Laurence Lamoulie
Chrysostome Baskiotis

ANALYSE NUMÉRIQUE

Fascicule 2

Algèbre linéaire

Année 2009 – 2010

INTRODUCTION

Dans la deuxième partie de ce cours nous présentons des méthodes de résolution des systèmes linéaires ainsi que des méthodes d'inversion matricielle.

1

MÉTHODES DE RÉSOLUTION DE SYSTÈMES LINÉAIRES

1.1 Introduction

L'étude des méthodes de résolution des systèmes linéaires est une étape obligatoire dans un cours d'analyse numérique. En effet, presque tous les calculs passent par la résolution d'un système. On en rencontre dans la discrétisation de problèmes aux limites, dans les problèmes d'approximation par exemple.

La résolution de systèmes n'est plus une opération destinée à être menée à la main. Du fait de l'apparition des ordinateurs, et du développement de méthodes qui leur sont adaptées, on peut envisager de résoudre de grands systèmes. Si la matrice est pleine, c'est à dire contient peu de zéros, on pourra "seulement" résoudre des systèmes à quelques centaines de milliers d'inconnues. Si la matrice est creuse, c'est à dire contient beaucoup de zéros, la résolution de systèmes à plusieurs millions d'inconnues est possible.

La résolution d'un système ne doit pas être comprise comme l'obtention de la solution exacte en un nombre fini d'étapes. Cette vision ne concerne que les méthodes dites "directes". On est souvent amené à accepter un compromis : obtenir une solution approchée en un nombre fini

d'étapes, sachant que la solution exacte serait obtenue en un nombre infini d'étapes, chose impossible à mettre en oeuvre. Ce compromis est le principe même des méthodes itératives. C'est devenu la solution la plus répandue pour résoudre les grands systèmes.

En plus de la taille de la matrice, ses propriétés sont aussi un élément déterminant : certaines assurent la convergence a priori de méthodes itératives. Dans le cas où elles ne sont pas vérifiées par une matrice, ou bien quand on ne pas peut démontrer qu'elles le sont, la méthode itérative est choisie aux risques et périls de l'utilisateur. C'est un risque à éviter, surtout dans des applications industrielles, parfois sensibles. Des navettes spatiales ont explosé pour moins que ça...

La résolution des systèmes linéaires est donc à la fois une question de théorie et une question de pratique : il faut choisir le bon algorithme, s'assurer que les conditions sont réunies pour l'utiliser, qu'il sera numériquement stable et assez rapide pour les besoins de la cause.

Il faut repenser la résolution de systèmes dans le cadre d'une mise en oeuvre informatique : nous allons voir sur deux exemples simples que les méthodes utilisables à la main ne sont pas, en général, adaptées à la résolution de systèmes en machine, notamment pour des questions de temps de calcul. Bien sur, de nombreux logiciels proposent des bibliothèques de résolution de systèmes linéaires, qui s'appuient sur des bibliothèques publiques d'algèbre linéaire de base (BLAS). Ces dernières ont été précieuses dans les progrès du calcul scientifique matriciel ; elles permettent d'effectuer de façon optimale les opérations algébriques de base. Les principaux logiciels sont soit du domaine public (LAPACK= Linear Algebra PACKage par exemple), soit des logiciels commerciaux (NAG = Numerical Algorithms Group, IMSL = International Mathematical and Statistical Library, MATLAB,...). Mais en tout état de cause, la possession d'une voiture de course ne permet pas d'avancer quand on ne sait pas la conduire...

1.1.1 Exemple 1 : Économie : analyse d'entrées-sorties

On veut déterminer l'équilibre entre la demande et l'offre de certains biens. Dans le modèle de production considéré, $m \geq n$ usines produisent n produits différents. Elles doivent faire face à une demande interne (l'entrée) nécessaire au fonctionnement propre des usines, ainsi qu'à une demande externe (la sortie) provenant des consommateurs.

La principale hypothèse du modèle de Leontieff (1930) [1] est que le modèle de production est linéaire, c'est à dire que la sortie est proportionnelle à l'entrée utilisée. Sous cette hypothèse, l'activité des usines est entièrement décrite par deux matrices : la matrice d'entrée $\mathbf{C} = (c_{ij}) \in \mathbb{R}^{n\times m}$ et la matrice de sortie $\mathbf{P} = (p_{ij}) \in \mathbb{R}^{n\times m}$. Le coefficient c_{ij} (resp. p_{ij}) représente la quantité du i^{ème} bien absorbé (resp. produit) pour la j^{ème} usine sur une période fixée. La matrice $\mathbf{A} = \mathbf{P} - \mathbf{C}$ est appelée matrice d'entrée-sortie : un a_{ij} positif (resp. négatif) désigne la quantité du i^{ème} bien produit (resp. absorbé) par la j^{ème} usine. Enfin, on peut raisonnablement supposer que le système de production satisfait à la demande du marché, qu'on peut représenter par un vecteur $b = (b_i) \in \mathbb{R}^n$ (vecteur de la demande finale). La composante b_i représente la quantité du i^{ème} bien absorbé sur le marché. L'équilibre est atteint lorsque le vecteur $x = (x_i) \in \mathbb{R}^m$ représentant

1. Wassily Leontieff a reçu en 1973 le prix Nobel d'économie pour ses travaux

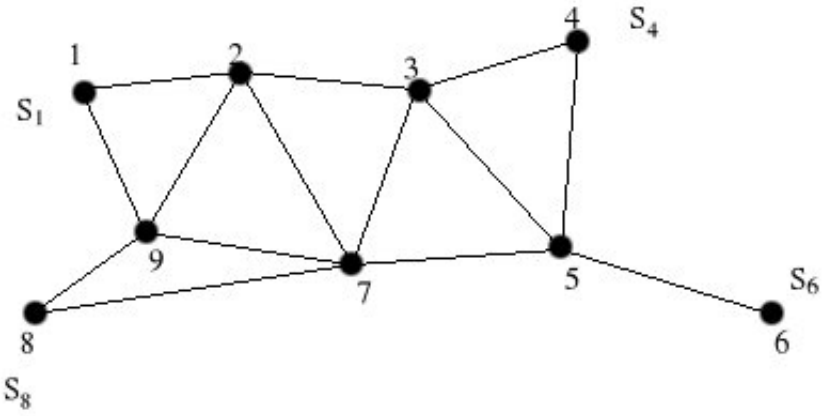
la production totale est égal à la demande totale, c'est à dire

$$\mathbf{Ax} = \mathbf{b}, \text{ où } \mathbf{A} = \mathbf{P} - \mathbf{C}$$

1.1.2 Exemple 2 : Résolution d'un problème de réseaux

Un réseau est un ensemble de nœuds P_i et d'arêtes $E_{i,j}$ reliant certains de ces nœuds :

- lignes électriques,
- canalisations d'eaux, égouts,...



Dans chaque arête circule un fluide ; à chaque nœud est associé un potentiel. L'intensité (ou le débit) du fluide est proportionnelle à la différence de potentiel entre les deux extrémités de l'arête où il circule ; c'est la loi d'Ohm pour les circuits électriques :

$$q_{i,j} = k_{i,j}(u_i - u_j)$$

Une loi physique de conservation (de Kirchoff dans le cas électrique) impose un équilibre : la somme algébrique des intensités en chaque nœud est égale à la valeur de la source (ou du puits) qu'il figure.

Au nœud P_i, on a dans le cas du circuit électrique :

$$S_i = \sum_j q_{i,j} = \sum_j k_{i,j}(u_i - u_j)$$

Cette somme peut être étendue aux nœuds adjacents de P_i, les équations d'équilibre s'écrivent :

$$\begin{cases} S_1 = k_{1,2}(u_1 - u_2) + k_{1,9}(u_1 - u_9) \\ 0 = k_{2,1}(u_2 - u_1) + k_{2,9}(u_2 - u_9) + k_{2,7}(u_2 - u_7) + k_{2,3}(u_2 - u_3) \\ \cdots = \cdots \\ 0 = k_{9,1}(u_9 - u_1) + k_{9,2}(u_9 - u_2) + k_{9,7}(u_9 - u_7) + k_{9,8}(u_9 - u_8) \end{cases}$$

de sorte que l'équilibre du système est connu en résolvant le système linéaire

$$\mathbf{Au} = \mathbf{S}$$

avec une matrice $\mathbf{A}$ dont les coefficients non nuls sont représentés ci-dessous par une étoile :

$$\mathbf{A} = \begin{bmatrix} * & * & & & & & & * \\ * & * & * & & & * & & * \\ & * & * & * & & * & & \\ & & * & * & * & & & \\ & & & * & * & * & * & \\ & & & & * & * & & \\ & * & * & & * & & * & * \\ & & & & & & * & * \\ * & * & & & & & * & * \end{bmatrix}$$

Le second membre est défini par $S^T = (S_1, 0, 0, S_4, 0, S_6, 0, S_8, 0)$.

1.1.3 Comment résoudre ces systèmes

La première idée qui vient est de résoudre ce système en utilisant les formules de Cramer, dites aussi "méthode des déterminants". Ces formules fournissent une solution exacte du système linéaire $\mathbf{Ax} = \mathbf{b}$ dans $\mathbb{R}^n$ avec $\mathbf{x} = (x_1, x_2, ...x_n)^T$ donnée par :

$$x_i = \frac{\det \mathbf{A}^{(i)}}{\det \mathbf{A}}$$

où $A^{(i)}$ désigne la matrice obtenue en substituant dans A la colonne i par le second membre du système.

Il y a donc $N+1$ déterminants à calculer, donnés dans le cas général pour une matrice $\mathbf{A}$ quelconque par :

$$\det(A) = \sum_{\sigma \in P_n} \epsilon(\sigma) a_{1,\sigma(1)} \times a_{2,\sigma(2)} \cdots \times a_{n,\sigma(n)}$$

où P_n désigne l'ensemble des permutations de $\{1, ...n\}$ qui compte $n!$ éléments et $\epsilon(\sigma)$ est $+1$ ou -1, la signature de la permutation. Le coût du calcul d'un tel déterminant est donc tel que :

$$c(n) \sim n \times n! \text{ opérations élémentaires}$$

et le coût global de la méthode est donc

$$C(n) \sim n \times (n-1)! \text{ opérations}$$

Le coût de la résolution d'un système 100×100 peut alors être évalué par la formule de Stirling ($n! \sim n^{n+1/2} e^{-n} \sqrt{2\pi}$). Cela conduit à l'évaluation :

$$C(n) \sim 9.4 \times 10^{161}$$

opérations élémentaires. Sur un ordinateur qui réalise 10^9 opérations flottantes par seconde (1 gigaflop), on devra attendre la solution pendant environ 3×10^{145} années !

1.2 Les systèmes faciles à résoudre

Afin de construire des méthodes qui simplifient la résolution d'un système sans passer par des calculs inappropriés pour une machine, voyons deux cas de systèmes dont la résolution est simple à programmer.

1.2.1 Systèmes diagonaux

Si **A** est une matrice diagonale, c'est à dire si

$$\mathbf{A} = (a_{i,j})_{1 \leq i,j \leq n} \text{ avec } a_{i,j} = 0 \text{ si } i \neq j$$

le système $Ax = b$ est immédiatement résolu du fait que

$$x_i = \frac{1}{a_{ii}} b_i$$

L'algorithme correspondant est donné par :

Algorithme : `Matrice diagonale`

{On suppose que $A_{kk} \neq 0$ }
Pour i ← 1 **à** n **faire**
$x_i \leftarrow b_i / a_{ii}$
FinPour

Le coût est $c(n) = n$ opérations élémentaires

1.2.2 Systèmes triangulaires

La matrice **A** d'un système triangulaire supérieur est telle que

$$\mathbf{A} = (a_{i,j})_{1 \leq i,j \leq n} \text{ avec } a_{i,j} = 0 \text{ si } i > j$$

Comme A est inversible,

$$a_{i,i} \neq 0 \text{ si } 1 \leq i \leq n$$

Le système s'écrit

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{nn}x_n &= b_n \end{aligned}$$

On le résout en remontant :

$$\begin{aligned} x_n &= \frac{b_n}{a_{n,n}} \\ x_{n-1} &= (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \\ &\vdots \\ x_1 &= (b_1 - a_{12}x_2 - \ldots - a_{1,n}x_n)/a_{1,1} \end{aligned}$$

On obtient alors la solution par un algorithme de *substitution rétrograde*, dit aussi simplement *algorithme de remontée* :

Algorithme : Algorithme de remontée

$x_n \leftarrow b_n/a_{n,n}$
Pour i ← n-1 **à** 1 **par pas de** -1 **faire**
$x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n} a_{i,j}x_j\right)/a_{i,i}$
FinPour

Le coût du calcul d'un $x_k = \left(b_k - a_{k,k+1}x_{k+1} - \ldots - a_{k,n}x_n\right)/a_{k,k}$ se décompose en :

- $(n-k)$ additions,
- $(n-k)$ multiplications,
- 1 division

Le coût total de remontée est donc de

$$\sum_{k=1}^{n}(n-k) = n^2 - \frac{n(n-1)}{2} \sim \frac{n^2}{2} \text{ additions + multiplications}$$

soit n^2 opérations élémentaires.

EXERCICE 1.1 *Établir l'algorithme de résolution d'un système triangulaire inférieur*

1.3 Méthodes directes

L'idée des méthodes directes est de remplacer la résolution d'un système du type $\mathbf{Ax} = \mathbf{b}$ dans lequel la matrice $\mathbf{A}$ est pleine, par un système ou plusieurs systèmes plus facile(s) à résoudre car creux (i.e. de matrice triangulaire ou diagonale). Le système diagonal serait idéal puisque c'est à la fois le plus rapide et le moins coûteux à résoudre. Mais diagonaliser $\mathbf{A}$, c'est rechercher ses éléments propres et déterminer $\mathbf{P}$ et $\mathbf{D}$ vérifiant $\mathbf{A} = \mathbf{PDP}^{-1}$. L'idée est séduisante mais malheureusement inapplicable car la recherche d'éléments propres est beaucoup plus difficile numériquement que la résolution d'un système. L'alternative est donc de remplacer $\mathbf{A}$ par le produit de deux matrices triangulaires, notées en général $\mathbf{L}$ et $\mathbf{U}$, respectivement triangulaire inférieure et triangulaire supérieure. En effet on résout alors successivement deux systèmes

triangulaires :

$$\mathbf{Ly} = \mathbf{b} \text{ puis } \mathbf{Ux} = \mathbf{y}$$

dont la solution $\mathbf{x}$ vérifie évidemment $\mathbf{Ax} = \mathbf{b}$. On va voir dans ce qui suit comment on s'y prend numériquement.

1.3.1 Elimination de Gauss sans recherche de pivot

1.3.1.1 Factorisation LU

Soit à résoudre le système : Trouver $\mathbf{x} \in \mathbb{R}^n$ tel que

$$\mathbf{Ax} = \mathbf{b}$$

pour $\mathbf{A} \in \mathbb{R}^{n\times n}$, et $\mathbf{b} \in \mathbb{R}^n$. On suppose que $\mathbf{A}$ est inversible.

Le but est de se ramener à un système triangulaire. On procède par étapes :

Étape 1 : Elimination de l'inconnue x_1 des lignes 2 à n

Sous l'hypothèse que $a_{11} \neq 0$ on peut l'utiliser pour éliminer l'inconnue x_1 des lignes 2 à n. Le terme a_{11} est appelé **pivot** et on note pour la suite :

$$\pi_1 = a_{11}$$

La ligne i devient alors :

$$0x_1 + (a_{i2} - \frac{a_{i1}}{\pi_1}a_{12})x_2 + \ldots + (a_{in} - \frac{a_{i1}}{\pi_1}a_{1n})x_n = b_i - \frac{a_{i1}}{\pi_1}b_1$$

ce que l'on écrit encore

$$a_{i2}^{(2)}x_2 + \ldots + a_{in}^{(2)}x_n = b_i^{(2)}, \forall i > 1$$

en posant

$$\begin{aligned} a_{i2}^{(2)} &= a_{i2} - \frac{a_{i1}}{\pi_1}a_{12} \\ &\vdots \\ a_{in}^{(2)} &= a_{in} - \frac{a_{i1}}{\pi_1}a_{1n} \\ b_i^{(2)} &= b_i - \frac{a_{i1}}{\pi_1}b_1 \end{aligned}$$

Si on pose aussi, pour $1 \leq j \leq n$

$$a_{1j}^{(2)} = a_{1j}, \forall 1 \leq j \leq n \text{ et } b_1^{(2)} = b_1$$

on a obtenu le système équivalent :

$$\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$$

avec

$$\mathbf{A}^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix}$$

On voit que

$$\mathbf{A}^{(2)} = \mathbf{M}^{(1)}\mathbf{A} \text{ où } \mathbf{M}^{(1)} = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -\frac{a_{n1}}{a_{11}} & 0 & \dots & 0 & 1 \end{bmatrix}$$

Le système s'écrit alors

$$\mathbf{A}^{(2)} = \mathbf{M}^{(1)}\mathbf{A}, \text{ et } \mathbf{b}^{(2)} = \mathbf{M}^{(1)}\mathbf{b}$$

Tout se passe comme si on avait prémultiplié à gauche le système initial par $\mathbf{M}^{(1)}$.

Étape k : Elimination de l'inconnue x_k des lignes $k+1$ à n

On suppose que l'on a pu itérer le procédé ci-dessus $k-1$ fois, c'est que l'on n'a jamais rencontré de pivot nul :

$$\pi_i = a_{ii}^{(i)} \neq 0 \text{ pour } 1 \leq i \leq k-1$$

On obtient alors le système équivalent :

$$\mathbf{A}^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$$

avec $\mathbf{A}^{(k)}$ de la forme

$$\mathbf{A}^{(k)} = \begin{bmatrix} a_{11}^{(k)} & a_{12}^{(k)} & \dots & \dots & \dots & \dots & \dots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & \ddots & & & & & a_{2n}^{(k)} \\ \vdots & \ddots & a_{33}^{(k)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a_{k-1,k-1}^{(k)} & \dots & \dots & a_{k-1,n}^{(k)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & \widetilde{\mathbf{A}^{(k)}} & & \\ 0 & \dots & \dots & 0 & 0 & & & \end{bmatrix} \tag{1.3.1}$$

où $\widetilde{\mathbf{A}^{(k)}}$ est une matrice carrée d'ordre $n-k+1$:

$$\widetilde{\mathbf{A}^{(k)}} = \begin{bmatrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix} \tag{1.3.2}$$

Comme précédemment, on doit effectuer une hypothèse sur la valeur du pivot :

$$\pi_k = a_{kk}^{(k)} \neq 0$$

On peut alors introduire la matrice

$$\mathbf{M}^{(k)} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & -\frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & -\frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{bmatrix}$$

(1.3.3)

Remarquons que l'inverse de $\mathbf{M}^{(k)}$ est $\mathbf{L}^{(k)}$:

$$\mathbf{L}^{(k)} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & \frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{bmatrix} \tag{1.3.4}$$

Soit

$$\mathbf{A}^{(k+1)} = \mathbf{M}^{(k)}\mathbf{A}^{(k)} \text{ et } \mathbf{b}^{(k+1)} = \mathbf{M}^{(k)}\mathbf{b}^{(k)}$$

alors

$$\mathbf{A}^{(k+1)}\mathbf{x} = \mathbf{b}^{(k+1)}$$

et

$$\mathbf{A}^{(k+1)} = \begin{bmatrix} a_{11}^{(k+1)} & a_{12}^{(k+1)} & \dots & \dots & \dots & \dots & \dots & a_{1n}^{(k+1)} \\ 0 & a_{22}^{(k+1)} & \ddots & & & & & a_{2n}^{(k+1)} \\ \vdots & \ddots & a_{33}^{(k+1)} & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \dots & \dots & 0 & a_{k,k}^{(k+1)} & \dots & \dots & a_{k,n}^{(k+1)} \\ \vdots & & & \vdots & 0 & & & \\ \vdots & & & \vdots & \vdots & \widetilde{A^{(k+1)}} & & \\ 0 & \dots & \dots & 0 & 0 & & & \end{bmatrix}$$

où $\widetilde{\mathbf{A}^{(k+1)}}$ est une matrice carrée d'ordre $n-k$:

$$\widetilde{\mathbf{A}^{(k+1)}} = \begin{bmatrix} a_{k+1,k+1}^{(k+1)} & \dots & a_{k+1n}^{(k+1)} \\ \vdots & & \vdots \\ a_{nk+1}^{(k+1)} & \dots & a_{nn}^{(k+1)} \end{bmatrix}$$

Après n-1 étapes : Si on itère $n-1$ fois, et si les pivots apparus sont tous non nuls, soit :

$$\pi_i = a_{ii}^{(i)} \neq 0 \text{ pour } 1 \leq i \leq n-1$$

on arrive au système triangulaire équivalent :

$$\mathbf{A}^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$$

avec

$$\mathbf{A}^{(n)} = \begin{bmatrix} \pi_1 & * & * & \cdots & * \\ 0 & \pi_2 & * & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & * \\ 0 & \cdots & \cdots & 0 & \pi_n \end{bmatrix}$$

qui est inversible, si de plus

$$\pi_n \neq 0$$

Bilan

Si on appelle $\mathbf{L}$ la matrice triangulaire inférieure avec des 1 sur la diagonale, et $\mathbf{L}^{(k)}$ la matrice définie en (1.3.4)

$$\mathbf{L} = \mathbf{L}^{(1)} \cdots \mathbf{L}^{(n-1)}$$

et U la matrice triangulaire supérieure

$$\mathbf{U} = \mathbf{A}^{(n)}$$

on a

$$\mathbf{A} = \mathbf{LU}$$

On dit qu'on a effectué une **factorisation de Gauss** ou **factorisation LU** de $\mathbf{A}$.

REMARQUE 1.3.1 *Il est aussi possible avec le même algorithme d'obtenir la factorisation* $\mathbf{LU}$ *d'une matrice de* $\mathcal{M}_m(\mathbb{C})$*, avec* $m \geq n$*, si les pivots qui apparaissent sont non nuls.*

1.3.1.2 Coût de la méthode d'élimination de Gauss

On estime le coût de l'élimination de x_k. Rappelons qu'à l'étape $k-1$, on obtient la matrice $\mathbf{A}_k$ donnée en (1.3.1) comportant le bloc $\widetilde{\mathbf{A}^{(k)}}$ donné par (1.3.2) :

$$\widetilde{\mathbf{A}^{(k)}} = \begin{bmatrix} a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & & \vdots \\ a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}$$

On construit tout d'abord $\mathbf{M}^{(k)}$ donné par (1.3.3) donnée par

$$\mathbf{M}^{(k)} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & 1 & \ddots & & & \vdots \\ \vdots & & & -\frac{a_{k+1,k}^{(k)}}{\pi_k} & \ddots & \ddots & & \vdots \\ \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & -\frac{a_{n,k}^{(k)}}{\pi_k} & 0 & \dots & 0 & 1 \end{bmatrix}$$

il faut pour cela diviser par le pivot sur $n-k$ lignes. On effectue donc $n-k$ **divisions**.

On construit ensuite le produit $\mathbf{A}^{(k+1)} = \mathbf{M}^{(k)}\mathbf{A}^{(k)}$ (puisque notre objectif est de déterminer à la fin $\mathbf{A}^{(n)}$). Pour cela il faut effectuer $(n-k)^2$ **additions et multiplications.**

Au total :

$$\sum_{k=1}^{n}(n-k)^2 = \frac{1}{3}n(n-1)(n-\frac{1}{2}) \sim \frac{n^3}{3} \textbf{additions et multiplications.}$$

$$\sum_{k=1}^{n}(n-k) = \frac{1}{2}n(n-1) \textbf{ divisions}$$

1.3.1.3 Utilisation de la factorisation LU

Calcul de déterminant

Une utilisation de la factorisation $\mathbf{LU}$ est le calcul de déterminant de $\mathbf{A}$: en effet, si $\mathbf{A}$ admet une factorisation $\mathbf{LU}$, on a

$$\det \mathbf{A} = \det(\mathbf{LU}) = \det \mathbf{L}. \det \mathbf{U}$$

Or $\mathbf{L}$ est triangulaire à diagonale unité donc de déterminant 1, ce qui donne finalement

$$\det \mathbf{A} = \det \mathbf{U} = \prod_{i=1}^{n} \pi_i$$

La factorisation $\mathbf{LU}$ permet donc de calculer le déterminant de $\mathbf{A}$ avec une complexité de l'ordre de $\frac{n^3}{3}$ additions et multiplications, au lieu de $n!$ avec la formule du déterminant !

Résolution de systèmes linéaires

N'oublions pas notre objectif initial : résoudre un système linéaire. Il est clair que la factorisation $\mathbf{LU}$ permet de résoudre successivement deux systèmes triangulaires :

$$\mathbf{Ly} = \mathbf{b} \text{ puis } \mathbf{Ux} = \mathbf{y}$$

dont la solution $\mathbf{x}$ vérifie $\mathbf{Ax} = \mathbf{b}$.

Au delà de ce point, si on dispose de plusieurs systèmes linéaires de seconds membres différents mais de même matrice $\mathbf{A}$:

$$\mathbf{Ax} = \mathbf{b}_i \text{ pour } i = 1, ...I$$

il suffit de calculer une seule fois les matrices $\mathbf{L}$ et $\mathbf{U}$ et de les stocker. On peut alors effectuer autant de descentes et de remontées qu'on a de systèmes pour les résoudre tous, sans pour autant refaire la factorisation LU.

Pour I systèmes à résoudre, la complexité est de l'ordre de $\frac{n^3}{3} + I.n^2$additions et multiplications plutôt que $I\frac{n^3}{3}$.

1.3.1.4 Et le stockage...

Pour une matrice de taille n, on remarque que le nombre de termes à connaître pour disposer des matrices $\mathbf{L}$ et $\mathbf{U}$ n'est que de n^2, puisque la diagonale unité de L n'est pas à stocker. De ce fait on peut utiliser la place mémoire disponible dans $\mathbf{A}$ pour y enregistrer les termes de $\mathbf{L}$ et $\mathbf{U}$.

Cette remarque explique que l'algorithme qui suit "écrase" la matrice $\mathbf{A}$ par sa décomposition $\mathbf{LU}$.

1.3.1.5 Algorithme

La factorisation $\mathbf{LU}$ présentée au paragraphe (1.3.1.1) n'est pas implémentée selon la méthode décrite : il existe une algorithme permettant de calculer directement les termes des matrices $\mathbf{L}$ et $\mathbf{U}$. On notera dans l'algorithme ci-dessous que l'ordre de caclul des coefficients n'est pas indifférent.

Voici l'algorithme réalisant la factorisation $\mathbf{LU}$ d'une matrice $\mathbf{A}$ et stockant cette factorisation dans la place mémoire occupée par $\mathbf{A}$: (la matrice $\mathbf{A}$ est perdue, on dit qu'on écrase $\mathbf{A}$).

Algorithme : `Factorisation` **LU**

```
Pour j ← 1 à n-1
  Pour i ← j+1 à n
     A(i,j) = A(i,j)/A(j,j) //construction de la j-ième colonne de L
     Pour k ← j+1 à n
       A(i,k) = A(i,k) - A(i,j) × A(j,k) //actualisation des n-j-1 dernières lignes de A
     FinPour
  FinPour
FinPour
```

1.3.1.6 Exercice

EXERCICE 1.2 *On considère la matrice de Vandermonde*

$$\mathbf{A} = (a_{ij}) \text{ avec } a_{ij} = x_i^{j-1}, i, j = 1, ...n$$

où les x_i *sont* n *abscisses distinctes.*

(1) Programmer la factorisation **LU** *de la matrice* $\mathbf{A}$ *pour des valeurs de* n *comprises entre* 10 *et* 60 *par pas de* 10. *On pourra utiliser des* x_i *définis par* $x_i = i$, *pour* $i = 1, ...n$.

(2) Calculer et représenter graphiquement le nombre d'opérations effectuées.

(3) Retrouver l'ordre de complexité de la méthode. On pourra utiliser l'option de la commande `plot2d logflag` *qui permet d'afficher des échelles logarithmiques, ainsi que la commande* `reglin` *fournissant les coefficients* a *et* b *de l'équation de la droite de régression linéaire appliquée à un nuage de points.*

1.3.1.7 CNS d'existence d'un factorisation LU

THÉORÈME 1.3.1 *Soit* $\mathbf{A}$ *une matrice de* $\mathcal{M}_m(\mathbb{C})$. *Pour* $1 \leq p \leq n$, *on note* $\mathbf{A}_p$ *le bloc*

$$\mathbf{A}_p = \begin{bmatrix} a_{11} & \cdots & \cdots & a_{1p} \\ a_{21} & & & a_{2p} \\ \vdots & & & \vdots \\ a_{p1} & \cdots & \cdots & a_{pp} \end{bmatrix}$$

La matrice $\mathbf{A}$ *admet une factorisation*

$$\mathbf{A} = \mathbf{LU}$$

où $\mathbf{L}$ *est triangulaire inférieure avec des* 1 *sur la diagonale, et* $\mathbf{U}$ *est triangulaire supérieure et inversible si et seulement si tous les blocs* $\mathbf{A}_p, 1 \leq p \leq n$ *sont inversibles. De plus, cette factorisation est unique. De, plus, si* $\mathbf{A}$ *est réelle,* $\mathbf{L}$ *et* $\mathbf{U}$ *le sont aussi.*

On pourra trouver la démonstration détaillée de ce résultat dans [YA].

1.3.1.8 Exercices

EXERCICE 1.3 *Supposons qu'on résolve* $\mathbf{Ax} = \mathbf{b}$ *avec*

$$\mathbf{A} = \begin{bmatrix} 1 & 1-\varepsilon & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix} \text{ et } \mathbf{b} = \begin{bmatrix} 5-\varepsilon \\ 6 \\ 13 \end{bmatrix}$$

(1) Déterminer pour quelles valeurs de ε la matrice $\mathbf{A}$ ne satisfait pas les hypothèse du théorème ci-dessus.

(2) Pour quelles valeurs de ε cette matrice est-elle singulière ?

(3) Est-il possible de calculer la factorisation dans ce cas ?

EXERCICE 1.4 *Montrer que la factorisation* $\mathbf{LU}$ *d'une matrice* $\mathbf{A}$ *peut être utilisée pour calculer la matrice inverse* $\mathbf{A}^{-1}$. *(On remarquera que la j-ième colonne de* $\mathbf{A}^{-1}$ *vérifie le système linéaire* $\mathbf{Ay}_j = \mathbf{e}_j$, $\mathbf{e}_j$ *étant le j-ième vecteur de la base canonique.)*

EXERCICE 1.5 *Considérons la matrice inversible*

$$\mathbf{A} = \begin{bmatrix} 1 & 1+0.5.10^{-15} & 3 \\ 2 & 2 & 20 \\ 3 & 6 & 4 \end{bmatrix}$$

(1) Effectuer la factorisation $\mathbf{LU}$ *de* $\mathbf{A}$ *à l'aide du programme de l'exercice .*

(2) Calculer le résidu $\mathbf{A} - \mathbf{LU}$, *et montrer qu'il vérifie :*

$$\mathbf{A} - \mathbf{LU} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

(3) Que peut-on en conclure ?

(4) Comparez le résultat que vous obtenez avec celui fourni par scilab avec la routine `lu`. *Analysez la différence.*

EXERCICE 1.6 *Soit la matrice* $\mathbf{A} \in \mathbb{R}^{n \times n}$. *On définit la matrice* $\mathbf{B} = [\mathbf{A}, \mathbf{I}_n] \in \mathbb{R}^{n \times 2n}$, *où* $\mathbf{I}_n$ *la matrice identité.*

Pour calculer l'inverse de $\mathbf{A}$ *on établit l'algorithme suivant*

```
Pour j ← 1 à n
  Pour i ← 1 à n
    Si  (i = j)
      Pour k ← 1 à n
        B'(j,k) = B(j,k)/B(j,j)
      FinPour
    Sinon
      Pour k ← 1 à n
        B'(i,k) = B(i,k) - B(i,j) * B'(j,k)
      FinPour
    FinSi
    B ← B'
  FinPour
FinPour
```

(1) *Montrer que le résultat est équivalent à multiplier à gauche* $\mathbf{B}$ *par une suite de matrices* $\mathbf{C}^{(i)}$ *que vous calculerez.*

(2) *Montrer qu'à la fin de l'algorithme les* n *premières colonnes de la matrice* $\mathbf{B}$ *forment la matrice identité* $\mathbf{I}_n$.

(3) *En déduire que les* n *dernières colonnes de* $\mathbf{B}$ *forment* $\mathbf{A}^{-1}$.

(4) *Application :* $\mathbf{A} = \begin{bmatrix} 2 & 4 & 2 \\ 1 & 0 & 3 \\ 3 & 1 & 2 \end{bmatrix}$ *en utilisant les programme* `inverMat`

1.4 Méthode de Cholesky

Dans le cas où la matrice $\mathbf{A}$ est hermitienne et définie positive, on peut toujours effectuer la factorisation décrite ci-dessus. de plus, on peut trouver une factorisation du type $\mathbf{A} = \mathbf{L}\mathbf{L}^*$ moins gourmande en place mémoire.

1.4.1 Existence de la factorisation

THÉORÈME 1.4.1 *Si* $\mathbf{A}$ *est hermitienne et définie positive, alors* $\mathbf{A}$ *admet une unique factorisation* $\mathbf{LU}$ *où* $\mathbf{L}$ *est triangulaire inférieure avec des* 1 *sur la diagonale et* $\mathbf{U}$ *est triangulaire supérieure et inversible.*

DÉMONSTRATION. Si $\mathbf{A}$ est hermitienne et définie positive, ses blocs $\mathbf{A}_p, 1 \leq p \leq n$ (voir théorème ci-dessus) le sont aussi, et on peut donc appliquer le théorème. ∎

THÉORÈME 1.4.2 *Si* $\mathbf{A}$ *est hermitienne et définie positive, alors il existe une unique matrice* $\mathbf{L}$ *triangulaire inférieure et inversible, avec des coefficients positifs sur la diagonale telle que*

$$\mathbf{A} = \mathbf{L}\mathbf{L}^*$$

Cette factorisation porte le nom de Cholesky (colonel de l'armée de Napoléon). De plus si $\mathbf{A}$ *est réelle, symétrique et définie positive,* $\mathbf{L}$ *est réelle.*

On pourra trouver la démonstration dans [YA].

REMARQUE 1.4.1 *Il est important de noter que les matrices* $\mathbf{L}$ *dans les factorisation* $\mathbf{LU}$ *et de Cholesky sont différentes.*

1.4.2 Algorithme

Algorithme : Algorithme de Cholesky

$L(1,1) \leftarrow \sqrt{A(1,1)}$ // Construction de l_{11}
Pour i ← 2 **à** n
  Pour j ← 1 **à** i-1
    $L(i,j) \leftarrow \frac{1}{L(j,j)} \left(A(i,j) - \sum_{k=1}^{j-1} L(i,k) \times L(j,k) \right)$
  FinPour
  $L(i,i) \leftarrow \left(a(i,i) - \sum_{k=1} i-1 L^2(i,k) \right)^{1/2}$
FinPour

1.4.3 Complexité

On montre facilement que l'on effectue :

- $\frac{n^3}{6}$ additions + multiplications
- $\frac{1}{2}n(n-1)$ divisions
- n évaluations de racines carrées.

REMARQUE 1.4.2 *L'intérêt de la méthode de Cholesky réside dans le fait qu'elle demande une place mémoire deux fois inférieure à celle de la factorisation LU, puisqu'on ne stocke que* $\mathbf{L}$ *et que sa complexité est aussi deux fois moindre.*

1.5 Elimination de Gauss avec recherche de pivot partiel

La méthode de Gauss peut-être bloquée si on tombe sur un pivot nul. C'est pour cela qu'a été mise au point la méthode dite "de Gauss avec pivotage partiel". Elle consiste à échanger les lignes du système lorsqu'on tombe sur un pivot nul, ce qui permet de continuer sans problème.

Elle repose sur l'utilisation de matrices de permutation, dont on rappelle ci-dessous la définition :

DÉFINITION 1.5.1 *On appelle permutation de {1,...,n} une bijection de {1,...,n} sur {1,...,n}.*

DÉFINITION 1.5.2 *Soit* σ *une permutation de {1,...,n}, on associe à* σ *une matrice* $\mathbf{P}$ *dite de permutation d'ordre* n *par*

$$P_{ij} = \delta_{\sigma(i)j}$$

c'est à dire

$$P_{ij} = 1 \text{ si } j = \sigma(i)$$
$$P_{ij} = 0 \text{ si } j \neq \sigma(i)$$

Les matrices de permutation ont des propriétés intéressantes, qui font que leur utilisation pour effectuer les pivotages conduit au résultat suivant :

THÉORÈME 1.5.1 *Soit* $\mathbf{A} \in \mathcal{M}_m(\mathbb{R})$, *inversible. Alors il existe :*
- *une matrice de permutation* $\mathbf{P}$,
- *une matrice triangulaire inférieure* $\mathbf{L}$, *avec des* 1 *sur la diagonale,*
- *une matrice triangulaire supérieure* $\mathbf{U}$ *inversible,*

telles que

$$\mathbf{PA} = \mathbf{LU}$$

REMARQUE 1.5.1 *Le pivotage partiel ne sert pas seulement à garantir l'obtention d'une factorisation. Il garantit aussi une meilleure stabilité que celle obtenue par la méthode* $\mathbf{LU}$ *dans le cas de matrices mal conditionnées. On pourra s'en convaincre en examinant le cas du système* $\mathbf{Ax} = \mathbf{b}$, *constitué de la matrice* $\mathbf{A}$ *:*

$$\mathbf{A} = \begin{bmatrix} 10^{-9} & 1 \\ 1 & 1 \end{bmatrix}$$

et du second membre $\mathbf{b}$ *:*

$$\mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Sa solution est

$$\mathbf{x} = \left[\frac{1}{1-10^{-9}}, \frac{1-2.10^{-9}}{1-10^{-9}} \right]^\top \simeq [1,1]^\top$$

Si on le résout sur une machine à 8 chiffres significatifs, la factorisation calculée est

$$\mathbf{U} = \begin{bmatrix} 10^{-9} & 1 \\ 0 & -10^{-9} \end{bmatrix} \text{ et } \mathbf{L} = \begin{bmatrix} 1 & 0 \\ 10^{-9} & 1 \end{bmatrix}$$

ce qui donne la solution

$$\mathbf{x} = [0,1]^\top$$

Si on utilise le pivotage partiel, on obtient :

$$\mathbf{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ et } \mathbf{L} = \begin{bmatrix} 1 & 0 \\ 10^{-9} & 1 \end{bmatrix}$$

ce qui donne la solution

$$\mathbf{x} = [1,1]^\top$$

En exploitant cette remarque, on aboutit à la méthode de Gauss dite de *pivot maximal*. Elle consiste à prendre, à chaque étape, comme pivot l'élément diagonal qui est le plus grand en

valeur absolue afin de limiter les erreurs d'arrondi inévitables dues à un résultat de division trop faible (pertes de chiffres significatifs).

EN SUBSTANCE

- ***La factorisation LU d'une matrice A consiste à calculer une matrice triangulaire inférieure L et une matrice triangulaire supérieure U telles que $\mathrm{A} = \mathrm{LU}$.***
- ***La factorisation LU, quand elle existe, n'est pas unique. Cependant, on peut la rendre unique en se donnant des conditions supplémentaires, par exemple en fixant les valeurs des éléments diagonaux de L à 1. Ceci s'appelle factorisation de Gauss.***
- ***La factorisation de Gauss existe et est unique si et seulement si les mineurs principaux de A d'ordre 1 à $n-1$ sont non nuls (autrement, au moins un pivot est nul).***
- ***Quand on trouve un pivot nul, un nouveau pivot peut être obtenu en échangeant des lignes (ou colonnes) convenablement choisies. C'est la stratégie du pivot.***
- ***Le calcul de la factorisation de Gauss nécessite de l'ordre de $2n^3/3$ opérations en général, et seulement de l'ordre de n opérations dans le cas d'un système tridiagonal.***
- ***Pour les amtrices symétriques définies positives, on peut utiliser la factorisation de Cholesky $\mathrm{A} = \mathrm{HH}^T$, où H est une matrice triangulaire inférieure. Le coût de calcul est de l'ordre de $n^3/3$ opérations.***
  La sensibilité du résultat aux perturbations des données dépend du conditionnement de la matrice du système : la solution calculée peut être imprécise quand la matrice est mal conditionnée.

1.6 Bibliographie

Les ouvrages ci-dessous sont disponibles sous forme de fichier téléchargeable sur le site du cours ou sur Arel

[CB] **Algèbre matricielle numérique**, *Claude Brezinski*

[YA] **Algèbre linéaire et analyse numérique matricielle**, *Yves Achdou*, téléchargeable à l'adresse http ://www.ann.jussieu.fr/~achdou/files/teaching/linalg/book.pdf

[AH1]**Analyse numérique matricielle, Cours de 3ème année**, *Alain Huard*, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

[AH2]**Analyse numérique des grands problèmes linéaires, Cours de 4ème année**, *Alain Huard*, téléchargeable à partir du site de l'INSA Toulouse www-gmm.insa-toulouse.fr

Les ouvrages ci-dessous sont disponibles en librairie

[QS] **Calcul scientifique, Cours, exercices corrigés et illustrations en Matlab et Octave**, Alfio Quarteroni, Fausto Saleri, Springer, 2206.

[AF] **Analyse numérique pour Ingénieurs**, André Fortin, Presses Internationales Polytechnique, 2001.

Table des matières