Analyse Numérique SV

Systèmes Linéaires

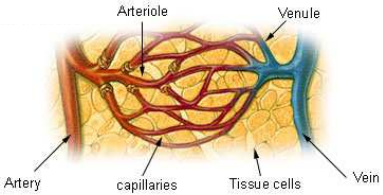
Simone Deparis

EPFL Lausanne – MATH

Printemps 2021

Exemple : Réseau de capillaires I

Les capillaires sont des vaisseaux sanguins microscopiques, la plus petite unité du système circulatoire. Les capillaires se regroupent en réseaux appelés lits capillaires comportant de 10 à 100 capillaires selon l'organe ou le tissu considéré.
Le sang arrive dans le lit capillaire par de petites artérioles. Dans le lit capillaire un échange de sang oxygéné et une élimination des déchets ont lieu.
Ensuite, des veinules recueillent le sang des capillaires, puis le transmettent à des veines qui le retournent au cœur.

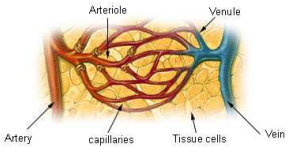


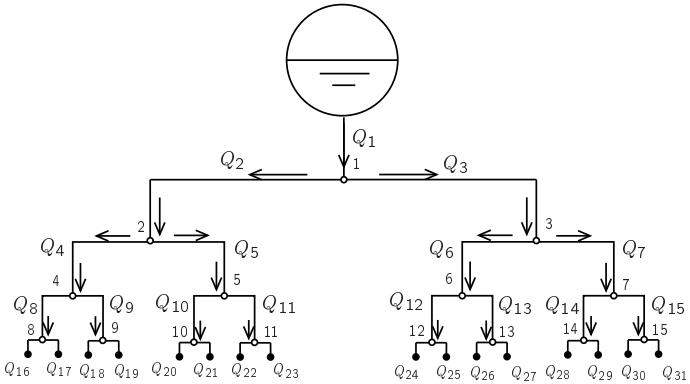
Exemple : Réseau de capillaires II

Nous considérons un modèle de lit capillaire.

- Nous pouvons modéliser une certaine portion du système capillaire par un réseau hydraulique où chaque capillaire est représenté par un tuyau.
- On appelle *nœuds* (petits cercles vides dans la figure de la page suivante) les points de rencontre de plusieurs capillaires.
- L'artère à partir de laquelle le réseau capillaire se développe est représentée comme un réservoir à une pression constante de 50mmHg.
- Nous supposons que les sorties du réseau (petits cercles noirs dans la figure) ont une pression constante (pression veineuse), fixée à la valeur de référence zéro (toutes les valeurs de pression se réfèrent donc à cette valeur).
- Le sang coule du réservoir aux sorties (on suppose de façon continue en temps) par effet du gradient de pression.

Exemple : Réseau de capillaires III





On veut trouver la distribution des pressions p_j, $j = 1, \cdots, 15$, et des débits Q_m, $m = 1, \cdots, 31$, dans le réseau capillaire. Pour ce faire, on considère que :

Exemple : Réseau de capillaires IV

- dans chaque branche m du réseau, $m = 1, \cdots, 31$, on a la relation suivante, dite loi constitutive, entre le débit de sang Q_m (mm^3/s) et la pression (mesurée en mmHg) aux deux nœuds d'extrémité i et j de la branche

$$Q_m = \frac{1}{R_m L_m}(p_i - p_j), \tag{1}$$

où R_m est la résistance hydraulique par unité de longueur ($\mathrm{mmHg\,s/mm}^4$) et L_m est la longueur (mm) de la m-ème branche ;

- dans chaque nœud j du réseau, $j = 1, \cdots 15$, on peut écrire le bilan des débits entrants et sortants. On a

$$\left(\sum_{m \text{ entrants}} Q_m \right) - \left(\sum_{m \text{ sortants}} Q_m \right) = 0,$$

où on a donné un signe négatif aux débits sortants du nœud.

Exemple : Réseau de capillaires V

Par exemple, pour le nœud 2 de la figure, le débit Q_2 est entrant et les débits Q_4 et Q_5 sont sortants. Le bilan doit être

$$Q_2 - Q_4 - Q_5 = 0.$$

En utilisant pour chaque débit la loi de constitution (1) dans le bilan ci-dessus, on peut écrire

$$\frac{1}{R_2 L_2}(p_1 - p_2) - \frac{1}{R_4 L_4}(p_2 - p_4) - \frac{1}{R_5 L_5}(p_2 - p_5) = 0.$$

Des relations analogues sont obtenues pour tous les autres nœuds du réseau.
A noter : si on considère le bilan pour le nœud 1, on a

$$\frac{1}{R_1 L_1}(p_r - p_1) - \frac{1}{R_2 L_2}(p_1 - p_2) - \frac{1}{R_3 L_3}(p_1 - p_3) = 0.$$

Exemple : Réseau de capillaires VI

Comme la pression p_r (pression du) est connue, on la mettra dans le membre de droite

$$\frac{1}{R_1 L_1} p_1 - \frac{1}{R_2 L_2}(p_1 - p_2) - \frac{1}{R_3 L_3}(p_1 - p_3) = -\frac{1}{R_1 L_1} p_r.$$

De façon similaire pour le nœud 8

$$\frac{1}{R_8 L_8}(p_4 - p_8) - \frac{1}{R_{16} L_{16}} p_8 - \frac{1}{R_{17} L_{17}} p_8 = 0,$$

où on observe que $p_{16} = p_{17} = 0$. On procède similairement pour les nœuds $9, \cdots, 15$.

En écrivant le bilan pour chaque nœud après y avoir remplacé la loi de constitution, on peut construire un système d'équations linéaires pour les pressions, du type

$$A\mathrm{p} = \mathrm{b}, \tag{2}$$

Exemple : Réseau de capillaires VII

où $\mathrm{p} = [p_1, p_2, \cdots, p_{15}]^T$ est le vecteur des pressions inconnues aux nœuds du réseau, $A \in \mathbb{R}^{15\times 15}$ est la matrice des coefficients du système et $\mathrm{b} \in \mathbb{R}^{15}$ est le vecteur des données.
Si on suppose que les branches du réseau ont toutes le même coefficient $R_m = R = 1$ et qu'à chaque niveau, la longueur devient la moitié de celle d'avant (si on pose $L_1 = 20$, on aura $L_2 = L_3 = 10$, $L_4 = L_5 = L_6 = L_7 = 5$ etc..), on trouve

$$\mathrm{b} = [-5/2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

et la matrice A suivante :

Exemple : Réseau de capillaires VIII

$$\begin{pmatrix}
-\frac{1}{4} & \frac{1}{10} & \frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{10} & -\frac{1}{2} & 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{10} & 0 & -\frac{1}{2} & 0 & 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{5} & 0 & -1 & 0 & 0 & 0 & 0.4 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{5} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0.4 & 0.4 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{5} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0.4 & 0 & 0 \\
0 & 0 & \frac{1}{5} & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0.4 \\
0 & 0 & 0 & 0.4 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2
\end{pmatrix}$$

Exemple : Réseau de capillaires IX

Déterminer les pressions et les débits dans le réseau revient donc à résoudre le système linéaire $Ap = b$.
Nous observons que, dans notre cas, la matrice A est symétrique et définie positive. Cette dernière propriété nous assure que la matrice A est non singulière et donc que le système admet une solution unique.
La solution est donnée par le vecteur

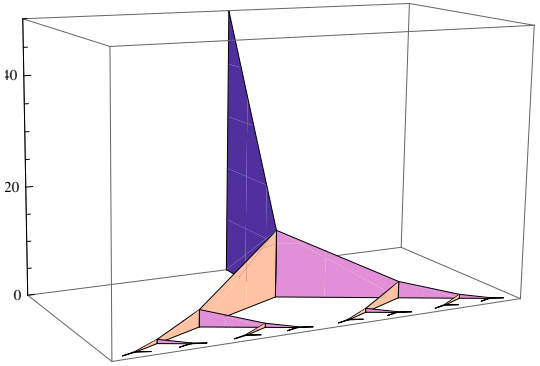
$$p = [12.46, 3.07, 3.07, 0.73, 0.73, 0.73, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15, 0.15]^T$$

Une fois les pressions trouvées, on peut calculer, grâce à la relation (1), les flux :

$$\begin{aligned} Q_1 &= 1.88 \\ Q_{2,3} &= 0.94 \\ Q_{4,\cdots,7} &= 0.47 \\ Q_{8,\cdots,15} &= 0.23 \\ Q_{16,\cdots,31} &= 0.12 \end{aligned}$$

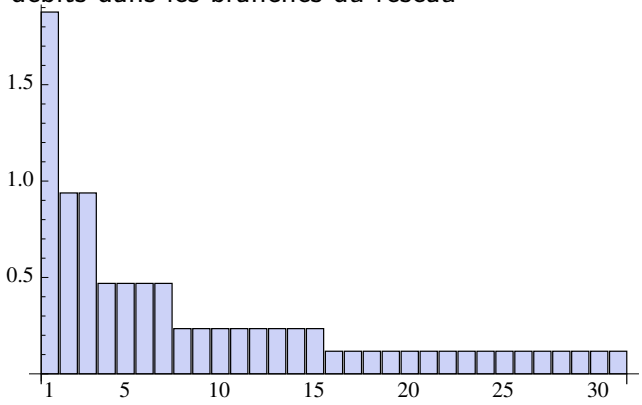
Exemple : Réseau de capillaires X

Distribution linéaire des pressions dans les branches du réseau calculée à partir des valeurs de la pression dans chaque nœud (solution du système)



Exemple : Réseau de capillaires XI

Distribution des débits dans les branches du réseau



Systèmes linéaires – méthodes directes

Considérons un système linéaire de n équations et n inconnues

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{cases} \tag{3}$$

Sous forme matricielle :

$$\mathrm{A}\mathrm{x} = \mathrm{b}$$

où

$$\mathrm{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n, \qquad \mathrm{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^n, \qquad A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

Système triangulaire inférieur

Il y a des systèmes qui sont particulièrement simples à résoudre. Un **système triangulaire inférieur** en est un exemple

$$\begin{cases} a_{11}x_1 & & & & = b_1 \\ a_{21}x_1 & + a_{22}x_2 & & & = b_2 \\ a_{31}x_1 & + a_{32}x_2 & + a_{33}x_3 & & = b_3 \\ \dots & & & & \\ a_{n1}x_1 & + a_{n2}x_2 & + \dots & + a_{nn}x_n & = b_n \end{cases} \tag{4}$$

avec $a_{ii} \neq 0$, $\forall i = 1, \dots, n$.

$$x_1 = \frac{b_1}{a_{11}}, \qquad x_2 = \frac{1}{a_{22}}\left(b_2 - a_{21}x_1\right), \qquad \cdots \qquad x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j\right)$$

Système triangulaire inférieur

Algorithme 1 : Algorithme de substitution directe (forward substitution)

pour $i = 1, \ldots, n$ **faire**

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

fin

Nombre d'opérations à faire :

- pour calculer x_i à l'i-ème pas, il faut faire $i - 1$ multiplications ; $i - 1$ additions/soustractions ; 1 division.
- Nombre total d'opérations :

$$N = \sum_{i=1}^{n} (2i - 1) = 2\frac{n(n+1)}{2} - n = n^2$$

L'algorithme de substitution directe coûte $O(n^2)$ opérations.

Système triangulaire supérieur

$$\begin{cases} a_{11}x_1 + & a_{12}x_2 + & \dots + & a_{1,n-1}x_{n-1} + & a_{1n}x_n = & b_1 \\ & a_{22}x_2 + & \dots + & a_{2,n-1}x_{n-1} + & a_{2n}x_n = & b_2 \\ & & & & \dots & \\ & & & a_{n-1,n-1}x_{n-1} + & a_{n-1,n}x_n = & b_{n-1} \\ & & & & a_{nn}x_n = & b_n \end{cases} \tag{5}$$

avec $a_{ii} \neq 0$, $\forall i = 1, \dots, n$

Algorithme 2 : Algorithme de substitution rétrograde (backward subst.)

pour $i = n, \dots, 1$ **faire**

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

fin

L'algorithme de substitution rétrograde coûte aussi $O(n^2)$ opérations.

Algorithme d'élimination de Gauss

Idée : transformer le système dans un système triangulaire supérieur à l'aide d'opérations élémentaires de combinaison linéaire de lignes de la matrice.

Résoudre le système linéaire

$$\begin{cases} 2x_1 & + x_2 & & = 4 \\ -4x_1 & + 3x_2 & - x_3 & = 2 \\ 4x_1 & - 3x_2 & + 4x_3 & = -2 \end{cases}$$

On fait des combinaisons linéaires de lignes pour

- éliminer d'abord l'inconnue x_1 de la 2^{eme} et 3^{eme} équations
- ensuite éliminer x_2 de la 3^{eme} équation.

Algorithme d'élimination de Gauss et factorisation LU

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	2	1	0	4
	-4	3	-1	2
	4	-3	4	-2

Algorithme d'élimination de Gauss et factorisation LU

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	2	1	0	4
$r_2 - \left(\frac{-4}{2}\right) r_1$	-4	3	-1	2
	4	-3	4	-2

L		
1	0	0
-2	1	0
		1

Algorithme d'élimination de Gauss et factorisation LU

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	2	1	0	4
$r_2 - \left(\frac{-4}{2}\right) r_1$	~~-4~~ 0	~~3~~ 5	-1	~~2~~ 10
	4	-3	4	-2

L		
1	0	0
-2	1	0
		1

Algorithme d'élimination de Gauss et factorisation LU

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	2	1	0	4
	0	5	-1	10
$r_3 - \left(\frac{4}{2}\right) r_1$	4	-3	4	-2

L		
1	0	0
-2	1	0
2		1

Algorithme d'élimination de Gauss et factorisation LU

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	2	1	0	4
	0	5	-1	10
$r_3 - \left(\frac{4}{2}\right) r_1$	~~4~~ 0	~~3~~ -5	4	~~2~~ -10

L		
1	0	0
-2	1	0
2		1

Algorithme d'élimination de Gauss et factorisation LU

	$A^{(2)}$			$b^{(2)}$
	2	1	0	4
	0	5	-1	10
	0	-5	4	-10

L		
1	0	0
-2	1	0
2		1

Algorithme d'élimination de Gauss et factorisation LU

	$A^{(2)}$			$\mathsf{b}^{(2)}$
	2	1	0	4
	0	5	-1	10
$r_3 - \left(\frac{-5}{5}\right) r_2$	0	-5	4	-10

L		
1	0	0
-2	1	0
2	-1	1

Algorithme d'élimination de Gauss et factorisation LU

	$A^{(2)}$			$b^{(2)}$
	2	1	0	4
	0	5	-1	10
$r_3 - \left(\frac{-5}{5}\right) r_2$	0	~~-5~~ 0	~~4~~ 3	~~-10~~ 0

L		
1	0	0
-2	1	0
2	-1	1

Algorithme d'élimination de Gauss et factorisation LU

	$A^{(3)} = U$			$\mathrm{b}^{(3)} = \mathrm{y}$
	2	1	0	4
	0	5	-1	10
	0	0	3	0

L		
1	0	0
-2	1	0
2	-1	1

Remarque : $LU = A$! !

Factorisation LU

Algorithme 3 : Algorithme d'élimination de Gauss (et factorisation LU)

Données : $A = \{a_{ij}\} \in \mathbb{R}^{n\times n}$, $\mathrm{b} = \{b_i\} \in \mathbb{R}^n$
Résultat : $U, L \in \mathbb{R}^{n\times n}$, $\mathrm{b}^{(n)} \in \mathbb{R}^n$
$A^{(1)} = A$;
pour $k = 1, \ldots, n-1$ **faire** // étapes de l'algorithme
  $l_{kk} = 1$;
  pour $i = k+1, \ldots, n$ **faire** // boucle sur les lignes
    $l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$;
    pour $j = k+1, \ldots, n$ **faire** // boucle sur les colonnes
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$;
    fin
    $b_i^{(k+1)} = b_i^{(k)} - l_{ik} b_k^{(k)}$;
  fin
fin
$U = A^{(n)}$, $L = \{l_{ij}\}$;

Nombre d'opérations $O(n^3)$ (trois boucles emboîtées)

Algorithme d'élimination de Gauss et factorisation LU

Une fois calculée la factorisation LU de A (coût $O(n^3)$), on peut résoudre le système linéaire $Ax = b$ par les étapes

1. $Ly = b$
2. $Ux = y$

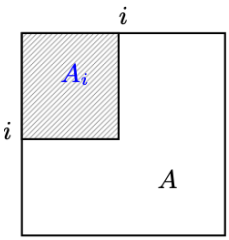
dont le coût est $O(n^2)$. (en fait $b = Ly = LUx = Ax$, donc x est bien la solution recherchée).

Remarque : si on doit résoudre deux systèmes linéaires $Ax_1 = b_1$ et $Ax_2 = b_2$ avec la même matrice, on calcule la factorisation LU une seule fois.

Condition nécessaire et suffisante

Des conditions restrictives sur A sont nécessaires pour assurer que la méthode d'élimination de Gauss (et la factorisation LU) puissent se réaliser sans permutations :

Soit d_i le déterminant de la i-ième sous-matrice principale A_i :



Si les mineurs principaux d_i de A sont non nuls pour $i = 1, \ldots, n-1$ alors les pivots correspondants $a_{ii}^{(i)}$ sont également non nuls.

Celle-ci est une **condition nécessaire et suffisante** pour que la méthode de Gauss puisse être appliquée sans permutations.

Conditions suffisantes

Voici des critères **suffisants** pour que la méthode de Gauss puisse être appliquée sans permutations.

1. Les matrices *à diagonale dominante par ligne*. Une matrice A est dite à diagonale dominante par ligne si
$$|a_{ii}| \geq \sum_{j=1,\dots,n; j\neq i} |a_{ij}|, \quad i = 1, \dots, n.$$
2. Les matrices *à diagonale dominante par colonne*. Une matrice A est dite à diagonale dominante par colonne si
$$|a_{jj}| \geq \sum_{i=1,\dots,n; i\neq j} |a_{ij}|, \quad j = 1, \dots, n.$$
3. Les matrices *symétriques définies positives*. Une matrice A est symétrique si $A = A^T$ $(a_{ij} = a_{ji} \quad \forall i, j)$; elle est définie positive si toutes ses valeurs propres sont positives, c'est-à-dire : $\lambda_i(A) > 0, \quad i = 1, \dots, n$.

Algorithme d'élimination de Gauss avec pivoting

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	1	2	3	4
	2	4	5	2
	7	8	9	20

Algorithme d'élimination de Gauss avec pivoting

	$A = A^{(1)}$			$\mathrm{b} = \mathrm{b}^{(1)}$
	1	2	3	4
$r_2 - (2)r_1$	2	4	5	2
	7	8	9	20

L		
1	0	0
2	1	0
		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	1	2	3	4
$r_2 - (2)r_1$	~~2~~ 0	~~4~~ 0	~~5~~ −1	~~2~~ −6
	7	8	9	20

L		
1	0	0
2	1	0
		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	1	2	3	4
	0	0	-1	-6
$r_3 - (7)r_1$	7	8	9	20

L		
1	0	0
2	1	0
7		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A = A^{(1)}$			$\mathsf{b} = \mathsf{b}^{(1)}$
	1	2	3	4
	0	0	-1	-6
$r_3 - (7)r_1$	~~7~~ 0	~~8~~ -6	~~9~~ -12	~~20~~ -8

L		
1	0	0
2	1	0
7		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A^{(2)}$			$b^{(2)}$
	1	2	3	4
	0	0	−1	−6
	0	−6	−12	−8

L		
1	0	0
2	1	0
7		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A^{(2)}$			$b^{(2)}$
	1	2	3	4
$r_2 \leftarrow r_3$	0	0	-1	-6
$r_3 \leftarrow r_2$	0	-6	-12	-8

L		
1	0	0
2	1	0
7		1

Q		
1	0	0
0	1	0
0	0	1

Algorithme d'élimination de Gauss avec pivoting

	$A^{(2)}$			$b^{(2)}$
	1	2	3	4
$r_2 \leftarrow r_3$	0	-6	-12	-8
$r_3 \leftarrow r_2$	0	0	-1	-6

L		
1	0	0
7	1	0
2		1

Q		
1	0	0
0	0	1
0	1	0

Algorithme d'élimination de Gauss avec pivoting

	$A^{(3)} = U$			$\mathsf{b}^{(3)} = \mathsf{y}$
	1	2	3	4
	0	-6	-12	-8
	0	0	-1	-6

L		
1	0	0
7	1	0
2	0	1

Q		
1	0	0
0	0	1
0	1	0

Algorithme d'élimination de Gauss avec pivoting

	$A^{(3)} = U$			$b^{(3)} = y$
	1	2	3	4
	0	-6	-12	-8
	0	0	-1	-6

L		
1	0	0
7	1	0
2	0	1

Q		
1	0	0
0	0	1
0	1	0

Remarque : $LU = QA$! !

Factorisation LU avec pivoting

$$LU = QA$$

Q est une matrice orthogonale car les colonnes sont orthonormées. Donc $Q^{-1} = Q^T$. Du coup, pour $P = Q^T$, on a

$$A = PLU.$$

La matrice Q ou P sert à mémoriser les permutations effectuées.

Dans la pratique il est bien d'effectuer le pivoting même si l'élément diagonal n'est pas nul. Le mieux c'est de choisir comme pivot le coefficient le plus grand de la colonne.

Factorisation LU avec pivoting

Algorithme 4 : Algorithme d'élimination de Gauss avec pivoting

Données : $A = \{a_{ij}\} \in \mathbb{R}^{n\times n}$, $\mathrm{b} = \{b_i\} \in \mathbb{R}^n$
Résultat : $U, L, P \in \mathbb{R}^{n\times n}$, $\mathrm{b}^{(n)} \in \mathbb{R}^n$
$A^{(1)} = A$, Q matrice identité;
pour $k = 1, \ldots, n-1$ **faire**
  trouver r tel que $|a_{rk}^{(k)}| = \max_{i=k,\ldots,n} |a_{ik}^{(k)}|$;
  échanger la ligne k avec la ligne r dans les matrices $A^{(k)}$, L et Q, ainsi que dans le vecteur $\mathrm{b}^{(k)}$;
  pour $i = k+1, \ldots, n$ **faire**
    $l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$;
    pour $j = k+1, \ldots, n$ **faire**
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$;
    fin
    $b_i^{(k+1)} = b_i^{(k)} - l_{ik} b_k^{(k)}$;
  fin
  $l_{kk} = 1$;
fin
$U = A^{(n)}$, $L = \{l_{ij}\}$, $P = Q^T$;

Factorisation LU avec pivoting

Théorème

Toute matrice $A \in \mathbb{R}^{n\times n}$ non singulière admet une factorisation $A = PLU$ où L est une matrice triangulaire inférieure, U une matrice triangulaire supérieure et P une matrice de permutation.

En Python, l'algorithme de factorisation LU est disponible avec la commande `P,L,U=scipy.linalg.lu(A)` et on obtient $A = PLU$ ou $P^TA = LU$

Une fois calculée la factorisation LU de A avec pivoting (coût $O(n^3)$), on peut résoudre le système linéaire $Ax = b$ ($\Leftrightarrow P^TAx = P^Tb$) par les étapes

1. $Ly = P^Tb$
2. $Ux = y$

dont le coût est $O(n^2)$.

D'autres factorisations

- Factorisation de Choleski
- Factorisation *QR*
- Factorisation *SVD*
- Diagonalisation

La factorisation de Cholesky

Dans le cas où A est une matrice $n \times n$ symétrique et définie positive, il existe une unique matrice triangulaire inférieure H avec les éléments diagonaux positifs telle que

$$\boxed{A = HH^T}.$$

Cette factorisation s'appelle la *factorisation de Cholesky*. Dans `scipy.linalg`, on peut utiliser la commande

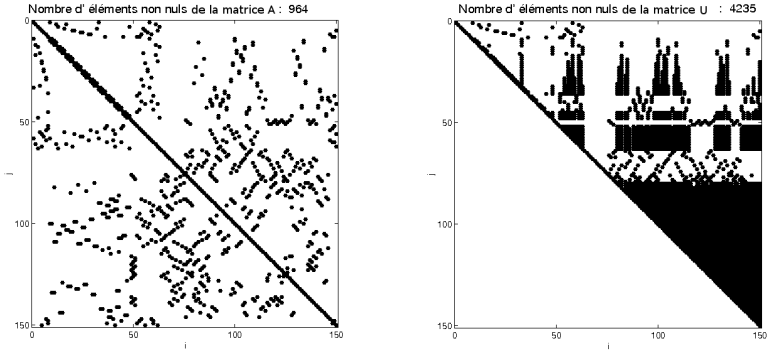
```
# lower : returns lower-triangular matrix, A = H H^T
H = cholesky(A, lower=True)
```

Critère de Sylvester : une matrice symétrique $A \in \mathbb{R}^{n \times n}$ est définie positive si et seulement si les mineurs principaux de A sont tous positifs.

Exemple

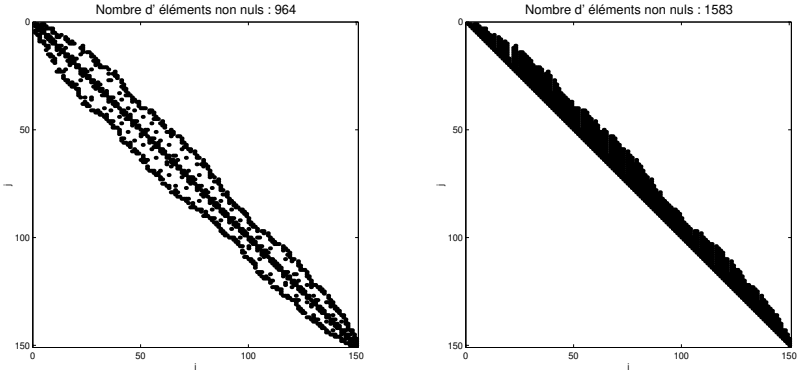
On considère le calcul des déformations d'une structure soumise à des forces données. La discrétisation par la méthode des éléments finis donne une matrice K de taille 150×150. (La même matrice dérive également de l'approximation du potentiel d'un champ électrique.) Cette matrice est symétrique définie positive. Le nombre d'éléments non nuls de K est égal à 964 et donc beaucoup plus petit que $(150)^2 = 22500$. Il s'agit, dans ce cas, d'une matrice *creuse*.

La figure de gauche montre la disposition des éléments non nuls de K, tandis que la figure de droite montre celle de la matrice H^T.



On remarque que le nombre d'éléments non nuls de H^T est beaucoup plus grand que celui de K (*phénomène du fill-in*). Ceci entraîne une occupation mémoire importante.
Pour réduire le phénomène de remplissage, on peut réordonner de façon intelligente les lignes et les colonnes de la matrice K ; on appelle cette procédure *réordonnancement* de la matrice. Il y a plusieurs algorithmes qui permettent de faire cela.

Par exemple la figure ci-dessous à gauche, montre un réordonnancement possible de la matrice K, tandis qu'à droite, on a affiché la disposition des éléments non nuls de la factorisation de Cholesky de la matrice K réordonnée.



Problèmes de précision : la matrice d'Hilbert I

Les erreurs d'arrondis peuvent causer des différences importantes entre la solution calculée par la méthode d'élimination de Gauss (MEG) et la solution exacte. Cela arrive si le *conditionnement* de la matrice du système est très grand.

La matrice de Hilbert de taille $n \times n$ est une matrice symétrique, définie par

$$A_{ij} = \frac{1}{i+j-1}, \qquad i,j = 1,\ldots,n$$

Dans `scipy.linalg`, on peut construire une matrice de Hilbert de taille n quelconque en utilisant la commande `A = hilbert(n)`. Par exemple, pour $n = 4$, on a :

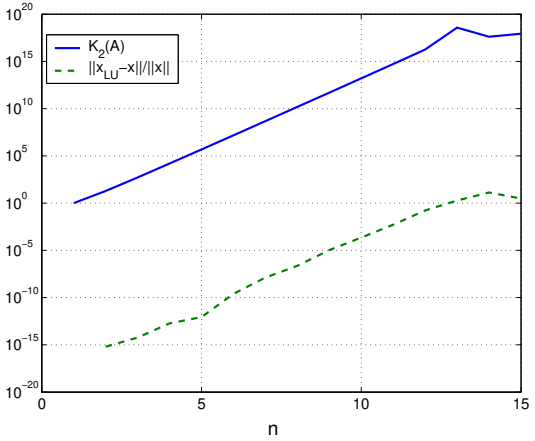
$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

Problèmes de précision : la matrice d'Hilbert II

On considère les systèmes linéaires $A_n x_n = b_n$ ou A_n est la matrice de Hilbert de taille n avec $n = 4, 6, 8, 10, 12, 14, \ldots$ tandis que b_n est choisi de sorte que la solution exacte soit $x_n = (1, 1, \cdots, 1)^T$.

Pour chaque n, on calcule le conditionnement de la matrice, on résout le système linéaire par la factorisation LU et on note x_n^{LU} la solution calculée. Le conditionnement obtenu ainsi que l'erreur $\|x_n - x_n^{LU}\|/\|x_n\|$ (où $\|\cdot\|$ est la norme euclidienne d'un vecteur, $\|x\| = \sqrt{x^T \cdot x}$) sont montrés ci-dessous.

Problèmes de précision : la matrice d'Hilbert III



Considérations sur la précision

Les méthodes qu'on a vu jusqu'à maintenant sont des méthodes qui permettent de trouver la solution d'un système linéaire en un nombre fini d'opérations. C'est pourquoi on les appelle *méthodes directes*. Toutefois, il y a des cas où ces méthodes ne fonctionnent pas de manière satisfaisante.

Définition

On définit le **conditionnement** d'une matrice M symétrique définie positive comme le rapport entre la valeur maximale et minimale de ses valeurs propres, i.e.

$$K(M) = \frac{\lambda_{max}(M)}{\lambda_{min}(M)}$$

On peut montrer que plus le conditionnement de la matrice est grand, plus la solution du système linéaire obtenue par une méthode directe peut être mauvaise. Par exemple, considérons un système linéaire $A\mathrm{x} = \mathrm{b}$.
Si on résout ce système avec un ordinateur, à cause des erreurs d'arrondis, on ne trouve pas la solution exacte mais une solution approchée $\hat{\mathrm{x}}$. On peut montrer la relation suivante :

$$\frac{\|\mathrm{x} - \hat{\mathrm{x}}\|}{\|\mathrm{x}\|} \leq K(A) \frac{\|\mathrm{r}\|}{\|\mathrm{b}\|} \tag{6}$$

où r est le résidu $\mathrm{r} = \mathrm{b} - A\hat{\mathrm{x}}$; on désigne par $\|\mathrm{v}\| = \left(\sum_{k=1}^{n} v_k^2\right)^{1/2}$ la norme euclidienne d'un vecteur v.
On remarque que, si le conditionnement de A est grand, la distance $\|\mathrm{x} - \hat{\mathrm{x}}\|$ entre la solution exacte et celle calculée numériquement peut être très grande même si le résidu est très petit.

Preuve de (6) : A étant symétrique définie positive, on peut considérer les n valeurs propres $\lambda_i > 0$ et les vecteurs propres unitaires associés $\{\mathrm{v}_i\}$, $i = 1, \ldots, n$. Ces derniers forment une base orthonormé de $\mathbb{R}^n$, c'est-à-dire $\mathrm{v}_i^T \mathrm{v}_j = \delta_{ij}$ pour $i, j = 1, \ldots, n$. Soit $\mathrm{w} \in \mathbb{R}^n$ quelconque. Si on l'exprime comme

$$\mathrm{w} = \sum_{i=1}^{n} w_i \mathrm{v}_i,$$

on a

$$\begin{aligned} \|A\mathrm{w}\|^2 &= (A\mathrm{w})^T (A\mathrm{w}) \\ &= (\lambda_1 w_1 \mathrm{v}_1^T + \ldots \lambda_n w_n \mathrm{v}_n^T)(\lambda_1 w_1 \mathrm{v}_1 + \ldots \lambda_n w_n \mathrm{v}_n) \\ &= \sum_{i,j=1}^{n} \lambda_i \lambda_j w_i w_j \mathrm{v}_i^T \mathrm{v}_j = \sum_{i,j=1}^{n} \lambda_i \lambda_j w_i w_j \delta_{ij} = \sum_{i=1}^{n} \lambda_i^2 w_i^2. \end{aligned}$$

Or, comme $\|w\|^2 = \sum_{i=1}^n w_i^2$, on obtient $\|Aw\|^2 \leq \lambda_{max}^2 \|w\|^2$, c.à d. $\|Aw\| \leq \lambda_{max}\|w\|$ où λ_{max} est la plus grande valeur propre de A.

Comme les valeurs propres de A^{-1} sont $1/\lambda_i$, de la même façon on trouve $\|A^{-1}w\| \leq \frac{1}{\lambda_{min}}\|w\|$ $\forall w \in \mathbb{R}^n$, où λ_{min} est la plus petite valeur propre de A. Donc, on a

$$\begin{aligned} \|x - \hat{x}\| &= \|A^{-1}r\| \leq \frac{1}{\lambda_{min}}\|r\|, \\ \|b\| &= \|Ax\| \leq \lambda_{max}\|x\|, \end{aligned}$$

d'où on trouve directement l'inégalité (6). ■