# TD3 - Systèmes linéaires numériques

Alexandra Bac

Méthodes numériques
Polytech Marseille - IRM 3A

## 1 TD

**Algorithme 1** Elimination de Gauss (pivot "partiel") - pour la résolution de systèmes linéaires.

**Entrées:** Matrice $[A|B] \in \mathcal{M}_{n,n+m}(\mathbb{R})$ avec $A$ inversible.
**Sorties:** Matrice triangulaire supérieure $[U|B']$ codant le système linéaire $UX = B'$ dont les solutions sont identiques à $AX = B$.

**pour** $i$ de 1 à $n$ **faire**
  Calculer l'indice $j$ du maximum de la colonne $\{|a_{ij}| \, ; \, j = i+1 \dots n\}$
  Echanger les lignes $i$ et $j$ de $[A|B]$
  **pour** $j$ de $i+1$ à $n$ **faire**
    $\alpha_j \leftarrow -a_{ji}/a_{ii}$
    $\text{ligne}_j([A|B]) \leftarrow \text{ligne}_j([A|B]) + \alpha_j \cdot \text{ligne}_i([A|B])$
  **fin pour**
**fin pour**
$[U|B'] \leftarrow [A|B]$

**Algorithme 2** Remontée triangulaire.

**Entrées:** Matrice $[U|B] \in \mathcal{M}_{n,n+m}(\mathbb{R})$ avec $U$ triangulaire supérieure inversible.
**Sorties:** Matrice $B'$ solution de $UX = B$.

$\text{ligne}_n(B') \leftarrow \text{ligne}_n(B)/u_{nn}$
**pour** $i$ de $n-1$ à 1 **faire**
  $\text{ligne}_i(B') \leftarrow \text{ligne}_i(B) - \sum_{j=i+1}^{n} \frac{u_{ij}}{u_{ii}} \cdot \text{ligne}_j(B')$
**fin pour**

**Exercice 1** (Systèmes linéaires : Gauss, Cholesky, Householder, conditionnement)**.** Dans cet exercice, on s'intéresse au comportement de Gauss, Householder et Cholesky et au temps d'exécution sur différentes matrices construites pour ces tests.

(i) On considère le code suivant :

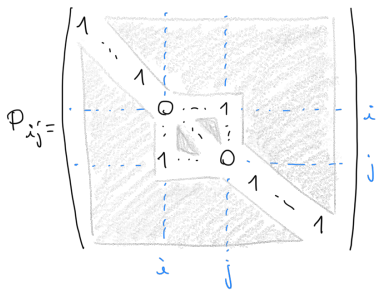
```
n = 20 ;
A = linspace(1e-3,1e5,n) ;
A = diag(A) ;
```

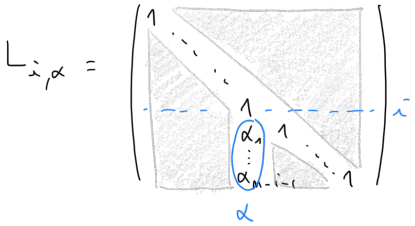
Quel est le conditionnement de la matrice $A$ ainsi générée ? Quelle est la forme de cette matrice ?

(ii) Puis, on considère le code suivant :

**Algorithme 3** Elimination de Gauss (factorisation PLU).

Soient $P_{ij}$ et $L_{i,\alpha}$ les matrices codant respectivement l'échange des lignes $i$ et $j$ et les combinaisons de pivot $l_i$ (avec $\alpha \in \mathbb{R}^n$ dont les $i$ premiers coefficients sont nuls). Ces matrices sont définies par :





On notera également $\sigma_{ij}$ la permutation échangeant les indices $i$ et $j$ d'un vecteur.

**Entrées:** Matrice $C = [A|B] \in \mathcal{M}_{n,n+m}(\mathbb{R})$ avec $A$ inversible.

**Sorties:** Matrice triangulaire supérieure $[U|B']$, matrice triangulaire inférieure $L$, matrice inversible $P$ (échange de lignes) telles que $L^{-1}P[A|B] = [U|B']$ (en particulier $PA = LU$).

1: Initialiser $\alpha$ une liste vide de vecteurs
2: $P \leftarrow \mathrm{Id}$
3: **pour** $i$ de 1 à $n$ **faire**
4: Calculer l'indice $j$ du maximum de la colonne $\{c_{ij}\,;\, j = i+1 \dots n\}$
5: Echanger les lignes $i$ et $j$ de $C$
6: $P \leftarrow P_{ij}P$
7: Appliquer $\sigma_{ij}$ à tous les vecteurs de $\alpha$
8: $tmp = (0, \dots, 0, -c_{i+1i}/c_{ii}, \dots, -c_{ni}/c_{ii})$
9: **pour** $j$ de $i+1$ à $n$ **faire**
10: $\mathrm{ligne}_j(C) \leftarrow \mathrm{ligne}_j(C) + tmp_j \cdot \mathrm{ligne}_i(C)$
11: **fin pour**
12: $\alpha \leftarrow [\alpha; tmp]$
13: **fin pour**
14: $[U|B'] \leftarrow C$
15: $L \leftarrow \prod_{i=1}^{n} L_{i,\alpha(i)}^{-1}$

```
M = rand(n,n) ;
[U,R]=qr(M) ;
```

Que pouvez-vous dire de la matrice $U$ ? (accessoirement de la matrice $R$).

(iii) On poursuit de la manière suivante :

```
B = U*A ;
```

Quel est le conditionnement de $B$ ? Vérifier avec Matlab. A quoi sert, par conséquent, l'ensemble du code précédent ?

(iv) Vous testerez, sur cette matrice, la décomposition LU ainsi que la décomposition QR. Quel est le conditionnement des matrices triangulaires supérieurs ainsi obtenues ?

(v) Comparez les temps de calcul respectifs de ces deux décompositions (fonctions `tic` et `toc`). Puis répétez ces tests pour $n$ croissants (50, 100, 500). Concluez

(vi) La matrice $B$ est-elle "elligible" à Cholesky ?

**Algorithme 4** Algorithme de Cholesky.

**Entrées:** Matrice $A \in \mathcal{M}_n(\mathbb{R})$ symétrique, définie, positive.
**Sorties:** Matrice $R \in \mathcal{M}_n(\mathbb{R})$ triangulaire inférieure telle que $A = RR^t$

Initialiser $R$ à 0
**pour** $i$ de 1 à $n$ **faire**
  $r_{ii} \leftarrow \sqrt{a_{ii} - \sum_{k=1}^{i-1}(r_{ik})^2}$
  **pour** $j$ de $i+1$ à $n$ **faire**
    $r_{ji} \leftarrow \frac{a_{ij} - \sum_{k=1}^{i-1} r_{ik} r_{jk}}{r_{ii}}$
  **fin pour**
**fin pour**

**Algorithme 5** Algorithme de Householder (décomposition QR).

Etant donné un vecteur $v$, la matrice de Householder $H(v)$ est donnée par

$$H(v) = \mathrm{Id} - 2\frac{v\,v^t}{v^t\,v}$$

Par ailleurs on notera $A|_i$ la sous-matrice de $A$ formée est lignes et colonnes de $i$ à $n$.

**Entrées:** Matrice $A \in \mathcal{M}_{n,m}(\mathbb{R})$ (dans la pratique, l'argument est en fait une matrice $C = [A|B]$).
**Sorties:** Matrices $Q$ orthogonale et $R$ triangulaire supérieure telles que $A = QR$.

Initialiser $Q \leftarrow \mathrm{Id}$.
$i \leftarrow 1$
**tant que** $A|_i \neq 0$ **faire**
  $a \leftarrow (a_{ii}, \ldots, a_{ni})^t$
  $v \leftarrow a + \|a\|(1, 0, \ldots, 0)^t$
  $Q_{\mathrm{tmp}} \leftarrow \left(\begin{array}{c|c} \mathrm{Id}_i & 0 \\ \hline 0 & H(v) \end{array}\right)$
  $Q \leftarrow Q\,Q_{\mathrm{tmp}}$
  $A|_i \leftarrow H(v)\,A|_i$
**fin tant que**
$Q \leftarrow Q^{-1}$
$R \leftarrow A$

(vii) Modifiez le code précédent pour générer une matrice appropriée et répétez les tests (conditionnements et temps d'exécution), cette fois sur LU, QR et Cholesky.

**Exercice 2** (Méthode des puissances itérées)**.** On considère la méthode des puissances itérées, qui permet de calculer rapidement la plus grande valeur propre d'une matrice $\boldsymbol{A}$ de dimensions $d \times d$ et un vecteur propre associé. Une version simplifiée est donnée par l'Algorithme 6. Elle permet de comprendre mathématiquement le fonctionnement de cette méthode. La version utilisée en pratique est donnée par l'Algorithme 7. On s'intéresse ici à la complexité et à la preuve de ces algorithmes. On supposera que $\boldsymbol{A}$ admet $d$ valeurs propres $|\lambda_1| > |\lambda_2| \geq \ldots > |\lambda_d|$ rangées dans l'ordre décroissant en valeur absolue et on note $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d$ les vecteurs propres associés. On suppose que $|\lambda_1|$ est strictement plus grande que les autres $|\lambda_i|$, ce qui n'est pas toujours le cas.

(i) Quelle est la complexité de l'Algorithme 6 en fonction du nombre d'itération $N$ et de la dimension $d$ ?

**Algorithme 6** Algorithme des puissances itérées (sans normalisation).

**Entrées:** Matrice $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, nombre d'itérations $N$
**Sorties:** Plus grande valeur propre $\lambda$ de $\boldsymbol{A}$ et vecteur propre $\boldsymbol{u}$ associé.

Initialiser $\boldsymbol{x}_0 \in \mathbb{R}^d$ aléatoirement
**pour** $n$ de 1 à $N$ **faire**
    $\boldsymbol{x}_n \leftarrow \boldsymbol{A}\boldsymbol{x}_{n-1}$
**fin pour**
$\boldsymbol{u} \leftarrow \frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2}$
$\lambda \leftarrow \frac{\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle}{\|\boldsymbol{u}\|_2^2}$
**renvoyer** $\lambda, \boldsymbol{u}$

**Algorithme 7** Algorithme des puissances itérées (avec normalisation).

**Entrées:** Matrice $\boldsymbol{A} \in \mathbb{R}^{d\times d}$, nombre d'itérations $N$
**Sorties:** Plus grande valeur propre $\lambda$ de $\boldsymbol{A}$ et vecteur propre $\boldsymbol{u}$ associé.

Initialiser $\boldsymbol{y}_0 \in \mathbb{R}^d$ aléatoirement
$\boldsymbol{x}_0 \leftarrow \frac{\boldsymbol{y}_0}{\|\boldsymbol{y}_0\|_2}$
**pour** $n$ de 1 à $N$ **faire**
    $\boldsymbol{y}_n \leftarrow \boldsymbol{A}\boldsymbol{x}_{n-1}$
    $\boldsymbol{x}_n \leftarrow \frac{\boldsymbol{y}_n}{\|\boldsymbol{y}_n\|_2}$
**fin pour**
$\boldsymbol{u} \leftarrow \boldsymbol{x}_N$
$\lambda \leftarrow \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle$
**renvoyer** $\lambda, \boldsymbol{u}$

(ii) En écrivant $\boldsymbol{x}_0$ sous la forme $\boldsymbol{x}_0 = \sum_{i=1}^n \alpha_i \boldsymbol{u}_i$, montrez par récurrence que pour tout $n \geq 0$, on a

$$\boldsymbol{x}_n = \sum_{i=1}^{d} \alpha_i \lambda_i^n \boldsymbol{u}_i$$

(iii) Montrez que $\frac{\boldsymbol{x}_n}{\lambda_1^n}$ tend vers $\alpha_1 \boldsymbol{u}_1$ quand $n$ tend vers $+\infty$.

(iv) Montrez qu'à convergence, on a $\frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2} = \boldsymbol{u}_1$ ou $\frac{\boldsymbol{x}_N}{\|\boldsymbol{x}_N\|_2} = -\boldsymbol{u}_1$ et $\frac{\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u}\rangle}{\|\boldsymbol{u}\|_2^2} = \lambda_1$.

(v) Pourquoi le nombre d'itérations $N$ nécessaire pour atteindre la convergence est-il faible ? Vous pouvez prendre par exemple $\lambda_2 = 0.5\lambda_1$ pour vous en persuader.

(vi) En considérant la limite de $\lambda_1^n$ quand $n$ tend vers $+\infty$, expliquez pourquoi en pratique, l'Algorithme 6 ne fonctionne pas bien (séparez les cas $|\lambda_1| < 1$ et $|\lambda_1| > 1$).

(vii) (optionnel) Pour prouver l'Algorithme 7, montrez par récurrence que pour tout $n \geq 0$, on a

$$\boldsymbol{x}_n = \frac{1}{\sqrt{\sum_{i=1}^d \alpha_i^2 \lambda_i^{2n}}} \sum_{i=1}^{d} \alpha_i \lambda_i^n \boldsymbol{u}_i$$

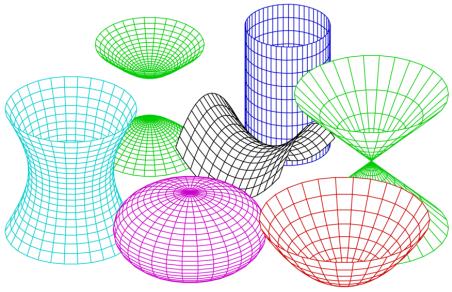
En factorisant $\lambda_1^n$ au numérateur et au dénominateur, montrez ensuite que la limite de $\boldsymbol{x}_n$ quand $n$ tend vers $+\infty$ est $\pm\boldsymbol{u}_1$.

## 2 TP

**Exercice 3** (Conditionnement interpolation de points par une quadrique)**.** Dans cet exercice on considère les quadriques ne passant pas par 0, surfaces de $\mathbb{R}^3$ dont l'équation implicite est un polynôme de degré 2 :

$$a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z = 1$$

on modélise ainsi la gamme de surfaces représentées ci-dessous.



L'équation ayant 9 paramètres, étant donnés 0 points, il existe une unique surface passant par ces 9 points (on dit qu'elle interpole les points).

Soient $\{X_1, \ldots, X_9\}$ ces 10 points de données de l'espace 3D avec $X_i = (x_i, y_i, z_i)$.

(i) Ecrire l'équation d'interpolation (donc l'équation que doivent vérifier les coefficients de la quadrique pour qu'elle passe par tous les points). En déduire que l'interpolation se réduit à la résolution d'un système linéaire.

(ii) Tirer aléatoirement 9 points de données (on générera une matrice aléatoire $M \in \mathcal{M}_{3,9}(\mathbb{R})$ donc les colonnes coderons les points).

(iii) Résoudre le système linéaire et construire la fonction :

$$f : (x, y, z) \mapsto a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z - 1$$

(iv) Représenter la surface en utilisant l'algorithme de "marching cubes" fourni en matériel de TP (on appellera la fonction `plotImplicit3D`) ainsi que les points de données.

(v) Perturber légèrement les points ($\varepsilon\%$). Comment la surface interpolante est-elle impactée ? Pourquoi cette instabilité ? Quel conditionnement permet de comprendre ce comportement ?

**Exercice 4** (Méthode des puissances itérées)**.**

(i) Implémentez l'Algorithme 7 via une fonction dont les entrées et les sorties coïncident avec celles de l'algorithme.

(ii) Testez-le sur une matrice dont vous connaissez la plus grande valeur propre et un vecteur propre associé.

Cet exercice permettra une application à la séparation d'objets mobiles et du fond fixe dans des vidéos à la séance 2.

**Exercice 5** (Valeurs propres d'une matrice)**.** Dans cet exercice, vous allez expérimenter (sans démonstration) une méthode d'approximation des valeurs propres d'une matrice.

(i) Pour cela, il faut tout d'abord pouvoir construire des matrices diagonalisables. On considère la boucle :

```
n = 10 ;
P = zeros(n,n) ;
while (det(P) ~= 0)
    P = rand(n,n) ;
end
```

quelle propriété est satisfaite par la matrice $P$ en fin de boucle ?

(ii) En déduire une construction d'une matrice aléatoire $A$ dont vous contrôlez les valeurs propres (donc diagonalisable).

(iii) On itère ensuite la boucle suivante :

```
for i=...
    [Q,R]=qr(A) ;
    A = R*Q ;
end
```

Itérer cette boucle et au fur et à mesure des itérations :

(a) Que devient la forme de la matrice $A$ (a-t-elle des parties nulles) ?

(b) Comparez sa diagonale (triée) aux valeurs propres de la matrice $A$ initiale (triées).

Vous avez ici l'embryon de l'algorithme de calcul des valeurs propres (suite dans Ciarlet - Introduction à l'analyse numérique matricielle et à l'optimisation).

**Exercice 6** (Régression linéaire)**.** On considère le jeu de données Diabetes fourni de façon similaire dans les fichiers `diabetes.mat` (Matlab/Octave) et `diabetes.npz` (Python) sur Ametice (données pour la régression). Il contient $N = 442$ exemples de patients atteint du diabète. Pour chaque exemple $i$, on dispose d'une part d'un vecteur $\boldsymbol{x}_i \in \mathbb{R}^d$ contenant $d = 11$ attributs relatifs au patient – âge, sexe, indice de masse corporelle, etc., voir variable `feature_names` – et d'autre part d'une mesure $y_i$ de la progression de la maladie au bout d'un an. L'objectif est d'apprendre la relation entre les attributs des patients et l'évolution de leur maladie. On suppose que cette relation est linéaire : on cherche un vecteur $\boldsymbol{w} \in \mathbb{R}^d$ tel que chaque $y_i = \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle = \sum_j \boldsymbol{w}[j]\, \boldsymbol{x}_i[j]$. Pour calculer $\boldsymbol{w}$, on propose de considérer les $d$ premiers exemples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{d}$ et de résoudre le système de $d$ équations à $d$ inconnues

$$\begin{cases} \boldsymbol{w}[1]\,\boldsymbol{x}_1[1] + \boldsymbol{w}[2]\,\boldsymbol{x}_1[2] + \ldots + \boldsymbol{w}[d]\,\boldsymbol{x}_1[d] &= y_1 \\ \boldsymbol{w}[1]\,\boldsymbol{x}_2[1] + \boldsymbol{w}[2]\,\boldsymbol{x}_2[2] + \ldots + \boldsymbol{w}[d]\,\boldsymbol{x}_2[d] &= y_2 \\ \vdots & \vdots \\ \boldsymbol{w}[1]\,\boldsymbol{x}_d[1] + \boldsymbol{w}[2]\,\boldsymbol{x}_d[2] + \ldots + \boldsymbol{w}[d]\,\boldsymbol{x}_d[d] &= y_d \end{cases}$$

(i) Écrivez le système d'équation sous la forme matricielle $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$ : que valent $\boldsymbol{X}$ et $\boldsymbol{y}$ et quelles sont leurs dimensions ?

(ii) Chargez les données, construisez $\boldsymbol{X}$ et $\boldsymbol{y}$ puis la solution $\boldsymbol{w}$ du système $\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$.

(iii) On souhaite évaluer dans quelle mesure les coefficients $\boldsymbol{w}$ estimés à partir de $d$ exemples permettent de prédire le diagnostic sur les $N - d$ autres exemples : calculez et commentez l'erreur

$$\sqrt{\frac{1}{N-d} \sum_{i=d+1}^{N} \left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle - y_i\right)^2}$$

(iv) recommencez la manipulation en calculant $\boldsymbol{w}$ à partir de $d$ autres exemples et l'erreur sur les exemples restants : retrouve-t-on le même vecteur $\boldsymbol{w}$ et la même erreur ? Qu'en pensez-vous ?

(v) calculez numériquement le conditionnement des matrices $\boldsymbol{X}$ que vous avez construites et commentez.

(vi) on souhaite exploiter plus de $d$ exemples pour apprendre $\boldsymbol{w}$ : en notant $n$ le nombre d'exemples ($n > d$), posez le système d'équations à résoudre et commentez sa résolution ; avez-vous des idées face au problème rencontré ?