j} = \frac{1}{1+h\lambda}(u_{1j} - hg_{0j})}$

si j = 0, $y_0 = 0$ alors $u_{i1} = u(x_i, y_0 + k) = u_{i0} + ku'_{i0} = u_{i0} + k(g_{i0} + \lambda u_{i0})$)

d'où $\underline{u_{i0} = \frac{1}{1+k\lambda}(u_{i1} - kg_{i0})}$

si j = m, $y_m = b$ alors $\underline{u_{im} = \frac{1}{1-k\lambda}(u_{im-1} + kg_{im})}$

si i = 0, $x_0 = 0$ alors $\underline{u_{0j} = \frac{1}{1+h\lambda}(u_{1j} - hg_{0j})}$

si i = n, $x_n = a$ alors $\underline{u_{nj} = \frac{1}{1-h\lambda}(u_{n-1j} + hg_{nj})}$

3 Travail attendu

1. On a fait toutes les méthodes avec un stockage plein et en utilisant tous les coefficients de la matrice, puis toujours avec un stockage plein mais en adaptant l'algorithme au profil bande de la matrice, et enfin on a modifié les algorithmes "adaptés plein" pour utiliser le stockage bande.
   Les algos de décomposition avec des permutations ne conserve pas le profil bande de A, on ne peut donc pas les utiliser avec le stockage bande.
   On a pas utilisé les stockages diagonal et creux car ils nécessitent d'utiliser des bibliothèques.

   Cholesky (SS^T) version 2

   ```
   pour j de 0 à (n-1)(m-1)-1
       somme = 0
       pour k de 0 à j-1
           somme += (S[j][k])²        // normalement on prend le module de S[j][k]
       S[j][j] = sqrt (A[j][j] - somme)
       pour i de j+1 à (n-1)(m-1)-1
           somme = 0
           pour k de 0 à j-1
               somme += S[i][k] * S[j][k]    // normalement c'est le conjugué de S[j][k]
           S[i][j] = (1/S[j][j]) * (A[i][j]-somme)
   ```

   Complexité : $\sim\infty \frac{n^3 m^3}{3}$

   L'algorithme de Cholesky utilise un procédé par colonnes.
   Comme on a déjà adapté l'algorithme au profil bande de la matrice, on peut facilement l'adapter à une matrice stockée en bande.

2. Méthode itérative :

   Nous allons utiliser la méthode de relaxation (SOR). On cherche à résoudre le système Ax = b de façon itérative. On décompose A = D-E-F avec :

— D une matrice diagonale, D = diag(diag(A))

— -E : $e_{ij} = \begin{cases} -a_{ij} & \text{si } j < i \\ 0 & \text{sinon.} \end{cases}$ une matrice triangulaire inférieure.

— -F : $f_{ij} = \begin{cases} -a_{ij} & \text{si } j > i \\ 0 & \text{sinon.} \end{cases}$ une matrice triangulaire supérieure.

On construit la suite des $x^{(n)}$ comme suit :

$(\frac{1}{\omega}$ D - E$)x^{(n+1)} = ((\frac{1}{\omega}$-1) D + F$)x^{(n)}$ + b

$x^{(n+1)} = (\frac{1}{\omega}D - E)^{-1}$ $((\frac{1}{\omega}$-1) D + F$)x^{(n)} + (\frac{1}{\omega}D - E)^{-1}$b

$x^{(n+1)} = B_{SOR}\ x^{(n)}$ + c

avec B_{SOR} la matrice d'itération $B_{SOR} = (\frac{1}{\omega}D - E)^{-1}((\frac{1-\omega}{\omega})D + F)$
et c = $(\frac{1}{\omega}D - E)^{-1}$b .

Voici en exemple de la matrice d'itération (en 4*4, i.e n=3 et m=3) :

A = $\begin{pmatrix} 36 & -9 & -9 & 0 \\ -9 & 36 & 0 & -9 \\ -9 & 0 & 36 & -9 \\ 0 & -9 & -9 & 36 \end{pmatrix}$, E = $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 \\ 0 & 9 & 9 & 0 \end{pmatrix}$, F = $\begin{pmatrix} 0 & 9 & 9 & 0 \\ 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, et

D = $\begin{pmatrix} 36 & 0 & 0 & 0 \\ 0 & 36 & 0 & 0 \\ 0 & 0 & 36 & 0 \\ 0 & 0 & 0 & 36 \end{pmatrix}$, on obtient donc :

$\frac{1}{\omega}$D - E = $\begin{pmatrix} \frac{36}{\omega} & 0 & 0 & 0 \\ -9 & \frac{36}{\omega} & 0 & 0 \\ -9 & 0 & \frac{36}{\omega} & 0 \\ 0 & -9 & -9 & \frac{36}{\omega} \end{pmatrix}$, et on en déduit

X = $(\frac{1}{\omega}D - E)^{-1}$ = $\begin{pmatrix} \frac{\omega}{36} & 0 & 0 & 0 \\ \frac{\omega}{144} & \frac{\omega}{36} & 0 & 0 \\ \frac{\omega}{144} & 0 & \frac{\omega}{36} & 0 \\ \frac{\omega}{288} & \frac{\omega}{144} & \frac{\omega}{144} & \frac{\omega}{36} \end{pmatrix}$, puis

$$Y = (\frac{1-\omega}{\omega})D + F = \begin{pmatrix} \frac{36(1-\omega)}{\omega} & 9 & 9 & 0 \\ 0 & \frac{36(1-\omega)}{\omega} & 0 & 9 \\ 0 & 0 & \frac{36(1-\omega)}{\omega} & 9 \\ 0 & 0 & 0 & \frac{36(1-\omega)}{\omega} \end{pmatrix},$$

on obtient donc la matrice d'itération :

$$B_{SOR} = XY = \begin{pmatrix} (1-\omega) & \frac{\omega}{4} & \frac{\omega}{4} & 0 \\ \frac{(1-\omega)}{4} & \frac{(16-15\omega)}{16} & \frac{\omega}{16} & \frac{\omega}{4} \\ \frac{(1-\omega)}{4} & \frac{\omega}{16} & \frac{(16-15\omega)}{16} & \frac{\omega}{4} \\ \frac{(1-\omega)}{8} & \frac{(8-7\omega)}{32} & \frac{(8-7\omega)}{32} & \frac{(8-7\omega)}{8} \end{pmatrix}$$

Ce qui donne pour i = 1 à n :

$$x_i^{(n+1)} = x_i^{(n)} + \frac{\omega}{a_{ii}}(b_i - \sum_{j<i} a_{ij}x_j^{(n+1)} - \sum_{j\geq i} a_{ij}x_j^{(n)})$$

Cette méthode converge quand le rayon spectral de B_{SOR} est inférieur à 1, $\rho(B_{SOR} < 1)$, ce qui est le cas quand $0 < \omega < 2$, car A est symétrique définie positive.
On choisit ω en "tatonnant".

L'algorithme en pseudo-code donne :

```
Relaxation

x=x⁰

tant que nb_iterations < max_iterations & ‖Ax -b‖ > residu_cible
    pour i de 1 à A.size()
        somme = 0
        pour j de 1 à A.size()
            somme += aij xj
        xi = xi + ω/aii (bi - somme)
```

3. Autres méthodes itératives :

   Les autres méthodes que l'on aurait pu utiliser sont la méthode de Richardson, de Jacobi, qui converge car A est à diagonale fortement dominante et irréductible.
   On aurait aussi pu utiliser la méthode de Gauss-Seidel qui converge car correspond au cas où $\omega = 1$ dans la méthode de relaxation.
   Mais la méthode de relaxation a une meilleure vitesse de convergence.

4 Résumé des résultats à obtenir

1. Voici le tableau du nombre de coefficients non nuls en fonction de n et m :

Nombres de coefficients non nuls et largeur de bande

		NNZ	Bande
m	n	5(n-1)(m-1) -2((m-1)+1)	2(m-1)+1
2	2	1E+00	3E+00
2^2	2^2	3,7E+01	7E+00
2^3	2^3	2,29E+02	1,5E+01
2^4	2^4	1,093E+03	3,1E+01
2^5	2^5	4,741E+03	6,3E+01
2^6	2^6	1,9717E+04	1,27E+02
2^7	2^7	8,0389E+04	2,55E+02
2^8	2^8	3,24613E+05	5,11E+02
2^9	2^9	1,304581E+06	1,023E+03
2^{10}	2^{10}	5,230597E+06	2,047E+03
2^{11}	2^{11}	2,0946949E+07	4,095E+03
2^{12}	2^{12}	8,3836933E+07	8,191E+03
2^{13}	2^{13}	3,35446021E+08	1,6383E+04
2^{14}	2^{14}	1,341980677E+09	3,2767E+04
2^{15}	2^{15}	5,368315909E+09	6,5535E+04

Ensuite le tableau de la mémoire occupée par la matrice A :

Taille mémoire de A (en bytes) avec différent stockage

		Plein	Bande	Diagonal	COO	CSC	CSR
m	n	$8n^2m^2$	$16nm^2$	40nm	120nm	88nm	88nm
2	2	1,28E+02	1,28E+02	1,6E+02	4,8E+02	3,52E+02	3,52E+02
2^2	2^2	2,048E+03	1,024E+03	6,4E+02	1,92E+03	1,408E+03	1,408E+03
2^3	2^3	3,2768E+04	8,192E+03	2,56E+03	7,68E+03	5,632E+03	5,632E+03
2^4	2^4	5,24288E+05	6,5536E+04	1,024E+04	3,072E+04	2,2528E+04	2,2528E+04
2^5	2^5	8,388608E+06	5,24288E+05	4,096E+04	1,2288E+05	9,0112E+04	9,0112E+04
2^6	2^6	1,34217728E+08	4,194304E+06	1,6384E+05	4,9152E+05	3,60448E+05	3,60448E+05
2^7	2^7	2,147483648E+09	3,3554432E+07	6,5536E+05	1,96608E+06	1,441792E+06	1,441792E+06
2^8	2^8	3,4359738368E+10	2,68435456E+08	2,62144E+06	7,86432E+06	5,767168E+06	5,767168E+06
2^9	2^9	5,49755813888E+11	2,147483648E+09	1,048576E+07	3,145728E+07	2,3068672E+07	2,3068672E+07
2^{10}	2^{10}	8,796093022208E+12	1,7179869184E+10	4,194304E+07	1,2582912E+08	9,2274688E+07	9,2274688E+07
2^{11}	2^{11}	1,40737488355328E+14	1,37438953472E+11	1,6777216E+08	5,0331648E+08	3,69098752E+08	3,69098752E+08
2^{12}	2^{12}	2,25179981368525E+15	1,099511627776E+12	6,7108864E+08	2,01326592E+09	1,476395008E+09	1,476395008E+09
2^{13}	2^{13}	3,6028797018964E+16	8,796093022208E+12	2,68435456E+09	8,05306368E+09	5,905580032E+09	5,905580032E+09
2^{14}	2^{14}	5,76460752303423E+17	7,0368744177664E+13	1,073741824E+10	3,221225472E+10	2,3622320128E+10	2,3622320128E+10
2^{15}	2^{15}	9,22337203685478E+18	5,62949953421312E+14	4,294967296E+10	1,2884901888E+11	9,4489280512E+10	9,4489280512E+10

Enfin voici le tableau des complexités de chaque méthode :

Nombre d'opérations et temps (à 10^9 op/s)

		Pivot de Gauss	LU	LU pivot partiel	SS¹ v1	SS¹ v2	LU adapté à A	SS¹ v1 adapté à A
m	n	$\frac{2n^3m^3}{3}$	$\frac{2n^3m^3}{3}$	$\frac{2n^3m^3}{3}$	$\frac{2n^3m^3}{3}$	$\frac{n^3m^3}{3}$	$2nm^3$	$2nm^3$
2	2	4,266667E+01	4,266667E+01	4,266667E+01	4,266667E+01	2,133333E+01	3,2E+01	3,2E+01
2^2	2^2	2,730667E+03	2,730667E+03	2,730667E+03	2,730667E+03	1,365333E+03	5,12E+02	5,12E+02
2^3	2^3	1,747627E+05	1,747627E+05	1,747627E+05	1,747627E+05	8,738133E+04	8,192E+03	8,192E+03
2^4	2^4	1,118481E+07	1,118481E+07	1,118481E+07	1,118481E+07	5,592405E+06	1,31072E+05	1,31072E+05
2^5	2^5	7,158279E+08	7,158279E+08	7,158279E+08	7,158279E+08	3,579139E+08	2,097152E+06	2,097152E+06
2^6	2^6	4,581298E+10	4,581298E+10	4,581298E+10	4,581298E+10	2,290649E+10	3,3554432E+07	3,3554432E+07
2^7	2^7	2,932031E+12	2,932031E+12	2,932031E+12	2,932031E+12	1,466016E+12	5,36870912E+08	5,36870912E+08
2^8	2^8	1,876500E+14	1,876500E+14	1,876500E+14	1,876500E+14	9,382499E+13	8,589934592E+09	8,589934592E+09
2^9	2^9	1,200960E+16	1,200960E+16	1,200960E+16	1,200960E+16	6,004800E+15	1,37438953472E+11	1,37438953472E+11
2^{10}	2^{10}	7,686143E+17	7,686143E+17	7,686143E+17	7,686143E+17	3,843072E+17	2,199023255552E+12	2,199023255552E+12
2^{11}	2^{11}	4,919132E+19	4,919132E+19	4,919132E+19	4,919132E+19	2,459566E+19	3,5184372088832E+13	3,5184372088832E+13
2^{12}	2^{12}	3,148244E+21	3,148244E+21	3,148244E+21	3,148244E+21	1,574122E+21	5,62949953421312E+14	5,62949953421312E+14
2^{13}	2^{13}	2,014876E+23	2,014876E+23	2,014876E+23	2,014876E+23	1,007438E+23	9,00719925474099E+15	9,00719925474099E+15
2^{14}	2^{14}	1,289521E+25	1,289521E+25	1,289521E+25	1,289521E+25	6,447604E+24	1,44115188075856E+17	1,44115188075856E+17
2^{15}	2^{15}	8,252934E+26	8,252934E+26	8,252934E+26	8,252934E+26	4,126467E+26	2,30584300921369E+18	2,30584300921369E+18

2. Voici les temps CPU de résolution qu'on obtient en fonction des différentes méthodes :

Temps CPU

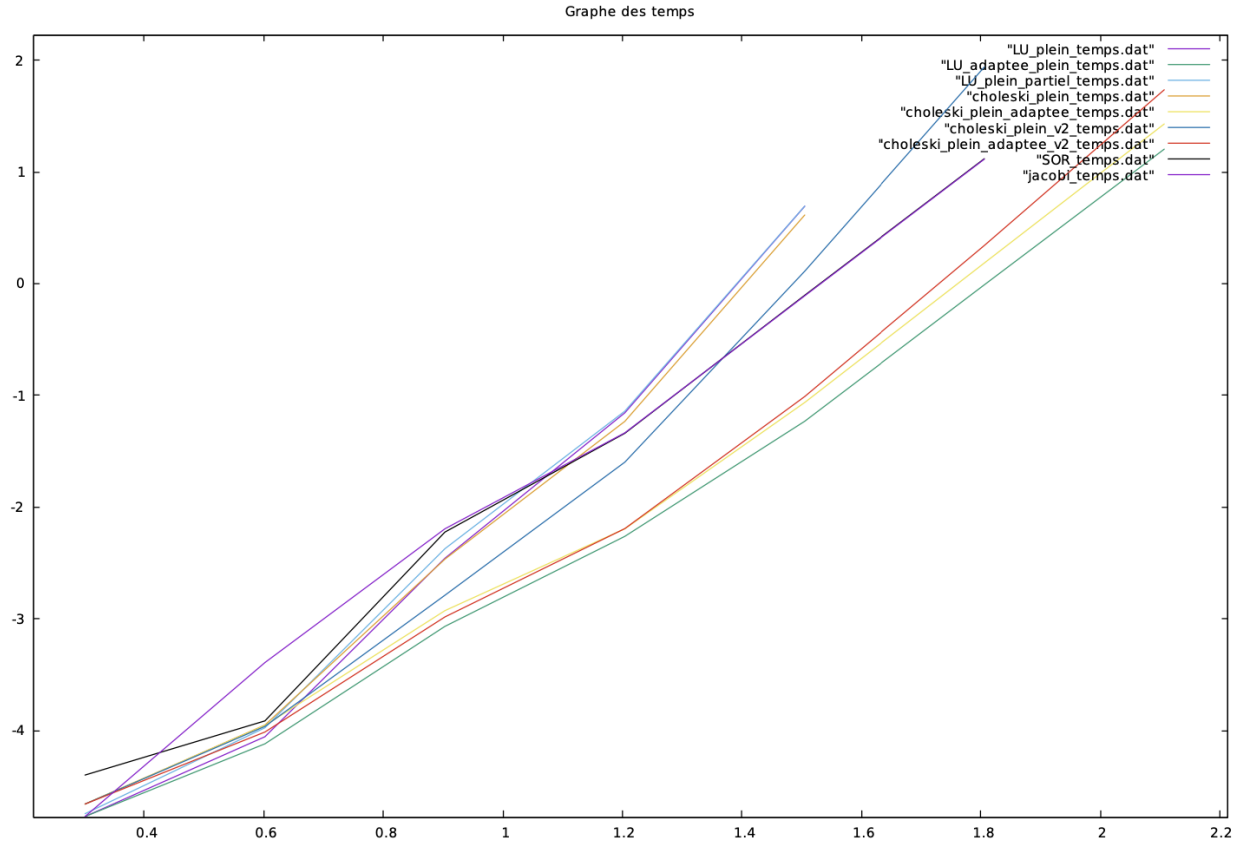
m	n	LU_plein	LU_plein_adaptée	LU_partiel_plein
2	2	1,70000422215637E-05	1,70000422215637E-05	1,79998961724559E-05
2^2	2^2	8,79994585505891E-05	7,59993713661527E-05	1,06001009186253E-04
2^3	2^3	3,48096775232401E-03	8,55992579805155E-04	4,24404519429134E-03
2^4	2^4	0,0696883208084927	5,47797442079277E-03	7,21855051518927E-02
2^5	2^5	4,92895666626519E+00	5,86340644429065E-02	4,93853345333669E+00
2^6	2^6		9,76335093675415E-01	
2^7	2^7		1,59734968335354E+01	
2^8	2^8			

Temps CPU

m	n	cholesky_plein	cholesky_plein_adaptée	cholesky_v2_plein	cholesky_v2_plein_adaptée
2	2	2,19998641983516E-05	2,19998641983516E-05	2,19998641983516E-05	2,19998641983516E-05
2^2	2^2	1,12000509933819E-04	1,10999238663628E-04	1,09000878956623E-04	9,69996126510999E-05
2^3	2^3	3,42003066432634E-03	1,18500450119363E-03	1,62498740804744E-03	1,03999200345037E-03
2^4	2^4	5,85612043909672E-02	6,38895751220507E-03	2,52481754814201E-02	6,4230307324544E-03
2^5	2^5	4,11382677051429E+00	8,62541500534819E-02	1,29212937572965E+00	9,75169253988599E-02
2^6	2^6		1,52778063170786E+00	8,69020457019489E+01	2,1971818356363E+00
2^7	2^7		2,68311940371618E+01		5,42825198688785E+01
2^8	2^8				

Temps CPU

m	n	Jacobi_plein	SOR_plein
2	2	1,70000422215637E-05	4,00000007987242E-05
2^2	2^2	4,06003708025499E-04	1,22000047566077E-04
2^3	2^3	6,38998737348277E-03	5,97998369714251E-03
2^4	2^4	4,61391936160978E-02	4,57772816714559E-02
2^5	2^5	7,78305321756825E-01	7,87335801498424E-01
2^6	2^6	1,30124777523885E+01	1,31183737487489E+01
2^7	2^7		



On peut observer que la méthode qui est la plus rapide est la décomposition LU adaptée.

Voici les résidus en norme 2 qu'on obtient en fonction de chaque méthode :

residu_2

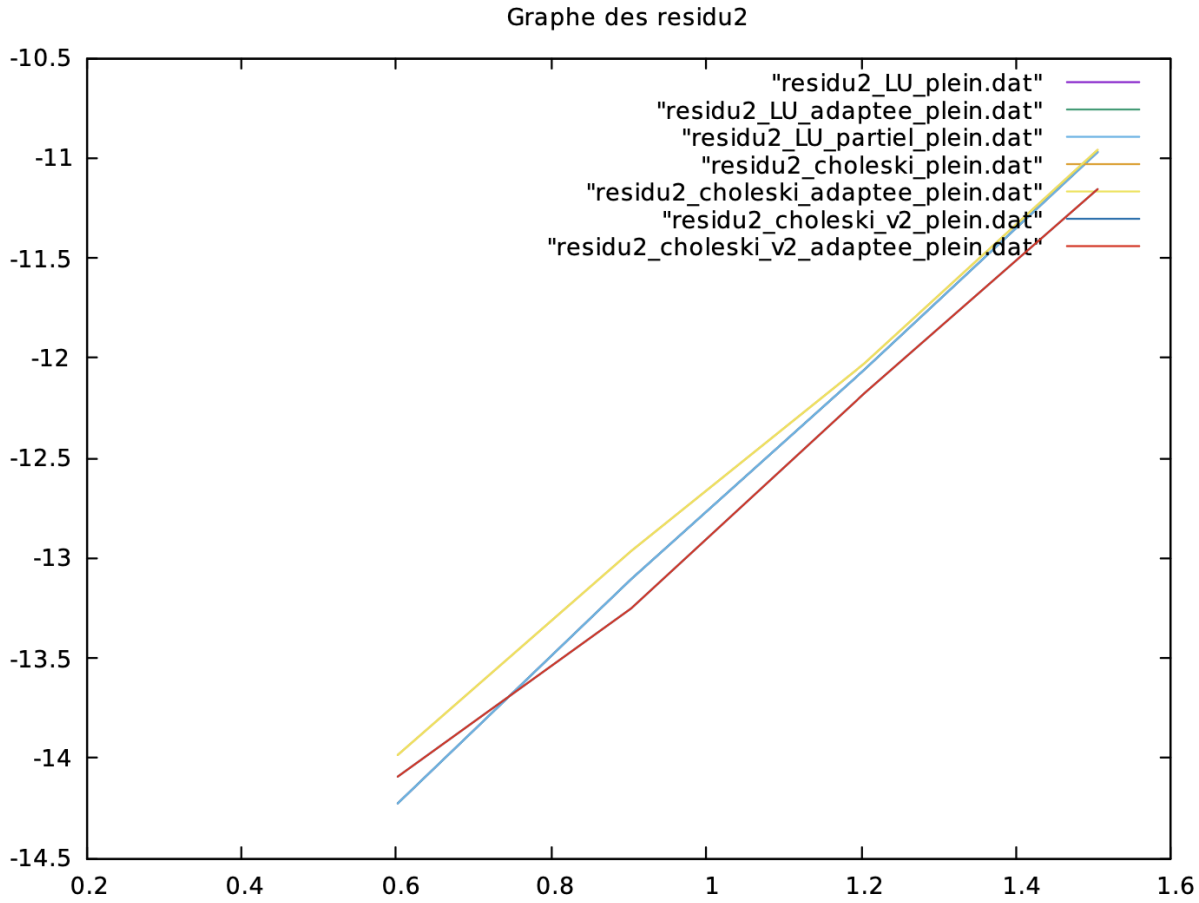
m	n	LU_plein	LU_plein_adaptée	LU_partiel_plein
2	2	0E+00	0E+00	0E+00
2^2	2^2	5,93335044380005E-15	5,93335044380005E-15	5,93335044380005E-15
2^3	2^3	7,8487410348873E-14	7,8487410348873E-14	7,8487410348873E-14
2^4	2^4	8,74782325961286E-13	8,74782325961286E-13	8,74782325961286E-13
2^5	2^5	1,0715193052376E-11	1,0715193052376E-11	1,0715193052376E-11

residu_2

m	n	cholesky_plein	cholesky_plein_adaptée	cholesky_v2_plein	cholesky_v2_plein_adaptée
2	2	0E+00	0E+00	0E+00	0E+00
2^2	2^2	1,03561897704499E-14	1,03561897704499E-14	8,07606861328975E-15	8,07606861328975E-15
2^3	2^3	1,08317842238459E-13	1,08317842238459E-13	5,60015438761557E-14	5,60015438761557E-14
2^4	2^4	9,38642036672135E-13	9,38642036672135E-13	6,65579596367229E-13	6,65579596367229E-13
2^5	2^5	1,0979940836496E-11	1,0979940836496E-11	6,98554026032695E-12	6,98554026032695E-12

residu_2

m	n	Jacobi_plein	SOR_plein
2	2	0E+00	0E+00
2^2	2^2	3,14825571207578E-09	9,59577375921581E-10
2^3	2^3	1,095014827602E-02	1,62644731544874E-08
2^4	2^4	8,91452074576625E+00	2,35619917722363E-01
2^5	2^5	1,04344605153935E+02	3,30986286225985E+01



Voici les résidus infinis qu'on obtient en fonction de chaque méthode :

residu_inf

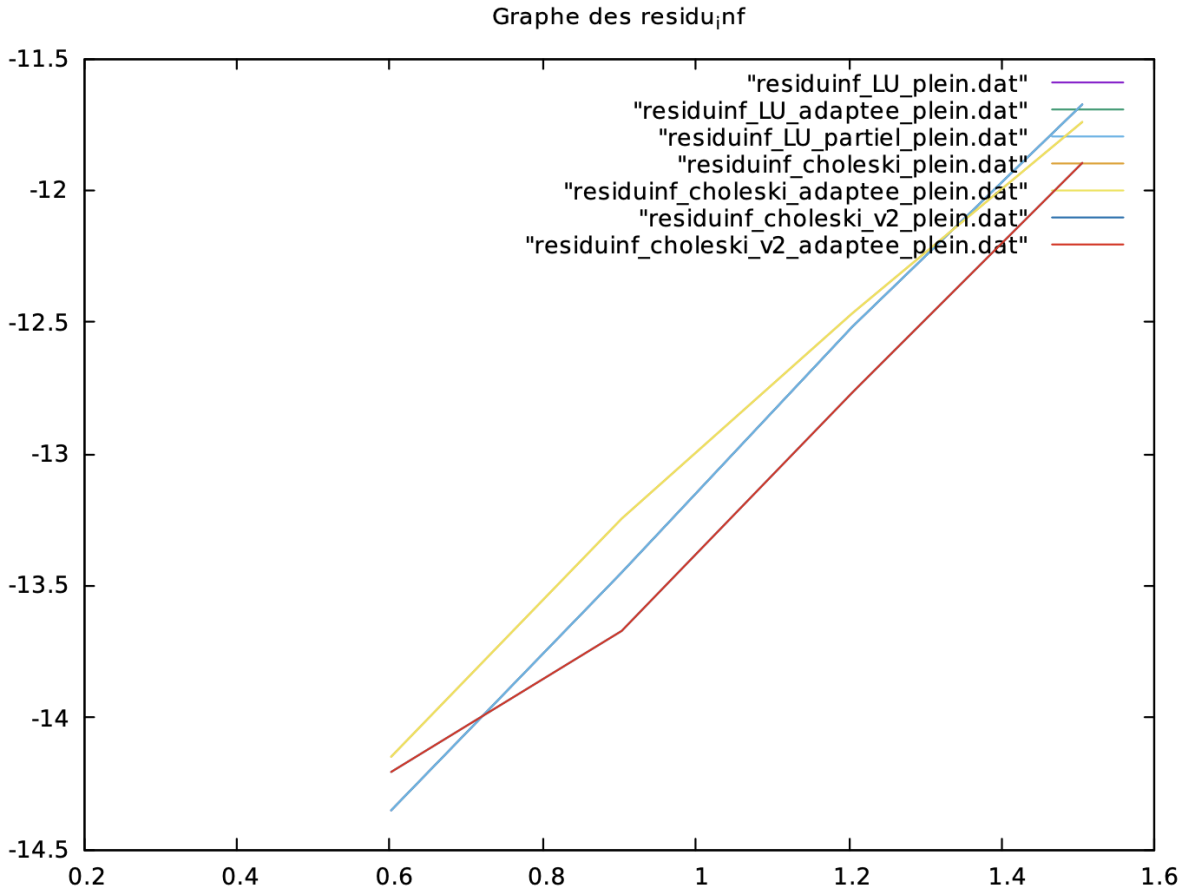
m	n	LU_plein	LU_plein_adaptée	LU_partiel_plein
2	2	0E+00	0E+00	0E+00
2^2	2^2	4,4411966136559E-15	4,4411966136559E-15	4,4411966136559E-15
2^3	2^3	3,55303920815271E-14	3,55303920815271E-14	3,55303920815271E-14
2^4	2^4	3,02761048092313E-13	3,02761048092313E-13	3,02761048092313E-13
2^5	2^5	2,11641095077086E-12	2,11641095077086E-12	2,11641095077086E-12

residu_inf

m	n	cholesky_plein	cholesky_plein_adaptée	cholesky_v2_plein	cholesky_v2_plein_adaptée
2	2	0E+00	0E+00	0E+00	0E+00
2^2	2^2	7,10558763369473E-15	7,10558763369473E-15	6,2172738929396E-15	6,2172738929396E-15
2^3	2^3	5,68460116849008E-14	5,68460116849008E-14	2,13157196671691E-14	2,13157196671691E-14
2^4	2^4	3,41035822653349E-13	3,41035822653349E-13	1,70529688990959E-13	1,70529688990959E-13
2^5	2^5	1,81718858478441E-12	1,81718858478441E-12	1,26969672678094E-12	1,26969672678094E-12

residu_inf

m	n	Jacobi_plein	SOR_plein
2	2	0E+00	0E+00
2^2	2^2	1,79382470115172E-09	9,50451586613739E-10
2^3	2^3	2,83002322734475E-03	4,21706213461198E-09
2^4	2^4	1,12898117391434E+00	2,95120922666639E-02
2^5	2^5	8,96045369771897E+00	2,19983952071769E+00



3. On a calculé les erreurs E_2 :

erreur_2

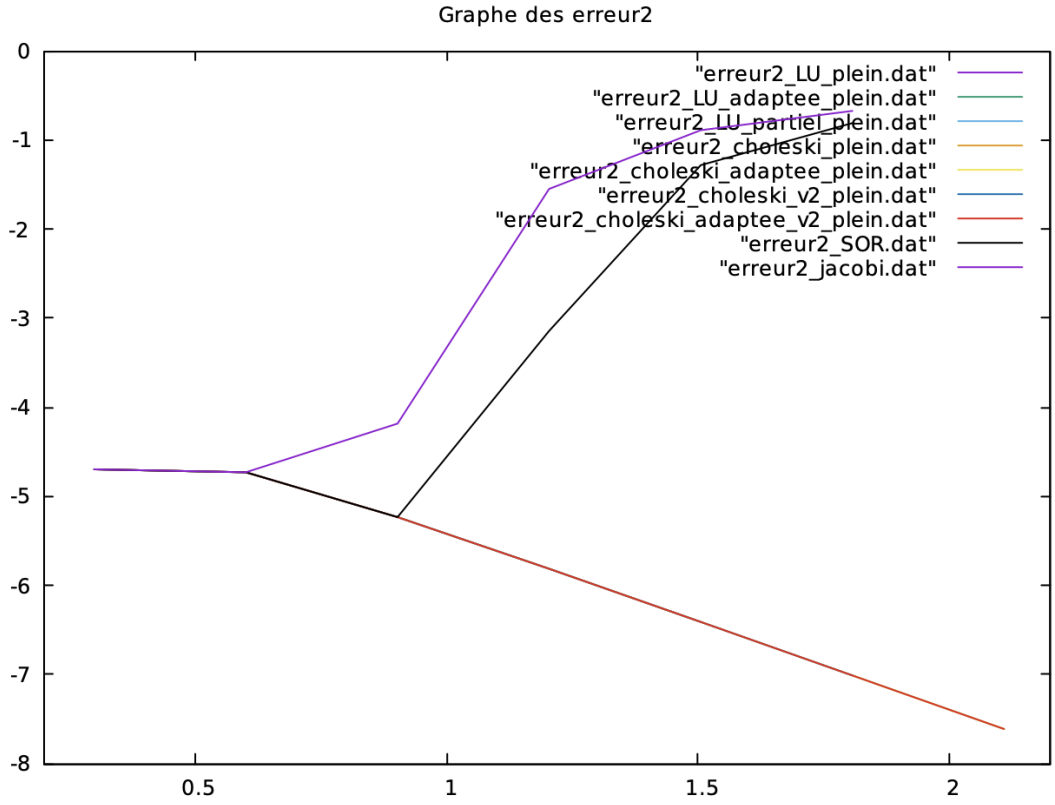
m	n	LU_plein	LU_plein_adaptée	LU_partiel_plein
2	2	2,00918533688238E-05	2,00918533688238E-05	2,00918533688238E-05
2^2	2^2	1,85686358779311E-05	1,85686358779311E-05	1,85686358779311E-05
2^3	2^3	5,83485408625782E-06	5,83485408625782E-06	5,83485408625782E-06
2^4	2^4	1,53263944977065E-06	1,53263944977065E-06	1,53263944977065E-06
2^5	2^5	3,87685894504053E-07	3,87685894504053E-07	3,87685894504053E-07
2^6	2^6		9,72030741129193E-08	
2^7	2^7		2,43181201569699E-08	

erreur_2

m	n	cholesky_plein	cholesky_plein_adaptée	cholesky_v2_plein	cholesky_v2_plein_adaptée
2	2	2,00918533688238E-05	2,00918533688238E-05	2,00918533688238E-05	2,00918533688238E-05
2^2	2^2	1,85686358779311E-05	1,85686358779311E-05	1,85686358779311E-05	1,85686358779311E-05
2^3	2^3	5,83485408625782E-06	5,83485408625782E-06	5,83485408625782E-06	5,83485408625782E-06
2^4	2^4	1,53263944977065E-06	1,53263944977065E-06	1,53263944977065E-06	1,53263944977065E-06
2^5	2^5	3,87685894504053E-07	3,87685894504053E-07	3,87685894504053E-07	3,87685894504053E-07
2^6	2^6		9,72030741129193E-08		9,72030741129193E-08
2^7	2^7		2,43181201569699E-08		2,43181201569699E-08

erreur_2

m	n	Jacobi_plein	SOR_plein
2	2	2,00918533688238E-05	2,00918533688238E-05
2^2	2^2	1,85686358779311E-05	1,85686358779311E-05
2^3	2^3	6,53100475566542E-05	5,83485408625782E-06
2^4	2^4	2,822149394343E-02	7,17546416715008E-04
2^5	2^5	1,28621331166245E-01	5,21038722080411E-02
2^6	2^6	2,12342047153279E-01	1,55057222393527E-01



On remarque que les erreurs se comportent comme h^2 et k^2 quand n et m tendent vers $+\infty$.

Les méthodes en bande ne marchent pas à cause de la transposée avec des matrices bandes (mais nos décompositions LU et cholesky en bande marchent). On a été limité dans les tests par le calcul du résidu qui prenait beaucoup de temps.

On a remarqué également que l'erreur 2 est multipliée par 2 lorsque m et n passent à la puissance de deux suivante.

La question du choix du stockage de la matrice de la méthode dépend de ce que l'on veut privilégier. Par exemple si on avait utilisé un stockage creux, cela n'aurait pas coûté beaucoup de mémoire.

On voit que les différences entre les méthodes directes ne sont pas flagrantes. Les méthodes cholesky v2 et v1 auraient dû être plus rapide que les décompositions LU, les méthodes "adaptées" devraient être plus rapides que celles utilisant toute la matrice A. Peut-être que si on avait réussi à aller sur de plus grands n et m, les différences auraient été plus flagrantes.