CERGY PARIS UNIVERSITÉ

Physique

Modélisation, méthodes numériques pour la physique

Ch. Oguey

LPTM, Institut ST
CY Cergy Paris Université

01/09/2021

Table des matières

Bibliographie

1. W Press, SA Teukolsky *et al.* : *Numerical Recipes: The Art of Scientific Computing* 3d ed. (Cambridge UP 2007)
2. R Landau, M Paez, C Bordeianu : *Computational physics: problem solving with computers* 3d ed. (Wiley 2015)
3. H Gould, J Tobochnik : *An Introduction to Computer Simulations* (Addison-Wesley 1996)
4. D. Frankel, B. Smit : *Understanding Molecular Simulations* 2nd ed. (Acad. Press 2002) doi.org/10.1016/B978-0-12-267351-1.X5000-7
5. G H Golub, Van Loan : *Matrix computations* (Johns Hopkins UP 1996)
6. M Plischke, B Bergersen : *Equilibrium Statistical Physics* 3d ed. (World Scient. 2006) doi.org/10.1142/5660
7. J Rappaz, M Picasso : *Introduction à l'analyse numérique* (PPUR 1998)
8. C Abrams, Molecular Simulations, Drexel University, Philadelphia

C. Oguey, CYU, F-95031 Cergy-Pontoise, oguey@u-cergy.fr

Chapitre 1

Résolution d'équations linéaires

L'objectif est de résoudre numériquement des systèmes d'équations linéaires. Nous aurons en vue trois aspects :

La méthode : En quoi consiste l'algorithme ? Comment fonctionne-t-il ?

La preuve : Vérifier que la méthode donne le bon résultat.

La mise en oeuvre : Traiter des cas concrets dans le cadre de projets réalisés en TD.

Dans ce chapitre, A est une matrice de m lignes par n colonnes à coefficients dans un corps K, typiquement $\mathbb{R}$ ou $\mathbb{C}$.

1.1 Combinaisons de lignes

I_m désigne la matrice identité m par m. Pour $1 \leq i, j \leq m$ entiers, E_{ij} est la matrice carrée nulle partout sauf en i, j où l'élément vaut $(E_{ij})_{ij} = 1$.

Proposition 1.1 *Pour tous $1 \leq i \neq j \leq m$ et $h \in K$, la matrice $I_m + hE_{ij}$ est inversible, de déterminant 1, et son inverse est $I_m - hE_{ij}$.*

Démonstration : comme la matrice est triangulaire ($i \neq j$), le déterminant est le produit des éléments diagonaux, 1. (On peut aussi développer le dét selon la ligne i, ou la colonne j). L'inverse suit de $(I_m + hE_{ij})(I_m - hE_{ij}) = I_m + hE_{ij} - hE_{ij} - h^2(E_{ij})^2 = I_m$ car $(E_{ij})^2 = 0$, facile à vérifier. □

Proposition 1.2 (Combinaisons de lignes ou colonnes) *Soient $1 \leq i \neq j \leq m$ et $h \in K$. La matrice $(I_m + hE_{ij})\,A$ se déduit de A en ajoutant à la ligne i de A h fois la ligne j de A, les autres lignes restant inchangées.*

Soient $1 \leq k \neq \ell \leq n$. La matrice $A(I_n + hE_{k\ell})$ se déduit de A en ajoutant à la colonne ℓ de A h fois la colonne k de A, les autres colonnes restant inchangées.

Démonstration : distribuer le produit ; $E_{ij}A$ est nulle partout sauf à la ligne i qui contient la ligne j de A. □

1.2 Réduction à la forme triangulaire – procédé d'élimination de Gauss

Proposition 1.3 (Lemme fondamental) *Il existe et on construit une matrice P inversible telle que $(PA)_{i1} = 0,\ \forall i = 2, \ldots, m$. (La première colonne de PA est nulle sous la diagonale). P est donnée par*

$P = (I_m - \frac{A_{m1}}{A_{11}}E_{m1}) \ldots (I_m - \frac{A_{21}}{A_{11}}E_{21})$ *si $A_{11} \neq 0$,*

$P = (I_m - \frac{A_{m1}}{A_{p1}}E_{m1}) \ldots (I_m - \frac{A_{21}}{A_{p1}}E_{21})(I_m + E_{1p})$ *si $A_{11} = 0$ et un élément $A_{p1} \neq 0$.*

$P = I_m$ *si $A_{21} = \ldots = A_{m1} = 0$ (1re colonne déjà nulle).*

Démonstration : directe par calcul. □

Théorème 1.1 (Procédé d'élimination de Gauss) *Il existe et on construit une matrice P, produit de matrices $I_m + hE_{ij}$ avec $1 \leq i \neq j \leq m$ et $h \in K$, telle que PA est triangulaire supérieur : $(PA)_{k\ell} = 0\ \forall\, k > \ell$.*

Démonstration : appliquer le lemme successivement à A et aux sous-matrices obtenues en omettant, à chaque étape $k = 1, \ldots, \min(m, n)$, la première ligne et la première colonne de la matrice précédente. Noter que le produit par les matrices $I_m + hE_{ij}$ ne change pas les lignes et colonnes omises. □

Remarques

- Puisque c'est un produit de $I_m + hE_{ij}$ avec $i \neq j$, la matrice P est inversible et de déterminant 1.
- Dans le cas $m = n$, il s'en suit que $\det(PA) = \det(A)$. Or le déterminant d'une matrice triangulaire (ici PA) est le produit des éléments diagonaux. La réduction de Gauss à la forme triangulaire fournit donc une méthode efficace pour calculer le déterminant.

Coût opératoire

Numériquement, il n'est pas intéressant d'implémenter l'opération $(I_m + hE_{ik})A$ comme le produit des deux matrices (avec une routine ad hoc). En effet, le produit général de deux matrices n par n réelles requiert $n^2(2n-1)$ flop (opérations élémentaires en virgule flottante), soit $2n^3 + O(n^2)$ flop si n est grand.

En revanche, calculer directement la combinaison de lignes,

```
Pour j=1,..,n
   A[i][j] = A[i][j] + h*A[k][j],
```

ne nécessite que $2n$ flop. Par conséquent, le programme sera bien plus efficace s'il suit cette dernière méthode : $2n \ll 2n^3$ si n est grand.

Algorithme 1.1 (Pivots dangereux)

```
Declarer et initialiser A (m x n)
Pour k=1,..,min(m,n)
|  pour i=k+1,..,m
|  |  poser h = A[i][k]/A[k][k];
|  |  pour j=k,..,n
|  |  |_  A[i][j] = A[i][j] - h*A[k][j];
|  |_
|_
```

Cet algorithme transforme, dans le tableau `A`, la matrice initiale A en matrice triangulaire supérieure $T = PA$.

L'élément de matrice A_{kk} est appelé *pivot*. Il délimite le coin supérieur gauche de la sous-matrice où l'on opère à l'étape k.

1.3 Équations linéaires

Étant donnés une matrice A et un vecteur $b \in K^m$ connus,

$$Ax = b \tag{1.1}$$

définit une équation vectorielle linéaire inhomogène pour le vecteur inconnu $x \in K^n$. Écrite en coordonnées, cette équation vectorielle équivaut à un système de m équations (scalaires) linéaires inhomogènes à n inconnues (scalaires).

Puisque P, du procédé de Gauss, est inversible, l'équation (1.1) est équivalente à

$$(PA)x = Pb, \tag{1.2}$$

où la matrice $T \equiv PA$ est triangulaire supérieure. Or un système triangulaire se résout facilement par retro-substitution.

Retro-substitution

Supposons $m \geq n$. Si les lignes $n+1, \ldots, m$ de $b' = Pb$ ne sont pas nulles, les équations sont incompatibles et le système n'admet aucune solution. Dans le cas contraire, la dernière ligne non nulle de l'équation $Tx = b'$ ne contient qu'une inconnue :

$$T_{nn}\, x_n = b'_n \quad \Rightarrow \quad x_n = b'_n / T_{nn}.$$

Insérer cette solution dans l'avant-dernière équation scalaire $T_{n-1n-1}\, x_{n-1} + T_{n-1n}\, x_n = b'_{n-1}$ donne

$$x_{n-1} = (b'_{n-1} - T_{n-1n}\, x_n)/T_{n-1n-1}$$

et ainsi de suite.

Algorithme 1.2 (Retro-substitution)

```
pour i=n, n-1,..,1
|  s = 0;
|  pour j=i+1,..,n
|  |_  s = s + T[i][j]*x[j];
|_ x[i]= (b[i]-s)/T[i][i];
```

La boucle interne calcule la somme $s = \sum_{j=i+1}^{n} T_{ij}\, x_j$. Cette somme requiert $2(n-i)$ opérations arithmétiques. Donc au total, l'algorithme 1.2 implique $\sum_{i=n}^{1}(2(n-i)+2) = 2\sum_{i=1}^{n} i = 2n(n+1)/2$, soit $\sim n^2$ flop en tout pour n grand.

1.4 Conditionnement des matrices

Une matrice, ou une équation linéaire $Ax = b$, est mal conditionnée si de petites variations des données (A, b) entraînent de grandes variations des solutions x.

Ex. : Résoudre chacun des systèmes

1) $x_1 + x_2 = 2$ et $x_1 + 1.00001x_2 = 1.99999$,
2) $x_1 + x_2 = 2$ et $x_1 + 1.00001x_2 = 2.00001$.

Quantitativement, commençons par lever une part d'arbitraire : multiplier les deux membres d'une équation par un même facteur non nul ne change pas les solutions. Il est toujours conseillé de *normaliser* les lignes pour que les plus grands coefficients en valeur absolue soient de l'ordre de 1 : $\max\{|A_{i1}|, .., |A_{in}|, |b_i|\} \simeq 1$. Dans ces conditions, si $|\det A|$ est petit, le système est mal conditionné.

Un critère plus robuste fait appel aux *valeurs singulières* $\sigma_1, \ldots, \sigma_{\min(m,n)}$ de A. Il s'agit d'une factorisation

$$A = U\, \mathrm{diag}_{mn}(\sigma_1, \ldots, \sigma_{\min(m,n)})\, V^\dagger$$

où U et V sont unitaires, diag_{mn} est diagonale (m par n) et les éléments diagonaux satisfont $\sigma_j \geq 0$. Le conditionnement est défini comme le rapport

$$\text{conditionnement} = \frac{\max\{\sigma_j\}}{\min\{\sigma_j\}},$$

au moins égal à un, égal à $+\infty$ si $\min\{\sigma_j\} = 0$. Plus ce conditionnement est grand, plus le système est mal conditionné.

Optimisation des pivots

Dans la pratique, on connaît rarement le conditionnement a priori. On a vu que, justement, le procédé de Gauss est une bonne méthode pour calculer le déterminant, notre premier critère. Donc il s'agit de capter un éventuel mauvais conditionnement en cours d'exécution. Une méthode courante consiste à maximiser et surveiller la valeur absolue des éléments diagonaux de la matrice triangulaire, autrement dit des pivots. En résumé :

1. Au départ, normaliser la matrice ligne par ligne.
2. A chaque étape k, choisir, comme pivot, l'élément de plus grande valeur absolue dans la colonne k sous la diagonale.

On ne change pas un système d'équations (donc les solutions) si l'on permute les équations ; autrement dit si l'on échange des lignes de la matrice prise avec le second membre : (A, b). Dans l'algorithme, à l'étape k, avant toute combinaison de lignes, on cherche dans la colonne k l'élément maximal en v.a., disons A_{pk} avec $k \leq p \leq m$. S'il est (approximativement) nul, on a affaire à un système singulier ou mal conditionné. Sinon, on échange les lignes k et p de (A, b), affectant ainsi la valeur maximale au pivot A_{kk}. Ensuite, on procède comme dans l'algorithme 1.1.

Noter que, dans l'algorithme 1.1, nous avons divisé par A_{kk} sans vérifier que le dénominateur n'était pas nul (ou proche de zéro). L'optimisation partielle corrige aussi cette faiblesse.

Algorithme 1.3 (Élimination de Gauss avec optimisation partielle)

```
precision=1e-6     // ajuster au contexte
Pour k=1,..,min(m,n)
|  trouver p entre k et m tel que
|                    |A[p][k]| soit max
|  si |A[p][k]| < precision
|  |  afficher "Pivot nul, mat.singuliere"
|  |_ sortir      // arrete l'execution
|  echanger les lignes k et p
|  pour i=k+1,..,m
```

```
|  |  h = A[i][k]/A[k][k];
|  |  pour j=k,..,n
|  |  |_  A[i][j] = A[i][j] - h*A[k][j];
|  |_
|_
```

Dans le cas $m = n$ régulier, le nombre d'opérations élémentaires dans cet algorithme est de l'ordre de

$$\sum_{k=1}^{n} 2(n-k)^2 \simeq 2\sum_{k=1}^{n} k^2 \simeq \frac{2}{3}n^3 \text{ flop},$$

en utilisant $\sum_{k=1}^{n} k^2 = \frac{1}{6}n(1+n)(1+2n)$.

Pour résoudre l'équation $Ax = b$, on enchaîne l'algorithme 1.3, qui la transforme en système triangulaire $Tx = b'$, et l'algorithme 1.2 de retro-substitution qui donne les solutions x. Pour n grand, le coût total de la résolution est dominé par la partie triangularisation : $\frac{2}{3}n^3 + n^2 \simeq \frac{2}{3}n^3$ flop.

Complément L'optimisation complète consisterait chercher l'élément maximum non seulement dans la colonne k, mais aussi dans la ligne k à droite de la diagonale. Échanger des colonnes de la matrice A change l'ordre des coordonnées du vecteur x. Il faut donc en garder mémoire pour l'affichage final des résultats.

En pratique, l'optimisation complète est rarement nécessaire. L'optimisation partielle suffira à nos besoins.

1.5 Calculer l'inverse

A toute permutation s de $\{1, \ldots, n\}$, on associe une matrice orthogonale définie par $S_{ij} = \delta_{is(j)}$. On vérifie que SA s'obtient en permutant les lignes de A par s.

Supposons la matrice A carrée : $m = n$. Le procédé d'élimination de Gauss optimisé transforme A en matrice triangulaire supérieure $T = PA$ où P est un produit de matrices 'combinaisons de lignes' et de matrices de permutations.

Si un élément diagonal T_{ii} est nul (ou quasi-nul compte tenu des arrondis), det$A$ l'est aussi et A n'a pas d'inverse.

Sinon, tous les éléments diagonaux sont significativement différents de zéro et l'on peut réduire T à la forme diagonale par combinaison de lignes (Gauss-Jordan).

Proposition 1.4 *Si A est inversible, il existe et on construit une matrice P, produit de matrices $I_m + hE_{ij}$ avec $1 \leq i \neq j \leq m$ et $h \in K$ et de matrices de permutations, telle que PA est diagonale : $(PA)_{k\ell} = 0 \; \forall k \neq \ell$.*

L'inverse de la matrice diagonale $D = PA$ se calcule facilement : $D^{-1} = \text{diag}(\frac{1}{D_{11}}, \ldots, \frac{1}{D_{nn}})$. De $D^{-1}D = I_n = D^{-1}PA$ on déduit que $A^{-1} = D^{-1}P$. Pour trouver P, il suffit d'opérer sur I_n les mêmes combinaisons de lignes et permutations que celles qu'on effectue sur A, car $PI_n = P$.

Algorithme 1.4 (Inverse par doublement de tableau et optimisation partielle)

```
precision=1e-6       // ajuster
n2=n*2;     // etendre A à n2 colonnes
definir A[:][n+1..n2] = I_n
Pour k=1,..,n
|  trouver p entre k et n tq
|                  |A[p][k]| soit max
|  si |A[p][k]| < precision
|  |   afficher "matrice singuliere"
|  |_  stop
|  echanger les lignes k et p
|  poser h = A[k][k]; // le pivot
|  pour j=k,..,n2
|  |_  A[k][j] = A[k][j]/h;
|  pour i=1,..,n et i!=k
|  |  poser h = A[i][k];
|  |  pour j=k,..,n2
|  |  |_  A[i][j] = A[i][j] - h*A[k][j];
|  |_
|_
```

A la fin, la moitié gauche du tableau contient l'identité et la moitié droite contient A^{-1}.
Coût : $\sum_{k=1}^{n} 2n(2n-k) \simeq 2n(2n^2 - n(n+1)/2) \simeq 3n^3$ flop.

1.6 Systèmes sur- et sous-déterminés, moindres carrés

1.6.1 Systèmes sur-déterminés

Exemple : Mesurer la longueur d'un barreau L en fonction de la températures T donne une série de points expérimentaux $(T_i, L_i), i = 1, \ldots, m$. On cherche une loi affine passant par ces points : $L = x_1 + x_2 T$. La détermination des coefficients donne lieu à un système surdéterminé d'équations linéaires :

$$\left.\begin{array}{c} x_1 + x_2 T_1 = L_1 \\ \cdots \\ x_1 + x_2 T_m = L_m \end{array}\right\} \Leftrightarrow Ax = b, \; (A,b) = \begin{pmatrix} 1 & T_1 & L_1 \\ : & : & : \\ 1 & T_m & L_m \end{pmatrix}.$$

En général, si $m > n$, il n'existe pas de solution exacte au système $Ax = b$. Mais il existe des solutions approchées, dont l'écart $r = Ax - b$ est petit. On cherche la solution dite de *moindres carrés* qui minimise $||r||^2 = \sum_{i=1}^{m} r_i^2$.

Proposition 1.5 *Soit $f(x) = ||r||^2 = ||Ax - b||^2$. Le vecteur $x \in K^n$ minimise f seulement si ${}^tAAx = {}^tAb$.*

Démonstration : développer $f(x) = (Ax - b) \cdot (Ax - b) = Ax \cdot Ax - b \cdot Ax - Ax \cdot b + b \cdot b = x \cdot {}^tAAx - 2x \cdot {}^tAb + ||b||^2$. Pour minimiser f, une condition nécessaire est que le gradient ∇f s'annule en x. Or $\nabla f(x) = 2\,{}^tAAx - 2\,{}^tAb$. □

On résout ${}^tAAx = {}^tAb$ par les méthodes vues aux § 1.3-1.4. Si A est de rang n, comme c'est souvent le cas, alors rang$({}^tAA) = n$ est maximal car ${}^tAA$ est carrée $n \times n$. Donc ${}^tAA$ est inversible et l'on peut écrire :

$$x = ({}^tAA)^{-1}\,{}^tAb.$$

1.6.2 Problèmes sous-déterminés

L'ensemble des solutions de $Ax = b$ forme un sous-espace affine de K^n. Si $m < n$ et, de plus, rang$(A) = m$, la solution la plus petite, minimisant $||x||^2$, est donnée par

$$x = {}^tA(A\,{}^tA)^{-1}b.$$

La démonstration de ce fait est laissée en exercice.