

Part I

Algorithme de triangularisation de Gauss

L'algorithme de Gauss est une généralisation de la méthode d'élimination qui permet la résolution à la main des petit système linéaire.

Remarques

- Une des spécifité de l'algorithme de Gauss est que les calculs successifs de coefficients concernent à la fois les matrices et les second membres, contrairement aux algorithme directs de type LU.
- Les matrices successifs obtenues par l'algorithme de Gauss ont tous le même déterminant, égal au déterminant de A. la dernière matrice calculé $A^{(n-1)}$ permet donc de calculé le déterminant de A : $\det(A) = \prod_{i=1}^n a_{ii}^{(n-1)}$.

$A^{(n-1)}$

1 Algorithme de Gauss basique

```

Pour j = 1 à n-1 faire          # parcours des colonnes
    Pour i = j+1 à n faire      # parcours des lignes
        Lij = Aij / Ajj          # pivot de Gauss
        bi = bi - Lij*bj
        Pour k = j à n faire
            Aik = Aik - Lij*ajk   # ligne i - ligne pivot * pivot
        Fin Pour
    Fin Pour
Fin Pour

```

Équivalent mathématique :

- $L_{ij}^{(k+1)} = \frac{A_{ij}^{(k)}}{A_{jj}^{(k)}}$ pour tout $i > k$
- $b_i^{(k+1)} = b_i^{(k)} - L_{ij}b_j^{(k)}$ pour tout $i > k$
- $A_{ik}^{(k+1)} = A_{ik}^{(k)} - L_{ij}A_{jk}^{(k)}$ pour tout $k > j$ équivalent à $L_i = L_i - pivot * L_{pivot}$

Attention, cette méthode ne marche que si tout les éléments diagonaux de A sont non nulle.

La complexité de l'algorithme d'éliminatioin de Gauss est $O(\frac{2n^3}{3})$.

6.5 COMPLEXITÉ

Chaque étape d'élimination requiert le calcul de $n - i$ divisions, $(n - i)^2$ additions, $(n - i)^2$ multiplications avec $1 \leq i \leq n - 1$. Pour l'ensemble des étapes on obtient

- $1 + \dots + n - 1 = \frac{n(n - 1)}{2}$ divisions,
- $1^2 + \dots + (n - 1)^2 = \frac{n(n - 1)(2n - 1)}{6}$ additions,
- $1^2 + \dots + (n - 1)^2 = \frac{n(n - 1)(2n - 1)}{6}$ multiplications,

donc un total de

$$\frac{n(n - 1)(2n - 1)}{3} + \frac{n(n - 1)}{2} \approx \frac{2n^3}{3}$$

opérations arithmétiques.

Nous n'avons pas tenu compte des opérations sur le second membre b du système $Ax = b$. Leur ordre de grandeur est $O(n)$.

Algorithmes et structures de données

2 Algorithme de Gauss pivot partiel

Comme on l'a vu, si un élément diagonale de A est nul alors il y a intérêt à inverser la colonne avec une colonne qui permettrait d'avoir un pivot que l'on peut calculé.

choix du pivot de Gauss :

Supposons que les nombres soient représenté en virgules flottante dans un échelle décimale avec 3 chiffres significatifs et que les résultats des opérations soient arrondies à 3 chiffres significatifs.

Soit le système $Ax = b$

$$A = \begin{bmatrix} 10^{-4} & 1 \\ 0 & 1 \end{bmatrix} \text{ et } b = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Choisissons 10^{-4} comme premier pivot. Après élimination de x_1 on a

$$A = \begin{bmatrix} 10^{-4} & 1 \\ 0 & 1 - 10^{-4} \end{bmatrix} \text{ et } b = \begin{bmatrix} 1 \\ 2 - 10^{-4} \end{bmatrix} \text{ Or } 1 - 10^{-4} \text{ donnera } -10^4 \text{ et } 2 - 10^{-4} \text{ donnera } -10^4.$$

Ainsi la solution du système serait $x_2 = \frac{-10^{-4}}{-10^{-4}} = 1$ et $x_1 = \frac{1-1}{10^{-4}} = 0$

Choisissons maintenant comme ligne pivot la deuxième ligne. On obtient

$A = \begin{bmatrix} 0 & 1 - 10^{-4} \\ 1 & 1 \end{bmatrix}$ et $b = \begin{bmatrix} 1 - 10^{-4} \\ 2 \end{bmatrix}$ Or $1 - 10^{-4}$ sera représenté par 1 et $1 - 2 * 10^{-4}$ par 1 aussi compte tenu des erreurs d'arrondies

On aura donc

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \text{ et } b = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ donc les solutions sont } x_1 = 1 \text{ et } x_2 = 1$$

La première solution est complètement fausse alors que la deuxième est excellente.

Le coefficient $L_{ij} = \frac{A_{ij}}{A_{jj}}$ est inversement proportionnel au pivot. Or ces coefficients interviennent en multiplicateur dans les formules de transformation des lignes. Les erreurs d'arrondies qui affectent déjà la matrice A^k pourront être amplifier. Cette amplification est d'autant plus importante que le pivot est petit.

À la $k^{ième}$ étape on choisie comme ligne pivot, une ligne d'indice p tel que $|A_{pj}| = \max_{i \in 1..n} |A_{ij}|$.

a) Pivotage partiel

La stratégie de pivotage partiel associée à l'algorithme de Gauss consiste, lorsque l'on s'apprête à effectuer la transformation n° k de la matrice et du second membre, à rechercher dans la $k^{\text{ème}}$ colonne quel est le « meilleur candidat à être pivot » : on recherche $a_{pk}^{(k-1)}$ tel que $|a_{pk}^{(k-1)}| = \underset{r \in [k, k+1, \dots, n]}{\text{Max}} |a_{rk}^{(k-1)}|$. Comme schématisé sur la figure II.6,

on ne recherche ce pivot optimal que dans les lignes situées en dessous du pivot « naturel » que propose l'algorithme de Gauss. Une fois ce pivot optimal trouvé, on intervertit la $k^{\text{ème}}$ ligne et la $p^{\text{ème}}$ ligne de la matrice $A^{(k-1)}$, ainsi que celles du second membre $b^{(k-1)}$, ce qui ne modifie pas le système d'équations, et on continue l'algorithme d'élimination.

Le pivotage partiel est de mise en œuvre très simple, et permet souvent d'améliorer très nettement la solution fournie par l'algorithme de Gauss sans pivotage.

La figure II.7 donne un organigramme succinct de l'algorithme de Gauss avec pivotage partiel, correspondant aux conventions de stockage définies au § 2.2.

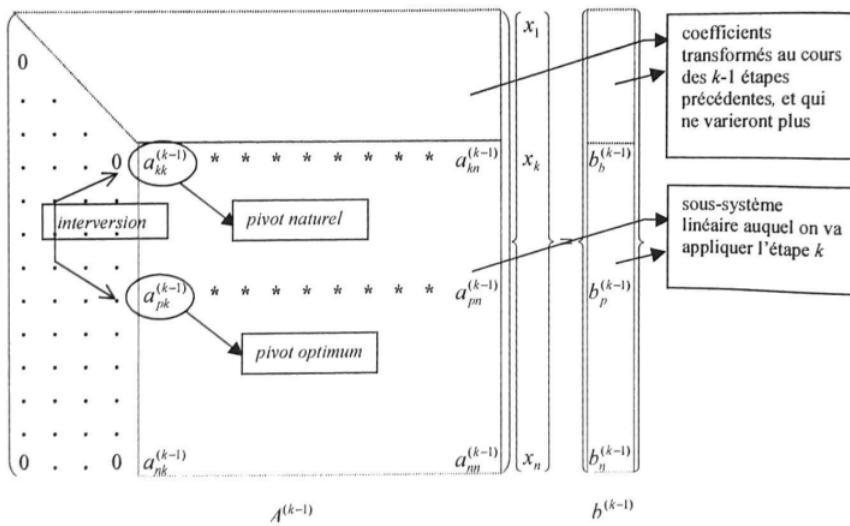


figure II.6 – Recherche de pivot optimal dans le système linéaire $A^{(k-1)}x = b^{(k-1)}$ (pivotage partiel)

Part II

Factorisation LU

3 factorisation LU

3.1 Principe

L'algorithme LU consiste à factoriser la matrice A du système linéaire $Ax = b$ en un produit de 2 matrices, afin de faciliter sa résolution ultérieure : $A = L \times U$ où :

- L (pour *low*) est une matrice triangulaire inférieure dont la diagonale est formée de coefficients égaux à 1,
- U (pour *up*) est une matrice triangulaire supérieure.

Une fois la factorisation effectuée, le système linéaire s'écrit $LUX = b$, et on obtient la solution après avoir résolu successivement deux systèmes triangulaires : $\begin{cases} Ly = b \\ UX = y \end{cases}$.

On peut vérifier que dans le problème de calcul des coefficients inconnus de L et U , le nombre d'équations est identique au nombre d'inconnues : si $A \in \mathfrak{M}_{n,n}(\mathbb{R})$, le nombre de coefficients de U à déterminer est $1+2+\dots+n = \frac{n(n+1)}{2}$, celui de L est $\frac{n(n+1)}{2}-n$, d'où un total égal à n^2 . Le nombre d'équations à saisir est égal au nombre de coefficients de A , c'est-à-dire également n^2 .

Remarque

Une caractéristique importante de l'algorithme LU est qu'il opère indépendamment du second membre b du système linéaire, ce qui est intéressant lorsque l'on doit résoudre successivement des systèmes linéaires ayant la même matrice, mais des seconds membres distincts, ce que l'on rencontre par exemple dans les discrétisations par différences finies ou éléments finis lorsque les conditions limites évoluent, ou lorsque l'on souhaite calculer l'inverse d'une matrice.

3.2 Exemples

Exemple 1

Soit $A = \begin{pmatrix} 2 & -1 & 0 \\ -4 & 3 & 1 \\ 4 & -1 & 2 \end{pmatrix}$, et les matrices inconnues $L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$ et

$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$. Le système des 9 équations à résoudre, consistant à identifier

chaque coefficient de A au coefficient correspondant du produit LU , s'écrit :

Identification de :		Calcul de :
$u_{11} = 2$	1 ^{er} pivot de A	$u_{11} = 2$ 1 ^{er} pivot de U
$l_{21}u_{11} = -4$	1 ^{ère} colonne de A	$l_{21} = -2$ 1 ^{ère} colonne de L
$l_{31}u_{11} = 4$		$l_{31} = 2$
$u_{12} = -1$	1 ^{ère} ligne de A	$u_{12} = -1$ 1 ^{ère} ligne de U
$u_{13} = 0$		$u_{13} = 0$
$l_{21}u_{12} + u_{22} = 3$	2 ^{ème} pivot de A	$u_{22} = 1$ 2 ^{ème} pivot de U
$l_{31}u_{12} + l_{32}u_{22} = -1$	2 ^{ème} colonne de A	$l_{32} = 1$ 2 ^{ème} colonne de L
$l_{21}u_{13} + u_{23} = 1$	2 ^{ème} ligne de A	$u_{23} = 1$ 2 ^{ème} ligne de U
$l_{31}u_{13} + l_{32}u_{23} + u_{33} = 2$	3 ^{ème} pivot de A	$u_{33} = 1$ 3 ^{ème} pivot de U

d'où

On constate que la résolution du système linéaire dont les coefficients des matrices L et U sont solutions est triviale. Elle permet de calculer dans l'ordre :

- un pivot de U
- la colonne correspondante de L
- la ligne correspondante de U
- le pivot suivant de U
- etc...

Ainsi, on obtient la factorisation $A = \begin{pmatrix} 2 & -1 & 0 \\ -4 & 3 & 1 \\ 4 & -1 & 2 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 2 & 1 & 1 \end{pmatrix}}_L \underbrace{\begin{pmatrix} 2 & -1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}}_U$, ainsi que

la valeur du déterminant de A : $\det A = \det U = 2$.

Exemple 2

On considère le cas où $A = \begin{pmatrix} 2 & -1 & 1 \\ 4 & -2 & 1 \\ 2 & 0 & 1 \end{pmatrix}$ et les matrices inconnues $L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}$.

$U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$. Le système des 9 équations à résoudre devient :

Identification de :

$$\begin{array}{ll} u_{11} = 2 & 1^{\text{er}} \text{ pivot de } A \\ l_{21}u_{11} = 4 & 1^{\text{ere}} \text{ colonne de } A \\ l_{31}u_{11} = 2 & \\ u_{12} = -1 & 1^{\text{ere}} \text{ ligne de } A \\ u_{13} = 1 & \\ l_{21}u_{12} + u_{22} = -2 & 2^{\text{eme}} \text{ pivot de } A \\ l_{31}u_{12} + l_{32}u_{22} = 0 & 2^{\text{eme}} \text{ colonne de } A \\ l_{21}u_{13} + u_{23} = 1 & 2^{\text{eme}} \text{ ligne de } A \\ l_{31}u_{13} + l_{32}u_{23} + u_{33} = 1 & 3^{\text{eme}} \text{ pivot de } A \end{array}$$

, d'où

Calcul de :

$$\begin{array}{ll} u_{11} = 2 & 1^{\text{er}} \text{ pivot de } U \\ l_{21} = 2 & 1^{\text{ere}} \text{ colonne de } L \\ l_{31} = 1 & \\ u_{12} = -1 & 1^{\text{ere}} \text{ ligne de } U \\ u_{13} = 1 & \\ u_{22} = 0 & \text{le } 2^{\text{eme}} \text{ pivot de } U \text{ est nul!} \\ \text{calcul des autres coefficients} & \\ & \text{impossible} \end{array}$$

On constate que dans ce cas, le 2^{eme} pivot de la matrice U est égal à 0, ce qui rend la suite de la décomposition impossible.

Ainsi, dans la factorisation LU comme dans l'algorithme de Gauss, l'apparition de pivots nuls peut rendre la suite du processus impossible. Et la présence de pivots petits est susceptible de provoquer des phénomènes d'instabilité, dans l'algorithme LU comme dans l'algorithme de Gauss.

Une stratégie de pivotage permet, dans la factorisation LU comme dans l'algorithme de Gauss, d'éviter l'apparition de pivots petits.

On applique par exemple le pivotage partiel (cf. § 2.6) à l'exemple considéré. La recherche de pivot maximum dans la 1^{ere} colonne conduit à intervertir les deux

premières lignes, et définir ainsi la matrice $A' = \begin{pmatrix} 4 & -2 & 1 \\ 2 & -1 & 1 \\ 2 & 0 & 1 \end{pmatrix}$. Le processus de

factorisation donne $A' = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & l'_{32} & 1 \end{pmatrix}}_{L'} \underbrace{\begin{pmatrix} 4 & -2 & 1 \\ 0 & 0 & u'_{23} \\ 0 & 0 & u'_{33} \end{pmatrix}}_{U'},$ et à nouveau la factorisation

s'avère impossible puisque le 2^{ème} pivot de U est nul. Il suffit alors d'inverser les 2 dernières lignes de A' , ainsi que les 2 derniers coefficients de la 1^{ère} colonne de L' :

$$A'' = \begin{pmatrix} 4 & -2 & 1 \\ 2 & 0 & 1 \\ 2 & -1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & l''_{32} & 1 \end{pmatrix}}_{L''} \underbrace{\begin{pmatrix} 4 & -2 & 1 \\ 0 & u''_{22} & u''_{23} \\ 0 & 0 & u''_{33} \end{pmatrix}}_{U''}.$$

Puis on continue la factorisation en calculant successivement u''_{22} , l''_{32} , u''_{23} , et enfin u''_{33} , pour aboutir à :

$$A'' = \begin{pmatrix} 4 & -2 & 1 \\ 2 & 0 & 1 \\ 2 & -1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix}}_{L''} \underbrace{\begin{pmatrix} 4 & -2 & 1 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \end{pmatrix}}_{U''}. \quad \text{On peut en déduire que}$$

$\det A'' = \det U'' = 2$ ($= \det A$ puisque $A'' = PA$ où $P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ est la matrice d'une permutation circulaire).

3.3 Algorithme

Comme illustré par l'exemple 2 du § 3.2, une matrice inversible ne possède pas toujours de factorisation LU.

Cependant, pour toute matrice $A \in \mathfrak{M}_{n,n}(\mathbb{R})$ inversible, il existe au moins une matrice de permutation P , telle que la factorisation $PA = LU$ soit possible, dans laquelle :

- $L \in \mathfrak{M}_{n,n}(\mathbb{R})$ est une matrice triangulaire inférieure dont la diagonale est formée de coefficients égaux à 1,
- U est une matrice triangulaire supérieure.

Lorsque la factorisation est possible, elle est unique.

Si l'objectif de la factorisation LU est la résolution du système linéaire $Ax = b$, celui-ci devient $LUX = Pb$, et on obtient la solution après avoir résolu successivement deux systèmes triangulaires : $\begin{cases} Ly = Pb \\ UX = y \end{cases}$.

La figure II.8 représente un algorithme de factorisation LU d'une matrice notée A ; seul est considéré le cas où la factorisation est possible, c'est-à-dire lorsqu'aucun des pivots rencontrés n'est nul. Pour la mise en œuvre, il est possible de stocker les coefficients de la matrice A , ainsi que les coefficients non connus a priori de L et U dans un même tableau, qui est noté C dans l'organigramme correspondant, cf. figure II.9.

Remarques

- Comme l'algorithme de Gauss, l'algorithme LU est instable, du fait des divisions par les pivots de la matrice U .
- La factorisation LU, présentée dans ce chapitre comme un algorithme direct de résolution de système linéaire, intervient également dans l'algorithme LR qui permet de calculer les valeurs propres d'une matrice (cf. chapitre V, § 3).

3.4 Complexité

La complexité de l'algorithme LU est, comme celle de l'algorithme de Gauss, un $O(n^3)$: pour les gros systèmes linéaires, elle varie comme le cube du nombre d'équations.

La complexité de la résolution des 2 systèmes triangulaires obtenus après factorisation étant chacune un $O(n^2)$, le coût du processus de résolution est donc négligeable devant le coût de factorisation pour les gros systèmes linéaires.

Ainsi, cet algorithme a un intérêt particulier lorsque l'on doit résoudre successivement plusieurs systèmes linéaires ayant la même matrice et des seconds membres distincts. Il suffit alors d'une seule étape de factorisation, suivie d'une succession de résolutions de systèmes triangulaires (2 pour chaque second membre).

```

 $u_{11} = a_{11}$            1er pivot de  $U$ 
 $l_{11} = 1$ 
pour  $i = 2, 3, \dots, n$ 
 $l_{i1} = \frac{a_{i1}}{u_{11}}$        1ere colonne de  $L$ 
 $l_{ii} = 0$ 
 $l_{ii} = 1$                  diagonale de  $L$ 
fin pour
pour  $j = 2, 3, \dots, n$ 
 $u_{1j} = a_{1j}$            1ere ligne de  $U$ 
 $u_{j1} = 0$ 
fin pour
pour  $k = 2, 3, \dots, n-1$ 
 $u_{kk} = a_{kk} - \sum_{p=1}^{k-1} l_{kp} u_{pk}$     $k^{\text{eme}}$  pivot de  $U$ 
pour  $i = k+1, k+2, \dots, n$ 
 $a_{ik} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}$ 
 $l_{ik} = \frac{a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}}{u_{kk}}$     $k^{\text{eme}}$  colonne de  $L$ 
 $l_{ki} = 0$ 
fin pour
pour  $j = k+1, k+2, \dots, n$ 
 $u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}$     $k^{\text{eme}}$  ligne de  $U$ 
 $u_{jk} = 0$ 
fin pour
fin pour
 $u_{nn} = a_{nn} - \sum_{p=1}^{n-1} l_{np} u_{pn}$    dernier pivot de  $U$ 

```

```

pour  $i = 2, 3, \dots, n$ 
 $c_{ii} \leftarrow \frac{c_{ii}}{c_{11}}$            1er colonne de  $L$ 
fin pour
pour  $k = 2, 3, \dots, n-1$ 
 $c_{kk} \leftarrow c_{kk} - \sum_{p=1}^{k-1} c_{kp} c_{pk}$     $k^{\text{eme}}$  pivot de  $U$ 
pour  $i = k+1, k+2, \dots, n$ 
 $c_{ik} \leftarrow c_{ik} - \sum_{p=1}^{k-1} c_{ip} c_{pk}$ 
 $c_{ik} \leftarrow \frac{c_{ik} - \sum_{p=1}^{k-1} c_{ip} c_{pk}}{c_{kk}}$     $k^{\text{eme}}$  colonne de  $L$ 
fin pour
pour  $j = k+1, k+2, \dots, n$ 
 $c_{kj} \leftarrow c_{kj} - \sum_{p=1}^{k-1} c_{kp} c_{pj}$     $k^{\text{eme}}$  ligne de  $U$ 
fin pour
fin pour
 $c_{nn} \leftarrow c_{nn} - \sum_{p=1}^{n-1} c_{np} c_{pn}$    dernier pivot de  $U$ 

```

figure II.9 – Organigramme de mise en œuvre de l'algorithme LU.
Stockage : tableau C initialisé avec les coefficients de A, puis stockant les coefficients sous-diagonaux de L et sur-diagonaux de U

figure II.8 – Algorithme de la factorisation LU d'une matrice A

3.5 Factorisation LDU

L'algorithme de factorisation LDU est une variante de l'algorithme LU, dans laquelle la matrice A est décomposée en un produit de 3 matrices : $A = L \times D \times U$.

- L est une matrice triangulaire inférieure dont la diagonale est formée de coefficients égaux à 1,
- D est une matrice diagonale,
- U est une matrice triangulaire supérieure dont la diagonale est formée de coefficients égaux à 1.

La complexité de cet algorithme est du même ordre de grandeur que celle de l'algorithme LU. Il trouve son intérêt lorsque la matrice A est symétrique, puisqu'alors $U = L^t$, et seul le calcul d'une des matrices L ou U est nécessaire.

```

pour k = 2, 3, ..., n-1
  ckk = ckk -  $\sum_{p=1}^{k-1} c_{kp} c_{pk}$    kème pivot de D
  pour i = k+1, k+2, ..., n
    cik =  $c_{ik} - \sum_{p=1}^{k-1} c_{ip} c_{pk}$    ckk   kème colonne de L
    fin pour
  pour j = k+1, k+2, ..., n
    cij = cij -  $\sum_{p=1}^{k-1} c_{ip} c_{pj}$    kème ligne de DU
    fin pour
  fin pour
  cnn = cnn -  $\sum_{p=1}^{n-1} c_{np} c_{pn}$    dernier pivot de D
  pour i = 1, 2, ..., n-1
    pour j = i+1, i+2, ..., n
      cij =  $c_{ij} / c_{ii}$    coefficients de U
    fin pour
  fin pour

```

```

l11 =  $\sqrt{a_{11}}$ 
pour i = 2, 3, ..., n
  li1 =  $a_{i1} / l_{11}$ 
fin pour
pour k = 2, 3, ..., n-1
  lkk =  $\sqrt{a_{kk} - \sum_{p=1}^{k-1} l_{kp}^2}$ 
  pour i = k+1, k+2, ..., n
    lik =  $a_{ik} - \sum_{p=1}^{k-1} l_{ip} l_{kp}$  / lkk
    fin pour
  fin pour
  lnn = ann -  $\sum_{p=1}^{n-1} l_{np}^2$ 

```

figure II.10 – Organigramme de mise en œuvre de l'algorithme LDU. Stockage : tableau C initialisé avec les coefficients de A , puis stockant les coefficients sous-diagonaux de L , diagonaux de D , et sur-diagonaux de U

figure II.11 – Algorithme de la factorisation de Cholesky d'une matrice symétrique définie positive A

4.3 Organigramme

Comme pour l'algorithme de Gauss, il est usuel de stocker la matrice et le second membre dans un unique tableau à n lignes et $n+1$ colonnes. Dans les formules de transformation de ce tableau entre l'étape k à l'étape $k+1$, on constate qu'il est possible, à condition d'utiliser un stockage unicolonne temporaire, d'écraser les anciens coefficients du tableau avec les nouveaux. Dans l'organigramme de la figure II.13, toutes les étapes successives de transformation du système linéaire sont stockées dans le même tableau, noté C .

4.4 Complexité

La complexité de l'algorithme de Householder est, comme celles de Gauss et LU, un $O(n^3)$.

5 BILAN COMPARATIF

Complexité

Toutes les méthodes directes ont une complexité du même ordre de grandeur, proportionnel au cube du nombre d'inconnues lorsque celui-ci est grand.

Instabilité

L'algorithme de Householder est stable, contrairement à l'algorithme de Gauss et aux algorithmes de type LU, dont l'instabilité peut être compensée par des stratégies de pivotage. Le pivotage partiel est de mise en œuvre très simple. Le pivotage total nécessite de gérer l'ordre des inconnues.

Influence du second membre

Les triangulations de Gauss et de Householder modifient à la fois les coefficients de la matrice et du second membre, alors que les algorithmes de type LU ne modifient que la matrice du système linéaire. Or la complexité de la résolution d'un système triangulaire est un $O(n^2)$, négligeable si n est grand devant la complexité des algorithmes de Gauss, Householder ou LU.

Il existe des situations dans lesquelles on doit résoudre successivement plusieurs systèmes linéaires ayant la même matrice et des seconds membres distincts. C'est par exemple le cas pour inverser une matrice, ou lorsque le système linéaire est issu de la discrétisation d'une équation aux dérivées partielles dont les conditions limites dépendent du temps. On peut alors privilégier les factorisations de type LU, qui ne doivent être appliqués qu'une seule fois, indépendamment du second membre.

Calcul de résidu et correction de solution

Soit le système linéaire noté $Ax = b_{T1}$ dont la solution exacte est notée s . On appelle

résidu du système linéaire $Ax = b$ associé à une solution approchée s' , le vecteur $b - As'$.

Après résolution du système par une méthode directe, il est prudent de calculer le résidu associé à la solution numérique obtenue, et de vérifier que sa norme est faible (relativement à une valeur de référence, norme de b par exemple). En effet, les procédures de triangulation puis de résolution du système triangulaire ont tendance à cumuler les erreurs d'arrondi, avec parfois des facteurs d'amplification d'erreur spectaculaires, dès soit à l'instabilité de la méthode, soit au mauvais conditionnement du système linéaire.

Ce calcul de résidu peut également permettre de corriger la solution numérique. En effet, si $\Delta s = s' - s$ est l'erreur associée à s' , le résidu associé à s' est $r = b - As - A\Delta s = -A\Delta s$. En résolvant le système $A\Delta s = -r$ (le second membre est changé par rapport au système initial, mais on garde la même matrice), on peut obtenir une valeur approchée de Δs , et donc corriger la solution.

Cette étape de correction est particulièrement peu coûteuse si l'on a résolu le système initial par un algorithme de type LU (LDU, LDL^t ...), car le changement de second membre du système ne remet pas en question la phase de factorisation de la matrice.

3.1 Factorisation Cholesky

3.6 Factorisation de Cholesky

L'algorithme de factorisation de Cholesky (cf. figure II.11), dont le principe est proche de celui de la factorisation LU, s'applique aux matrices symétriques définies positives, pour lesquelles elle est toujours possible. Il s'agit de calculer une matrice L , triangulaire inférieure, telle que $A = LL'$. Contrairement à la factorisation LU, la diagonale de la matrice L n'est pas imposée.

Théorème 6.6 Pour toute matrice $A \in \mathbb{GL}_n$

1. Une décomposition LU de A existe si et seulement si $\det A(1 : k, 1 : k) \neq 0$ pour tout $k = 1 \dots n$.

Théorème 6.10 Pour toute matrice $A \in \mathbb{GL}_n$ il existe une matrice de permutation P , une matrice triangulaire inférieure à diagonale unité L et une matrice triangulaire supérieure U telles que $PA = LU$.

Théorème 7.5 Pour une matrice $A \in \mathbb{C}^{n \times n}$, il y a équivalence entre

1. A est définie positive,
2. A est hermitienne et les matrices $A(1 : k, 1 : k)$, $1 \leq k \leq n$, ont un déterminant positif.

Corollaire 7.6 Toute matrice définie positive possède une décomposition LU .

Théorème 7.10 Soit $A \in \mathbb{C}^{n \times n}$ une matrice définie positive. Il existe une unique matrice $L \in \mathbb{C}^{n \times n}$ triangulaire inférieure telle que $l_{ii} > 0$ pour tout i et $A = LL^*$ (décomposition de Cholesky).

7.5 COMPLEXITÉ DE LA DÉCOMPOSITION

La décomposition de Cholesky s'obtient par l'algorithme suivant qui décrit l'identification des deux membres de l'équation matricielle :

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & \bar{l}_{21} & \dots & \bar{l}_{n1} \\ & l_{22} & \dots & \bar{l}_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{pmatrix} = A.$$

On obtient :

Algorithme de décomposition de Cholesky

```

 $l_{11} = \sqrt{a_{11}}$ 
pour  $j = 2 : n$ 
     $l_{j1} = a_{j1}/l_{11}$ 
fin
pour  $i = 2 : n$ 
     $l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} |l_{ik}|^2}$ 
    pour  $j = i + 1 : n$ 
         $l_{ji} = (a_{ji} - \sum_{k=1}^{i-1} l_{jk}\bar{l}_{ik}) / l_{ii}$ 
    fin
fin

```

Chaque étape de cet algorithme requiert $(2i - 1)(n - i + 1)$ opérations arithmétiques : $+$, $-$, \times , $/$, $|.|^2$ et $\sqrt{..}$. Puisque $1 \leq i \leq n$, le compte total est de $(n - 1)n(n + 1)/3 \approx n^3/3$ opérations arithmétiques soit moitié moins que pour la décomposition LU.

3.2 factorisation QR

Chapitre 8

La décomposition QR

8.1 MATRICES DE STIEFEL

Nous verrons, lors de l'étude de la méthode des moindres carrés, qu'il est utile de considérer la décomposition QR d'une matrice rectangulaire. Une telle décomposition requiert des matrices unitaires « incomplètes » que nous introduisons ici.

Définition 8.1 *On appelle matrice de Stiefel toute matrice $S \in \mathbb{C}^{m \times n}$ ($m \geq n$) dont les n vecteurs-colonne constituent une famille orthonormée dans \mathbb{C}^m . On note $\mathbb{S}_{t_{mn}}$ l'ensemble de ces matrices.*

Remarque 8.1. Une matrice unitaire est une matrice de Stiefel carrée : $\mathbb{S}_{t_{nn}} = \mathbb{U}_n$.

Proposition 8.2 *$S \in \mathbb{C}^{m \times n}$ est une matrice de Stiefel si et seulement si $S^*S = I_n$. Dans ce cas, SS^* est la projection orthogonale de \mathbb{C}^m sur le sous-espace $\text{Im } S$ engendré par les vecteurs-colonne de S .*

Démonstration. La première assertion est évidente : S^*S est en effet la matrice dont les entrées sont les produits scalaires $\langle s_j, s_i \rangle$ où les s_i sont les colonnes de S . Pour prouver la seconde assertion (voir paragraphe 1.13 sur les projections orthogonales) on note que $SS^*s_i = s_i$ pour toute colonne s_i de S et que $SS^*x = 0$ pour tout $x \in (\text{Im } S)^\perp$. En effet cette dernière condition est équivalente à $S^*x = 0$.

8.2 DÉCOMPOSITION QR

Dans tout ce chapitre nous supposons que $m \geq n$.

Définition 8.3 On appelle décomposition QR d'une matrice $A \in \mathbb{C}^{m \times n}$ avec $m \geq n$ une identité $A = QR$ où $Q \in \mathbb{S}_{mn}$ est une matrice de Stiefel et où $R \in \mathbb{C}^{n \times n}$ triangulaire supérieure.

Il n'y a pas unicité de ce type de décomposition puisque, par exemple, $1 = z\bar{z}$ pour tout nombre complexe z de module 1. Toutefois on a :

Proposition 8.4 Toute matrice $A \in \mathbb{C}^{m \times n}$ de rang n possède une et une seule décomposition $A = QR$ avec $r_{ii} > 0$ pour tout $i = 1 \dots n$.

Démonstration. Puisque A est de rang n la matrice $A^*A \in \mathbb{C}^{n \times n}$ est définie positive (théorème 7.2). Elle possède donc une décomposition de Cholesky $A^*A = R^*R$ (théorème 7.10) où $R \in \mathbb{C}^{n \times n}$ est triangulaire supérieure et à diagonale positive. Prenons $Q = AR^{-1}$. On a :

$$Q^*Q = R^{-*}A^*AR^{-1} = R^{-*}(R^*R)R^{-1} = I_n$$

ce qui prouve que $Q \in \mathbb{S}_{mn}$ et que $A = QR$.

Pour prouver l'unicité on part d'une seconde décomposition : $A = Q'R'$. On a $A^*A = R'^*R' = R^*R$ donc $R' = R$ puisque la décomposition de Cholesky est unique d'où $Q = Q'$.

Remarque 8.2. On rencontre, dans la littérature consacrée à l'algèbre linéaire, deux définitions de la décomposition QR. La première est celle donnée ci-dessus $A = Q_1R_1$ avec $Q_1 \in \mathbb{S}_{mn}$ et $R_1 \in \mathbb{C}^{n \times n}$ triangulaire supérieure. Une seconde définition est $A = Q_2R_2$ avec $Q_2 \in \mathbb{U}_m$ et $R_2 \in \mathbb{C}^{m \times n}$ triangulaire supérieure. Dans le premier cas Q_1 est rectangulaire et R_1 carrée, dans le second cas c'est l'inverse.

1. Les deux définitions coïncident lorsque $m = n$ c'est-à-dire lorsque A est carrée,
2. Le lien entre ces deux définitions est le suivant : $Q_1 = Q_2(1 : m, 1 : n)$ et $R_1 = R_2(1 : n, 1 : n)$,
3. Seule la première définition assure l'unicité de la décomposition.

Cette première définition apparaît naturellement avec la méthode de Gram-Schmidt, la seconde avec Givens et Householder. C'est la raison pour laquelle nous les conservons toutes deux.

Un des intérêts de la décomposition QR est que le conditionnement de la matrice A n'est pas détruit comme cela peut être le cas pour LU. Si $A = QR$ le système linéaire

$Ax = b$ est équivalent à $Rx = Q^*b$ qui est un système triangulaire. Par le théorème 5.4 A et R ont le même conditionnement $\text{cond}_2(A) = \text{cond}_2(R)$.

Nous allons maintenant décrire plusieurs procédés pour calculer cette décomposition.

8.3 L'ORTHONORMALISATION DE GRAM-SCHMIDT

8.3.1 Description du procédé

Soient a_1, \dots, a_n des vecteurs linéairement indépendants d'un espace hermitien (ou préhilbertien) \mathbb{E} . Le procédé d'*orthonormalisation de Gram-Schmidt* calcule n vecteurs q_1, \dots, q_n qui ont les deux vertus suivantes :

- Ils sont orthonormés,
- Pour tout i , les sous-espaces de \mathbb{E} engendrés par a_1, \dots, a_i et q_1, \dots, q_i sont les mêmes.

On obtient un tel résultat par l'algorithme suivant :

Algorithme de Gram-Schmidt

```

 $q_1 = a_1 / \|a_1\|_2$ 
pour  $i = 1 : n - 1$ 
     $p_{i+1} = a_{i+1} - \sum_{k=1}^i \langle a_{i+1}, q_k \rangle q_k$ 
     $q_{i+1} = p_{i+1} / \|p_{i+1}\|_2$ 
fin

```

Ce procédé est à la base de la construction des polynômes orthogonaux. Notons que la base orthonormée obtenue dépend de l'ordre dans lequel sont pris les vecteurs a_i . Voir l'exercice 8.9 pour un procédé d'orthonormalisation indépendant de l'ordre des a_i .

Soit $A = (a_1 \dots a_n) \in \mathbb{C}^{m \times n}$ une matrice de rang n dont les a_i sont ses vecteurs colonne. L'orthonormalisation de Gram-Schmidt appliquée aux a_i donne la matrice $Q = (q_1 \dots q_n) \in \mathbb{S}_{mn}$ et la matrice $R \in \mathbb{C}^{n \times n}$ triangulaire supérieure à diagonale positive telles que $A = QR$ par l'algorithme qui suit :

Décomposition QR par l'algorithme de Gram-Schmidt

```

 $r_{11} = \|a_1\|_2$ 
 $q_1 = a_1 / \|a_1\|_2$ 
pour  $i = 1 : n - 1$ 
     $p_{i+1} = a_{i+1}$ 
    pour  $k = 1 : i$ 
         $r_{k,i+1} = \langle a_{i+1}, q_k \rangle$ 
         $p_{i+1} = p_{i+1} - r_{k,i+1} q_k$ 
    fin
     $r_{i+1,i+1} = \|p_{i+1}\|_2$ 
     $q_{i+1} = p_{i+1} / r_{i+1,i+1}$ 
fin

```

8.3.2 Interprétation géométrique de G-S

Notons $Q_i \in \mathbb{S}_{t_{mi}}$ la matrice de Stiefel constituée des i premières colonnes de Q : q_1, \dots, q_i . L'identité

$$p_{i+1} = a_{i+1} - \sum_{k=1}^i \langle a_{i+1}, q_k \rangle q_k$$

montre que $p_{i+1} - a_{i+1} \in \text{Im } Q_i$ et que $p_{i+1} \in (\text{Im } Q_i)^\perp$, autrement dit p_{i+1} est la projection orthogonale de a_{i+1} sur $(\text{Im } Q_i)^\perp$. Ainsi

$$p_{i+1} = P_i a_{i+1}$$

où $P_i = I_m - Q_i Q_i^*$ est la matrice de la projection orthogonale sur $(\text{Im } Q_i)^\perp$ (voir le paragraphe 1.13). L'algorithme de Gram-Schmidt s'écrit alors

Algorithme de Gram-Schmidt (formulation géométrique)

```

 $q_1 = a_1 / \|a_1\|_2$ 
pour  $i = 1 : n - 1$ 
     $p_{i+1} = P_i a_{i+1}$ 
     $q_{i+1} = p_{i+1} / \|p_{i+1}\|_2$ 
fin

```

8.3.3 Complexité de G-S

La complexité de cet algorithme est donnée par le compte suivant du nombre d'opérations arithmétiques :

- $2m$ pour le calcul de r_{11} ,
- m pour le calcul de q_1 ,
- $2mi$ pour le calcul de $r_{k i+1}$,
- $2mi$ pour le calcul de p_{i+1} ,
- $2m$ pour le calcul de $r_{i+1 i+1}$,
- m pour le calcul de q_{i+1} .

On obtient un total de

$$3m + \sum_{i=1}^{n-1} (4mi + 3m) \approx 2mn^2 \text{ opérations arithmétiques.}$$

8.3.4 Instabilité numérique de G-S

Donnons un exemple numérique. On prend une matrice A de Vandermonde 10×10 (voir le paragraphe 16.4) définie à partir de 10 points aléatoires. Le conditionnement de A est important : $\text{cond}_2(A) = 1.0022 \cdot 10^8$. Soit $b \in \mathbb{R}^{10}$ un vecteur aléatoire. On résout le système $Ax = b$ grâce à la décomposition QR de A : $x = R^{-1}Q^*b$. Les résultats numériques obtenus par l'algorithme de Gram-Schmidt sont les suivants (\bar{x} désigne la solution approchée du système) :

- défaut d'orthonormalité $\|Q^*Q - I_{10}\|_2 = 7.1789 \cdot 10^{-4}$,
- norme de l'erreur $\|x - \bar{x}\|_2 = 71.7977$.

L'égalité $A = QR$ est par ailleurs satisfaite à l'ordre de l'unité d'arrondi $\approx 10^{-16}$.

On constate une perte significative d'orthogonalité des vecteurs q_i et la solution du système calculée grâce à cette décomposition est très différente de la solution exacte.

Remarque 8.3. La raison de cette instabilité numérique peut être comprise en raisonnant sur deux vecteurs a_1 et a_2 . Le procédé de Gram-Schmidt va générer $q_1 = a_1 / \|a_1\|_2$ puis le vecteur $p_2 = a_2 - \langle a_2, q_1 \rangle q_1$ et enfin $q_2 = p_2 / \|p_2\|_2$. Lorsque a_1 et a_2 sont deux vecteurs quasi proportionnels, les vecteurs a_2 et $\langle a_2, q_1 \rangle q_1$ sont à peu près égaux, leur différence sera donc petite et q_2 sera obtenu comme quotient de deux petites quantités d'où des instabilités numériques.

Prenons $a_1 = (-0.88061, -0.47384)^T$, $a_2 = (-0.881, -0.474)^T$ et calculons avec cinq décimales. On obtient : $q_1 = a_1$, $\langle a_2, q_1 \rangle = 1.0004$, $p_2 = (-0.00004, 0.00003)^T$ ce qui conduit à un angle $(\widehat{q_1}, p_2) = 66.4$ degrés bien loin des 90 degrés souhaités.

8.3.5 L'algorithme de Gram-Schmidt modifié

On peut remédier en partie à ce défaut en ordonnant différemment le calcul des vecteurs p_i . On décompose l'opérateur de projection orthogonale P_i de la manière suivante :

$$P_i = I_m - Q_i Q_i^* = (I_m - q_i q_i^*) \dots (I_m - q_1 q_1^*).$$

Il est facile de prouver que ces deux expressions de P_i sont égales. Cependant, du fait des erreurs d'arrondi, les deux formes ne sont pas numériquement équivalentes. Ce nouveau calcul donne lieu à l'algorithme de Gram-Schmidt modifié :

Algorithme de Gram-Schmidt modifié

```

 $q_1 = a_1 / \|a_1\|_2$ 
pour  $i = 1 : n - 1$ 
     $z = a_{i+1}$ 
    pour  $k = 1 : i$ 
         $z = z - \langle z, q_k \rangle q_k$ 
    fin
     $q_{i+1} = z / \|z\|_2$ 
fin

```

L'algorithme de Gram-Schmidt modifié appliqué au système considéré au paragraphe 8.3.4 donne le résultat suivant :

- défaut d'orthogonalité $\|Q^*Q - I_{10}\|_2 = 1.5229 \cdot 10^{-9}$,
- norme de l'erreur $\|x - \bar{x}\|_2 = 0.0970$.

L'orthogonalité des vecteurs est mieux satisfaite que par l'algorithme de Gram-Schmidt et la solution du système meilleure, bien qu'encore assez médiocre. On verra au paragraphe 8.5.3 que la décomposition QR obtenue grâce à la méthode de Householder donne de meilleurs résultats.

8.3.6 Procédé de réorthogonalisation

Une autre méthode fréquemment utilisée pour améliorer l'orthogonalité des colonnes de Q consiste à appliquer une seconde fois l'algorithme de Gram-Schmidt à la matrice Q que l'on vient de calculer et dont on a vu qu'elle ne vérifiait qu'imparfaitement l'égalité $Q^*Q = I_n$. On obtient la décomposition $Q = Q'R'$ et donc $A = Q'(R'R)$. La matrice $R'R$ est triangulaire supérieure et Q' est la matrice de Stiefel de cette

nouvelle décomposition QR de A . En général les colonnes de Q' vérifient les relations d'orthogonalité avec une bonne précision et il n'est pas nécessaire de poursuivre au-delà ce processus.

La complexité des calculs de cette méthode a plus que doublé par rapport à l'algorithme de Gram-Schmidt puisqu'il faut aussi considérer le produit des matrices triangulaires supérieures $R'R$.

Donnons une illustration de l'efficacité de cette méthode pour le système numérique déjà considéré :

- défaut d'orthonormalité $\|Q'^* Q' - I_{10}\|_2 = 6.7759 \cdot 10^{-16}$,
 - norme de l'erreur $\|x - \bar{x}\|_2 = 1.1819 \cdot 10^{-11}$.

Les résultats sont meilleurs que ceux obtenus par les transformations de Household (voir paragraphe 8.5.3).

8.4 ROTATIONS DE GIVENS

Nous n'allons considérer, dans ce paragraphe, que des matrices réelles, le cas complexe fait l'objet de l'exercice 8.8. Une *rotation de Givens* consiste en une rotation d'angle $-\theta$ dans le plan $Ox_i x_j$. Elle est donnée par $G(i, j, \theta)x = y$ avec

$$G(i, j, \theta) = \begin{pmatrix} & i & & j & \\ & | & & | & \\ 1 & & & & \\ \ddots & & & & \\ 1 & & & & \\ & \cos \theta & & \sin \theta & \\ & 1 & \dots & & \\ & \vdots & & \ddots & \vdots \\ & -\sin \theta & \dots & \cos \theta & \\ & & & 1 & \\ & & & & \ddots \\ & & & & 1 \end{pmatrix}$$

c'est-à-dire

$$y_k = \begin{cases} x_k & k \neq i, j, \\ x_i \cos \theta + x_j \sin \theta & k = i, \\ -x_i \sin \theta + x_j \cos \theta & k = j. \end{cases}$$

Il est clair qu'il s'agit là d'une transformation orthogonale. On peut forcer y_j à être nul en prenant

$$\cos \theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}} \text{ et } \sin \theta = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}.$$

Une succession de telles rotations permet d'appliquer un vecteur $a \in \mathbb{R}^m$ sur le vecteur

$$\begin{pmatrix} \pm \|a\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Il suffit, pour ce faire, d'effectuer le produit suivant de rotations de Givens :

$$G(1, 2, \theta_2)G(2, 3, \theta_3)\dots G(m-1, m, \theta_m)a$$

l'angle θ_k est choisi de façon à annuler la k -ième composante du vecteur

$$G(k, k+1, \theta_{k+1})\dots G(m-1, m, \theta_n)a$$

comme indiqué dans l'algorithme suivant :

Algorithme de Givens pour la transformation d'un vecteur

```

pour  $i = m : -1 : 2$ 
     $r = \sqrt{a_{i-1}^2 + a_i^2}$ 
     $a_{i-1} = r$ 
     $a_i = 0$ 
fin
  
```

Notons enfin que l'on peut contrôler le signe de la première coordonnée $\pm \|a\|_2$ en remplaçant θ par $\pi + \theta$.

Pour obtenir la décomposition QR d'une matrice $A \in \mathbb{R}^{m \times n}$ on applique la méthode précédente aux différentes colonnes de A comme indiqué dans le schéma suivant :

$$\begin{pmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{pmatrix} \xrightarrow{G(3,4)} \begin{pmatrix} \times & \times \\ \times & \times \\ \times & \times \\ 0 & \times \end{pmatrix} \xrightarrow{G(2,3)} \begin{pmatrix} \times & \times \\ \times & \times \\ 0 & \times \\ 0 & \times \end{pmatrix} \xrightarrow{G(1,2)} \begin{pmatrix} \times & \times \\ 0 & \times \\ 0 & \times \\ 0 & \times \end{pmatrix} \xrightarrow{G(3,4)}$$

$$\begin{pmatrix} \times & \times \\ 0 & \times \\ 0 & \times \\ 0 & 0 \end{pmatrix} \xrightarrow{G(2,3)} \begin{pmatrix} \times & \times \\ 0 & \times \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Noter que l'ordre choisi dans la composition des rotations de Givens ne perturbe pas les zéros déjà acquis. L'algorithme correspondant est le suivant :

Algorithme de Givens

```

pour  $j = 1 : n - 1$ 
    pour  $i = m : -1 : j - 1$ 
         $r = \sqrt{a_{i-1,j}^2 + a_{ij}^2}$ 
         $c = a_{i-1,j}/r$ 
         $s = a_{ij}/r$ 
         $a_{i-1,j} = r$ 
         $a_{ij} = 0$ 
        pour  $k = j + 1 : n$ 
             $a_{i-1,k} = ca_{i-1,k} + sa_{ik}$ 
             $a_{ik} = -sa_{i-1,k} + ca_{ik}$ 
        fin
    fin
fin

```

Ceci montre que

Théorème 8.5 Toute matrice $A \in \mathbb{R}^{m \times n}$ possède une décomposition $A = QR$ avec $R \in \mathbb{R}^{m \times n}$ triangulaire supérieure et $Q \in \mathbb{O}_m$, produit d'au plus $n(2m - n - 1)/2$ rotations de Givens. L'algorithme de Givens calcule la matrice R en $\approx 3mn^2 - n^3$ opérations arithmétiques.

Démonstration. Le calcul de r , c et s compte pour 6 opérations, puis 6 autres opérations dans la boucle k d'où un total de $6(n - j) + 6$ ops. La boucle sur i puis celle sur j conduisent à un total de

$$\sum_{j=1}^{n-1} (6(n - j) + 6)(m - j + 2) \approx 3mn^2 - n^3.$$

8.5 LA MÉTHODE DE HOUSEHOLDER

8.5.1 Symétries orthogonales

Définition 8.6 Soit $w \in \mathbb{C}^m$ un vecteur unitaire c'est-à-dire tel que

$$\|w\|_2^2 = \langle w, w \rangle = w^*w = 1.$$

On appelle matrice de Householder associée à w

$$H_w = I_m - 2ww^*.$$

Théorème 8.7 H_w est la symétrie orthogonale par rapport à l'hyperplan

$$\mathcal{H}_w = \{u \in \mathbb{C}^m : \langle u, w \rangle = 0\}.$$

Elle est hermitienne et unitaire : $H_w = H_w^* = H_w^{-1}$.

Démonstration. $H_w(w) = (I_m - 2ww^*)w = w - 2w(w^*w) = -w$ parce que $w^*w = 1$. De plus, pour tout $u \in \mathcal{H}_w$, $H_w(u) = (I_m - 2ww^*)u = u - 2w(w^*u) = u$ puisque $w^*u = \langle u, w \rangle = 0$. Ceci prouve que H_w est bien la symétrie orthogonale par rapport à l'hyperplan \mathcal{H}_w . Notons enfin que $H_w^* = (I_m - 2ww^*)^* = I_m - 2ww^* = H_w$ et $H_w^*H_w = (I_m - 2ww^*)^2 = I_m - 4ww^* + 4ww^*ww^* = I_m$ parce que $w^*w = 1$. Ceci prouve que la symétrie orthogonale par rapport à l'hyperplan \mathcal{H}_w est hermitienne et unitaire.

Étant donné deux vecteurs dans \mathbb{R}^m de même longueur, on peut les rabattre l'un sur l'autre par une symétrie orthogonale. En termes plus imagés, l'un est « l'image miroir » de l'autre. Pour des vecteurs complexes cela n'est vrai qu'à un changement d'argument près :

Théorème 8.8 Soient u_1 et $u_2 \in \mathbb{C}^m$ de mêmes normes et indépendants. Il existe $w \in \mathbb{C}^m$, unitaire, et un nombre complexe θ de module 1 tels que

$$H_w(u_1) = \theta u_2.$$

Démonstration. On prend

$$\theta = \exp(-i \arg(u_1^* u_2))$$

et

$$w = \frac{u_1 - \theta u_2}{\|u_1 - \theta u_2\|_2}.$$

L'indépendance de u_1 et u_2 fait que $u_1 - \theta u_2 \neq 0$. On a : $H_w(u_1) =$

$$u_1 - 2 \frac{(u_1 - \theta u_2)(u_1 - \theta u_2)^*}{(u_1 - \theta u_2)^*(u_1 - \theta u_2)} u_1 = u_1 - (u_1 - \theta u_2) \frac{u_1^* u_1 - \bar{\theta} u_2^* u_1}{u_1^* u_1 - \Re(\theta u_1^* u_2)}.$$

Le choix de θ fait de $\theta u_1^* u_2$ un nombre réel positif donc

$$\Re(\theta u_1^* u_2) = \theta u_1^* u_2 = \bar{\theta} u_2^* u_1$$

de sorte que

$$H_w(u_1) = u_1 - (u_1 - \theta u_2) = \theta u_2.$$

Lorsque les vecteurs u_1 et u_2 ne sont pas de même norme, il suffit de considérer le théorème précédent avec $u_1/\|u_1\|_2$ et $u_2/\|u_2\|_2$. On obtient le résultat suivant

Corollaire 8.9 *Soient u_1 et $u_2 \in \mathbb{C}^m$ indépendants. Il existe $w \in \mathbb{C}^m$ unitaire et un nombre complexe $k \neq 0$ tels que*

$$H_w(u_1) = ku_2.$$

Le cas réel se traite de façon similaire :

Théorème 8.10 *Soient u_1 et $u_2 \in \mathbb{R}^m$ de mêmes normes et indépendants. Posons*

$$w_+ = \frac{u_1 + u_2}{\|u_1 + u_2\|_2}$$

et

$$w_- = \frac{u_1 - u_2}{\|u_1 - u_2\|_2}.$$

Alors

$$H_{w_-}(u_1) = u_2 \text{ et } H_{w_+}(u_1) = -u_2.$$

Corollaire 8.11 *Soient u_1 et $u_2 \in \mathbb{R}^m$ indépendants. Il existe $w \in \mathbb{R}^m$, unitaire, et un nombre réel positif (resp. négatif) k tels que*

$$H_w(u_1) = ku_2.$$

8.5.2 QR via Householder

La méthode de Householder pour calculer la décomposition QR d'une matrice $A \in \mathbb{C}^{m \times n}$ repose sur la construction suivante :

Théorème 8.12 *Il existe une matrice unitaire $Q \in \mathbb{U}_m$ produit d'au plus n matrices de Householder ($n-1$ matrices si $m=n$) et une matrice triangulaire supérieure $R \in \mathbb{C}^{m \times n}$ à diagonale positive ou nulle telles que $A = QR$.*

Démonstration. Soit a_1 le premier vecteur-colonne de A et soit e_1 le premier vecteur de la base canonique de \mathbb{C}^m : $e_1 = (1, 0, \dots, 0)^T$. Si $a_1 = k e_1$, il n'y a rien à faire et l'on pose $H_1 = I_m$. Sinon, en vertu du corollaire 8.9, il existe une matrice de Householder H_1 et un scalaire $k_1 \neq 0$ tels que

$$H_1 A = \begin{pmatrix} k_1 & a'_{12} & \cdots & a'_{1n} \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix}.$$

Après j telles étapes on obtient

$$H_j \dots H_1 A = \begin{pmatrix} k_1 & & \cdots & \\ 0 & \ddots & & \\ & & k_j & \cdots \\ \vdots & & 0 & \\ & & \vdots & A_j \\ 0 & 0 & & \end{pmatrix}$$

où A_j est une matrice $(m-j) \times (n-j)$. Appliquons à A_j la même procédure que pour A : notons $\tilde{a}_{j+1} \in \mathbb{C}^{m-j}$ la première colonne de A_j et $\tilde{e}_1 \in \mathbb{C}^{m-j}$ le premier vecteur de la base canonique de \mathbb{C}^{m-j} . Si \tilde{a}_{j+1} est proportionnel à \tilde{e}_1 on pose $H_{j+1} = I_m$ et on passe à l'étape suivante. Sinon, on calcule une matrice de Householder $\tilde{H}_{j+1} \in \mathbb{C}^{(m-j) \times (m-j)}$ et un scalaire $k_{j+1} \in \mathbb{C}$ tels que

$$\tilde{H}_{j+1} \tilde{a}_{j+1} = k_{j+1} \tilde{e}_1.$$

À la matrice de Householder $\tilde{H}_{j+1} \in \mathbb{C}^{(m-j) \times (m-j)}$ nous associons la matrice $H_{j+1} \in \mathbb{C}^{m \times m}$ donnée par

$$H_{j+1} = \begin{pmatrix} I_j & 0 \\ 0 & \tilde{H}_{j+1} \end{pmatrix}.$$

Il est facile de voir que lorsque \tilde{H}_{j+1} est la matrice de Householder associée au vecteur $\tilde{w} \in \mathbb{C}^{m-j}$, H_{j+1} est la matrice de Householder associée au vecteur $w \in \mathbb{C}^m$ tel que

$$w = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \tilde{w} \end{pmatrix}.$$

La multiplication à gauche de $H_j \dots H_1 A$ par H_{j+1} ne modifie que le bloc A_j , qui est transformé en $\tilde{H}_{j+1} A_j$, et laisse inchangé le reste de la structure d'où

$$H_{j+1} H_j \dots H_1 A = \begin{pmatrix} k_1 & & \cdots & \\ 0 & \ddots & & \\ \vdots & & k_{j+1} & \cdots \\ 0 & & 0 & \\ & & \vdots & A_{j+1} \\ 0 & & 0 & \end{pmatrix}.$$

Le processus s'arrête après n telles étapes. On obtient alors une matrice triangulaire supérieure $H_n \dots H_1 A = R$ d'où $A = QR$ avec $Q = H_1 \dots H_n$. Lorsque $m = n$ la dernière étape n'est plus nécessaire parce que la matrice A_n est de taille 1×1 donc déjà triangulaire supérieure ; $n - 1$ matrices de Householder suffisent.

Remarque 8.4. Par construction des matrices de Householder H_j , les différentes colonnes q_j de la matrice $Q = H_1 \dots H_n$ sont données par $q_j = Q e_j = H_1 \dots H_j e_j$.

8.5.3 L'algorithme et sa complexité

La triangulation d'un système via la méthode de Householder est résumée dans l'algorithme suivant que nous donnons pour des matrices réelles. La matrice R obtenue a une diagonale positive ou nulle.

Décomposition QR par la méthode de Householder

```

pour  $j = 1 : n$ 
     $\alpha^2 = \sum_{i=j+1}^m |a_{ij}|^2$ 
    si  $\alpha \neq 0$ 
         $\beta = \sqrt{|a_{jj}|^2 + \alpha^2}$ 
         $w_j = a_{jj} - \beta$ 
        pour  $i = j + 1 : m$ 
             $w_i = a_{ij}$ 
        fin
         $H = I_{m-j+1} - 2 \frac{ww^T}{\|w\|_2^2}$ 
         $A(j : m, j : n) = HA(j : m, j : n)$ 
    fin
fin

```

Le calcul du produit de matrices $HA(j : m, j : n)$ demande en principe $2(m - j + 1)^2(n - j + 1)$ opérations arithmétiques. Compte tenu de la structure particulière de H , on peut l'effectuer en $\approx 4(m - j + 1)(n - j + 1)$ opérations. Il suffit de procéder de la façon suivante :

1. On calcule le produit vecteur ligne-matrice $w^T A(j : m, j : n)$: ceci nécessite $\approx 2(m - j + 1)(n - j + 1)$ opérations,
2. On évalue $v = -2 \frac{w}{\|w\|^2}$ puis $v (w^T A(j : m, j : n))$ ce qui demande $\approx (m - j + 1)(n - j + 1)$,
3. On ajoute le résultat obtenu à $A(j : m, j : n)$ d'où $(m - j + 1)(n - j + 1)$ opérations supplémentaires,
4. Comme $1 \leq j \leq n$ on obtient au total (les opérations oubliées dans ce compte ont un ordre de grandeur inférieur)

$$\sum_{j=1}^n 4(m - j + 1)(n - j + 1) \approx 2mn^2 - \frac{2}{3}n^3 \text{ opérations arithmétiques.}$$

Cet algorithme est donc moins coûteux que l'algorithme de Gram-Schmidt ($2mn^2$) mais il faut noter que ce dernier calcule les matrices Q et R à la fois alors que les algorithmes de Givens et de Householder ne fournissent que la matrice R .

Une évaluation possible de Q utilise la remarque 8.4 : les différentes colonnes q_j de la matrice $Q = H_1 \dots H_n$ sont données par $q_j = Qe_j = H_1 \dots H_j e_j$. Ce calcul

requiert $\approx 2mn^2 - \frac{2}{3}n^3$ opérations arithmétiques, mais il suppose que l'on stocke les matrices $H_1 \dots H_n$.

Une dernière possibilité consiste à effectuer le produit $Q^* = H_n \dots H_1$ ce qui évite de conserver les différentes matrices de Householder mises en oeuvre. Cette stratégie coûte $\approx 4(m^2n - mn^2 + n^3/3)$ opérations arithmétiques.

Considérons à nouveau l'exemple du paragraphe 8.3.4 en utilisant la décomposition QR par les matrices de Householder (fonction qr de Matlab). On obtient le résultat suivant :

- défaut d'orthonormalité $\|Q^* Q - I_{10}\|_2 = 1.1917 \cdot 10^{-15}$,
- norme de l'erreur $\|x - \bar{x}\|_2 = 1.4980 \cdot 10^{-8}$.

On constate l'excellente précision numérique des calculs obtenus avec la méthode de Householder.

8.6 RÉDUCTION À LA FORME DE HESSENBERG

Définition 8.13 Une matrice $H \in \mathbb{C}^{n \times n}$ est dite de Hessenberg lorsque $h_{ij} = 0$ pour tout $i > j + 1$.

Une matrice de Hessenberg H est de la forme

$$H = \begin{pmatrix} \times & \cdots & \cdots & \cdots & \times \\ \times & \ddots & & & \vdots \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \times & \times \end{pmatrix}. \quad (8.1)$$

On va montrer que toute matrice A est unitairement semblable à une matrice de Hessenberg. Cette décomposition peut être obtenue grâce aux matrices de Householder et de Givens déjà considérées pour la décomposition QR. Nous donnons ici la décomposition par les matrices de Householder.

Théorème 8.14 Étant donné une matrice $A \in \mathbb{C}^{n \times n}$, il existe une matrice unitaire Q produit d'au plus $n - 2$ matrices de Householder telle que $Q^* A Q$ soit une matrice de Hessenberg.

Démonstration. Si la première colonne a_1 de A est du type $a_1 = (a_{11}, a_{21}, 0, \dots, 0)^T$ alors on pose $H_2 = I_n$. Sinon, les vecteurs $\tilde{a}_2 = (a_{21}, \dots, a_{n1})^T$ et $\tilde{e}_1 = (1, 0, \dots, 0)^T \in \mathbb{C}^{n-1}$ sont indépendants

donc, par le corollaire 8.9, il existe une matrice de Householder $\tilde{H}_2 \in \mathbb{C}^{(n-1) \times (n-1)}$ et un scalaire k_1 tels que

$$\tilde{H}_2 \tilde{a}_2 = k_1 \tilde{e}_1.$$

On pose

$$H_2 = \begin{pmatrix} 1 & 0 \\ 0 & \tilde{H}_2 \end{pmatrix}$$

qui, comme nous l'avons vu, est aussi une matrice de Householder. On a

$$H_2 A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ k_1 & & & & \\ 0 & & & & \\ \vdots & & & & A_1 \\ 0 & & & & \end{pmatrix}.$$

La multiplication à droite de $H_2 A$ par H_2^* ne modifie pas la première colonne de $H_2 A$. On obtient la matrice

$$H_2 A H_2^* = \begin{pmatrix} a_{11} & \tilde{a}_{12} & \tilde{a}_{13} & \dots & \tilde{a}_{1n} \\ k_1 & & & & \\ 0 & & & & \\ \vdots & & & & \tilde{A}_1 \\ 0 & & & & \end{pmatrix}$$

On recommence ce même procédé sur la matrice \tilde{A}_1 et après $n - 2$ étapes on obtient

$$H_{n-1} \dots H_2 A H_2^* \dots H_{n-1}^* = H$$

où H est de Hessenberg (à ne pas confondre avec les matrices de Householder H_j). Il suffit de poser $Q = H_2^* \dots H_{n-1}^* = H_2 \dots H_{n-1}$ pour avoir la décomposition de Hessenberg $A = Q H Q^*$.

La forme algorithmique du théorème 8.14 est donnée ici pour des matrices réelles :

Forme Hessenberg par la méthode de Householder

```

pour  $j = 1 : n - 2$ 
 $\alpha^2 = \sum_{i=j+2}^n |a_{ij}|^2$ 
si  $\alpha \neq 0$ 
 $\beta = \sqrt{|a_{j+1,j}|^2 + \alpha^2}$ 
 $w_{j+1} = a_{j+1,j} - \beta$ 
pour  $i = j + 2 : n$ 
 $w_i = a_{ij}$ 
fin
 $H = I_{n-j} - 2 \frac{ww^T}{\|w\|_2^2}$ 
 $A(j+1:n, j:n) = HA(j+1:n, j:n)$ 
 $A(1:n, j+1:n) = A(1:n, j+1:n)H$ 
fin
fin

```

Le décompte des opérations est le suivant :

- $\approx \frac{4}{3}n^3$ opérations pour le produit des matrices $HA(j+1:n, j:n)$ (voir paragraphe 8.5.3),
- $\sum_{j=1}^{n-2} 4n(n-j) \approx 2n^3$ opérations pour le produit des matrices $A(1:n, j+1:n)H$,

soit $\approx \frac{10}{3}n^3$ opérations pour le calcul de la matrice de Hessenberg. Il faut ajouter à cela $\approx \frac{4}{3}n^3$ opérations pour le calcul de la matrice Q (voir paragraphe 8.5.3).

Remarque 8.5.

1. La matrice obtenue par cet algorithme est du type Hessenberg et, lorsque le cas $\alpha = 0$ ne s'est jamais présenté dans son déroulement, les entrées $h_{i+1,i}$ sont toutes positives.
2. La décomposition de Hessenberg peut aussi s'obtenir à l'aide des matrices de Givens.

8.7 TRIDIAGONALISATION D'UNE MATRICE HERMITIENNE

Définition 8.15 Une matrice T est tridiagonale¹ si $T_{ij} = 0$ lorsque $|i - j| > 1$.

La décomposition de Hessenberg d'une matrice hermitienne conduit à une matrice (hermitienne) tridiagonale. Nous donnons ici le procédé de tridiagonalisation qui utilise les matrices de Householder.

Théorème 8.16 Étant donné une matrice $A \in \mathbb{C}^{n \times n}$ hermitienne, il existe une matrice unitaire Q produisant d'au plus $n - 2$ matrices de Householder telle que QAQ^* soit une matrice tridiagonale hermitienne.

Démonstration. On procède comme dans la preuve du théorème 8.14. Le résultat de la multiplication à gauche de A par la matrice de Householder H_2 ne modifie pas la première ligne de A . Puisque la matrice A est hermitienne cette ligne est la conjuguée de la première colonne de A et $a_{11} = \bar{a}_{11}$. On a

$$H_2 A = \begin{pmatrix} a_{11} & \bar{a}_{21} & \bar{a}_{31} & \dots & \bar{a}_{n1} \\ k_1 & & & & \\ 0 & & & & \\ \vdots & & & A_1 & \\ 0 & & & & \end{pmatrix}.$$

En multipliant à droite par $H_2^* = H_2$ on a donc

$$H_2 A H_2^* = \begin{pmatrix} a_{11} & \bar{k}_1 & 0 & \dots & 0 \\ k_1 & & & & \\ 0 & & & & \\ \vdots & & & \tilde{A}_1 & \\ 0 & & & & \end{pmatrix}$$

où \tilde{A}_1 est elle-même hermitienne. On recommence avec \tilde{A}_1 ce qui vient d'être fait avec A et après $n - 2$ telles étapes on a :

$$H_{n-2} \dots H_2 A H_2^* \dots H_{n-2}^* = T$$

tridiagonale hermitienne. Il suffit de poser $Q = H_2^* \dots H_{n-2}^* = H_2 \dots H_{n-2}$ pour avoir la décomposition $A = QTQ^*$.

Cette décomposition s'obtient également grâce aux transformations de Givens.

1. D'après la définition générale (voir exercice 6.5) une matrice tridiagonale est une matrice bande de largeur 3.

8.8 L'ALGORITHME D'ARNOLDI

La décomposition $A = QHQ^*$ sous la forme de Hessenberg n'est pas unique. Déterminons le nombre de variables réelles indépendantes d'une matrice $Q \in \mathbb{U}_n$ unitaire et d'une matrice $H \in \mathbb{C}^{n \times n}$ de Hessenberg.
Pour $Q \in \mathbb{U}_n$ nous obtenons le décompte suivant :

- $\dim_{\mathbb{R}} \mathbb{C}^{n \times n} = 2n^2$,
- $n(n - 1)/2$ conditions d'orthogonalité et donc $n(n - 1)$ équations réelles,
- n conditions de normalité et donc n équations réelles.

Cette heuristique « montre que » $\dim_{\mathbb{R}} \mathbb{U}_n = 2n^2 - n(n - 1) - n = n^2$.² Pour une matrice $H \in \mathbb{C}^{n \times n}$ de Hessenberg, on a $(1+2+\dots+n)+(n-1)$ coefficients complexes et donc $n^2 + 3n - 2$ coefficients réels. Sachant que $A \in \mathbb{C}^{n \times n}$ admet $2n^2$ coefficients réels, nous disposons donc de $3n - 2$ paramètres réels arbitraires. Cette possibilité est utilisée par l'algorithme d'Arnoldi qui calcule une décomposition de Hessenberg de A ayant la première colonne de Q fixée ($2n - 1$ paramètres réels) et les arguments des coefficients $h_{j+1,j}$ fixés, par exemple $h_{j+1,j} > 0$ ($n - 1$ paramètres réels).

Considérons un vecteur unitaire $q_1 \in \mathbb{C}^n$. Nous souhaitons construire une matrice unitaire $Q = (q_1 q_2 \dots q_n)$ ayant q_1 pour première colonne et une matrice H de Hessenberg à sous-diagonale positive ($h_{i+1,i} > 0$ pour tout $i = 1 \dots n - 1$) telles que $A = QHQ^*$, c'est-à-dire

$$AQ = QH. \quad (8.2)$$

La première colonne de (8.2) montre que

$$Aq_1 = h_{11}q_1 + h_{21}q_2$$

on a donc $h_{21}q_2 = Aq_1 - h_{11}q_1$. La condition d'orthogonalité $\langle q_1, q_2 \rangle = 0$ permet de déterminer

$$h_{11} = \langle Aq_1, q_1 \rangle = q_1^* Aq_1.$$

Par ailleurs, la condition $\|q_2\|_2 = 1$ implique

$$|h_{21}| = \|Aq_1 - h_{11}q_1\|_2.$$

Si $Aq_1 - h_{11}q_1 \neq 0$ est distinct de zéro, on peut alors choisir

$$h_{21} = \|Aq_1 - h_{11}q_1\|_2 > 0$$

2. Il s'agit là de la dimension de \mathbb{U}_n en tant que sous-variété différentiable de \mathbb{R}^{2n^2} . Nous admettons ici que les relations d'orthonormalité $Q^* Q = I_n$ sont indépendantes.

126

$$q_2 = (Aq_1 - h_{11}q_1)/h_{21}.$$

et

On aurait pu tout aussi bien prendre $h_{21} = z\|Aq_1 - h_{11}q_1\|_2$ avec $z \in \mathbb{C}$ de module 1. Supposons avoir construit les j premières colonnes de Q et de H avec les conditions requises. La relation (8.2) appliquée à la colonne j donne

$$Aq_j = \sum_{i=1}^{j+1} h_{ij}q_i$$

et l'on obtient

$$h_{j+1,j} q_{j+1} = Aq_j - \sum_{i=1}^j h_{ij}q_i. \quad (8.3)$$

Puisque les vecteurs q_i ($i = 1, \dots, j$) sont orthonormés, les coefficients h_{ij} ($i = 1, \dots, j$) sont déterminés par les contraintes d'orthogonalité $\langle q_{j+1}, q_i \rangle = 0$, ce qui donne

$$h_{ij} = \langle Aq_j, q_i \rangle = q_i^* Aq_j.$$

Si $Aq_j - \sum_{i=1}^j h_{ij}q_i \neq 0$ alors on choisit

$$h_{j+1,j} = \|Aq_j - \sum_{i=1}^j h_{ij}q_i\|_2 > 0$$

et

$$q_{j+1} = (Aq_j - \sum_{i=1}^j h_{ij}q_i)/h_{j+1,j}.$$

Si le processus se poursuit jusqu'à l'étape $n-1$, c'est-à-dire si $Aq_j - \sum_{i=1}^j h_{ij}q_i \neq 0$ pour $j = 1, \dots, n-1$, alors (q_1, \dots, q_n) est une base orthonormée de \mathbb{C}^n . Dans cette base, écrivons $Aq_n = \sum_{i=1}^n h_{in}q_i$. Les coefficients h_{in} sont encore donnés par $h_{in} = q_i^* Aq_n$ et la matrice H est enfin déterminée. On a ainsi défini l'algorithme d'Arnoldi :

Algorithme d'Arnoldi

Entrée : $A, H \in \mathbb{C}^{n \times n}$, $H = 0$, $q_1 \in \mathbb{C}^n$ de norme 1

```

pour  $j = 1 : n - 1$ 
     $z = Aq_j$ 
     $p_{j+1} = z$ 
    pour  $i = 1 : j$ 
         $h_{ij} = q_i^* z$ 
         $p_{j+1} = p_{j+1} - h_{ij} q_i$ 
    fin
     $h_{j+1,j} = \|p_{j+1}\|_2$ 
    si  $h_{j+1,j} = 0$ 
         $k = j$ 
        stop
    fin
     $q_{j+1} = p_{j+1}/h_{j+1,j}$ 
fin
 $k = n$ 
pour  $i = 1 : n$ 
     $h_{in} = q_i^* Aq_n$ 
fin

```

Sortie : k , $Q_k = (q_1 \dots q_k) \in \mathbb{S}t_{nk}$, $H_k \in \mathbb{C}^{k \times k}$ de Hessenberg avec $h_{j+1,j} > 0$.

D'une manière générale, lorsque l'algorithme s'arrête

- Si $k \leq n - 1$ et $h_{k+1,k} = 0$, on a $AQ_k = Q_k H_k$ avec $Q_k \in \mathbb{S}t_{nk}$, $H_k \in \mathbb{C}^{k \times k}$ de Hessenberg et $AQ_j = Q_{j+1} \tilde{H}_j$ pour tout $j < k$ avec $\tilde{H}_j = H_k(1:j+1, 1:j)$. Les colonnes de Q_k engendrent un sous-espace de \mathbb{C}^n , de dimension k , invariant par A .
- Si $k = n - 1$ et $h_{k+1,k} \neq 0$, on a obtenu à la fin de la boucle k l'égalité $AQ_{n-1} = Q_n \tilde{H}_{n-1}$ avec $\tilde{H}_{n-1} \in \mathbb{C}^{n \times (n-1)}$ que l'on transforme en $AQ_n = Q_n H_n$ au travers de la dernière boucle de l'algorithme.

Nous résumons les propriétés que nous venons de décrire dans la

Proposition 8.17 *L'algorithme d'Arnoldi calcule un entier $k \leq n$, une matrice de Stiefel $Q_k \in \mathbb{S}t_{nk}$ et une matrice de Hessenberg $H_k \in \mathbb{C}^{k \times k}$ telles que $AQ_k = Q_k H_k$*

Algorithme d' Arnoldi modifié

Entrée : $A, H \in \mathbb{C}^{n \times n}$, $H = 0$, $q_1 \in \mathbb{C}^n$ de norme 1

```

pour  $j = 1 : n - 1$ 
     $z = Aq_j$ 
    pour  $i = 1 : j$ 
         $h_{ij} = q_i^* z$ 
         $z = z - h_{ij} q_i$ 
    fin
     $h_{j+1,j} = \|z\|_2$ 
    si  $h_{j+1,j} = 0$ 
         $k = j$ 
        stop
    fin
     $q_{j+1} = z/h_{j+1,j}$ 
fin
 $k = n$ 
pour  $i = 1 : n$ 
     $h_{in} = q_i^* Aq_n$ 
fin

```

Sortie : k , $Q_k = (q_1 \dots q_k) \in \mathbb{S}t_{nk}$, $H_k \in C^{k \times k}$ de Hessenberg avec $h_{j+1,j} > 0$.

Définition 8.18 Une matrice de Hessenberg H dont les coefficients $h_{j+1,j}$ sont distincts de zéro est dite non réduite. La matrice H_k donnée par l'algorithme d'Arnoldi en est un exemple.

La complexité de l'algorithme d'Arnoldi est celle de l'algorithme de Gram-Schmidt auquel il faut ajouter le coût des produits Aq_j , $j = 1, \dots, k$. Chaque produit nécessite $2n^2$ opérations. L'algorithme d'Arnoldi requiert donc $\approx 2nk^2 + 2n^2k$ opérations arithmétiques.

8.9 L'ALGORITHME DE LANCZOS

Lorsque la matrice A est hermitienne, la décomposition de Hessenberg $A = QHQ^*$ montre que la matrice H est également hermitienne et donc tridiagonale. Notons T

et $h_{j+1,j} > 0$ pour tout $j = 1, \dots, k-1$. Les colonnes q_j de Q_k engendrent un sous-espace de dimension k de \mathbb{C}^n invariant par A . On a les égalités

$$\begin{aligned} A Q_j &= Q_{j+1} \tilde{H}_j \\ A Q_j &= Q_j H_j + h_{j+1,j} q_{j+1} e_j^T \\ Q_j^* A Q_j &= H_j \end{aligned} \quad (8.4)$$

pour tout $j < k$, en notant $Q_j = Q_k(1:n, 1:j)$, $H_j = H_k(1:j, 1:j)$, $\tilde{H}_j = H_k(1:j+1, 1:j)$ et e_j le j -ième vecteur de base de \mathbb{R}^j .

Cet algorithme est un cas particulier d'algorithme de Gram-Schmidt : il s'agit de l'orthonormalisation de la base $(q_1, Aq_1, Aq_2, \dots, Aq_{k-1})$ où chaque q_j est construit grâce aux vecteurs q_1, \dots, q_{j-1} et Aq_{j-1} . À ce titre on observe les mêmes défauts numériques que dans l'algorithme de Gram-Schmidt : lorsque j croît, les vecteurs q_j calculés ne vérifient pas la contrainte d'orthogonalité de manière satisfaisante. On modifie les calculs de la même façon que dans l'algorithme de Gram-Schmidt (paragraphe 8.3.5). L'égalité (8.3) s'écrit

$$h_{j+1,j} q_{j+1} = P_j A q_j$$

où $P_j = (I_n - Q_j Q_j^*)$ et $Q_j = (q_1 \dots q_j) \in \mathbb{S}t_{nj}$. Le projecteur orthogonal P_j admet la décomposition

$$P_j = \prod_{i=1}^j (I_n - q_i q_i^*)$$

et les matrices $(I_n - q_i q_i^*)$ commutent entre elles. Nous obtenons ainsi l'algorithme d'Arnoldi modifié.

Algorithme d' Arnoldi modifié

Entrée : $A, H \in \mathbb{C}^{n \times n}$, $H = 0$, $q_1 \in \mathbb{C}^n$ de norme 1

```

pour  $j = 1 : n - 1$ 
     $z = Aq_j$ 
    pour  $i = 1 : j$ 
         $h_{ij} = q_i^* z$ 
         $z = z - h_{ij} q_i$ 
    fin
     $h_{j+1,j} = \|z\|_2$ 
    si  $h_{j+1,j} = 0$ 
         $k = j$ 
        stop
    fin
     $q_{j+1} = z/h_{j+1,j}$ 
fin
 $k = n$ 
pour  $i = 1 : n$ 
     $h_{in} = q_i^* Aq_n$ 
fin

```

Sortie : k , $Q_k = (q_1 \dots q_k) \in \mathbb{S}t_{nk}$, $H_k \in C^{k \times k}$ de Hessenberg avec $h_{j+1,j} > 0$.

Définition 8.18 Une matrice de Hessenberg H dont les coefficients $h_{j+1,j}$ sont distincts de zéro est dite non réduite. La matrice H_k donnée par l'algorithme d'Arnoldi en est un exemple.

La complexité de l'algorithme d'Arnoldi est celle de l'algorithme de Gram-Schmidt auquel il faut ajouter le coût des produits Aq_j , $j = 1, \dots, k$. Chaque produit nécessite $2n^2$ opérations. L'algorithme d'Arnoldi requiert donc $\approx 2nk^2 + 2n^2k$ opérations arithmétiques.

8.9 L'ALGORITHME DE LANCZOS

Lorsque la matrice A est hermitienne, la décomposition de Hessenberg $A = QHQ^*$ montre que la matrice H est également hermitienne et donc tridiagonale. Notons T

cette matrice

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \bar{\beta}_1 & \alpha_2 & \beta_2 & \\ & \ddots & \ddots & \ddots \\ & & \bar{\beta}_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \bar{\beta}_{n-1} & \alpha_n \end{pmatrix}.$$

Nous allons suivre la même démarche qu'au paragraphe précédent en considérant les relations existant entre les vecteurs-colonne des matrices A et Q déduites de la relation

$$AQ = TQ \quad (8.5)$$

et les propriétés d'orthogonalité des colonnes de Q .

On suppose que le vecteur unitaire q_1 est donné. La première colonne de l'égalité

(8.5) donne

$$Aq_1 = \alpha_1 q_1 + \bar{\beta}_1 q_2$$

d'où l'on déduit $\bar{\beta}_1 q_2 = Aq_1 - \alpha_1 q_1$. Le vecteur $Aq_1 - \alpha_1 q_1$ colinéaire à q_2 doit être perpendiculaire à q_1 . Le choix du coefficient α_1 est imposé par cette contrainte : on a $\langle Aq_1 - \alpha_1 q_1, q_1 \rangle = 0$ et donc $\alpha_1 = q_1^* Aq_1$. Par ailleurs, $\|q_2\|_2 = 1$ implique $|\bar{\beta}_1| = \|Aq_1 - \alpha_1 q_1\|_2$. Si $\|Aq_1 - \alpha_1 q_1\|_2$ est distinct de zéro, on peut alors choisir $\beta_1 = \|Aq_1 - \alpha_1 q_1\|_2 > 0$ et $q_2 = (Aq_1 - \alpha_1 q_1)/\beta_1$. De cette dernière égalité on déduit aussi que $q_2^* q_2 = 1 = q_2^*(Aq_1 - \alpha_1 q_1)/\beta_1 = q_2^* Aq_1/\beta_1$ et donc $\beta_1 = q_2^* Aq_1 = q_1^* Aq_2$ puisque A est hermitienne et β_1 réel.

Plus généralement, pour les colonnes successives $j = 2, \dots, n-1$, on a

$$Aq_j = \beta_{j-1} q_{j-1} + \alpha_j q_j + \bar{\beta}_j q_{j+1},$$

que l'on écrit sous forme d'une récurrence à trois termes :

$$\bar{\beta}_j q_{j+1} = Aq_j - \beta_{j-1} q_{j-1} - \alpha_j q_j. \quad (8.6)$$

Supposons que les vecteurs $q_i, i = 1, \dots, j$ soient orthonormés et que $\beta_{j-1} \in \mathbb{R}$ vérifie $\beta_{j-1} = q_j^* Aq_{j-1} = q_{j-1}^* Aq_j$. Le produit scalaire par q_j des deux membres de l'égalité (8.6) donne $\alpha_j = q_j^* Aq_j$. Le produit par q_{j-1} donne $\beta_{j-1} = q_{j-1}^* Aq_j$ qui est déjà vérifié par hypothèse. En considérant la norme, on obtient

$$|\beta_j| = \|Aq_j - \beta_{j-1} q_{j-1} - \alpha_j q_j\|_2.$$

Si $\|Aq_j - \beta_{j-1} q_{j-1} - \alpha_j q_j\|_2 \neq 0$ on pose $\beta_j = \|Aq_j - \beta_{j-1} q_{j-1} - \alpha_j q_j\|_2$ et

$$q_{j+1} = (Aq_j - \beta_{j-1} q_{j-1} - \alpha_j q_j)/\beta_j.$$

Cette égalité donne en outre $\beta_j = q_{j+1}^* A q_j = q_j^* A q_{j+1}$. On poursuit ainsi le calcul tant que β_j est distinct de zéro.
De cette récurrence, nous déduisons l'algorithme de Lanczos qui calcule les vecteurs q_j à partir d'un vecteur q_1 .

Algorithme de Lanczos

Entrée : $A, T \in \mathbb{C}^{n \times n}$, $T = 0$, $q_1 \in \mathbb{C}^n$ de norme 1, $q_0 = 0$, $\beta_0 = 0$

```

pour  $j = 1 : n - 1$ 
     $z = Aq_j$ 
     $\alpha_j = q_j^* z$ 
     $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$ 
     $\beta_j = \|z\|_2$ 
    si  $\beta_j = 0$ 
         $k = j$ 
        stop
    fin
     $q_{j+1} = z / \beta_j$ 
fin
 $k = n$ 
 $\alpha_n = q_n^* A q_n$ 
fin
```

Sortie : k , $Q_k = (q_1 \dots q_k) \in \mathbb{S}t_{nk}$, $T_k = T(1:k, 1:k) \in \mathbb{C}^{k \times k}$ tridiagonale

La complexité de l'algorithme de Lanczos est dominée par les produits $Aq_j, j = 1, \dots, k$ ce qui donne $\approx 2n^2k$ opérations.

Remarque 8.6. Une récurrence à trois termes se rencontre aussi dans la construction des polynômes orthogonaux tels que, par exemple, les polynômes de Legendre, de Chebyshev ou de Jacobi utilisés dans les formules de quadrature de Gauss. L'orthogonalité des polynômes y est donnée au sens du produit scalaire

$$\langle f, g \rangle = \int_I f(t)g(t)\omega(t) dt,$$

où I est un intervalle et $\omega : I \rightarrow]0, +\infty[$ une fonction poids.

3.3 Splitting

CHAPITRE 7

Méthodes itératives de relaxation

7.1. Les méthodes itératives

7.1.1 Introduction

Nous allons étudier les méthodes itératives pour résoudre un système linéaire $Ax = b$.

Une méthode itérative engendre une suite de vecteurs qui doit tendre vers la solution de l'équation $Ax = b$.

Dans les méthodes de Jacobi et de Gauss-Seidel, le passage d'un vecteur $x^{(k)}$ de la suite au suivant se fait en corrigeant successivement (ou simultanément) une (ou plusieurs) composante de ce vecteur. Ces corrections s'effectuent en annulant une (ou plusieurs) composante du résidu $r^{(k)} = b - Ax^{(k)}$.

Après avoir introduit ces deux méthodes classiques par points (ou par blocs), on généralisera la méthode de Gauss-Seidel pour obtenir les méthodes de relaxation.

Ensuite, on étudiera la convergence de ces méthodes, notamment pour deux grandes classes de matrices : celles qui sont à diagonale dominante et celles qui sont hermitiennes et définies positives.

Un dernier paragraphe sera consacré aux méthodes de directions alternées.

Ces méthodes itératives ont en commun que chaque itération nécessite un nombre d'opérations de même ordre de grandeur que celui nécessaire à effectuer le produit de la matrice par un vecteur. Il n'y a pas de problème de stockage.

7.1.2 La méthode de Jacobi

Soit $A = (a_{ij})$ une matrice complexe régulière et d'ordre N .

Soit \bar{x} la solution de l'équation $Ax = b$ où b est un vecteur donné de \mathbb{C}^N .

A partir d'un vecteur $x^{(0)} \in \mathbb{C}^N$, on construit la suite de vecteurs $\{x^{(k)}\}$ de la manière suivante :

pour $i = 1 \text{ à } N$, on calcule :

$$(1) \quad x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k)} \right).$$

Cette méthode de Jacobi n'est définie que si $a_{ii} \neq 0$ pour $i = 1 \text{ à } N$.

On peut encore écrire (1) en retranchant $x_i^{(k)}$ aux deux membres de la relation précédente, soit :

$$(2) \quad x_i^{(k+1)} - x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^N a_{ij} x_j^{(k)} \right) = \frac{r_i^{(k)}}{a_{ii}},$$

où $r_i^{(k)}$ est la $i^{\text{ème}}$ composante du vecteur $r^{(k)}$, vecteur résidu à la $k^{\text{ème}}$ itération, c'est-à-dire :

$$\begin{aligned} r^{(k)} &= b - Ax^{(k)}, \\ \text{Posons :} \quad \delta x^{(k+1)} &= x^{(k+1)} - x^{(k)}, \end{aligned}$$

alors (2) devient : $\delta x_i^{(k+1)} = \frac{r_i^{(k+1)}}{a_{ii}}$ pour $i = 1 \text{ à } N$.

Cherchons à écrire ces relations sous forme vectorielle.

Pour cela, on décompose A en une somme de trois matrices $A = D - E - F$ de la manière suivante :

$$A = \begin{bmatrix} & & -F \\ & D & \\ -E & & \end{bmatrix},$$

où :

$-D$ est la partie diagonale de A :

$$(D)_{ii} = a_{ii} \quad \text{pour } i = 1 \text{ à } N,$$

$$(D)_{ij} = 0 \quad \text{pour } i \neq j,$$

$-E$ est la partie strictement triangulaire inférieure :

$$(-E)_{ij} = a_{ij} \quad \text{pour } i > j,$$

$$= 0 \quad \text{pour } i \leq j,$$

$-F$ est la partie strictement triangulaire supérieure :

$$42 \quad (-F)_{ij} = a_{ij} \quad \text{pour } i < j,$$

$$= 0 \quad \text{pour } i \geq j.$$

Alors, comme on a supposé $a_{ii} \neq 0$ pour $i = 1$ à N , D est inversible. La relation (2) s'écrit :

$$\delta x^{(k+1)} = D^{-1} r^{(k)},$$

soit :

$$x^{(k+1)} - x^{(k)} = D^{-1} (b - Ax^{(k)}),$$

(4)

$$x^{(k+1)} = D^{-1} (E + F) x^{(k)} + D^{-1} b$$

Comme $E + F = D - A$, on obtient encore :

$$x^{(k+1)} = (I - D^{-1} A) x^{(k)} + D^{-1} b$$

Voici un exemple de la programmation d'une itération de l'algorithme,

```

    POUR I = 1 à N FAIRE
        S := B(I)
        POUR J = 1 à N FAIRE
            S := S - A(I,J) * X(J)
        FIN DE BOUCLE J
        Y(I) := X(I) + S/A(I,I)
    FIN DE BOUCLE I
    POUR I = 1 à N FAIRE
        X(I) := Y(I)
    FIN DE BOUCLE I

```

Remarque 1

La programmation précédente a pour base la relation (2).
On peut diminuer le nombre d'opérations en programmant la relation (1).

```

    ┌── POUR I = 1 à N FAIRE
    │   S := B(I)
    │   ┌── POUR J = 1 à I - 1 FAIRE
    │   │   S := S - A(I,J) * X(J)
    │   └── FIN DE BOUCLE J
    │   ┌── POUR J = I + 1 à N FAIRE
    │   │   S := S - A(I,J) * X(J)
    │   └── FIN DE BOUCLE J
    │   Y(I) := S/A(I,I)
    └── FIN DE BOUCLE I
    ┌── POUR I = 1 à N FAIRE
    │   X(I) := Y(I)
    └── FIN DE BOUCLE I

```

Exemple 2

Soit à résoudre l'équation :

$$\begin{bmatrix} 10 & 1 \\ 2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \end{bmatrix} \text{ dont la solution est } x_1 = 1, x_2 = 1.$$

Initialisons avec le vecteur $x^{(0)} = 0$.

On obtient successivement :

$$x^{(1)} = \begin{bmatrix} \frac{11}{10} \\ \frac{12}{10} \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} \frac{98}{100} \\ \frac{98}{100} \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} \frac{1002}{1000} \\ \frac{1004}{1000} \end{bmatrix}, \quad x^{(4)} = \begin{bmatrix} \frac{9996}{10000} \\ \frac{9992}{10000} \end{bmatrix}.$$

Soit à résoudre l'équation :

$$\begin{bmatrix} 1 & 10 \\ 10 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \end{bmatrix} \text{ dont la solution est } x_1 = 1, x_2 = 1.$$

Pour $x^{(0)} = 0$, on obtient :

$$x^{(1)} = \begin{bmatrix} 11 \\ 6 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} -49 \\ -49 \end{bmatrix}, \quad x^{(3)} = \begin{bmatrix} 501 \\ 251 \end{bmatrix}, \quad x^{(4)} = \begin{bmatrix} -2499 \\ -2499 \end{bmatrix}.$$

On constate à travers ces deux exemples que la suite engendrée par la méthode de Jacobi peut se "rapprocher" de la solution ou au contraire s'en "éloigner". Il nous faudra étudier la convergence de la méthode.

7.1.3 La méthode de Gauss-Seidel

Au lieu d'attendre une itération entière pour corriger chaque composante, on peut le faire au fur et à mesure.

Supposons qu'à l'intérieur de la $(k+1)$ ème itération, on ait déjà obtenu $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{i-1}^{(k+1)}$, alors en supposant toujours $a_{ii} \neq 0$, on calcule :

$$(6) \quad x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^N a_{ij} x_j^{(k)} \right).$$

Comme précédemment, en retranchant $x_i^{(k)}$ aux deux nombres, on obtient :

$$(7) \quad \delta x_i^{(k+1)} = x_i^{(k+1)} - x_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^N a_{ij} x_j^{(k)} \right).$$

En écrivant (6) sous la forme :

$$\sum_{j=1}^i a_{ij} x_j^{(k+1)} = b_i - \sum_{j=i+1}^N a_{ij} x_j^{(k)},$$

on obtient en écriture vectorielle :

$$(D - E) x^{(k+1)} = b + F x^{(k)},$$

ou encore :

$$(8) \quad x^{(k+1)} = (D - E)^{-1} F x^{(k)} + (D - E)^{-1} b.$$

Voici la programmation d'une itération de l'algorithme :

```

    - POUR I = 1 à N FAIRE
        S := 0
        - POUR J = 1 à N FAIRE
            S := S + A(I,J) * X(J)
        FIN DE BOUCLE J
        R := B(I) - S
        X(I) := X(I) + R/A(I,I)
    FIN DE BOUCLE I

```

Remarque 3

Dans la méthode de Gauss-Seidel, on n'a pas besoin de deux tableaux X et Y comme dans la méthode de Jacobi. D'où un gain en stockage.

Très souvent la méthode de Gauss-Seidel convergera plus rapidement que celle de Jacobi car on utilise les nouvelles valeurs des composantes dès qu'elles sont calculées.

On remarque que la méthode de Jacobi peut s'écrire (cf. (3))

$$Dx^{(k+1)} = (b - Ax^{(k)} + Dx^{(k)}) = r^{(k)} + Dx^{(k)}.$$

Nous étudierons en détail ce type de méthode faisant intervenir le résidu au chapitre suivant.

Par contre, dans la méthode de Gauss-Seidel $\delta x^{(k+1)}$ ne s'exprime pas directement en fonction du résidu de $x^{(k)}$.

Pourtant, en faisant intervenir les vecteurs $\tilde{x}^{(k,i)}$ intermédiaires entre $x^{(k)}$ et $x^{(k+1)}$, soit :

$$\tilde{x}_j^{(k,i)} = \begin{cases} x_j^{(k+1)} & \text{pour } j < i, \\ x_j^{(k)} & \text{pour } j \geq i, \end{cases}$$

on obtient, $r^{(k,i)} = b - A\tilde{x}^{(k,i)}$ de telle sorte que :

$$\delta x_i^{(k+1)} = \frac{r_i^{(k,i)}}{a_{ii}}.$$

Alors comme $r_i^{(k+1)} = b_i - \sum_{j=1}^N a_{ij} x_j^{(k+1)}$,

on a :

$$\begin{aligned} r_i^{(k+1)} - r_i^{(k,i)} &= - \sum_{j=i}^N a_{ij} (x_j^{(k+1)} - x_j^{(k)}) = - \sum_{j=i}^N \frac{a_{ij}}{a_{jj}} r_j^{(k,i)} \\ &= - r_i^{(k,i)} - \sum_{j=i+1}^N \frac{a_{ij}}{a_{jj}} r_j^{(k,i)}. \end{aligned}$$

Donc :

$$r_i^{(k+1)} = - \sum_{j=i+1}^N \frac{a_{ij}}{a_{jj}} r_j^{(k,i)},$$

ce qui s'écrit encore en posant $\tilde{r}_j^{(k)} = r_j^{(k,i)}$,

$$r^{(k+1)} = F D^{-1} \tilde{r}^{(k)}$$

Remarque 4

Si on renomme les équations et les inconnues en effectuant une même permutation d'indices, la méthode de Jacobi engendrera les mêmes $x^{(k+1)}$ (cf. lemme 36).

Il n'en est pas de même en général avec la méthode de Gauss-Seidel. Certains choix peuvent être meilleurs que d'autres. Nous étudierons cela dans le cadre d'un problème modèle. (cf. paragraphe 6).

Les caractéristiques de ces deux méthodes itératives sont les suivantes :

1/ Le nombre d'opérations est pratiquement le même que celui du calcul de Ax pour x donné ce qui est avantageux si la matrice est creuse. Par contre, on ne tire aucun profit lorsque A est une matrice symétrique.

2/ On stocke en mémoire A, b et un vecteur (deux pour la méthode de Jacobi). Pour stocker A , il suffit d'enregistrer les termes non nuls. Dans certains cas particuliers, par exemple l'équation de Poisson discrétisée par différences finies, la règle de formation des équations est suffisante.

3/ Dans le cas où les méthodes convergent, il faudra que celles-ci soient suffisamment rapides pour que le nombre d'itérations ne soit pas trop grand.

4/ Ces méthodes sont très simples à programmer.

7.1.4 Les méthodes de relaxation

On peut généraliser les deux méthodes précédentes, Jacobi et Gauss-Seidel, en introduisant un paramètre réel ω . Soit $x_i^{(k)}$ calculé et $\hat{x}_i^{(k+1)}$ obtenu à partir de $x^{(k)}$ par l'une des deux méthodes précédentes. On définit alors la combinaison linéaire :

(9)

$$x_i^{(k+1)} = \omega \hat{x}_i^{(k+1)} + (1 - \omega) x_i^{(k)}$$

Si la méthode de base est celle de Jacobi, on obtient pour $i = 1 \text{ à } N$

(10)

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k)} \right) + (1 - \omega) x_i^{(k)}$$

Soit en multipliant par a_{ii} :

$$a_{ii} x_i^{(k+1)} = \omega \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^N a_{ij} x_j^{(k)} \right) + (1 - \omega) a_{ii} x_i^{(k)}$$

ou, en notation matricielle :

$$\begin{aligned} D x^{(k+1)} &= (1 - \omega) D x^{(k)} + \omega (b + (E+F) x^{(k)}) , \\ D x^{(k+1)} &= D x^{(k)} + \omega (-D+E+F) x^{(k)} + \omega b , \end{aligned}$$

d'où :

(11)

$$x^{(k+1)} = (1 - \omega D^{-1} A) x^{(k)} + \omega D^{-1} b$$

De même, en explicitant (9) dans le cas de la méthode de Gauss-Seidel, on construit la méthode appelée *méthode de relaxation* (S.O.R. en langue anglaise pour *successive over relaxation*).

Pour $i = 1 \text{ à } N$:

(12a)

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^N a_{ij} x_j^{(k)} \right) + (1 - \omega) x_i^{(k)}$$

(12b)

$$x_i^{(k+1)} - x_i^{(k)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^N a_{ij} x_j^{(k)} \right)$$

(12c)

$$x_i^{(k+1)} - x_i^{(k)} = \frac{\omega}{a_{ii}} r_i^{(k,i)},$$

soit : $a_{ii} x_i^{(k+1)} + \omega \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} = \omega b_i - \omega \sum_{j=i+1}^N a_{ij} x_j^{(k)} + (1-\omega) a_{ii} x_i^{(k)}$

ce qui s'écrit : $(D - \omega E) x^{(k+1)} = \omega b + ((1-\omega) D + \omega F) x^{(k)}$.

Soit :

(13)

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

ou encore :

(14)

$$x^{(k+1)} = \left(\frac{D}{\omega} - E\right)^{-1} b + \left(\frac{D}{\omega} - E\right)^{-1} \left[\left(\frac{1}{\omega} - 1\right) D + F\right] x^{(k)}.$$

La généralisation, par relaxation, de la méthode de Jacobi est très peu utilisée parce que, en général, elle n'apporte aucun gain. Par contre, la méthode de relaxation SOR est d'un emploi courant car, comme on le démontrera, elle améliore souvent la rapidité de la convergence.

Remarque 5

La programmation de SOR, pour ω choisi, est aussi simple que celle de Gauss-Seidel.

```

    POUR I = 1 à N FAIRE
        S := 0
        POUR J = 1 à N FAIRE
            S := S + A(I,J) * X(J)
        FIN DE BOUCLE J
        R := B(I) - S
        X(I) := X(I) + OMEGA * R/A(I,I)
    FIN DE BOUCLE I

```

Remarque 6

L'introduction du paramètre ω peut s'interpréter en remarquant que l'on a

$$x_i^{(k+1)} = x_i^{(k)} + \omega \frac{r_i^{(k,i)}}{a_{ii}}.$$

L'idée est que si la "correction" apportée à la $i^{\text{ème}}$ composante va "dans le bon sens", on a intérêt à l'augmenter en la multipliant par un facteur plus grand que 1. Au contraire, si on risque d'osciller ou de diverger, il faut alors "amortir" cette correction en la multipliant par un facteur plus petit que 1.

Le paramètre ω considéré est indépendant de k et de i . Pour certaines méthodes que l'on étudiera dans la suite du cours, il dépendra de k (par exemple dans la méthode de TCHEBYCHEFF).

Remarque 7

Comme précédemment (cf. remarque 3), posons :

$$\tilde{r}_i^{(k)} = r_i^{(k,i)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^N a_{ij} x_j^{(k)}$$

que l'on calcule en cours d'itération.

$$\text{On a : } x_i^{(k+1)} - x_i^{(k)} = \omega \frac{\tilde{r}_i^{(k)}}{a_{ii}},$$

ou encore :

$$\delta x^{(k+1)} = \omega D^{-1} \tilde{r}^{(k)},$$

$$\text{alors : } r_i^{(k+1)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^N a_{ij} x_j^{(k+1)}.$$

$$\begin{aligned} \text{Donc : } r_i^{(k+1)} - \tilde{r}_i^{(k)} &= - \sum_{j=i}^N a_{ij} (x_j^{(k+1)} - x_j^{(k)}) \\ &= -\omega \sum_{j=i}^N \frac{a_{ij}}{a_{jj}} \tilde{r}_j^{(k)} = -\omega \tilde{r}_i^{(k)} - \omega \sum_{j=i+1}^N \frac{a_{ij}}{a_{jj}} \tilde{r}_j^{(k)}. \end{aligned}$$

$$\text{D'où : } r_i^{(k+1)} = -\omega \sum_{j=i+1}^N \frac{a_{ij}}{a_{jj}} \tilde{r}_j^{(k)} + (1-\omega) \tilde{r}_i^{(k)}.$$

$$50 \quad r^{(k+1)} = (\omega F D^{-1} + (1-\omega) I) \tilde{r}^{(k)}.$$

7.2. Etude générale

7.2.1 La convergence

Les méthodes précédentes peuvent se formuler de la manière générale suivante :

- on décompose A sous la forme $A = M - N$

$$Ax = b \Leftrightarrow Mx = Nx + b,$$

- si M est une matrice régulière, on définit la méthode itérative par :

$$Mx^{(k+1)} = Nx^{(k)} + b$$

(19)

$$x^{(k+1)} = M^{-1} Nx^{(k)} + M^{-1} b .$$

Remarquons que $M^{-1} N = M^{-1} (M - A) = I - M^{-1} A$.

Alors :

(20)

$$x^{(k+1)} = (I - M^{-1} A) x^{(k)} + M^{-1} b .$$

Soit \bar{x} , la solution de $Ax = b$ qui vérifie :

$$(21) \quad \bar{x} = (I - M^{-1}A)\bar{x} + M^{-1}b.$$

Notons $e^{(k)}$ l'erreur à la $k^{\text{ème}}$ itération, c'est-à-dire :

$$e^{(k)} = \bar{x} - x^{(k)}.$$

On obtient en retranchant (20) de (21)

$$(22) \quad e^{(k+1)} = (I - M^{-1}A)e^{(k)}.$$

Donc :

$$(23) \quad e^{(k)} = (I - M^{-1}A)^k e^{(0)}.$$

Posons $B = M^{-1}N = I - M^{-1}A$ la matrice qui caractérise la méthode itérative liée à la décomposition $A = M - N$.

On appellera **B la matrice de l'itération**.

On a :

$$(24) \quad e^{(k)} = B^k e^{(0)}.$$

Soit $c = M^{-1}b$, alors (20) peut encore s'écrire :

$$x^{(k+1)} = Bx^{(k)} + c.$$

Théorème 8

La suite définie par $x^{(0)} \in \mathbb{C}^N$ et $x^{(k+1)} = Bx^{(k)} + c$ converge vers

$\bar{x} = (I - B)^{-1}c = (M^{-1}A)^{-1}M^{-1}b = A^{-1}b$ quel que soit $x^{(0)}$ si et seulement si $\rho(B) < 1$.

Preuve :

Pour que la suite $x^{(k)}$ converge vers \bar{x} quel que soit $x^{(0)} \in \mathbb{C}^N$, il faut et il suffit que $e^{(k)}$ converge vers 0 quel que soit $e^{(0)}$.

D'après le corollaire (I.103), il faut et il suffit que $\rho(B) < 1$. ■

7.2.2. Les matrices de l'itération

Dans les exemples précédents : Jacobi, Gauss-Seidel, relaxation par points ou par blocs, on a :

$$\text{pour Jacobi : } \begin{aligned} M &= D & B &= D^{-1} (E + F) = I - D^{-1} A, \\ N &= E + F \end{aligned}$$

$$\text{Gauss-Seidel } \begin{aligned} M &= D - E & B &= (D - E)^{-1} F = (I - D^{-1} E)^{-1} D^{-1} F, \\ N &= F \end{aligned}$$

$$\text{Relaxation } \begin{aligned} M &= \frac{D}{\omega} - E & B &= (D - \omega E)^{-1} [(1 - \omega) D + \omega F] \\ N &= \left(\frac{1}{\omega} - 1 \right) D + F = (I - \omega D^{-1} E)^{-1} [(1 - \omega) I + \omega D^{-1} F] \end{aligned}$$

Ces méthodes s'expriment en fonction de $L = D^{-1} E$ et $U = D^{-1} F$ qui sont respectivement deux matrices strictement triangulaires inférieures ou supérieures. Pour les méthodes par points, on a :

$$L = \begin{bmatrix} 0 & & & & & \\ -\frac{a_{21}}{a_{22}} & & & & & \\ -\frac{a_{31}}{a_{33}} & -\frac{a_{32}}{a_{33}} & & & & \\ \vdots & & & & & \\ -\frac{a_{N1}}{a_{NN}} & -\frac{a_{N2}}{a_{NN}} & \dots & -\frac{a_{N,N-1}}{a_{NN}} & & 0 \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \dots & -\frac{a_{1N}}{a_{11}} & \\ & -\frac{a_{23}}{a_{22}} & \dots & -\frac{a_{2N}}{a_{22}} & & \\ & & \ddots & & & \\ & & & -\frac{a_{N-1,N}}{a_{N-1,N-1}} & & \\ & & & & & 0 \end{bmatrix}.$$

1

Il est d'usage de noter respectivement les matrices des itérations par :

$J = L + U$ pour Jacobi,

$\mathcal{L}_\omega = (I - \omega L)^{-1} ((1 - \omega) I + \omega U)$ pour la relaxation.

En particulier pour $\omega = 1$, $\mathcal{L}_1 = (I - L)^{-1} U$ pour Gauss-Seidel.

Remarque 9

Il est naturel que les méthodes précédentes s'expriment en fonction $L = D^{-1}E$ et $U = D^{-1}F$ car le système $Ax = b$ est équivalent à $D^{-1}Ax = D^{-1}b$. Alors $D^{-1}A$ se décompose en $D^{-1}A = D^{-1}(D-E-F) = I - L - U$.

Remarque 10

Si on reprend les deux exemples traités au début du chapitre, dans le premier cas la matrice de l'itération de Jacobi est

$$\begin{bmatrix} 0 & -\frac{1}{10} \\ -\frac{2}{10} & 0 \end{bmatrix}$$

dont les valeurs propres sont $\lambda = \pm \frac{\sqrt{2}}{10}$, alors $\rho(J) < 1$ donc la méthode converge. Par contre dans le deuxième exemple $J = \begin{bmatrix} 0 & 10 \\ -5 & 0 \end{bmatrix}$, $\rho(J) = \sqrt{50} > 1$, la méthode diverge.

Théorème 11

Pour toute matrice A , le rayon spectral de la matrice de la méthode itérative de relaxation \mathcal{L}_ω est supérieur ou égal à $|\omega - 1|$.

Preuve :

La matrice de l'itération de la méthode de relaxation est :

$$\mathcal{L}_\omega = (I - \omega L)^{-1} [(1 - \omega) I + \omega U].$$

Les valeurs propres λ de \mathcal{L}_ω vérifient l'équation caractéristique

$$p(\lambda) = \det(\lambda I - \mathcal{L}_\omega) = a_0 \lambda^N + a_1 \lambda^{N+1} + \dots + a_N = 0 \quad \text{où } a_0 = 1.$$

Comme $\det(I - \omega L) = 1$ car L est triangulaire strictement inférieure on a :

$$p(\lambda) = \det(I - \omega L) \det(\lambda I - \mathcal{L}_\omega) = \det(\lambda(I - \omega L) - (1 - \omega)I - \omega U) = 0.$$

D'après les relations existant entre les coefficients du polynôme et ses racines, on sait que :

$$\prod_{i=1}^N \lambda_i = (-1)^N \frac{a_N}{a_0} = (-1)^N a_N.$$

Comme $a_N = p(0) = \det(-(1 - \omega)I - \omega U) = (\omega - 1)^N$,

$$\prod_{i=1}^N |\lambda_i| = |\omega - 1|^N.$$

Par définition du rayon spectral, on a pour $i = 1 \text{ à } N$

$$\rho(\mathcal{L}_\omega) \geq |\lambda_i| \quad \text{donc} \quad \rho^N(\mathcal{L}_\omega) \geq \prod_{i=1}^N |\lambda_i|,$$

d'où le résultat

$$\rho(\mathcal{L}_\omega) \geq |\omega - 1|. \quad \blacksquare$$

Corollaire 12

Pour toute matrice A , une condition nécessaire de convergence de la méthode de relaxation est que :

$$0 < \omega < 2$$

Preuve :

Pour que $\rho(\mathcal{L}_\omega) < 1$ il faut que $|\omega - 1| < 1$ c'est-à-dire $0 < \omega < 2$. \blacksquare

7.3. Les matrices à diagonale dominante

Comme nous l'avons constaté, pour certains problèmes modèles la matrice A du système linéaire à résoudre est à diagonale dominante, ou hermitienne définie positive.

Pour ces deux grandes classes de matrices, nous allons étudier la convergence des différentes méthodes. Nous commençons par l'étude des matrices à diagonale dominante dont les principales propriétés ont été vues au chapitre I, paragraphe 2.6.

Théorème 19

A étant une matrice, soit à diagonale strictement dominante, soit irréductible et à diagonale fortement dominante, alors la méthode de Jacobi est convergente.

Preuve :

Nous savons que la matrice A est régulière (cf. proposition 1.60 et 1.61). Montrons que la méthode de Jacobi est bien définie.

Si $a_{ii} = 0$, alors $\sum_{\substack{j=1 \\ j \neq i}}^N |a_{ij}| = 0$, donc $\forall j, 1 \leq j \leq N, a_{ij} = 0$.

A posséderait une ligne de zéros et serait singulière.

Donc, $\forall i$, $1 \leq i \leq N$, $a_{ii} \neq 0$, la méthode de Jacobi ainsi que les méthodes de relaxation sont bien définies.

La matrice d'itération est :

$$J = D^{-1}(E + F),$$

où :

$$J_{ii} = 0,$$

$$J_{ij} = -\frac{a_{ij}}{a_{ii}} \quad \text{pour } i \neq j.$$

On a :

$$\sum_{j=1}^N |J_{ij}| = \sum_{\substack{j=1 \\ j \neq i}}^N \left| \frac{a_{ij}}{a_{ii}} \right|.$$

Si A est à diagonale strictement dominante alors on a pour tout i , $1 \leq i \leq N$:

$$\sum_{j=1}^N |J_{ij}| < 1, \text{ donc } \max_{1 \leq i \leq N} \sum_{j=1}^N |J_{ij}| = \|J\|_\infty < 1.$$

La méthode converge car :

$$\rho(J) \leq \|J\|_\infty < 1.$$

Si A est à diagonale fortement dominante et irréductible, on a de la même manière :

$$\rho(J) \leq \|J\|_\infty \leq 1.$$

Mais, si λ était valeur propre de module égal à 1, alors en utilisant le deuxième théorème de Gerchgorin, tous les cercles de Gerchgorin passeraient par λ , ce qui est en contradiction avec l'hypothèse qu'il existe au moins un indice pour lequel l'inégalité est stricte.

On ne peut donc avoir une valeur propre du module 1, donc $\rho(J) < 1$. La méthode de Jacobi converge. ■

Corollaire 23

Soit A une matrice hermitienne décomposée par points ou par blocs sous la forme $A = D - E - F$ avec **D définie positive**.

La méthode de relaxation par points ou par blocs, pour $0 < \omega < 2$ est convergente, si et seulement si A est définie positive.

Preuve :

Soit $A = D - E - F$, la décomposition usuelle de A par points ou par blocs. Comme A est hermitienne, dans les deux cas, on a $D = D^*$ et $F = E^*$.

Pour la méthode de relaxation, on a $M = \frac{D}{\omega} - E$, d'où $M^* + M - A = \frac{2-\omega}{\omega} D$ qui est hermitienne.

Pour $0 < \omega < 2$ et puisque D est définie positive, $M^* + M - A$ est définie positive, donc la méthode converge si et seulement si A est définie positive (cf. théorème 22). ■

Corollaire 24

Soit A une matrice hermitienne et définie positive, alors la méthode de relaxation converge si et seulement si $0 < w < 2$.

Preuve :
Il suffit de montrer que D est définie positive (cf. corollaire 23). C'est vrai par points ou par blocs (cf. proposition 1.14) lorsque A est définie positive.

Remarque 25

Soit A une matrice hermitienne avec D non définie positive, alors la méthode de relaxation peut néanmoins converger.

Par exemple :

$$A = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix},$$

d'où $\mathcal{L}_\omega = (1 - \omega)I$ puisque A est diagonale.
On a alors $\rho(\mathcal{L}_\omega) < 1$ lorsque $0 < \omega < 2$.

3.4 Méthode des descentes

CHAPITRE 8

Méthodes de gradient conjugué

8.0. Introduction

A chaque itération d'une méthode de descente, on détermine un vecteur p_k et un scalaire α_k , ce qui permet de calculer x_{k+1} par :

(1)

$$x_{k+1} = x_k + \alpha_k p_k ,$$

avec l'objectif de minimiser une fonctionnelle.

Avant d'introduire l'importante méthode du gradient conjugué pour résoudre un système linéaire dont la matrice est symétrique et définie positive, il est nécessaire d'aborder sommairement les méthodes du gradient.

Comme ces méthodes convergent d'autant plus vite que le conditionnement de la matrice est faible, nous étudierons certaines techniques de préconditionnement pour accélérer la rapidité de la convergence. On obtient alors des méthodes dont la convergence est souvent beaucoup plus rapide que celles du type relaxation.

Pour terminer ce chapitre, nous étudierons sommairement quelques extensions des méthodes de gradient conjugué à des systèmes dont la matrice n'est plus nécessairement symétrique et définie positive.

8.1. Principe des méthodes de descente

8.1.1 Choix de la fonctionnelle à minimiser

Soit A une matrice symétrique et définie positive. On note l'équivalence entre la solution \bar{x} de $Ax = b$ et le vecteur qui réalise le minimum de la fonctionnelle J définie par :

(2)

$$J(x) = (Ax | x) - 2(b | x)$$

En effet, la fonctionnelle J est quadratique et définie positive.

Son minimum unique est obtenu en annulant le gradient g de J .

Or $g(x) = 2(Ax - b) = -2r(x)$ (cf. chapitre 1 ; 4, §)

où $r(x) = b - Ax = A\bar{x} - Ax$ est le résidu du système $Ax = b$

Il est encore équivalent de minimiser J ou de minimiser E défini par :

(3)

$$E(x) = (A(x - \bar{x}) | x - \bar{x}) = (Ae(x) | e(x))$$

où $e(x) = x - \bar{x}$,

car

$$\begin{aligned} E(x) &= (Ax | x) - 2(x | A\bar{x}) + (A\bar{x} | \bar{x}) \\ &= J(x) + (A\bar{x} | \bar{x}) . \end{aligned}$$

Comme $(A\bar{x} | \bar{x})$ est une constante, E et J atteignent leur minimum au même point. A étant symétrique et définie positive $(Ax | y)$ est un produit scalaire et $E(x) = \|e(x)\|_A^2$ où $\|e\|_A = (Ae | e)^{1/2}$ est la norme associée à ce produit scalaire.

On remarque également que $E(x)$ peut s'exprimer en fonction du résidu $r(x) = A\bar{x} - Ax$ par

(4)

$$E(x) = (r(x) | A^{-1}r(x)) .$$

Pour minimiser la fonctionnelle E , les méthodes de "descente" sont construites en choisissant à la $k^{\text{ième}}$ itération une direction de descente $p_k \neq 0$ et un scalaire α_k de manière que $E(x_{k+1}) < E(x_k)$.

8.1.2 Choix optimal de α_k dans une direction fixée p_k

Supposons que la direction p_k soit fixée. Le choix local optimal de α_k consiste, à chaque itération, à choisir α_k de façon à minimiser $E(x_{k+1})$ dans la direction p_k .

Le α_k optimal est donc tel que :

$$E(x_k + \alpha_k p_k) = \min_{\alpha \in \mathbb{R}} E(x_k + \alpha p_k) .$$

Or,

$$E(x_k + \alpha p_k) = (A(x_k + \alpha p_k - \bar{x}) | x_k + \alpha p_k - \bar{x})$$

$$= E(x_k) - 2\alpha(r_k | p_k) + \alpha^2(Ap_k | p_k) ,$$

est un trinôme du second degré en α dont le terme de plus haut degré $(A p_k | p_k)$ est strictement positif quel que soit le choix de $p_k \neq 0$, car A est une matrice définie positive.

Son minimum est atteint pour :

(5)

$$\alpha_k = \frac{(r_k | p_k)}{(A p_k | p_k)} .$$

Alors :

(6)

$$x_{k+1} = x_k + \frac{(r_k | p_k)}{(A p_k | p_k)} p_k .$$

Proposition 1

Quel que soit le choix de $p_k \neq 0$ pour α_k optimal, on a les deux

relations suivantes :

(7) $\forall k \geq 0$

$$r_{k+1} = r_k - \alpha_k A p_k ,$$

(8)

$$(p_k | r_{k+1}) = 0 .$$

Preuve :

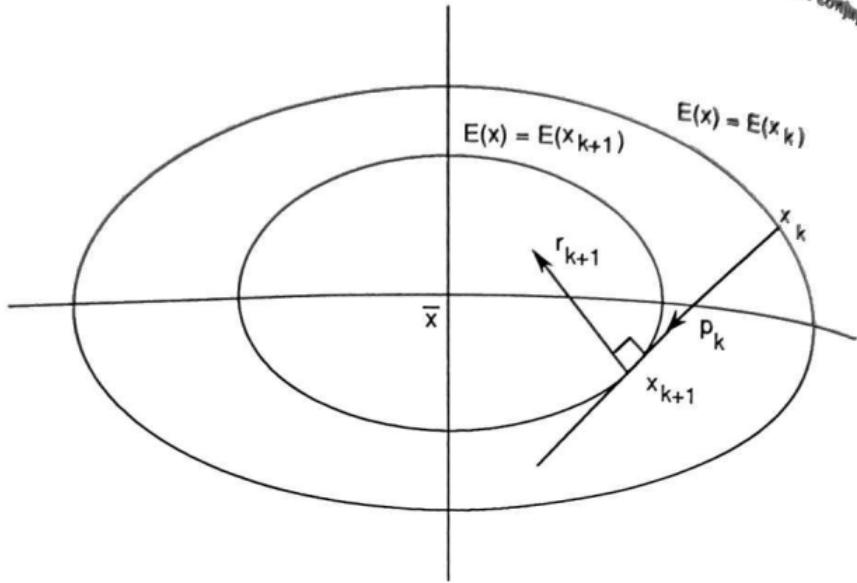
$$r_{k+1} = b - A x_{k+1} = b - A (x_k + \alpha_k p_k) = r_k - \alpha_k A p_k .$$

$$(p_k | r_{k+1}) = (p_k | r_k - \alpha_k A p_k) = (p_k | r_k) - \frac{(r_k | p_k)}{(A p_k | p_k)} (p_k | A p_k) = 0 . \blacksquare$$

Donnons une interprétation géométrique dans \mathbb{R}^2 des méthodes de descente.

$E(x) = \text{constante positive}$ est l'équation d'une ellipse.

Pour les différentes valeurs de (x_k) , on obtient une famille d'ellipses $E(x) = E(x_k)$ concentriques autour du minimum \bar{x} de la fonctionnelle, et qui représentent les courbes de niveau :



Le vecteur p_k est tangent à l'ellipse $E(x) = E(x_{k+1})$. Comme r_{k+1} est orthogonal à p_k , r_{k+1} est orthogonal à la tangente de la courbe de niveau.

Reportons la valeur de α_k dans $E(x_{k+1})$, on obtient :

$$E(x_{k+1}) = E(x_k) - \frac{(r_k \mid p_k)^2}{(A p_k \mid p_k)} ,$$

$$E(x_{k+1}) = E(x_k) \left[1 - \frac{1}{E(x_k)} \frac{(r_k \mid p_k)^2}{(A p_k \mid p_k)} \right] ,$$

ou encore, puisque $E(x_k) = (r_k \mid A^{-1} r_k)$:

$$(9) \quad E(x_{k+1}) = E(x_k)(1 - \gamma_k) \quad \text{où} \quad \gamma_k = \frac{(r_k \mid p_k)^2}{(A p_k \mid p_k)(A^{-1} r_k \mid r_k)} .$$

Sauf pour $p_k = 0$ (cas que l'on élimine) ou $r_k = 0$ (auquel cas, x_k est la solution cherchée), le nombre

$$\gamma_k = \frac{(r_k \mid p_k)^2}{(A p_k \mid p_k)(A^{-1} r_k \mid r_k)}$$

est toujours défini et positif car A étant une matrice symétrique et définie positive, on a $(A p_k \mid p_k) > 0$ et $(A^{-1} r_k \mid r_k) > 0$.

Lemme 2

Quel que soit le choix de $p_k \neq 0$ pour α_k optimal local, on a la relation suivante valable pour $k \geq 0$:

$$(10) \quad \gamma_k = \frac{(r_k | p_k)^2}{(A p_k | p_k)(A^{-1} r_k | r_k)} \geq \frac{1}{K(A)} \left(\frac{r_k}{\|r_k\|} \middle| \frac{p_k}{\|p_k\|} \right)^2,$$

où $K(A)$ est le nombre conditionnement de la matrice A .

Preuve :

On a $(A p_k | p_k) \leq \lambda_1 \|p_k\|^2$ où λ_1 est la plus grande valeur propre de A .

$$(A^{-1} r_k | r_k) \leq \frac{1}{\lambda_N} \|r_k\|^2 \text{ où } \lambda_N \text{ est la plus petite valeur propre de } A.$$

Donc :

$$\frac{(A p_k | p_k)(A^{-1} r_k | r_k)}{\|p_k\|^2 \|r_k\|^2} \leq \frac{\lambda_1}{\lambda_N} = K(A).$$

D'où :

$$\gamma_k \geq \frac{1}{K(A)} \frac{(r_k | p_k)^2}{\|r_k\|^2 \|p_k\|^2}.$$

Ce lemme va nous permettre de choisir des directions de descente.

Théorème 3

Pour α_k optimal local, toute direction p_k qui vérifie, $\forall k \geq 0$:

$$(11) \quad \left(\frac{r_k}{\|r_k\|} \middle| \frac{p_k}{\|p_k\|} \right)^2 \geq \mu > 0 \quad \text{où } \mu \text{ est indépendant}$$

de k , implique que la suite (x_k) converge vers la solution \bar{x} qui minimise $E(x)$.

Preuve :

D'après (9) (10) et (11), on a :

$$E(x_{k+1}) \leq E(x_k) \left(1 - \frac{\mu}{K(A)}\right).$$

$$\text{Donc } E(x_k) \leq \left(1 - \frac{\mu}{K(A)}\right)^k E(x_0).$$

D'après (11) et l'inégalité de Schwarz, on a $0 < \mu \leq 1$.

Comme $K(A) \geq 1$, on a $0 \leq 1 - \frac{\mu}{K(A)} < 1$.

Donc : $\lim_{k \rightarrow +\infty} E(x_k) = 0$.

Or : $E(x_k) \geq \lambda_N \|x_k - \bar{x}\|^2$

donc : $\lim_{k \rightarrow +\infty} \|x_k - \bar{x}\|^2 = 0$. avec $\lambda_N > 0$,

Dans le cadre de α_k optimal local, ce théorème est une condition suffisante de convergence qui signifie que pour tout k , p_k doit être non orthogonal à r_k . Il en résulte un premier choix évident $p_k = r_k$, car alors :

$$\mu = \left(\frac{r_k}{\|r_k\|} \middle\| \frac{p_k}{\|p_k\|} \right)^2 = 1.$$

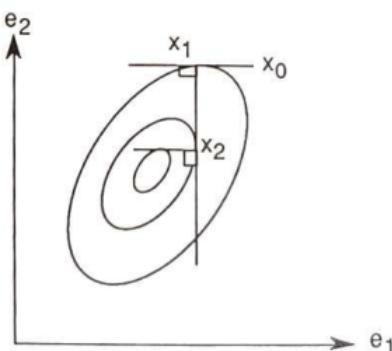
C'est ce que l'on va étudier au paragraphe suivant.

Remarque 4

Si l'on choisit comme direction de descente les vecteurs unitaires e_i des axes de coordonnées de l'espace à N dimensions dans l'ordre naturel e_1, e_2, \dots, e_N puis on recommence cycliquement, on obtient la méthode de Gauss-Seidel.

En effet $x_{k+1} = x_k + \alpha_k e_i$ où $\alpha_k = \frac{(r_k \mid e_i)}{(Ae_i \mid e_i)} = \frac{(b - Ax_k \mid e_i)}{(a_{ii})}$.

On sait alors que lorsque la matrice A est symétrique et définie positive la méthode est convergente.



8.2. Les méthodes du gradient

Dans ce paragraphe, comme le suggère le résultat précédent, nous allons choisir le gradient (ou ce qui revient au même, le résidu) comme direction de descente, d'abord dans le cadre de α_k optimal local, mais aussi pour α_k constant.

8.2.1 La méthode du gradient à paramètre optimal

C'est la méthode où $x_{k+1} = x_k + \alpha_k r_k$,

avec

$$(12) \quad \boxed{\alpha_k = \frac{(r_k | p_k)}{(A p_k | p_k)} = \frac{\|r_k\|^2}{(A r_k | r_k)}} .$$

Alors : $E(x_{k+1}) = E(x_k) \left(1 - \frac{\|r_k\|^4}{(A r_k | r_k)(A^{-1} r_k | r_k)} \right)$.

Grâce à l'inégalité de Kantorovitz (cf. chapitre 1 ; 4,3), on a :

$$\frac{\|r_k\|^4}{(A r_k | r_k)(A^{-1} r_k | r_k)} \geq \frac{4 \lambda_1 \lambda_N}{(\lambda_1 + \lambda_N)^2} = \frac{\lambda_N}{\left(\frac{\lambda_1}{\lambda_N} + 1\right)^2} = \frac{4 K(A)}{(K(A) + 1)^2} .$$

Alors :

$$E(x_{k+1}) \leq E(x_k) \left(1 - \frac{4 K(A)}{(K(A) + 1)^2} \right) = E(x_k) \left(\frac{K(A) - 1}{K(A) + 1} \right)^2 .$$

Donc :

$$(13) \quad \boxed{E(x_k) \leq E(x_0) \left(\frac{K(A) - 1}{K(A) + 1} \right)^{2k}} .$$

Comme $E(x_k) \geq \lambda_N \|x_k - \bar{x}\|^2$, on a :

$$(14) \quad \|x_k - \bar{x}\| \leq \beta \quad \left(\frac{K(A) - 1}{K(A) + 1} \right)^k \quad \text{où} \quad \beta = \left(\frac{E(x_0)}{\lambda_N} \right)^{1/2} .$$

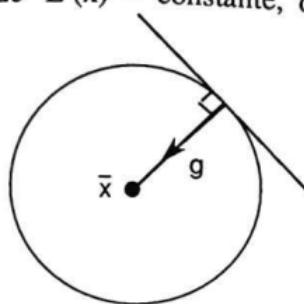
On a donc le théorème suivant :

Théorème 5

La méthode du gradient à paramètre local optimal est convergente, sa rapidité de convergence dépend de : $\frac{K(A) - 1}{K(A) + 1}$.

Remarque 6

Plus $K(A)$ est proche de 1, plus la méthode convergera vite. Lorsque $K(A) = 1$, toutes les valeurs propres de A sont égales. On a $A = \lambda I$ et $E(x) = \lambda \|x - \bar{x}\|^2$. Lorsque $E(x) = \text{constante}$, on obtient l'équation d'une sphère.



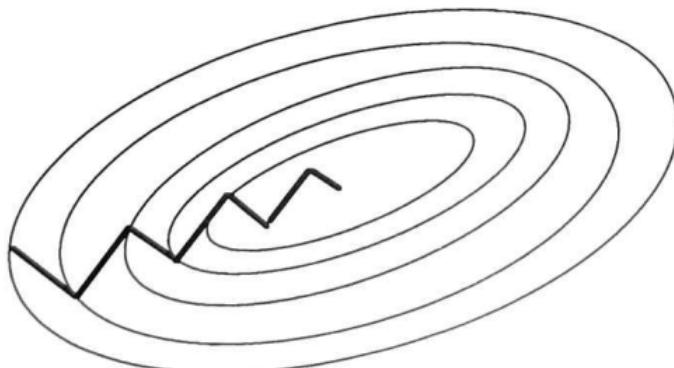
De n'importe quel point de la sphère, le gradient pointe vers le centre. On converge en une itération.

Par contre, lorsque $K(A)$ est grand, les valeurs propres extrêmes sont très différentes : l'ellipsoïde est alors très aplati.

La convergence est lente, pour que $\frac{E(x_k)}{E(x_0)} = \varepsilon$, il suffit que :

$$\left(\frac{K(A) - 1}{K(A) + 1}\right)^{2k} \leq \varepsilon, \text{ soit encore que : } k \sim \frac{K(A)}{4} \ln \frac{1}{\varepsilon}.$$

Le nombre d'itérations est proportionnel à $K(A)$.



8.2.2 Un exemple

Considérons le problème simple suivant : résoudre le système :

$$\begin{cases} \frac{1}{2}x = 0 \\ \frac{c}{2}y = 0 \end{cases} \quad \text{qui s'écrit} \quad \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{c}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

où c est une constante donnée supérieure à 1.

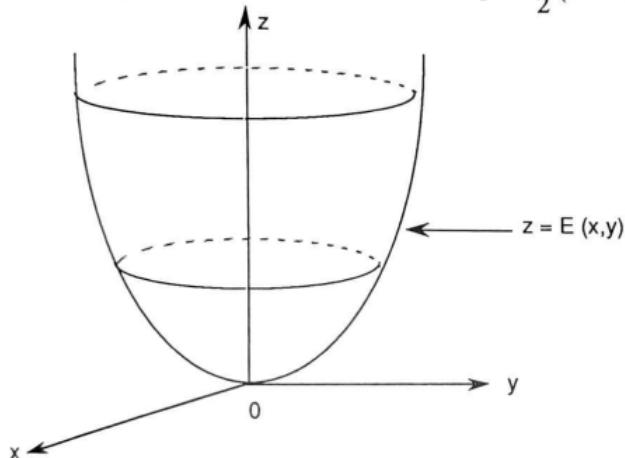
La solution est évidemment $x = y = 0$. Le nombre de conditionnement de A vaut $K(A) = c$ (dans ce paragraphe x et y désignent deux variables réelles).

Ce problème a pour solution le minimum de la fonctionnelle :

$$E(x,y) = \left(A \begin{pmatrix} x \\ y \end{pmatrix} \right) \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{2} (x^2 + c y^2)$$

Représentons géométriquement le problème de minimisation de $E(x,y)$.

$E(x,y) = \text{constante positive}$ est l'équation d'une ellipse : $\frac{1}{2}(x^2 + c y^2) = \text{Cte}$.



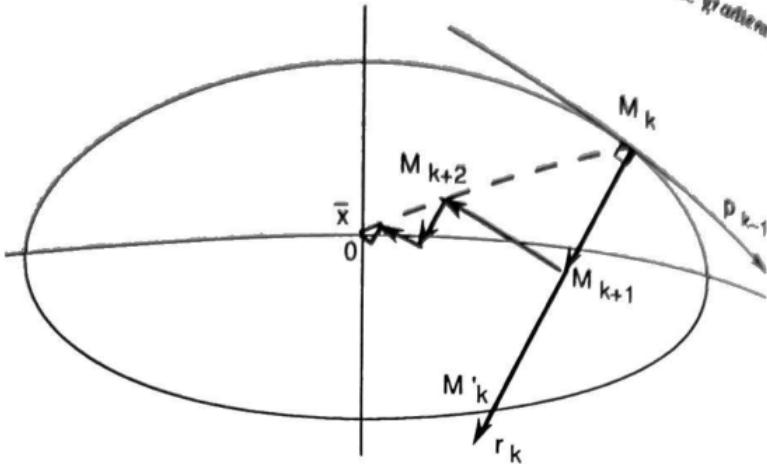
Pour différents choix de cette constante positive, on obtient une famille d'ellipses qui représentent les courbes de niveau de la fonctionnelle E .

Etudions la méthode du gradient à paramètre local optimal sur cet exemple. Le résidu $r = b - Ax$ est le vecteur $-\frac{1}{2} \begin{pmatrix} x \\ cy \end{pmatrix}$. Le paramètre est le nombre :

$$\alpha = \frac{\|r\|^2}{(A r | r)} = \frac{x^2 + c^2 y^2}{\frac{x^2}{2} + \frac{c^3 y^2}{2}}.$$

Connaissant le point $M_k(x_k, y_k)$, on construit le point $M_{k+1}(x_{k+1}, y_{k+1})$

$$M_{k+1} \begin{cases} x_{k+1} = x_k - \frac{\alpha_k}{2} x_k = \left(1 - \frac{\alpha_k}{2}\right) x_k = \frac{c^2(c-1)y_k^2}{x_k^2 + c^3 y_k^2} x_k, \\ y_{k+1} = y_k - c \frac{\alpha_k}{2} y_k = \left(1 - c \frac{\alpha_k}{2}\right) y_k = \frac{(1-c)x_k^2}{x_k^2 + c^3 y_k^2} y_k. \end{cases}$$

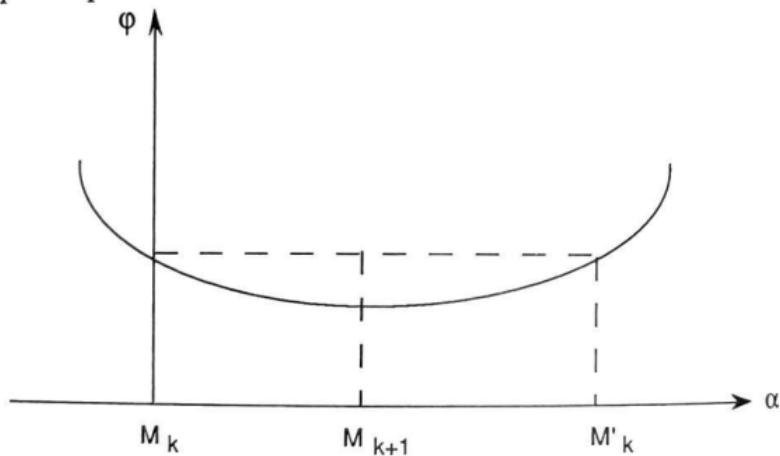


On peut construire géométriquement le point M_{k+1} à partir de M_k .
On sait que la direction r_k est orthogonale à celle de p_{k-1} .

Traçons l'ellipse passant par M_k . Soit M'_k son intersection avec la droite de direction r_k passant par M_k .

Alors M_{k+1} est le milieu de la corde $M_k M'_k$. En effet, soit $\varphi(\alpha) = E(M_k + \alpha r_k)$ qui est un trinôme du second degré en α .

α_k est le minimum de $\varphi(\alpha)$. Or, comme $E(M_k) = E(M'_k)$, le minimum est obtenu pour le point milieu



Démontrons que M_{k+2} et M_k sont alignés avec 0.

En effet, posons $t_k = \frac{y_k}{x_k}$ qui est la pente de $0 M_k$.

$$t_{k+1} = \frac{y_{k+1}}{x_{k+1}} = -\frac{1}{c^2 t_k^2} \quad t_k = -\frac{1}{c^2 t_k}$$

Donc :

$$t_{k+2} = -\frac{1}{c^2 t_{k+1}} = -\frac{1}{-c^2 \frac{1}{c^2 t_k}} = t_k$$

Les itérés successifs sont situés sur deux droites passant par l'origine. Soit t la pente d'une de ces deux droites.
Il est commode pour évaluer le facteur moyen τ de réduction de l'erreur de calculer :

$$\begin{aligned}\tau^2 &= \frac{y_{k+2}}{y_k} = \frac{x_{k+2}}{x_k} = \left(1 - \frac{\alpha_{k+1}}{2}\right)\left(1 - \frac{\alpha_k}{2}\right) = \frac{c^2(c-1)\left(\frac{1}{c^2 t}\right)^2}{1+c^3\left(\frac{1}{c^2 t}\right)^2} \cdot \frac{c^2(c-1)t^2}{1+c^3 t^2} \\ &= \frac{(c-1)^2}{\left(1 + \frac{c^3}{c^4 t^2}\right)(1+c^3 t^2)} = \frac{(c-1)^2}{\frac{1}{c t^2}(c t^2 + 1)(1+c^3 t^2)}.\end{aligned}$$

Or :

$$\begin{aligned}\frac{1}{c t^2}(c t^2 + 1)(1+c^3 t^2) &= c^3 t^2 + 1 + c^2 + \frac{1}{c t^2} = (c+1)^2 - 2c + c^3 t^2 + \frac{1}{c t^2} \\ &= (c+1)^2 + c\left(c^2 t^2 - 2 + \frac{1}{c^2 t^2}\right) = (c+1)^2 \left(1 + \frac{c}{(c+1)^2} \left(ct - \frac{1}{ct}\right)^2\right).\end{aligned}$$

D'où : $\tau^2 = \frac{(c-1)^2}{(c+1)^2} \frac{1}{1 + \frac{c}{(c+1)^2} \left(ct - \frac{1}{ct}\right)^2}.$

Pour $t = \frac{1}{c}$, $\frac{1}{1 + \frac{c}{(c+1)^2} \left(ct - \frac{1}{ct}\right)^2}$ est maximum,

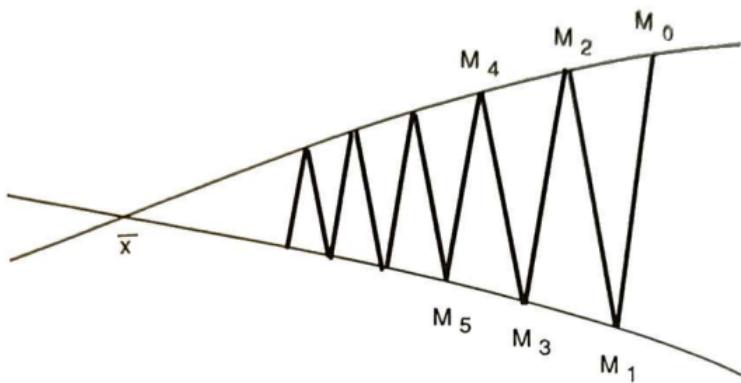
donc $\tau(t)$ est maximum et vaut :

$$\tau = \frac{c-1}{c+1} = \frac{K(A)-1}{K(A)+1}.$$

Par contre, si le point de départ est situé sur un des axes de l'ellipse ($t = 0$ ou $t = \infty$), alors la méthode converge en une seule itération.

On constate que le facteur de réduction de l'erreur peut être égal à $\frac{K(A)-1}{K(A)+1}$.

Sauf dans le cas où le point de départ est situé sur un des axes de l'ellipse, la convergence aura lieu en zigzag, avec un facteur de réduction de l'erreur d'autant plus voisin de 1 que $K(A)$ est grand.



Ces observations permettent d'envisager d'autres méthodes qui sont plus "performantes".

Si l'on veut diminuer le nombre d'opérations, on peut considérer la méthode où α_k serait constant au cours des itérations, ce qui se traduit dans l'exemple par :

$$x_{k+1} = x_k - \alpha x_k = (1 - \alpha) x_k ,$$

$$y_{k+1} = y_k - \alpha c y_k = (1 - \alpha c) y_k .$$

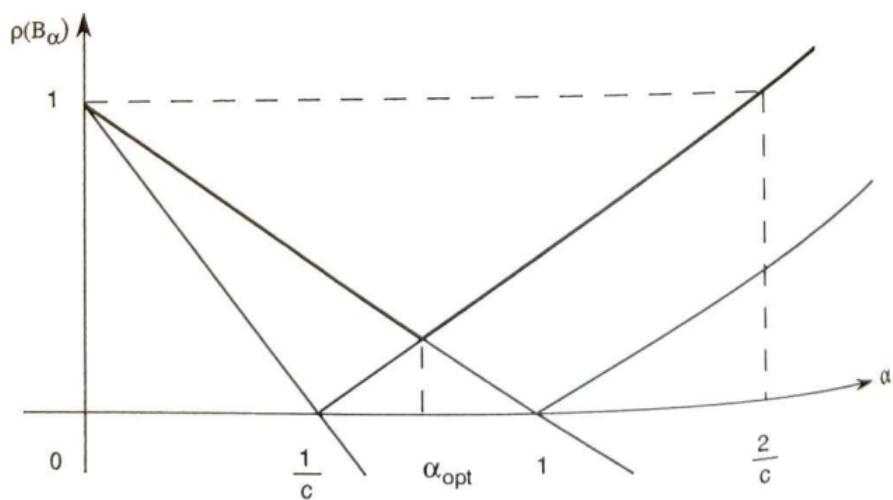
Pour déterminer α , on considère la matrice de l'itération

$$B_\alpha = \begin{bmatrix} 1-\alpha & 0 \\ 0 & 1-\alpha c \end{bmatrix} .$$

La condition nécessaire et suffisante de convergence est que :

$$\rho(B_\alpha) < 1 . \text{ Or } \rho(B_\alpha) = \max(|1 - \alpha|, |1 - \alpha c|) .$$

Le choix optimal de α est celui qui rend $\rho(B_\alpha)$ le plus petit possible.



Méthodes du gradient

Sur le graphe, on constate que la convergence a lieu pour $0 < \alpha < \frac{2}{c}$.

Le choix optimal est :

$$\alpha_{\text{opt}} = \frac{2}{1+c} \quad \text{pour lequel} \quad \rho(B\alpha_{\text{opt}}) = 1 - \frac{2}{1+c} = \frac{c-1}{c+1} = \frac{K(\Lambda) - 1}{K(\Lambda) + 1}.$$

On retrouve pour $c=1$ le même facteur de réduction de l'erreur que dans la méthode du gradient à paramètre optimal, ce qui prouve que α_k optimal peut être aussi mauvais que α constant optimal. Nous allons étudier le cas général de cette méthode au paragraphe suivant.

Comme les droites passant par M_k , de directions r_k , ne semblent pas donner entière satisfaction, on en cherchera de nouvelles. On souhaiterait qu'elles passent par le point qui rend minimum la forme quadratique, c'est-à-dire par le centre de l'ellipse.

Reprendons les notations habituelles.

Comme on a pour α_k optimal local $(p_k | r_{k+1}) = 0$ et que :

$$r_{k+1} = b - A x_{k+1} = A \bar{x} - A x_{k+1},$$

$$(p_k | A(r_{k+1})) = 0.$$

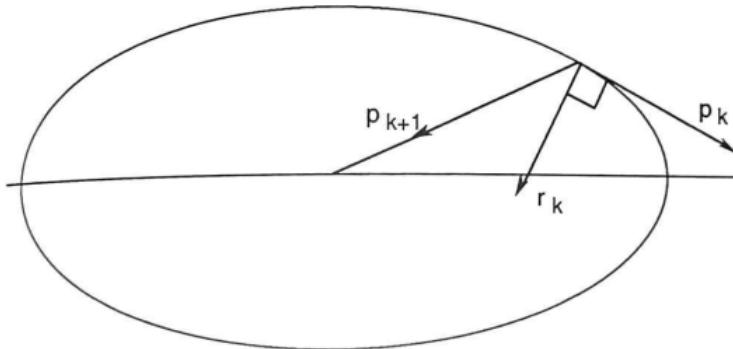
on obtient :

Si on veut que $x_{k+2} = \bar{x}$, il faut que la direction $p_{k+1} = \frac{1}{\alpha_{k+1}}(x_{k+2} - x_{k+1})$

$$(p_k | A p_{k+1}) = (A p_k | p_{k+1}) = 0.$$

soit telle que : Nous étudierons cette condition au paragraphe sur le gradient conjugué et nous

vérifierons alors que dans le cas d'une ellipse, on converge en deux itérations au plus.



8.2.3 La méthode du gradient à paramètre constant (méthode de Richardson)

Comme on l'a remarqué dans l'exemple précédent, il peut être inutile d'optimiser α_k à chaque itération, compte tenu de l'effet zigzag et du coût de calcul de α_k .

On prend toujours comme direction de descente celle du gradient c'est-à-dire celle du résidu au point considéré et on choisit α indépendant de k de façon que la suite des points (x_k) converge vers la solution \bar{x} .

$$\begin{aligned}x_{k+1} &= x_k + \alpha r_k, \\r_k &= b - Ax_k = A\bar{x} - Ax_k.\end{aligned}$$

avec :

L'erreur à la $k+1$ ^{ième} itération e_{k+1} peut s'exprimer en fonction de l'erreur à la k ^{ième} itération, en effet :

$$\begin{aligned}e_{k+1} &= x_{k+1} - \bar{x} = x_k + \alpha r_k - \bar{x} = x_k + \alpha (A\bar{x} - Ax_k) - \bar{x} \\&= (x_k - \bar{x}) - \alpha (Ax_k - A\bar{x}) = (I - \alpha A) e_k.\end{aligned}$$

Donc, on obtient :

(15)

$$e_k = (I - \alpha A)^k e_0.$$

La condition nécessaire et suffisante de convergence (cf. chapitre 7, théorème 8) est que : $\rho = (I - \alpha A) < 1$.

Il faut et il suffit que les N valeurs propres positives λ_i de A vérifient :

$$|1 - \alpha \lambda_i| < 1,$$

c'est-à-dire $\forall i = 1, 2, \dots, N, 0 < \alpha < \frac{2}{\lambda_i}$.

Donc, en classant les valeurs propres de A par ordre décroissant :

$$0 < \lambda_N \leq \lambda_{N-1} \leq \dots \leq \lambda_1,$$

il faut et il suffit que :

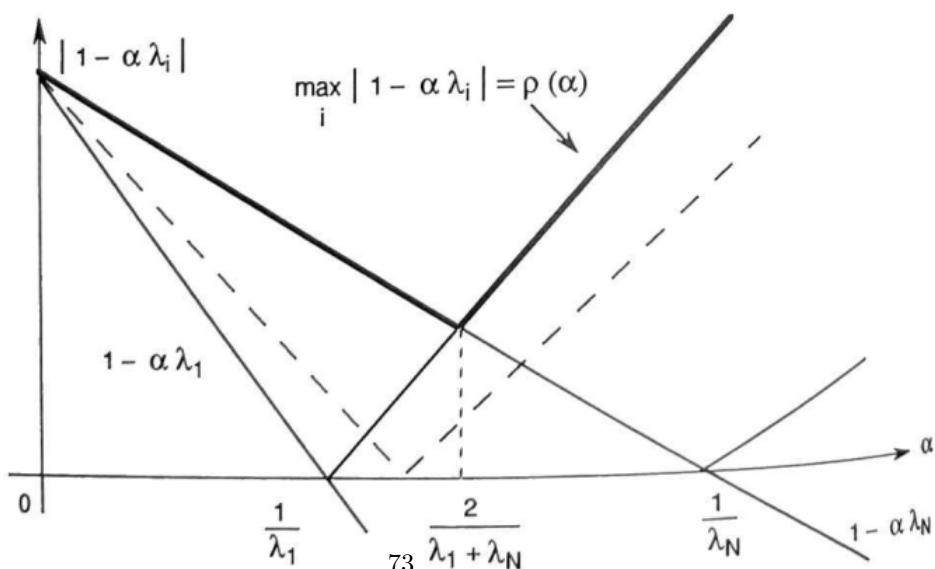
(16)

$$0 < \alpha < \frac{2}{\lambda_1}.$$

Le meilleur choix de α est celui qui minimise $\rho(I - \alpha A)$.

Or, $\rho(I - \alpha A) = \max_i |1 - \alpha \lambda_i| = \max(|1 - \alpha \lambda_1|, |1 - \alpha \lambda_N|)$.

Comme on le voit sur la figure, α est la solution de $1 - \alpha \lambda_1 = \alpha \lambda_N - 1$.



(17)

$$\alpha_{\text{opt}} = \frac{2}{\lambda_1 + \lambda_N}$$

Pour cette valeur α_{opt} , on obtient $\rho(I - \alpha_{\text{opt}} A) = \frac{\lambda_1 - \lambda_N}{\lambda_N + \lambda_1} = \frac{\frac{\lambda_1 - 1}{\lambda_N} - 1}{\frac{\lambda_1 + 1}{\lambda_N}}$,

(18)

$$\rho(I - \alpha_{\text{opt}} A) = \frac{K(A) - 1}{K(A) + 1}$$

Remarque 7

En prenant α optimal dans la méthode du gradient à paramètre constant, le facteur de réduction de l'erreur est de l'ordre de $\frac{K(A) - 1}{K(A) + 1}$, comme dans le pire des cas de la méthode du gradient à paramètre optimal. (Cf. exemple 8, 2, 2). Même si on augmente peu le nombre d'itérations, il est nécessaire de connaître λ_1 et λ_N , ce qui n'est pas le cas en pratique.

Remarque 8

L'équation $Ax = b$ est équivalente, si la matrice diagonale $D = \text{diag}(A)$ est inversible à $D^{-1}Ax = D^{-1}b$. Posons $A' = D^{-1}A$ et $b' = D^{-1}b$.

Pour ce nouveau système, dont la matrice est telle que les éléments diagonaux sont égaux à 1, la méthode de Jacobi est définie par la relation :

$$x_{k+1} = (I - A')x_k + b'$$

La méthode de Jacobi, dans ce cas, coïncide avec la méthode du gradient avec paramètre constant égal à 1.

8.3. Les méthodes du gradient conjugué

8.3.1 Introduction

On désire déterminer de nouvelles directions de descente p_k . On choisit α_k minimum local. Comme on l'a démontré en (8) $(p_{k-1} | r_k) = 0$, on va chercher p_k dans le plan formé par les deux directions orthogonales r_k et p_{k-1} .

Posons :

$$(19) \quad p_k = r_k + \beta_k p_{k-1}$$

β_k va être déterminé afin que le facteur de réduction de l'erreur soit le plus grand possible dans $E(x)$.

Or, d'après (9) :

$$E(x_{k+1}) = E(x_k) \left(1 - \frac{(r_k | p_k)^2}{(A p_k | p_k)(A^{-1} r_k | r_k)} \right).$$

Donc β_k sera choisi de façon que $\frac{(r_k | p_k)^2}{(A p_k | p_k)(A^{-1} r_k | r_k)}$ soit maximum.

$$\text{Or : } (r_k | p_k) = (r_k | r_k + \beta_k p_{k-1}) = \|r_k\|^2 + \beta_k (r_k | p_{k-1}) = \|r_k\|^2.$$

Donc :

$$(20) \quad (r_k | p_k) = \|r_k\|^2.$$

On choisit $p_0 = r_0$ (donc $\beta_0 = 0$) pour que cette relation soit vérifiée quelle que soit $k \geq 0$.

La détermination du maximum se réduit à minimiser $(A p_k | p_k)$.

$$\begin{aligned} (A p_k | p_k) &= (A(r_k + \beta p_{k-1}) | r_k + \beta p_{k-1}) \\ &= \beta^2 (A p_{k-1} | p_{k-1}) + 2\beta (A p_{k-1} | r_k) + (A r_k | r_k) \end{aligned}$$

Pour que ce trinôme soit minimum, il faut choisir β_k vérifiant :

$$\beta_k (A p_{k-1} | p_{k-1}) + (A p_{k-1} | r_k) = 0.$$

Donc :

$$(21) \quad \boxed{\beta_k = -\frac{(A p_{k-1} | r_k)}{(A p_{k-1} | p_{k-1})}}.$$

On en déduit que $(A p_{k-1} | r_k + \beta_k p_{k-1}) = 0$, c'est-à-dire :

$$(22) \quad \boxed{(A p_{k-1} | p_k) = 0}.$$

Lorsque deux vecteurs u, v vérifient la relation $(A u | v) = 0$, on dit qu'ils sont *A-conjugués*. Comme A est symétrique et définie positive, $(A u | v)$ définit un produit scalaire. La relation pour deux vecteurs d'être *A-conjugués* signifie qu'ils sont orthogonaux pour ce produit scalaire.

Proposition 9

On a les relations suivantes, valables si $r_i \neq 0$, $i = 0 à k$:

$$(23) \quad (r_{k+1} | r_k) = 0 \quad k \geq 0 ,$$

$$(24) \quad \beta_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2} \quad \text{pour } k \geq 1 .$$

Preuve:
$$\begin{aligned} (r_{k+1} | r_k) &= (r_k - \alpha_k A p_k | r_k) = \|r_k\|^2 - \alpha_k (A p_k | r_k) \\ &= \|r_k\|^2 - \alpha_k (A p_k | p_k - \beta_k p_{k-1}) \\ &= \|r_k\|^2 - \alpha_k (A p_k | p_k) + \alpha_k \beta_k (A p_k | p_{k-1}) = 0 , \end{aligned}$$

en tenant compte de: $(A p_k | p_{k-1}) = 0 ,$

ainsi que de: $\alpha_k = \frac{(r_k | p_k)}{(A p_k | p_k)} = \frac{\|r_k\|^2}{(A p_k | p_k)} .$

$$A p_{k-1} = \frac{1}{\alpha_{k-1}} (r_{k-1} - r_k) \quad \text{d'après (7), } k \geq 1 ,$$

$$(A p_{k-1} | r_k) = \frac{1}{\alpha_{k-1}} (r_{k-1} - r_k | r_k) = -\frac{1}{\alpha_{k-1}} \|r_k\|^2 ,$$

$$(A p_{k-1} | p_{k-1}) = \frac{1}{\alpha_{k-1}} (r_{k-1} - r_k | p_{k-1}) = \frac{1}{\alpha_{k-1}} (r_{k-1} | p_{k-1}) = \frac{1}{\alpha_{k-1}} \|r_{k-1}\|^2 ,$$

donc: $\beta_k = -\frac{(r_k | A p_{k-1})}{(p_{k-1} | A p_{k-1})} = \frac{\|r_k\|^2}{\|r_{k-1}\|^2} .$

Comme $2 r_k = -g_k$ où g_k est le gradient de la fonctionnelle, on a également:

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2} .$$

■

8.3.2 L'algorithme

On initialise en choisissant x_0 et $p_0 = r_0$:

$$\left\{ \begin{array}{l} x_0, \\ p_0 = r_0 = b - Ax_0, \\ \text{Pour } k = 0, 1 \dots \end{array} \right.$$

$$(25) \quad \left\{ \begin{array}{l} \alpha_k = \frac{\|r_k\|^2}{(A p_k | p_k)}, \\ x_{k+1} = x_k + \alpha_k p_k, \\ r_{k+1} = r_k - \alpha_k A p_k, \\ \beta_{k+1} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}, \\ p_{k+1} = r_{k+1} + \beta_{k+1} p_k. \end{array} \right.$$

Le test d'arrêt des itérations porte sur $\|r_k\|$ comme d'habitude. A chaque itération, le nombre d'opérations est le suivant : (c est le nombre moyen de coefficients non nuls par ligne de A)

	multiplications divisions	additions soustractions
calcul de $q = Ap$	Nc	
produit scalaire ($q p$)	N	$N(c-1)$
α	1	$N-1$
x	N	N
r	N	N
calcul de $\ r\ ^2$	N	$N-1$
β	1	
p	N	N

soit au total $(c+5)N+2$ multiplications,
 $(c+4)N-2$ additions.

Donc un nombre d'opérations voisin de $2cN$ par itération.
 Le coût essentiel est celui du produit Ap .

Pour un nombre d'itérations k de l'ordre de N , on aboutit à un nombre d'opérations voisin de $2cN^2$ ce qui est relativement important notamment lorsque c est grand (si $c=N$, on a $2N^3$ opérations, alors que la méthode de Cholesky n'en nécessite que $\frac{N^3}{3}$).
 77

En fait, on va démontrer que grâce au préconditionnement de A , le nombre d'itérations sera nettement inférieur à N . Cette méthode est alors une des meilleures adaptées à la résolution de système linéaire dont la matrice est symétrique, définie positive et creuse.

Au préalable, on va démontrer les résultats essentiels qui justifient le choix des directions de descente.

Remarque 10

En remplaçant $p_k = r_k + \beta_k p_{k-1} = r_k + \beta_k \left(\frac{x_k - x_{k-1}}{\alpha_{k-1}} \right)$ dans la relation

$$x_{k+1} = x_k + \alpha_k p_k ,$$

on obtient :

$$x_{k+1} = x_k + \alpha_k r_k + \frac{\alpha_k \beta_k}{\alpha_{k-1}} (x_k - x_{k-1}) ,$$

$$x_{k+1} = x_{k-1} + \left(1 + \frac{\alpha_k \beta_k}{\alpha_{k-1}} \right) (x_k - x_{k-1}) + \alpha_k r_k ,$$

soit en posant :

$$\gamma_{k+1} = 1 + \frac{\alpha_k \beta_k}{\alpha_{k-1}} ,$$

$$x_{k+1} = x_{k-1} + \gamma_{k+1} (x_k - x_{k-1}) + \alpha_k (b - A x_k) .$$

On constate que x_{k+1} est calculé à partir de x_k et de x_{k-1} .

8.3.3 Propriétés de l'algorithme

Théorème 11

Dans la méthode du gradient conjugué, en choisissant $p_0 = r_0 = b - A x_0$,

on a les relations suivantes valables pour tout $k \geq 1$ à condition que

$r_i \neq 0$ pour $0 \leq i \leq k$.

$$(26) \quad (r_k | p_i) = 0 \quad \text{pour } i \leq k-1 ,$$

$$(27) \quad \mathcal{E}(r_0, r_1, \dots, r_k) = \mathcal{E}(r_0, A r_0, \dots, A^k r_0) ,$$

$$(28) \quad \mathcal{E}(p_0, p_1, \dots, p_k) = \mathcal{E}(r_0, A r_0, \dots, A^k r_0) ,$$

$$(29) \quad (p_k | A p_i) = (A p_k | p_i) = 0 \quad \text{pour } i \leq k-1 ,$$

$$(30) \quad (r_k | r_i) = 0 \quad \text{pour } i \leq k-1 ,$$

3.5 Krylov + gradient conj + gmres

The image shows the cover of a chapter titled "Chapitre 10" in a dark blue circle. The title of the chapter is "Généralités sur les espaces de Krylov". Below the title, there is a box containing the word "Introduction". Inside this box, a text box contains the following text:

Les méthodes itératives calculent une correction de la solution approchée à l'aide du vecteur résidu, qui est la différence entre le produit de la matrice par le vecteur solution approchée et le second membre. L'opération fondamentale à chaque itération est donc le calcul d'un produit matrice-vecteur. Les espaces de Krylov sont des espaces affines construits à partir de la solution initiale et de produits successifs par la matrice en partant du résidu initial. Ces espaces sont très pertinents pour chercher la solution approchée du problème en ne réalisant que des produits matrice-vecteur, des produits scalaires ou des combinaisons linéaires de vecteurs.

Objectifs

- Connaître les propriétés des espaces de Krylov.
- Définir l'algorithme d'Arnoldi pour construire une base numériquement stable d'un espace de Krylov.
- Expliquer pourquoi il faut orthogonaliser la base de l'espace de Krylov et pourquoi la recherche de la solution approchée dans l'espace de Krylov est la meilleure méthodologie de résolution itérative.

Plan

- 1 Espaces de Krylov
- 2 Construction de la base d'Arnoldi

1 Espaces de Krylov

Si x_0 est une valeur de départ approchée de la solution de l'équation

$$Ax = b \quad (10.1)$$

le vecteur résidu associé à x_0 vaut $g_0 = Ax_0 - b$.

DÉFINITION 10.1

On appelle espace de Krylov d'ordre p , noté \mathcal{K}_p , l'espace vectoriel généré par g_0 et ses $p - 1$ produits itérés par A :

Chapitre 10 • Généralités sur les espaces de Krylov

$$\mathcal{K}_p = \text{Vect}\{g_0, Ag_0, A^2g_0, \dots, A^{p-1}g_0\}.$$

Les espaces de Krylov forment évidemment une famille croissante de sous-espaces nécessaires et suffisantes. On va noter p_{\max} , la dimension maximale des espaces de Krylov pour un x_0 donné.

Lemme 10.1

Si $A^pg_0 \in \mathcal{K}_p$ alors $A^{p+q}g_0 \in \mathcal{K}_p$ pour tout $q > 0$.

Démonstration. La démonstration se fait par récurrence. Si pour un $q \geq 0$, $A^pg_0, A^{p+1}g_0, \dots, A^{p+q}g_0 \in \mathcal{K}_p$, alors :

$$A^{p+q}g_0 = \sum_{k=0}^{p-1} \alpha_k A^k g_0,$$

et donc :

$$\begin{aligned} A^{p+q+1}g_0 &= \sum_{k=0}^{p-2} \alpha_k A^{k+1}g_0 + \alpha_{p-1} A^p g_0 \\ &= \sum_{k=0}^{p-2} \alpha_k A^{k+1}g_0 + \alpha_{p-1} \sum_{k=0}^{p-1} \beta_k A^k g_0 \\ &= \sum_{k=0}^{p-1} \gamma_k A^k g_0 \end{aligned}$$

Lemme 10.2

La suite des espaces de Krylov \mathcal{K}_p est strictement croissante de 1 à p_{\max} puis stagne à partir de $p = p_{\max}$.

Démonstration. Si p est le plus petit entier pour lequel A^pg_0 est dépendant des vecteurs précédents, alors, les vecteurs $(g_0, Ag_0, A^2g_0, \dots, A^{p-1}g_0)$ sont linéairement indépendants et donc \mathcal{K}_p est de dimension q , pour tout $q \leq p$. En particulier \mathcal{K}_p est de dimension p .

De plus, $A^pg_0 \in \mathcal{K}_p$ et, d'après le lemme (10.1), tous les vecteurs $A^{p+q}g_0$ appartiennent à \mathcal{K}_p , pour tout $q > 0$, de sorte que $\mathcal{K}_{p+q} = \mathcal{K}_p$, pour tout $q > 0$.

On a donc : $\mathcal{K}_1 \subsetneq \dots \subsetneq \mathcal{K}_p = \mathcal{K}_{p+q}$, pour tout $q > 0$. Et, par définition de p_{\max} , nécessairement $p = p_{\max}$.

2 Construction de la base d'Arnoldi

Ceci conduit au théorème fondamental suivant.

Théorème 10.3

La solution du système linéaire $Ax = b$ appartient à l'espace affine $x_0 + \mathcal{K}_{p_{max}}$.

Démonstration. D'après les deux lemmes (10.1) et (10.2), les vecteurs

$$(g_0, Ag_0, A^2g_0, \dots, A^{p_{max}-1}g_0)$$

sont linéairement indépendants et :

$$A^{p_{max}}g_0 = \sum_{k=0}^{p_{max}-1} \alpha_k A^k g_0 \quad (10.2)$$

Dans l'équation (10.2), le coefficient α_0 n'est pas nul, sans quoi, en multipliant les deux termes de l'équation par A^{-1} , on obtient :

$$A^{p_{max}-1}g_0 = \sum_{k=1}^{p_{max}-1} \alpha_k A^{k-1}g_0$$

ce qui est impossible, étant donné la propriété d'indépendance des vecteurs. En divisant les deux termes de l'équation (10.2) par α_0 et en faisant passer le membre de gauche à droite, on obtient :

$$\begin{aligned} g_0 + \sum_{k=1}^{p_{max}-1} \frac{\alpha_k}{\alpha_0} A^k g_0 - \frac{1}{\alpha_0} A^{p_{max}} g_0 &= 0 \Leftrightarrow \\ Ax_0 - b + \sum_{k=1}^{p_{max}-1} \frac{\alpha_k}{\alpha_0} A^k g_0 - \frac{1}{\alpha_0} A^{p_{max}} g_0 &= 0 \Leftrightarrow \\ A(x_0 + \sum_{k=1}^{p_{max}-1} \frac{\alpha_k}{\alpha_0} A^{k-1}g_0 - \frac{1}{\alpha_0} A^{p_{max}-1}g_0) &= b \end{aligned}$$

■

Remarque Les équivalences ci-dessus montrent réciproquement que, si $x \in x_0 + \mathcal{K}_p$, alors les vecteurs $(g_0, Ag_0, A^2g_0, \dots, A^pg_0)$ sont linéairement dépendants. D'après le lemme (10.2), cela signifie que la suite des espaces \mathcal{K}_p a atteint son point de stagnation et donc $\mathcal{K}_p = \mathcal{K}_{p_{max}}$.

2 Construction de la base d'Arnoldi

En pratique, construire les espaces de Krylov revient à en déterminer des bases. La base naturelle $(g_0, Ag_0, A^2g_0, \dots, A^{p-1}g_0)$ ne peut en aucun cas être utilisée, du fait de sa

dégénérescence numérique. En effet, si la matrice A est diagonalisable, avec m valeurs propres distinctes $\lambda_1, \dots, \lambda_m$, alors :

$$g_0 = \sum_{i=1}^m \alpha_i v_i, \quad \text{et} \quad A^p g_0 = \sum_{i=1}^m \alpha_i \lambda_i^p v_i.$$

La suite $A^p g_0$ se comporte donc comme $\alpha_{\max} \lambda_{\max}^p v_{\max}$, si λ_{\max} est la plus grande valeur propre. Numériquement, dès que le facteur $(\frac{\alpha_{\max}}{\alpha_i})(\frac{\lambda_{\max}}{\lambda_i})^p$ dépasse l'ordre de grandeur 10^r , où r est le nombre de chiffres significatifs de la représentation des réels en machine, le terme $\alpha_i \lambda_i^p v_i$ disparaît complètement. Ce qui arrive très vite en pratique. Pour un rapport de 10 entre les valeurs propres, il suffit de 16 itérations pour la représentation classique des nombres réels en 64 bits. Les vecteurs $A^p g_0$ deviennent donc très rapidement colinéaires numériquement. On peut démontrer aisément le même phénomène pour n'importe quelle matrice, en utilisant la mise sous forme de Jordan. Par ailleurs, selon que les normes des valeurs propres sont inférieures ou supérieures à 1, les termes $\alpha_i \lambda_i^p v_i$ deviennent trop petits ou trop grands pour être représentables.

La solution pour éviter la dégénérescence numérique de la base naturelle de l'espace de Krylov consiste à mettre en œuvre une procédure d'orthonormalisation. La base d'Arnoldi est ainsi construite par application de la méthode d'orthonormalisation de Gram-Schmidt modifiée aux vecteurs obtenus par produits successifs par la matrice, pour le produit scalaire euclidien. La méthode de Gram-Schmidt modifiée s'écrit donc :

- Initialisation de la base d'Arnoldi :

$$\begin{aligned} g_0 &= Ax_0 - b \\ v_1 &= \frac{1}{\|g_0\|} g_0 \end{aligned}$$

- Construction du vecteur numéro $j + 1$ de la base d'Arnoldi :

```
w = Av_j
for i = 1 to j
    alpha_i = (w.v_i)
    w = w - alpha_i v_i
end for
v_{j+1} = w / \|w\|
```

Si on note h_{ij} le coefficient d'orthogonalisation de Av_j par rapport à v_i et $h_{j+1,j}$ la norme du vecteur w obtenu par orthogonalisation du vecteur Av_j par rapport aux vecteurs v_i , les formules de construction de la base d'Arnoldi s'écrivent :

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i \tag{10.3}$$

Soit V_p , la matrice rectangulaire à n lignes et p colonnes, dont les colonnes sont les p premiers vecteurs de la base d'Arnoldi, alors, l'orthonormalité des vecteurs s'écrit matriciellement :

$$V_p^T V_p = I_p \tag{10.4}$$

2 Construction de la base d'Arnoldi

De même, l'équation (10.3) définissant les vecteurs d'Arnoldi s'écrit :

$$AV_p = V_{p+1} H_{p+1p} \quad (10.5)$$

où la matrice H_{p+1p} est une matrice à $p+1$ lignes et p colonnes dont les coefficients h_{ij} sont les coefficients d'orthonormalisation de l'équation (10.3), pour $i \leq j+1$, les autres coefficients étant nuls.

$$H_{p+1p} = \begin{pmatrix} h_{11} & h_{12} & \vdots & \vdots & h_{1j} & \vdots & h_{1p} \\ h_{21} & h_{22} & \vdots & \vdots & h_{2j} & \vdots & h_{2p} \\ 0 & h_{32} & \vdots & \vdots & h_{3j} & \vdots & h_{3p} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & & & \ddots & h_{j+1j} & \vdots & h_{j+1p} \\ \vdots & & & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & h_{p+1p} \end{pmatrix} \quad (10.6)$$

Le bloc diagonal principal de H_{p+1p} est une matrice carrée de dimension p , notée H_p , qui, d'après les équations (10.4) et (10.5), vérifie :

$$H_p = V_p^t A V_p = \begin{pmatrix} h_{11} & h_{12} & \vdots & \vdots & \vdots & h_{1p} \\ h_{21} & h_{22} & \vdots & \vdots & \vdots & h_{2p} \\ 0 & h_{32} & \vdots & \vdots & \vdots & h_{3p} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & h_{pp-1} & h_{pp} \end{pmatrix} \quad (10.7)$$

DÉFINITION 10.2

Une telle matrice, dont la partie triangulaire inférieure est nulle, à l'exception des termes situés sur la première sous-diagonale, est dite de « forme Hessenberg » supérieure.

L'équation (10.7) montre que la matrice H_p est la matrice, dans la base des vecteurs d'Arnoldi, de la projection dans l'espace de Krylov \mathcal{K}_p de l'application linéaire associée à la matrice A .

Méthodes avec orthogonalisation complète

Introduction

Pour construire une base orthogonale de l'espace de Krylov, il faut a priori réorthogonaliser chaque nouveau vecteur par rapport à tous les vecteurs précédents qui doivent donc être conservés. Cette opération coûte cher en complexité arithmétique et en mémoire. Cependant, dans le cas d'une matrice symétrique, la construction de la base d'Arnoldi devient une récurrence courte. À partir de cette observation il est possible de définir des méthodes pour lesquelles on peut calculer par récurrence courte non seulement des bases de l'espace de Krylov mais aussi une solution optimale pour une norme adaptée dans le cas où la matrice est en plus définie positive. La convergence de ces méthodes dépend du conditionnement de la matrice. Pour l'améliorer, on peut utiliser toute approximation, même de faible précision, du produit par l'inverse de la matrice pour préconditionner le système linéaire.

Objectifs

Connaitre les méthodes de Krylov optimales pour les matrices symétriques définies positives.

Identifier la norme la plus adaptée pour chercher la solution optimale dans le cas d'une matrice symétrique définie positive.

Définir l'algorithme de Lanczos pour calculer une base de l'espace de Krylov par récurrence courte et la méthode du gradient conjugué pour avoir en plus la solution optimale par récurrence courte.

Expliquer l'effet du conditionnement sur la convergence des méthodes et comment y remédier.

Plan

- 1 Construction de la base de Lanczos pour des matrices symétriques
- 2 Méthode de Lanczos
- 3 Méthode du gradient conjugué
- 4 Comparaison avec la méthode du gradient
- 5 Principe du préconditionnement pour des matrices symétriques définies positives

1 Construction de la base de Lanczos pour des matrices symétriques

Dans le cas particulier où la matrice A est symétrique, la matrice H_p l'est aussi, d'après l'équation (10.7). Comme elle est aussi de forme Hessenberg supérieure, on en déduit immédiatement qu'elle est tri-diagonale symétrique.

L'algorithme de construction de la base d'Arnoldi se simplifie alors énormément, puisque l'on sait que les coefficients h_{ij} sont nuls, si $i < j - 1$ et que le coefficient h_{j-1j} est égal à h_{jj-1} , précédemment calculé. L'algorithme de construction d'une base orthonormée des espaces de Krylov qui en découle s'appelle algorithme de *Lanczos*.

• Initialisation de la base de Lanczos :

$$g_0 = Ax_0 - b$$

$$v_1 = \frac{1}{\|g_0\|} g_0$$

• Construction du vecteur numéro $j + 1$ de la base de Lanczos :

$$w = Av_j$$

$$h_{j-1j} = h_{jj-1}$$

$$w = w - h_{j-1j}v_{j-1}$$

$$h_{jj} = (w, v_j)$$

$$w = w - h_{jj}v_j$$

$$h_{j+1j} = \|w\|$$

$$v_{j+1} = \frac{1}{h_{j+1j}} w$$

L'algorithme de Lanczos possède la propriété remarquable de permettre la construction d'une base orthonormée des espaces de Krylov à l'aide d'une récurrence courte. Il suffit en effet de disposer des seuls vecteurs v_{j-1} et v_j pour pouvoir calculer v_{j+1} et le coût de calcul est indépendant du numéro de l'itération.

2 Méthode de Lanczos

Une fois déterminée une base V_p de l'espace de Krylov \mathcal{K}_p à l'itération p , il reste à déterminer la solution approchée x_p du système (10.1) dans l'espace affine $x_0 + \mathcal{K}_p$. Puisque V_p est une base de \mathcal{K}_p , x_p s'écrit sous la forme suivante, où z_p est un vecteur de dimension p :

$$x_p = x_0 + V_p z_p \quad (11.1)$$

Les vecteurs erreur et résidu correspondants s'écrivent donc respectivement :

$$e_p = x_p - x = x_0 - x + V_p z_p = e_0 + V_p z_p \quad (11.2)$$

$$\begin{aligned} e_p &= x_p - x = x_0 - x + V_p z_p = e_0 + V_p z_p \\ g_p &= Ax_p - b = Ae_p = Ae_0 + AV_p z_p = g_0 + AV_p z_p \end{aligned} \quad (11.3)$$

2 Méthode de Lanczos

Une méthode de Krylov est définie, d'une part par l'algorithme de construction de la base des espaces de Krylov, et, d'autre part, par le critère d'optimalité choisi pour déterminer la solution approchée x_p . A priori, l'idéal serait de minimiser l'erreur ou le résidu pour une norme adaptée.

Dans le cas où la matrice A est symétrique définie positive, le choix fait pour la « méthode de Lanczos » consiste à minimiser la norme de l'erreur pour le produit scalaire associé à A , ce qui revient à minimiser le produit scalaire des vecteurs erreur et résidu et donc, en quelque sorte, à contrôler les deux quantités simultanément :

$$\mathcal{E}(x_p) = \|x_p - x\|_A^2 = (A(x_p - x) \cdot (x_p - x)) = (g_p \cdot e_p) \quad (11.4)$$

La solution approchée ainsi définie possède des propriétés très intéressantes.

Théorème 11.1

La solution approchée x_p de la méthode de Lanczos est la projection de x dans $x_0 + \mathcal{K}_p$, pour le produit scalaire associé à A .

Démonstration. D'après l'équation (11.4), x_p est l'élément de $x_0 + \mathcal{K}_p$ dont la distance à x est minimale pour le produit scalaire associé à A . ■

Corollaire 11.1

Le vecteur résidu $g_p = Ax_p - b$ de la méthode de Lanczos est orthogonal à \mathcal{K}_p .

Démonstration. Les propriétés d'une projection dans un espace affine impliquent :

$$(A(x_p - x) \cdot w_p) = 0, \quad \forall w_p \in \mathcal{K}_p. \quad ■$$

On va maintenant montrer comment calculer x_p en pratique. D'après les équations (11.2) et (11.3), on a :

$$\begin{aligned} \mathcal{E}(x_p) &= (A(e_0 + V_p z_p) \cdot (e_0 + V_p z_p)) \\ &= (AV_p z_p \cdot V_p z_p) + 2(g_0 \cdot V_p z_p) + (g_0 \cdot e_0) \end{aligned} \quad (11.5)$$

de sorte que, pour minimiser $\mathcal{E}(x_p)$, il suffit de minimiser la quantité qui dépend de z_p dans l'équation (11.5) et qui s'écrit :

$$\begin{aligned} (AV_p z_p \cdot V_p z_p) + 2(g_0 \cdot V_p z_p) &= (V_p^t A V_p z_p \cdot z_p) + 2(V_p^t g_0 \cdot z_p) \\ &= 2\left(\frac{1}{2}(V_p^t A V_p z_p \cdot z_p) + (V_p^t g_0 \cdot z_p)\right) \end{aligned} \quad (11.6)$$

Finalement, le problème revient à minimiser la quantité :

$$\begin{aligned} J_p(z_p) &= \frac{1}{2}(V_p^t A V_p z_p \cdot z_p) + (V_p^t g_0 \cdot z_p) \\ &= \frac{1}{2}(T_p z_p \cdot z_p) - (y_p \cdot z_p) \end{aligned} \quad (11.7)$$

où $T_p = H_p$ est la matrice tri-diagonale symétrique des coefficients d'orthogonalisation calculés avec la base de Lanczos V_p et, y_p , le vecteur de dimension p dont le coefficient numéro i est égal au produit scalaire $-(v_i, g_0)$.

On est ramené à un problème de minimisation en dimension finie classique, pour lequel on va rappeler les résultats d'existence et d'unicité de la solution.

Lemme 11.2

Soit A une matrice symétrique définie positive, la fonctionnelle quadratique

$$\mathcal{J}(x) = \frac{1}{2}(Ax, x) - (b, x)$$

est strictement convexe.

Démonstration. La fonctionnelle quadratique évaluée en $(\alpha x + (1 - \alpha)y)$ donne :

$$\begin{aligned} \mathcal{J}(\alpha x + (1 - \alpha)y) &= \frac{1}{2}\alpha^2(Ax, x) + \alpha(1 - \alpha)(Ax, y) \\ &\quad + \frac{1}{2}(1 - \alpha)^2(Ay, y) - \alpha(b, x) - (1 - \alpha)(b, y) \\ &= \alpha\mathcal{J}(x) + (1 - \alpha)\mathcal{J}(y) \\ &\quad + \frac{1}{2}[(\alpha^2 - \alpha)(Ax, x) + 2\alpha(1 - \alpha)(Ax, y) \\ &\quad + ((1 - \alpha)^2 - (1 - \alpha))(Ay, y)] \\ &= \alpha\mathcal{J}(x) + (1 - \alpha)\mathcal{J}(y) \\ &\quad + \frac{1}{2}\alpha(\alpha - 1)[(Ax, x) - 2(Ax, y) + (Ay, y)] \end{aligned}$$

Puisque A est définie positive,

$$(Ax, x) - 2(Ax, y) + (Ay, y) = (A(x - y), (x - y)) > 0, \quad \text{si } x \neq y.$$

Si $\alpha \in]0, 1[$, alors $\alpha(\alpha - 1) < 0$, et donc :

$$\frac{1}{2}\alpha(\alpha - 1)[(Ax, x) - 2(Ax, y) + (Ay, y)] < 0.$$

Théorème 11.3

La fonctionnelle $\mathcal{J}(x) = \frac{1}{2}(Ax, x) - (b, x)$ admet un minimum absolu au point x qui vérifie $Ax = b$.

Démonstration. \mathcal{J} est strictement convexe et bornée inférieurement puisque $\mathcal{J}(x) \rightarrow +\infty$ lorsque $\|x\| \rightarrow +\infty$. Elle est évidemment différentiable et sa différentielle vaut :

$$D\mathcal{J}(x).y = (Ax, y) - (b, y) = ((Ax - b), y) \tag{11.8}$$

La fonctionnelle a donc bien un minimum absolu au point unique où sa différentielle s'annule, c'est-à-dire au point où $(Ax - b)$ vaut 0.

2 Méthode de Lanczos

Remarque L'équation (11.8) montre que le gradient de \mathcal{J} en un point x vérifie :

$$\nabla \mathcal{J}(x) = Ax - b \quad (11.9)$$

Ceci justifie la notation g_p , pour le vecteur $Ax_p - b$, qui est effectivement le gradient de la fonctionnelle \mathcal{J} associée à la matrice du système linéaire (10.1), et qui sera appelé vecteur gradient dans la suite.

Corollaire 11.2

Le minimum de la quantité $\mathcal{E}(x_p)$ définie à l'équation (11.5) est atteint au point

$$x_p = x_0 + V_p z_p, \quad z_p \text{ étant la solution du système :}$$

$$T_p z_p = y_p \quad (11.10)$$

Démonstration. D'après le théorème 11.3 et les équations (11.6) et (11.7), il suffit de démontrer que la matrice T_p est définie positive, ce qui découle immédiatement du fait que A l'est aussi et que les colonnes de V_p forment une famille libre de vecteurs. D'où il ressort :

$$(T_p z_p, z_p) = (AV_p z_p, V_p z_p) \geq \alpha \|V_p z_p\|^2 \geq \alpha \beta \|z_p\|^2.$$

La matrice T_p est très facile à factoriser, puisqu'elle est tri-diagonale symétrique définie positive. Elle admet en particulier une factorisation de Crout, de la forme $T_p = L_p D_p L_p^t$, où L_p est une matrice triangulaire inférieure bi-diagonale avec des coefficients égaux à 1 sur la diagonale. Qui plus est, comme T_{p-1} est le bloc diagonal principal de T_p , la factorisation de T_p se calcule immédiatement à partir de celle de T_{p-1} .

$$\begin{aligned} T_p &= \begin{pmatrix} &&& \begin{pmatrix} 0 \\ \vdots \\ 0 \\ t_{p-1p} \\ t_{pp} \end{pmatrix} \\ T_{p-1} &&& \\ \begin{pmatrix} 0 & \cdots & 0 & t_{pp-1} \end{pmatrix} & \end{pmatrix} \\ &= \begin{pmatrix} &&& \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ L_{p-1} &&& \\ \begin{pmatrix} 0 & \cdots & 0 & t_{pp-1} \end{pmatrix} & \end{pmatrix} \begin{pmatrix} &&& \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u_{p-1p} \\ d_p \end{pmatrix} \\ D_{p-1} L_{p-1}^t &&& \\ \begin{pmatrix} 0 & \cdots & 0 & 0 \end{pmatrix} & \end{pmatrix} \end{aligned}$$

Par identification, $u_{p-1p} = t_{p-1p}$, $t_{pp-1} = t_{pp-1}/d_{p-1}$ et $d_p = t_{pp} - t_{pp-1} u_{p-1p}$.

Chapitre 11 • Méthodes avec orthogonalisation complète

De même, la mise à jour du vecteur y_p ne demande que le calcul de la dernière composante, $y_p(p) = -(v_p \cdot g_0)$, les $p - 1$ premières étant les mêmes (que celles du vecteur y_{p-1}).

Finalement, l'itération p de la méthode de Lanczos consiste à déterminer la dernière composante du vecteur y_p , calculer le nouveau vecteur v_{p+1} et compléter la factorisation de Crout de T_p à partir de celle de T_{p-1} selon l'algorithme suivant.

$$\begin{aligned}y_p(p) &= -(v_p \cdot g_0) \\w &= Av_p \\w &= w - t_{pp-1}v_{p-1} \\t_{pp} &= (w \cdot v_p) \\w &= w - t_{pp}v_p \\t_{p+1p} &= \|w\| \\v_{p+1} &= \frac{1}{t_{p+1p}}w \\l_{pp-1} &= t_{pp-1}/d_{p-1} \\d_p &= t_{pp} - l_{pp-1}t_{pp-1}\end{aligned}$$

Pour déterminer la solution approchée x_p , il suffit de résoudre le système linéaire (11.10), en utilisant la factorisation de Crout de T_p , puis de calculer x_p selon la formule (11.1).

Le choix du critère d'optimalité $\mathcal{E}(x_p)$ de l'équation (11.4) permet donc de déterminer très simplement la solution approchée en résolvant un système linéaire associé à la matrice T_p , qui est tri-diagonale symétrique. Néanmoins, la méthode souffre d'un inconvénient grave : alors que les vecteurs de la base de Lanczos se calculent par une récurrence courte, le calcul de la solution approchée demande de conserver tous les vecteurs de la base. Le coût de chaque itération augmente donc linéairement avec le nombre d'itérations, tant en terme du volume des données que du nombre d'opérations arithmétiques.

3 Méthode du gradient conjugué

La méthode de Lanczos serait bien plus intéressante si la remise à jour de la solution approchée pouvait être réalisée à l'aide d'une récurrence courte. C'est-à-dire qu'il faudrait que les premières composantes de z_p soient identiques à celles de z_{p-1} , ce qui conduirait à une formule de type :

$$x_p = x_{p-1} + \alpha_p v_p$$

Pour y parvenir, il faudrait une base W_p de l'espace de Krylov \mathcal{K}_p pour laquelle la matrice de projection $W_p^T A W_p$ soit diagonale. Ainsi, les $p - 1$ premières composantes de la solution du problème d'optimisation d'ordre p :

$$W_p^T A W_p x_p = y_p \quad (11.11)$$

seraient effectivement identiques à celle du problème d'ordre $p - 1$.

3 Méthode du gradient conjugué

Par ailleurs, on ne peut pas calculer effectivement $E(x_p)$, puisque l'erreur initiale e_0 est évidemment inconnue. Le seul moyen de contrôler la convergence consiste à calculer la norme du gradient, $g_p = Ax_p - b$. Le test d'arrêt des itérations de la méthode porte donc sur le résidu adimensionnalisé :

$$\|Ax_p - b\|/\|b\| < \epsilon. \quad (11.12)$$

Il faut donc de toutes les manières calculer les gradients successifs. Or, d'après l'équation (11.3), le vecteur g_p appartient à \mathcal{K}_{p+1} et on a démontré au corollaire 11.1, qu'il est dans \mathcal{K}_p^\perp . Par construction, le complémentaire orthogonal de \mathcal{K}_p dans \mathcal{K}_{p+1} est la droite générée par v_{p+1} , et donc $g_p = \delta v_{p+1}$. Plutôt que la base orthonormée des vecteurs v_p , on peut utiliser la base orthogonale des vecteurs gradients successifs. Ceux-ci tendent vers 0, mais les itérations seront de toute façon arrêtées par le test défini par l'équation (11.12), bien avant que n'apparaissent des problèmes de représentation.

Soit donc G_p , la matrice rectangulaire à n lignes et p colonnes, dont les colonnes sont les p premiers vecteurs gradients, $(g_0, g_1, \dots, g_{p-1})$. Comme on vient de le voir, $G_p = V_p \Delta_p$, Δ_p étant une matrice diagonale. La matrice de projection dans la base G_p est aussi tridiagonale symétrique définie positive :

$$G_p^t A G_p = \Delta_p^t V_p^t A V_p \Delta_p = \Delta_p^t T_p \Delta_p = \tilde{T}_p \quad (11.13)$$

Elle admet donc une factorisation de Crout $\tilde{T}_p = \tilde{L}_p \tilde{D}_p \tilde{L}_p^t$. On en déduit :

$$G_p^t A G_p = \tilde{L}_p \tilde{D}_p \tilde{L}_p^t \Leftrightarrow \tilde{L}_p^{-1} G_p^t A G_p \tilde{L}_p^{-t} = \tilde{D}_p \quad (11.14)$$

L'équation (11.14) implique que les vecteurs colonnes de la matrice :

$$W_p = G_p \tilde{L}_p^{-t} \quad (11.15)$$

qui sont des combinaisons linéaires des vecteurs colonnes de G_p , forment une famille orthogonale pour le produit scalaire associé à A . W_p est donc une base A -orthogonale de \mathcal{K}_p .

Puisque la matrice de projection $W_p^t A W_p$ est diagonale, W_p fournit la base ad-hoc pour que le problème d'optimisation (11.4) puisse être résolu par une récurrence courte. De plus, l'équation (11.15) montre que les vecteurs de W_p se calculent à l'aide d'une récurrence courte à l'aide de ceux de G_p , puisque :

$$W_p \tilde{L}_p^t = G_p \quad (11.16)$$

En effet, si, afin de conserver une certaine cohérence dans les indices, on note $(w_0, w_1, \dots, w_{p-1})$ les vecteurs colonnes de W_p et $(-\gamma_0, -\gamma_1, \dots)$ les coefficients sous-diagonaux de la matrice \tilde{L}_p , l'équation (11.16) signifie :

$$w_0 = g_0 \text{ et } -\gamma_{j-1} w_{j-1} + w_j = g_j, \forall j > 0 \quad (11.17)$$

Les différentes relations connues entre les vecteurs x_p , g_p et w_p permettent finalement d'écrire directement une nouvelle méthode sans passer par le calcul des vecteurs de

Chapitre 11 • Méthodes avec orthogonalisation complète

Lanczos ni la factorisation de Crout de la matrice de projection T_p . En effet, x_0 étant choisi, on a nécessairement, d'après l'équation (11.17) :

$$g_0 = Ax_0 - b \text{ et } w_0 = g_0$$

Du fait des propriétés de la base W_p , on a :

$$x_p = x_{p-1} + \rho_{p-1}w_{p-1} \Leftrightarrow g_p = g_{p-1} + \rho_{p-1}Aw_{p-1} \quad (11.18)$$

On sait que g_p est orthogonal à \mathcal{K}_p et donc en particulier à w_{p-1} . Cette relation d'orthogonalité définit le coefficient ρ_{p-1} de manière unique, puisque d'après l'équation (11.18) :

$$(g_p \cdot w_{p-1}) = (g_{p-1} \cdot w_{p-1}) + \rho_{p-1}(Aw_{p-1} \cdot w_{p-1}) = 0 \Leftrightarrow \rho_{p-1} = -\frac{(g_{p-1} \cdot w_{p-1})}{(Aw_{p-1} \cdot w_{p-1})} \quad (11.19)$$

Enfin, d'après l'équation (11.17) le nouveau vecteur w_p se calcule à l'aide de w_{p-1} et de g_p :

$$w_p = g_p + \gamma_{p-1}w_{p-1}$$

Le coefficient γ_{p-1} est lui aussi déterminé de manière unique par la relation de A -orthogonalité entre w_p et \mathcal{K}_p , et plus particulièrement entre w_p et w_{p-1} :

$$(w_p \cdot Aw_{p-1}) = (g_p \cdot Aw_{p-1}) + \gamma_{p-1}(w_{p-1} \cdot Aw_{p-1}) = 0 \Leftrightarrow \gamma_{p-1} = -\frac{(g_p \cdot Aw_{p-1})}{(Aw_{p-1} \cdot w_{p-1})} \quad (11.20)$$

La méthode ainsi définie est la méthode du « gradient conjugué », ainsi nommée car la base des vecteurs w_p est construite par conjugaison, ce qui signifie la A -orthogonalisation, des vecteurs gradients.

- Initialisation du gradient conjugué :

$$\begin{aligned} g_0 &= Ax_0 - b \\ w_0 &= g_0 \end{aligned}$$

- Itération numéro p du gradient conjugué :

```

 $v = Aw_{p-1}$ 
 $\rho_{p-1} = -(g_{p-1} \cdot w_{p-1})/(v \cdot w_{p-1})$ 
 $x_p = x_{p-1} + \rho_{p-1}w_{p-1}$ 
 $g_p = g_{p-1} + \rho_{p-1}v$ 
if  $(g_p \cdot g_p)/(b \cdot b) < \epsilon^2$  then
    Fin
end if
 $\gamma_{p-1} = -(g_p \cdot v)/(v \cdot w_{p-1})$ 
 $w_p = g_p + \gamma_{p-1}w_{p-1}$ 

```

À chaque itération, il suffit de calculer un produit matrice-vecteur, $v = Aw_{p-1}$, quatre produits scalaires, $(g_{p-1} \cdot w_{p-1})$, $(v \cdot w_{p-1})$, $(g_p \cdot g_p)$ et $(g_p \cdot v)$, et trois combinaisons linéaires de vecteurs, $x_p = x_{p-1} + \rho_{p-1}w_{p-1}$, $g_p = g_{p-1} + \rho_{p-1}v$ et $w_p = g_p - \gamma_{p-1}w_{p-1}$.

L'algorithme du gradient conjugué est optimal dans ce sens que les différents vecteurs sont calculés par une récurrence courte, et de ce fait le coût des itérations est constant.

④ Comparaison avec la méthode du gradient

A priori, en ne faisant que p produits par la matrice A , on ne peut pas construire en général un meilleur espace d'approximation que $x_0 + \mathcal{K}_p$, et la solution approchée x_p calculée par la méthode est le point le plus proche dans cet espace affine de la solution x , au sens du produit scalaire associé à A .

Remarque L'algorithme du gradient conjugué construit deux bases de l'espace de Krylov \mathcal{K}_p . La base des vecteurs gradients $(g_0, g_1, \dots, g_{p-1})$ qui sont orthogonaux, et la base des vecteurs $(w_0, w_1, \dots, w_{p-1})$, qualifiés de directions de « descente », qui sont conjuguées, c'est-à-dire A -orthogonaux. Par construction, g_p est orthogonal à \mathcal{K}_p et w_p est A -orthogonal à \mathcal{K}_p .

④ Comparaison avec la méthode du gradient

La méthode du gradient est une méthode d'optimisation appliquée à la minimisation de la fonctionnelle $J(x) = \frac{1}{2}(Ax.x) - (b.x)$. Partant d'un état x_{p-1} , dans une direction w_{p-1} , on cherche le point $x_p = x_{p-1} + \rho_{p-1}w_{p-1}$ qui minimise J sur la droite affine. La fonction $f(\rho) = J(x_{p-1} + \rho w_{p-1})$ est la combinaison d'une fonction affine et d'une fonction strictement convexe, bornée inférieurement et différentiable. Elle est donc elle aussi strictement convexe, bornée inférieurement et dérivable et atteint son minimum au point où sa dérivée s'annule. Sa dérivée vaut :

$$\begin{aligned} f'(\rho) &= D\mathcal{J}(x_{p-1} + \rho w_{p-1}) \cdot w_{p-1} \\ &= (\nabla \mathcal{J}(x_{p-1} + \rho w_{p-1}) \cdot w_{p-1}) \\ &= ((Ax_{p-1} + \rho Aw_{p-1} - b) \cdot w_{p-1}) \\ &= ((g_{p-1} + \rho Aw_{p-1}) \cdot w_{p-1}) \\ &= (g_{p-1} \cdot w_{p-1}) + \rho(Aw_{p-1} \cdot w_{p-1}) \end{aligned}$$

Et donc le minimum de f est atteint au point ρ_{p-1} défini par :

$$f'(\rho_{p-1}) = 0 \Leftrightarrow \rho_{p-1} = -\frac{(g_{p-1} \cdot w_{p-1})}{(Aw_{p-1} \cdot w_{p-1})}$$

On retrouve la même équation que pour le gradient conjugué (11.19), ce qui donne deux définitions équivalentes de ρ_{p-1} , comme le coefficient tel que le gradient g_p est orthogonal à la direction w_{p-1} ou comme celui pour lequel x_p minimise J sur la droite affine passant par x_{p-1} et de direction w_{p-1} .

Reste à définir la méthode de sélection de la direction w_{p-1} . L'objectif est de faire décroître $J(x)$ le plus rapidement possible. D'après la définition de la différentielle de J à l'équation (11.8), la direction de plus forte variation de J à partir du point x_{p-1} est celle du gradient en ce point : $g_{p-1} = Ax_{p-1} - b$.

La méthode du gradient consiste à choisir comme direction de descente, $w_{p-1} = g_{p-1}$. La méthode du gradient est a priori moins rapide que la méthode du gradient conjugué,

Chapitre 11 • Méthodes avec orthogonalisation complète

puisque celle-ci, en assurant l'optimisation par rapport à des directions A -orthogonales, fait en sorte de calculer le minimum dans tout l'espace généré par les directions successives, alors que la méthode du gradient ne réalise cette optimisation que direction par direction et en quelque sorte « oublie » les itérations précédentes.

On peut aisément montrer que la méthode du gradient assure une convergence géométrique du carré de la distance de x à x_p pour le produit scalaire associé à A , définie à l'équation (11.4). En effet, pour la méthode du gradient :

$$x_p = x_{p-1} + \rho_{p-1} g_{p-1} \text{ avec } \rho_{p-1} = -\frac{(g_{p-1} \cdot g_{p-1})}{(A g_{p-1} \cdot g_{p-1})}$$

Les vecteurs erreurs et gradients successifs vérifient les relations :

$$x_p - x = x_{p-1} - x + \rho_{p-1} g_{p-1} \Leftrightarrow g_p = g_{p-1} + \rho_{p-1} A g_{p-1}$$

D'où :

$$\begin{aligned} \mathcal{E}(x_p) &= (A(x_p - x) \cdot (x_p - x)) = (g_p \cdot A^{-1} g_p) \\ &= (g_{p-1} \cdot A^{-1} g_{p-1}) + 2\rho_{p-1} (g_{p-1} \cdot g_{p-1}) + \rho_{p-1}^2 (A g_{p-1} \cdot g_{p-1}) \\ &= (g_{p-1} \cdot A^{-1} g_{p-1}) - \frac{(g_{p-1} \cdot g_{p-1})}{(A g_{p-1} \cdot g_{p-1})} (g_{p-1} \cdot g_{p-1}) \\ &= (g_{p-1} \cdot A^{-1} g_{p-1}) \left(1 - \frac{(g_{p-1} \cdot g_{p-1})}{(A g_{p-1} \cdot g_{p-1})} \frac{(g_{p-1} \cdot g_{p-1})}{(g_{p-1} \cdot A^{-1} g_{p-1})} \right) \\ &= \mathcal{E}(x_{p-1}) \left(1 - \frac{(g_{p-1} \cdot g_{p-1})}{(A g_{p-1} \cdot g_{p-1})} \frac{(g_{p-1} \cdot g_{p-1})}{(g_{p-1} \cdot A^{-1} g_{p-1})} \right) \end{aligned}$$

Afin de majorer le facteur de droite de la dernière ligne de l'équation (11.21), il faut majorer des quantités de la forme :

$$\frac{(Av \cdot v)}{(v \cdot v)} \leq \frac{(\|Av\| \|v\|)}{\|v\|^2} \leq \frac{\|Av\|}{\|v\|} \leq \|A\|$$

L'équation (11.21) conduit finalement à la majoration suivante :

$$\mathcal{E}(x_p) \leq \mathcal{E}(x_0) \left(1 - \frac{1}{\|A\| \|A^{-1}\|} \right)$$

et donc, par récurrence :

$$\mathcal{E}(x_p) \leq \mathcal{E}(x_0) \left(1 - \frac{1}{\|A\| \|A^{-1}\|} \right)^p$$

La quantité $\|A\| \|A^{-1}\| = \kappa(A)$ est le conditionnement de la matrice A . Comme on l'a vu au chapitre 4, paragraphe 4.2.4, pour une matrice symétrique définie positive, $\|A\| = \lambda_{max}$, λ_{max} étant la plus grande valeur propre de la matrice, et $\kappa(A) = \lambda_{max}/\lambda_{min}$.

L'estimation du taux de convergence de l'algorithme du gradient se base sur l'évolution de $\mathcal{E}(x_p)$ d'une itération à l'autre, puisque la solution approchée x_p ne dépend que de l'état x_{p-1} issu de l'itération précédente. La situation est différente avec la méthode

6 Principe du préconditionnement pour des matrices symétriques définies positives

du gradient conjugué, puisque $\mathcal{E}(x_p)$ est le minimum atteint sur tout l'espace $x_0 + \mathcal{K}_p$.
Plus précisément :

$$x_p = x_0 + P_{p-1}(A)g_0 \quad (11.22)$$

où P_{p-1} est le polynôme de degré $p - 1$ pour lequel $\mathcal{E}(x_p)$ est minimum.

En utilisant des polynômes particuliers, les polynômes de Tchebychev, on peut prouver le résultat suivant sur la vitesse de convergence du gradient conjugué.

Théorème 11.4

Avec la méthode du gradient conjugué, $\mathcal{E}(x_p)$ converge géométriquement, avec un taux de convergence égal à $\left(1 - \frac{1}{\sqrt{\kappa(A)}}\right)^2$.

Ce résultat, qui ne sera pas démontré ici car la démonstration, assez technique, repose sur l'interprétation en terme de construction d'un polynôme optimal de la méthode du gradient conjugué. Le choix qui a été fait dans ce livre de la construction des méthodes à partir de la base d'Arnoldi et de l'orthogonalisation permet une approche plus simple. Le résultat indiqué ci-dessus indique que la méthode du gradient conjugué converge beaucoup plus rapidement que la méthode du gradient. Ce qui est d'autant plus remarquable qu'une itération de gradient conjugué ne diffère de celle du gradient que par le fait que la direction de descente est obtenue en orthogonalisant, pour le produit scalaire associé à A , le gradient par rapport à la direction de descente précédente. Cette conjugaison ne coûte qu'un produit scalaire et une combinaison linéaire de vecteurs en plus.

1 Méthode GMRES

Dans le cas d'une matrice A quelconque, la matrice H_p de l'équation (11.7) n'est pas diagonale tridiagonale. On ne peut donc plus espérer une récurrence courte pour fabriquer une base orthogonale de l'espace de Krylov \mathcal{K}_p . Par ailleurs, A ne définit pas un produit scalaire et le critère d'optimalité $\mathcal{E}(x_p)$ (11.4) n'est plus pertinent. Le choix logique consiste à minimiser le carré de la norme du vecteur résidu qui, contrairement au vecteur errant, est effectivement calculable.

$$\begin{aligned}\mathcal{R}(x_p) &= (A(x_p - x), A(x_p - x)) \\ &= (g_p, g_p) \\ &= \|x_p - x\|_A^2 \\ &= (A^T A(x_p - x), (x_p - x))\end{aligned}$$

On va retrouver des propriétés semblables à celles du théorème 11.1 et de son corollaire 11.1, pour la solution approchée qui minimise $\mathcal{R}(x_p)$.

Théorème 12.1

La solution approchée x_p qui minimise $\mathcal{R}(x_p)$ dans $x_0 + \mathcal{K}_p$ est la projection de x , pour le produit scalaire associé à $A^T A$.

Démonstration. D'après l'équation (12.1), x_p est l'élément de $x_0 + \mathcal{K}_p$ dont la distance à x est minimale pour le produit scalaire associé à $A^T A$.

Corollaire 12.1

Le vecteur résidu $g_p = Ax_p - b$ est orthogonal à $A\mathcal{K}_p$.

Démonstration. Les propriétés d'une projection dans un espace affine impliquent :

$$(A^T A(x_p - x), w_p) = (A(x_p - x), Aw_p) = (g_p, Aw_p) = 0, \quad \forall w_p \in \mathcal{K}_p.$$

On a naturellement introduit le produit scalaire associé à la matrice $A^T A$ qui est symétrique définie positive dès lors que A est inversible. Il pourrait sembler judicieux de remplacer le système initial donné par l'équation (10.1), en utilisant le même type d'approche que pour le préconditionnement, par un système équivalent associé à une matrice symétrique définie positive, appelé « équation normale ».

$$A^T A x = A^T b \tag{12.2}$$

L'intérêt est que l'on peut appliquer la méthode du gradient conjugué à ce système. C'est malheureusement, le plus souvent, une très mauvaise idée. En effet, le conditionnement de la matrice $A^T A$ peut être égal au carré celui de la matrice A . C'est en particulier

1 Méthode Givens

évidemment le cas pour une matrice symétrique. La convergence de l'algorithme sera donc extrêmement lente, si la matrice A n'est pas très bien conditionnée.

Pour calculer x_p dans $x_0 + \mathcal{K}_p$, pour le système initial donné par l'équation (11.1), on dispose de la base orthonormée d'Arnoldi, V_p , et x_p et g_p vérifient respectivement les équations (11.1) et (11.3). D'après les propriétés de la base d'Arnoldi, et en particulier l'équation (10.5), on a :

$$g_p = Ax_p - b = g_0 + AV_p z_p = g_0 + V_{p+1} H_{p+1,p} z_p \quad (12.3)$$

Or, le premier vecteur de la base d'Arnoldi vérifie $\|g_0\|v_1 = g_0$, ce qui, d'après l'équation (12.3), nous donne une autre écriture du critère d'optimalité (12.1) :

$$\mathcal{R}(x_p) = (g_p \cdot g_p) = \|V_{p+1} (g_0 \|e_{p+1}^1 + H_{p+1,p} z_p)\|^2$$

où e_{p+1}^1 est le premier vecteur de la base canonique de dimension $p+1$, alors que z_p est un vecteur de dimension p . Et donc, vu que les colonnes de V_{p+1} forment une famille orthonormée :

$$\mathcal{R}(x_p) = \|g_0 \|e_{p+1}^1 + H_{p+1,p} z_p\|^2 \quad (12.4)$$

Ceci nous ramène de nouveau à un problème classique d'optimisation en dimension p , plus précisément à un problème de « moindres carrés » : on veut minimiser la norme d'une quantité de dimension $p+1$ alors que l'on ne dispose que d'un nombre inférieur de paramètres de contrôle, à savoir ici les p coefficients de z_p .

Pour résoudre ce problème de moindres carrés, la technique la mieux adaptée, vu la dimension réduite du problème et la forme de la matrice $H_{p+1,p}$, est la méthode QR de Givens qui consiste à construire, à l'aide de matrices de rotations planes, une matrice orthogonale Q_{p+1} telle que :

$$Q_{p+1} H_{p+1,p} = \begin{pmatrix} R_p \\ (0 \dots 0) \end{pmatrix}$$

où R_p est une matrice triangulaire supérieure. Une fois cette factorisation réalisée, le problème de minimisation se résout directement. En effet, puisque la matrice Q_{p+1} est orthogonale :

$$\|g_0 \|e_{p+1}^1 + H_{p+1,p} z_p\| = \|g_0 \|Q_{p+1} e_{p+1}^1 + Q_{p+1} H_{p+1,p} z_p\|$$

Si on note $y_{p+1} = -\|g_0\|Q_{p+1} e_{p+1}^1$, on a :

$$\|g_0 \|Q_{p+1} e_{p+1}^1 + Q_{p+1} H_{p+1,p} z_p\| = \begin{pmatrix} (-y_{p+1}(1)) \\ \vdots \\ (-y_{p+1}(p)) \\ (-y_{p+1}(p+1)) \end{pmatrix} + \begin{pmatrix} R_p z_p \\ 0 \end{pmatrix} \quad (12.5)$$

Chapitre 12 • Méthodes avec orthogonalisation exacte

D'après l'équation (12.5) il apparaît clairement que le minimum est atteint lorsque les p premières composantes s'annulent, c'est-à-dire pour z_p solution du système

$$R_p z_p = \tilde{y}_p = \begin{pmatrix} y_{p+1}(1) \\ \vdots \\ y_{p+1}(p) \end{pmatrix} \quad (12.6)$$

De plus, la valeur du minimum est connue, sans avoir besoin de résoudre effectivement le système (12.5), et vaut $|y_{p+1}(p+1)|$. Par construction $|y_{p+1}(p+1)|$ est la valeur de la norme de $g_p = Ax_p - b$, pour $x_p = x_0 + V_p z_p$, z_p étant la solution du système (12.5).

Reste donc finalement à expliciter la technique de calcul de la matrice Q_{p+1} pour définir complètement cette nouvelle méthode de Krylov, appelée GMRES (General Minimum RESidual).

Une matrice de rotation de Givens est une matrice de rotation dans un plan défini par deux vecteurs de la base canonique. Pour la méthode GMRES, seules serviront des rotations dans un plan défini par deux vecteurs successifs de la base canonique, de la forme suivante :

$$G^{\theta_k} = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & \begin{pmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{pmatrix} & 0 \\ 0 & 0 & I_{n-k-1} \end{pmatrix}$$

Vu que seules les lignes k et $k+1$ de G^{θ_k} diffèrent de la matrice I , les lignes de la matrice $N = G^{\theta_k} M$ de numéro différent de k et de $k+1$ sont identiques à celle de A . Quant aux termes situés dans les lignes k et $k+1$, ils sont, colonne par colonne, le résultat du produit par la rotation d'angle θ_k :

$$\begin{pmatrix} n_{kj} \\ n_{k+1j} \end{pmatrix} = \begin{pmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{pmatrix} \begin{pmatrix} m_{kj} \\ m_{k+1j} \end{pmatrix} = \begin{pmatrix} \cos(\theta_k)m_{kj} - \sin(\theta_k)m_{k+1j} \\ \sin(\theta_k)m_{kj} + \cos(\theta_k)m_{k+1j} \end{pmatrix}$$

Considérons les deux premières lignes de la matrice $H_{p+1,p}$ dans l'équation (10.6). On cherche la rotation de Givens G^{θ_1} qui fait disparaître le premier terme sous-diagonal, c'est-à-dire telle que :

$$\begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{21} \end{pmatrix} = \begin{pmatrix} r_{11} \\ 0 \end{pmatrix}$$

En pratique, il n'est pas utile de calculer explicitement θ_1 mais seulement $\cos(\theta_1)$ et $\sin(\theta_1)$, que l'on notera c_1 et s_1 , et qui doivent vérifier :

$$\begin{aligned} c_1^2 + s_1^2 &= 1 \\ s_1 h_{11} + c_1 h_{21} &= 0 \end{aligned} \quad (12.7)$$

● Méthode GMRES

L'équation (12.7) admet deux couples de solutions différant par leurs signes. On en choisit arbitrairement un des deux. La matrice $G^{\theta_j} H_{p+1,p}$ ne diffère de $H_{p+1,p}$ que par ses deux premières lignes. Elle a donc la même structure, mais avec un coefficient (2, 1) nul. La construction de la matrice Q_{p+1} se fait de manière récurrente. Si on a annulé les termes sous-diagonaux des $j - 1$ premières colonnes, on a obtenu une matrice de la forme suivante :

$$\tilde{H}_{p+1,p} = \begin{pmatrix} r_{11} & r_{12} & \vdots & r_{1j-1} & r_{1j} & \vdots & r_{1p} \\ 0 & r_{22} & \vdots & r_{2j-1} & r_{2j} & \vdots & r_{2p} \\ 0 & 0 & \vdots & r_{3j-1} & r_{3j} & \vdots & r_{3p} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & & & 0 & \tilde{h}_{jj} & \vdots & \tilde{h}_{jp} \\ \vdots & & & 0 & h_{j+1,j} & \vdots & h_{j+1,p} \\ \vdots & & & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & h_{p+1,p} \end{pmatrix} \quad (12.8)$$

Pour faire disparaître le coefficient $h_{j+1,j}$, il suffit de multiplier à gauche la matrice de l'équation (12.8) par la matrice de rotation de Givens G^{θ_j} , définie par les facteurs c_j et s_j satisfaisant l'équation :

$$\begin{aligned} c_j^2 + s_j^2 &= 1 \\ s_j \tilde{h}_{jj} + c_j h_{j+1,j} &= 0 \end{aligned} \quad (12.9)$$

Cette transformation ne modifie que les lignes j et $j + 1$ de la matrice. Les lignes de numéro compris entre 1 et $j - 1$ ne seront donc pas affectées par les transformations ultérieures, ce qui justifie la notation utilisée dans l'équation (12.8) pour la matrice $\tilde{H}_{p+1,p}$ issue des $j - 1$ premières rotations de Givens : les coefficients des $j - 1$ premières lignes sont ceux de la matrice R de la factorisation QR ; les coefficients des lignes de numéro compris entre $j + 1$ et $p + 1$ sont encore ceux de la matrice $H_{p+1,p}$ initiale ; seuls les coefficients de la ligne numéro j ont des valeurs différentes de leurs valeurs initiales mais doivent encore subir les effets de la rotation de Givens G^{θ_j} . En particulier :

$$\begin{pmatrix} c_j & -s_j \\ s_j & c_j \end{pmatrix} \begin{pmatrix} \tilde{h}_{jj} \\ h_{j+1,j} \end{pmatrix} = \begin{pmatrix} r_{jj} \\ 0 \end{pmatrix}$$

La matrice $H_{p+1,p}$ s'obtient, à l'itération p , par l'ajout d'une colonne à la matrice H_{pp-1} . De ce fait, la factorisation QR de la matrice H_{pp-1} donne une factorisation QR partielle de la matrice $H_{p+1,p}$. En effet si la factorisation QR de la matrice H_{pp-1} s'écrit :

$$G_p^{\theta_p} \cdots G_p^{\theta_1} H_{pp-1} = \begin{pmatrix} R_{p-1} \\ (0 \cdots 0) \end{pmatrix}$$

Chapitre 12 • Méthodes avec orthogonalisation exacte

où les matrices $G_p^{\theta_i}$ sont des matrices de rotation de Givens de dimension $p \times p$,

$$G_{p+1}^{\theta_{p+1}} \cdots G_{p+1}^{\theta_1} H_{p+1,p} = \begin{pmatrix} R_{p-1} & \begin{pmatrix} r_{1,p} \\ \vdots \\ r_{p-1,p} \\ \tilde{h}_{pp} \\ h_{p+1,p} \end{pmatrix} \\ (0 \cdots 0) & \end{pmatrix}$$

Il suffit donc, pour compléter la factorisation QR de la matrice H , d'appliquer posteriori les $p-1$ rotations de Givens précédentes à la colonne numéro p puis de calculer la rotation de Givens $G_p^{\theta_p}$ en déterminant deux coefficients c_p et s_p tels que

$$\begin{aligned} c_p^2 + s_p^2 &= 1 \\ s_p \tilde{h}_{pp} + c_p h_{p+1,p} &= 0 \end{aligned}$$

De même, les $p-1$ premières composantes du vecteur $y_{p+1} = -\|g_0\|Q_{p+1,p}e_p^1$ sont identiques à celles du vecteur $y_p = -\|g_0\|Q_p e_p^1$. Les deux dernières composantes sont calculées en appliquant tout simplement la rotation d'angle θ_p :

$$\begin{pmatrix} y_{p+1}(p) \\ y_{p+1}(p+1) \end{pmatrix} = \begin{pmatrix} c_p & -s_p \\ s_p & c_p \end{pmatrix} \begin{pmatrix} y_p(p) \\ 0 \end{pmatrix}$$

Finalement, l'algorithme GMRES consiste à calculer itérativement la base d'Arnoldi de l'espace de Krylov, la factorisation QR de la matrice H et le vecteur y . Comme on l'a vu précédemment, $|y_{p+1}(p+1)|$ est égal à $\|g_p\|$. Il n'est donc pas utile de calculer la solution approchée à chaque itération, mais seulement une fois que la convergence a été atteinte.

Il est maintenant possible de détailler complètement la mise en œuvre de l'algorithme GMRES, en notant respectivement (β_p) le dernier coefficient du vecteur \tilde{y}_p , égal à $y_{p+1}(p)$, et r_{p+1} , le coefficient $y_{p+1}(p+1)$, dont la valeur absolue est égale à la norme du résidu.

Au niveau de la mise en œuvre informatique, il est inutile de conserver la matrice H , puisque seule sa dernière colonne est active et peut donc n'être stockée que de manière temporaire dans un vecteur utilitaire. Dans l'algorithme tel qu'il est décrit ci-dessous, ce vecteur, noté α , sert aussi à réaliser le calcul des coefficients de la colonne numéro p de R , de sorte qu'à l'issue du produit par les $p-1$ premières rotations de Givens α , et α_{p+1} représentent respectivement \tilde{h}_{pp} et $h_{p+1,p}$. Par ailleurs, il n'est nécessaire de stocker que la partie triangulaire supérieure de la matrice R .

- Initialisation de GMRES :

$$\begin{aligned} g_0 &= Ax_0 - b \\ r_1 &= -\|g_0\| \\ v_1 &= \frac{1}{\|g_0\|} g_0 \end{aligned}$$

- Construction du vecteur numéro $p+1$ de la base d'Arnoldi :

$$w = Av_p$$

❶ Méthode GMRES

```

for  $i = 1$  to  $p$ 
     $a_i = (w, v_i)$ 
     $w = w - a_i v_i$ 
end for
 $\alpha_{p+1} = \|w\|$ 
 $v_{p+1} = \frac{1}{\alpha_{p+1}}w$ 

```

• Calcul du produit de la colonne numéro p de la matrice H par les $p - 1$ premières rotations de Givens :

$$\begin{pmatrix} r_{ip} \\ \alpha_{i+1} \end{pmatrix} = \begin{pmatrix} c_i & -s_i \\ s_i & c_i \end{pmatrix} \begin{pmatrix} \alpha_i \\ \alpha_{i+1} \end{pmatrix}$$

• Calcul de la nouvelle rotation de Givens et remise à jour de la dernière colonne de R

• Calcul du vecteur y :

$$c_p = \sqrt{\frac{1}{1 + (\frac{\alpha_{p+1}}{\alpha_p})^2}}$$

$$s_p = -c_p \frac{\alpha_{p+1}}{\alpha_p}$$

$$\begin{pmatrix} r_{pp} \\ 0 \end{pmatrix} = \begin{pmatrix} c_p & -s_p \\ s_p & c_p \end{pmatrix} \begin{pmatrix} \alpha_p \\ \alpha_{p+1} \end{pmatrix}$$

$$\begin{pmatrix} \beta_p \\ r_{p+1} \end{pmatrix} = \begin{pmatrix} c_p & -s_p \\ s_p & c_p \end{pmatrix} \begin{pmatrix} r_p \\ 0 \end{pmatrix}$$

• Test d'arrêt :

```

if  $|r_{p+1}|/\|b\| < \epsilon$  then
    Fin
end if

```

Ce n'est que lorsque le test de convergence est satisfait que l'on résout effectivement le système (12.6), avec $y_{p+1}(i) = \beta_i$, en reprenant les notations de l'algorithme ci-dessus. Il est alors possible de calculer la solution approchée $x_p = x_0 + V_p z_p$.

Il faut remarquer qu'avec cette méthode, il faut calculer $p + 1$ vecteurs d'Arnoldi pour trouver la solution optimale dans l'espace de Krylov de dimension p . On verra au paragraphe 3 une méthode, appelée ORTHODIR, qui ne présente pas ce décalage mais qui nécessite en contrepartie le stockage des produits des vecteurs de base par la matrice.

Le problème principal avec la méthode GMRES est que l'on n'a pas de récurrence courte pour calculer les vecteurs de la base d'Arnoldi qu'il faut donc tous conserver, aussi bien pour orthonormaliser chaque nouveau vecteur par rapport aux précédents, que pour calculer la solution à convergence. Le plus souvent, il n'est pas possible, pour des questions de place mémoire comme de coût de calcul, de faire plus qu'un certain nombre m d'itérations, fixé a priori. Si la convergence n'est pas atteinte en m itérations, on calcule la solution approchée x_m et on redémarre à partir de x_m comme solution initiale. Cette méthode est notée $GMRES(m)$ dans la littérature. Contrairement à la méthode GMRES, elle peut ne pas converger si m n'est pas assez grand.