1 Résolution numérique de systèmes linéaires

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

$$\underbrace{\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}}_{\vec{x}} = \underbrace{\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}}_{\vec{b}}$$

Si peu d'éléments sont non-nuls alors A est dite **creuse**

1.1 Condition d'arrêt

$$\boxed{||\vec{r}_k|| = \left|\left|\vec{b} - A\vec{x}_k\right|\right|}$$

Tolérance fixe τ (par exemple 10^{-5})

$$||\vec{r}_k|| \leq \tau \left|\left|\vec{b}\right|\right|$$

Erreur :

$$\vec{e}_k = \vec{x} - \vec{x}_k$$

On peut aussi utiliser une condition d'arrêt sur l'erreur $\vec{e}_k$ au lieu du résidu $\vec{r}_k = \vec{b} - A\vec{x}_k$

1.1.1 Lien entre résidu et erreur

$$\frac{||\vec{x} - \vec{x}_k||}{||\vec{x}_k||_p} \leq \underbrace{||A||_p \left|\left|A^{-1}\right|\right|_p}_{\kappa_p(A)} \frac{\left|\left|\vec{b} - A\vec{x}_k\right|\right|_p}{\left|\left|\vec{b}\right|\right|_p}$$

Autant de digits valides dans la mantisse que

$$N_{\text{digits}} = |\log_{10}(\varepsilon)| - \log_{10}(\kappa(A)_p)$$

Avec ε la précision machine (1e-16 en général)

1.1.2 Perturbation

Perturbation sur A

$$\boxed{\frac{||\delta\vec{x}_A||}{||\vec{x} + \delta\vec{x}_A||} \leq \left|\left|\vec{A}\right|\right| \cdot \left|\left|\vec{A}^{-1}\right|\right| \cdot \frac{||\delta A||}{||A||}}$$

Perturbation sur A et $\vec{b}$:

$$\boxed{\frac{||\delta\vec{x}||}{||\vec{x} + \delta\vec{x}||} \leq \frac{\left|\left|\vec{A}\right|\right| \cdot \left|\left|\vec{A}^{-1}\right|\right|}{1 - ||A|| \cdot ||A^{-1}|| \frac{||\delta A||}{||A||}} \cdot \left(\frac{||\delta A||}{||A||} + \frac{\left|\left|\delta\vec{b}\right|\right|}{\left|\left|\vec{b}\right|\right|} \right)}$$

1.2 Normes

$$||\vec{v}||_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}$$

1. Vecteurs
   (a) 1-norme : somme des composantes
   (b) 2-norme : norme euclidienne
   (c) max-norme : $p \to \infty$ max des valeurs absolues
2. Matrices
   (a) 1-norme : $\max(\sum |A|)$ (la somme est colonne par colonne).
   (b) 2-norme : ou max des valeurs propres de $A^T A$
   (c) max-norme : $\max(\sum |A|)$ (la somme est ligne par ligne).

1. $||\vec{v}|| = 0 \longleftrightarrow \vec{v} = \vec{0}$
2. $||\lambda \vec{v}|| = |\lambda| \cdot ||\vec{v}||$
3. $||\vec{v} + \vec{u}|| \leq ||\vec{v}|| + ||\vec{u}||$

1.3 Méthodes directes

1.3.1 Élimination de Gauss sans pivot

Effectuer des combinaisons linéaires des lignes pour obtenir une matrice triangulaire supérieure. Matrice augmentée :

$$\left(\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0.1 & 6 & 6.1 \\ 0 & 2.5 & 5 & 2.5 \end{array}\right)$$

1. Commencer par la colonne de gauche
2. Si nécessaire, permuter les lignes pour avoir un non-nul comme premier élément
3. Faire les combinaisons linéaires des lignes pour annuler les éléments inférieurs
4. Passer à la colonne suivante

1.3.2 Élimination de Gauss avec pivot

On fait le premier pas normalement (0 en dessous du premier élément). Mais avant d'effectuer le prochain pas, on échange les lignes pour avoir le plus grand élément en diagonale. A la fin on se retrouve avec, par exemple :

$$A = \begin{pmatrix} 10 & x & x \\ 0 & 6 & x \\ 0 & 0 & 0.01 \end{pmatrix}$$

1.4 LU

$$A\vec{x} = \vec{b} \longrightarrow L\underbrace{U\vec{x}}_{\vec{y}} = \vec{b}$$

L'avantage est que si on change b, il n'y a pas besoin de tout recommencer.

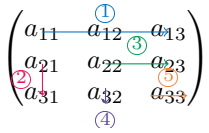
Avec pivotement Si on applique le pivotement, on a une matrice de permutation P

$$LU = PA$$

1.4.1 Doolittle

Méthode équivalente à l'élimination de Gauss

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$



$$(1) \begin{cases} u_{11} & = a_{11} \\ u_{12} & = a_{12} \\ u_{13} & = a_{13} \end{cases} \qquad (2) \begin{cases} l_{21} & = \frac{a_{21}}{u_{11}} \\ l_{31} & = \frac{a_{31}}{u_{11}} \end{cases}$$

$$(3) \begin{cases} u_{22} & = a_{22} - l_{21}u_{12} \\ u_{23} & = a_{23} - l_{21}u_{13} \end{cases} \qquad (4) \left\{ l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} \right.$$

$$(5) \left\{ u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} \right.$$

Formules générales (au cas ou matrice plus petite ou plus grande) :

$$u_{km} = a_{km} - \sum_{j=1}^{k-1} l_{kj}u_{jm} \qquad m = k, k+1, k+2, ..., n$$

$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk} \right) \qquad i = k+1, k+2, ...$$

1.4.2 Cholesky

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} = \begin{pmatrix} \hat{l}_{11} & 0 & 0 \\ \hat{l}_{21} & \hat{l}_{22} & 0 \\ \hat{l}_{31} & \hat{l}_{32} & \tilde{l}_{33} \end{pmatrix} \cdot \begin{pmatrix} \hat{l}_{11} & \hat{l}_{21} & \hat{l}_{31} \\ 0 & \hat{l}_{22} & \hat{l}_{32} \\ 0 & 0 & \tilde{l}_{33} \end{pmatrix}$$

$$\hat{l}_{11} = \sqrt{a_{11}} \qquad \hat{l}_{21} = \frac{a_{12}}{\sqrt{a_{11}}}$$

$$\hat{l}_{31} = \frac{a_{13}}{\sqrt{a_{11}}} \qquad \hat{l}_{22} = \sqrt{a_{22} - \frac{a_{12}}{\sqrt{a_{11}}}}$$

$$\hat{l}_{32} = \frac{a_{23} - \frac{a_{13}a_{12}}{a_{11}}}{\sqrt{a_{22} - \frac{a_{12}}{a_{11}}}} \qquad \hat{l}_{33} = \sqrt{a_{33} - \hat{l}_{31}^2 - \hat{l}_{32}^2}$$

1.4.3 Permutations

Attention, si on utilise des permutations, alors

$$PA\vec{x} = P\vec{b}$$

1.4.4 Méthode QR

R est triangulaire supérieur (en dessous-gauche de la diagonale : que des 0)

$$A = QR \longrightarrow R\vec{x} = Q^{-1}\vec{b} \longrightarrow R\vec{x} = \underbrace{Q_t\vec{b}}_{\vec{c}}$$

$$\begin{pmatrix} \vec{a}_1 & \cdots & \vec{a}_n \end{pmatrix} = \begin{pmatrix} \vec{q}_1 & \cdots & \vec{q}_m \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Avec les propriétés de Q :

- $Q^TQ = I \quad Q^{-1} = Q^T$
- $\det(Q) = \pm 1$

On commence avec une matrice $A_{n\times n}$

1. Prendre le premier vecteur colonne de A : $\vec{x}_1$
2. Déterminer $\vec{y}$

$$y = -\rho \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = -\text{signe}(a_{ii})||x_i|| \left.\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}\right\} \text{taille de } x$$

3. Déterminer $\vec{v}$

$$\vec{v} = \vec{x} - \vec{y}$$

$$\gamma = \frac{||\vec{v}||^2}{2}$$

4. Faire la décomposition pour obtenir H_1

$$\tilde{H} = I - \frac{2}{||v||^2} vv^T$$

$$H = \begin{pmatrix} I & 0 \\ 0 & \tilde{H} \end{pmatrix} \quad \text{(si nécessaire)}$$

5. Construire $H_1 A$ puis prendre v_2 (première colonne de $H_1 A$ **sans la première ligne et sans la première colonne**
6. Recommencer jusqu'à avoir terminé
7. à la fin, déterminer R et Q

$$H_n H_{n-1} \cdots H_2 H_1 A = R \longrightarrow A = \underbrace{H_1 H_2 \cdots H_{n-1}^T H_n^T}_{Q} R$$

1.4.5 Système creux

: Réseaux hydrauliques, thermiques, électriques, etc... Possibilité de faire du "fill-in" si n ou b sont grands mais le coup sera inutilement grand.

1.4.6 Valeur κ

$$\kappa(A)_p = ||A||_p \left|\left|A^{-1}\right|\right|_p$$

En général on utilisera κ_2 si ce n'est pas précisé (mais des fois on calcule κ_∞

1.5 Résolution au sens des moindres carrés

On résout le système

$$A^T A\vec{x} = A^T \vec{b}$$

Si A est mal conditionné, $A^T A$ est doublement mal conditionné

$$\kappa(A^T A) \approx \kappa^2(A)$$

Pour obtenir le nombre de digits corrects, on fait :

$$N_{\text{digits}} = |\log_{10}(\epsilon)| - \log_{10}(\kappa)$$

Avec QR , on a une méthode plus stable

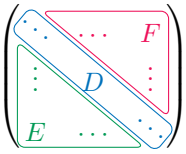
$$\underset{m\times n}{A} = \underset{m\times m}{Q} \underset{m\times n}{\begin{pmatrix} R^* \\ 0 \end{pmatrix}}$$

1.5.1 Minimisation

$$\left|\left|\vec{b} - A\vec{x}\right|\right| = \left|\left|QQ^T\vec{b} - QR\vec{x}\right|\right| = \left|\left|\underbrace{Q^T\vec{b}}_{\vec{c}} - R\vec{x}\right|\right|$$

1. Trouver Q et R avec la méthode au dessus
2. Calculer $\vec{c}$
$$\vec{c} = Q^T\vec{b}$$
3. Trouver R^* (partie "supérieure" de R, donc sans les 0 en bas)
4. Trouver c^* (partie "supérieure" de $\vec{c}$, en ignorant la valeur du bas pour avoir la même hauteur que R^*)
5. Poser le système
$$R^*\vec{x} = \vec{c}^*$$
6. Résoudre à la main, en inversant, c'est égal

1.6 Méthodes itératives



1.6.1 Méthode Jacobi

On commence avec un $\vec{x}(0) = \vec{0}$

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

$$\boxed{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^{(k+1)} = \begin{pmatrix} \frac{1}{a_{11}} \cdot (b_1 - a_{12}x_2^{(k)} \cdots - a_{1n}x_n^{(k)}) \\ \frac{1}{a_{22}} \cdot (b_2 - a_{21}x_1^{(k)} \cdots - a_{2n}x_n^{(k)}) \\ \vdots \\ \frac{1}{a_{nn}} \cdot (b_n - a_{n1}x_1^{(k)} \cdots - a_{n\ n-1}x_{n-1}^{(k)}) \end{pmatrix}}$$

Alternativement :

$$\boxed{\vec{x}^{(k)} = D^{-1}\left(b - (A - D)\vec{x}^{(k-1)}\right)}$$

$$N = D^{-1} = \text{diag}^{-1}(A)$$

Convergence assurée sur A est **strictement diagonalement dominante**1.8

1.6.2 Méthode de Gauss-Seidel

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

$$\boxed{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}^{(k+1)} = \begin{pmatrix} \frac{1}{a_{11}} \cdot (b_1 - a_{12}x_2^{(k)} \cdots - a_{1n}x_n^{(k)}) \\ \frac{1}{a_{22}} \cdot (b_2 - a_{21}x_1^{(k+1)} \cdots - a_{2n}x_n^{(k)}) \\ \vdots \\ \frac{1}{a_{nn}} \cdot (b_n - a_{n1}x_1^{(k+1)} \cdots - a_{nn-1}x_n^{(k+1)}) \end{pmatrix}}$$

Alternativement :

$$\boxed{\vec{x}^{(k)} = (E + D)^{-1} \left(\vec{b} - F\vec{x}^{(k-1)} \right)}$$

$$N = (D + E)^{-1}$$

Gauss-Seidel est bien meilleur que la méthode de Jacobi.
Convergence assurée sur A est **strictement diagonalement dominante**1.8

1.6.3 Méthode SOR

$$\vec{x}^{(k+1)} = (D + \omega E)^{-1} \cdot \left(\omega\vec{b} - (\omega F + (\omega - 1)D)\vec{x}^{(k)} \right)$$

1.6.4 Itération simple

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + N \left(\vec{b} - A\vec{x}^{(k)} \right)$$

$$\max_i |\lambda_i| < 1 \longrightarrow \text{ convergence assurée pour tout } \vec{b}$$

1.7 Convergence

De manière générale on a une expression de la forme

$$\vec{x}^{(k)} = \vec{x}^{(k-1)} + N \left(\vec{b} - A\vec{x}^{(k-1)} \right)$$

Avec un N qui se rapproche de A^{-1} pour que le système soit rapide (mais sans être trop dur à calculer).

$$\boxed{\rho(I - NA) = \max \left(\text{valeurs propres}(I - NA) \right)}$$

$$\rho < 1 \longrightarrow \text{ convergence garantie}$$

1.7.1 Erreur

$$e^{(k)} = \underbrace{(I - NA)}_{\text{matrice d'itération}} \quad e^{(k-1)}$$

$$\left|\left| \vec{e}^{(k)} \right|\right| \leq \rho^k \left|\left| \vec{e}^{(0)} \right|\right|$$

Relation entre le nombre de décimales souhaitées D et le nombre d'itérations n

$$n \geq \frac{\log_{10}(10^{-D})}{\log_{10}(\rho)} = -\frac{D}{\log_{10}(\rho)}$$

1.8 Matrice strictement diagonalement dominante

$$\sum_{\forall j \neq i} |a_{ij}| < |a_{ii}|$$

a_{ii} est la plus grande valeur sur chaque ligne

1.9 Méthodes de minimisation $A\vec{x} - \vec{b}$

$$\boxed{\vec{r} = \vec{b} - Ax}$$

Gradient

$$\nabla G(\vec{x}) = \begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix}$$

Si on multiplie par la transposée de $\vec{n}$ on obtient la dérivée directionnelle.

$$\frac{\partial G}{\partial \vec{n}} = \vec{n}^T \nabla G(\vec{x})$$

Il existe deux méthodes de résolutions et deux algorithmes (F fonctionne toujours et G fonctionne si A est symétrique et semi-définie positive, voir définition plus bas)

1.9.1 Méthode 1 : $F(x) = \frac{1}{2}||b - A\vec{x}||_2^2$

On cherche à minimiser le résidu $\vec{r}$ (fonctionne seulement si $A\vec{x} = \vec{b}$ possède au moins une solution)

$$\vec{d} = A^T(\vec{b} - A\vec{x})$$

Algorithme de la plus grande pente

$$\vec{d_k} = -\nabla F = A^T(\vec{b} - A\vec{x_k})$$
$$\vec{x}_{k+1} = \vec{x}_k + \vec{d_k}$$

Algorithme des gradients conjuguées (pas sur que cet algorithme peut être utilisé avec cette méthode) On utilise le résidu précédent et le résidu actuel pour optimiser encore plus l'itération

$$\vec{d}_{k+1} = \vec{r}_{k+1} + \beta_k \vec{d_k}$$
$$\beta = \frac{\vec{r}_{k+1}^T \vec{r}_{k+1}}{\vec{r}_k^T \vec{r}_k}$$
$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d_k}$$

1.9.2 Méthode 2 : $G(x) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}$

Deux notations possibles :

$$G(x) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b} = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{b}^T\vec{x}$$

A doit être semi-définie positive (valeurs propres de A plus grandes ou égales à 0)

$$\vec{x}^T A\vec{x} \geq 0$$
$$\vec{d} = \vec{r} = \vec{b} - A\vec{x}$$

Algorithme de la plus grande pente (zig-zag)

$$\vec{d_k} = -\nabla F = A^T(\vec{b} - A\vec{x}_k)$$

$$\vec{x}_{k+1} = \vec{x}_k + \vec{d_k}$$

Si on utilise α optimal, alors toutes les itérations sont à angle droit. Calcul de α optimal :

$$\alpha_k = \frac{\vec{d_k}^T \vec{r}_k}{\vec{d_k}^T A \vec{d_k}}$$

Algorithme des gradients conjuguées (réponse après n itérations, avec n la largeur de la matrice A) On utilise le résidu précédent et le résidu actuel pour optimiser encore plus l'itération

$$\vec{d}_{k+1} = \vec{r}_{k+1} + \beta_k \vec{d_k}$$

$$\beta = \frac{\vec{r}_{k+1}^T \vec{r}_{k+1}}{\vec{r}_k^T \vec{r}_k}$$

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d_k}$$