Modélisation et résolutions numérique et symbolique de problèmes via les logiciels Maple et MATLAB (MODEL)

Cours n°8 : Calcul matriciel (2/2)

Stef Graillat

Université Pierre et Marie Curie (Paris 6)

Résumé du cours précédent

- Stockage des matrices
- Manipulation efficace des matrices : les BLAS
- Décomposition LU et résolution de systèmes linéaires

Objectifs

On considère des calculs utilisant des matrices denses (c'est à dire avec peu d'éléments nuls)

Nous avons besoin de

1. manipuler efficacement ces matrices sur un ordinateur
2. choisir la décomposition adéquate pour résoudre un problème donné

Plan du cours

1. Manipulation des matrices
   1. comment sont stockées les matrices en machine ?
   2. outils de base pour la manipulation de matrice : les BLAS
2. Décomposition de matrice et applications
   1. LU
   2. QR
   3. Réduction (diagonalisation, etc.)
   4. SVD (décomposition en valeurs singulières)
3. Logiciels

Décompositions matricielles et leurs applications

Nous allons étudier les 4 décompositions suivantes :

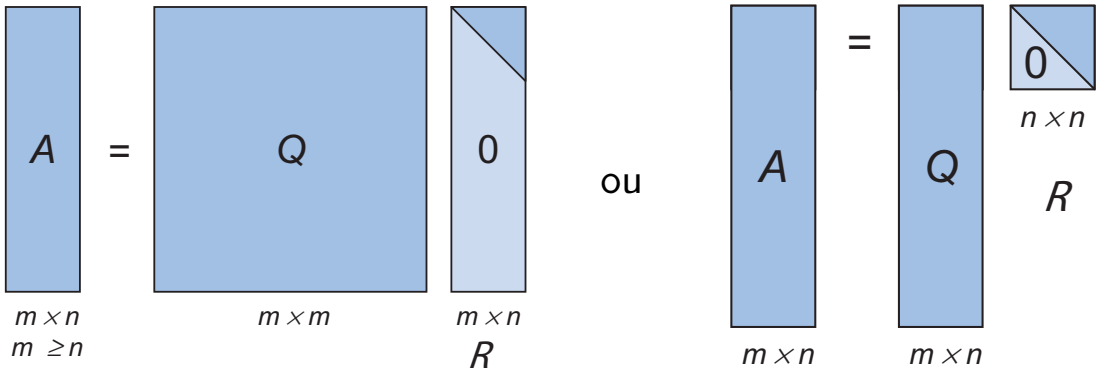
1. LU
2. QR
3. Réduction (diagonalisation, etc.)
4. SVD (décomposition en valeurs singulières)

La décomposition QR

Définition 1 (décomposition QR)

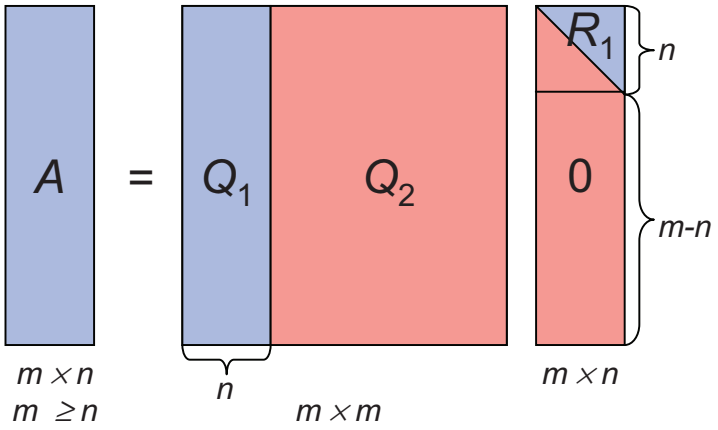
La décomposition QR du matrice A de de taille $m \times n$ avec $m \geq n$ est définie par $A = QR$ avec

- *Q est une matrice unitaire de taille $m \times m$ (orthogonale si A est réelle) et R est une matrice de taille $m \times n$ avec des zéros sous la diagonale*
- *Q est une matrice unitaire de taille $m \times n$ (orthogonale si A est réelle) et R est une matrice triangulaire supérieure de taille $n \times n$*



La décomposition QR (suite)

La version compacte (réduite) de la décomposition QR est possible car une partie de la matrice Q de taille $m \times m$ n'est pas nécessaire dans la décomposition



$$A = Q_1 R_1 + Q_2 0 = Q_1 R_1$$

Méthodes de calcul de la décomposition QR

Il y a principalement 3 méthodes différentes pour calculer une décomposition QR :

1. la méthode de Givens
2. la méthode de Gram-Schmidt
3. la méthode de Householder

Nous allons étudier les 2 premières en cours. La 3e sera vue en TD.

Matrices de Givens

On suppose ici que les matrices sont réelles (le cas complexe sera vu en TD)

On peut introduire un zéro dans une matrice en multipliant par une matrice orthogonale simple (une rotation)

Une matrice de Givens s'écrit sous la forme

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

avec $c^2 + s^2 = 1$ (on peut interpréter géométriquement c et s comme le cosinus et le sinus d'un angle θ)

La multiplication d'un vecteur par G correspondant à une rotation de ce même vecteur de l'angle θ

Comment utiliser les matrices de Givens ?

Problème : étant donné un vecteur $z \neq 0$ de longueur 2, trouver une matrice G telle que $Gz = xe_1$ avec $x = \|z\|$

Solution :

$$Gz = \begin{pmatrix} cz_1 + sz_2 \\ -sz_1 + cz_2 \end{pmatrix} = xe_1$$

En multipliant la premiere équation par c, la seconde par s et en soustrayant, on obtient $(c^2 + s^2)z_1 = cx$ et donc $c = z_1/x$.

De la même façon, on trouve $s = z_2/x$

Puisque $c^2 + s^2 = 1$, on déduit que $z_1^2 + z_2^2 = x^2$, et donc

$$\begin{aligned} c &= \frac{z_1}{\sqrt{z_1^2 + z_2^2}} \\ s &= \frac{z_2}{\sqrt{z_1^2 + z_2^2}} \end{aligned}$$

La méthode de Givens

Grâce aux matrices de Givens, nous savons comment mettre une composante d'un vecteur à 0. Notons G_{ij} la matrice identité de taille $n \times n$ avec les i-ième et j-ième colonnes modifiées de façon à inclure la rotation de Givens.

Exemple pour $n = 6$:

$$G_{25} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & 0 & s & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -s & 0 & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

La multiplication d'un vecteur par cette matrices laissent toutes les lignes du vecteurs inchangées sauf les lignes 2 et 5

La méthode de Givens (suite)

Algorithme 1 (Algorithme de Givens)

Initialiser Q à la matrice identité de taille $m \times m$
Initialiser R à la matrice A de taille $m \times n$
for $i=1:n$
  for $j=i+1:m$
    - *construire la matrice G_{ij} pour mettre un zéro en position (j,i) dans la matrice en utilisant la valeur en (i,i)*
    - $R = G_{ij} R$
    - $Q = Q G_{ij}^T$
  end
end

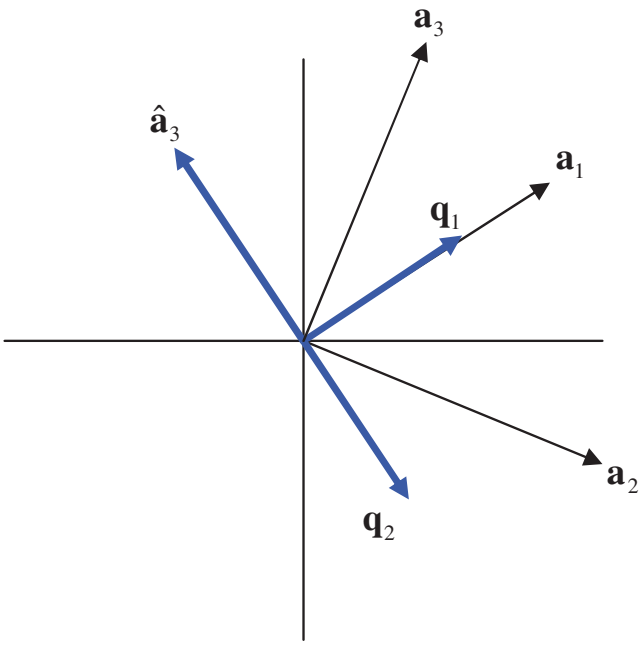
La méthode de Gram-Schmidt

À partir des colonnes $[a_1, \ldots, a_n]$ de la matrice A, on crée une base orthonormale $\{q_1, \ldots, q_n\}$ et on stocke les coefficients dans une matrice triangulaire supérieure R

Algorithme 2 (Algorithme de Gram-Schmidt)

$r_{1,1} = \|a_1\|$
$q_1 = a_1 / r_{1,1}$
for $k=1:n-1$
  $q_{k+1} = a_{k+1}$
  for $i=1:k$
    $r_{i,k+1} = q_i^* q_{k+1}$
    $q_{k+1} = q_{k+1} - r_{i,k+1} q_i$
  end
  $r_{k+1,k+1} = \|q_{k+1}\|$
  $q_{k+1} = q_{k+1} / r_{k+1,k+1}$
end

La méthode de Gram-Schmidt (suite)



Coût de la décomposition QR

- la méthode de Givens : $2mn^2 - 2/3n^3$ multiplications
- la méthode de Gram-Schmidt : mn^2 multiplications
- la méthode de Householder : $mn^2 - 1/3n^3$ multiplications

Utilisation de la décomposition QR

En MATLAB : `[Q,R] = qr(A)` pour A de taille $m \times n$ avec $m \geq n$

- `qr(A,0)` renvoie la matrice Q compactée
- la décomposition QR permet d'obtenir une base de l'image d'une matrice A de rang plein (les n premières colonnes de Q) et le noyaux de A^* (les $m - n$ dernières colonnes de Q)
- on utilise la décomposition QR pour résoudre des problèmes des moindres carrés

Résolution d'un système linéaire : $Ax = b$
$\rightarrow$ si $A = QR$ alors $Rx = Q^*b$

Problème des moindres carrés

Étant donnée A de taille $m \times n$ (avec $m > n$), on cherche

$$\min_x \|Ax - b\|$$

- minimiser $\|Ax - b\|$ est équivalent à minimiser $\|Ax - b\|^2$
- la norme d'une vecteur est invariante par multiplication par Q^*, c'est-à-dire $\|y\| = \|Q^* y\|$ pour tout y
- si on décompose y en

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$

alors $\|y\|^2 = \|y_1\|^2 + \|y_2\|^2$

Problème des moindres carrés (suite)

Si $A = QR$, on définit

$$c = Q^* b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

avec c_1 de taille n et c_2 de taille $m - n$.
Alors

$$\begin{aligned} \|b - Ax\|^2 &= \|Q^*(b - Ax)\|^2 \\ &= \|c - Rx\|^2 \\ &= \|c_1 - R_1 x\|^2 + \|c_2 - 0x\|^2 \\ &= \|c_1 - R_1 x\|^2 + \|c_2\|^2 \end{aligned}$$

Le minimun est obtenu en prenant pour x solution de $R_1 x = c_1$

En MATLAB, il faut utiliser `x = A\b`

Problème des moindres carrés (suite)

Données (t_i, f_i) représentant la quantité de polluant dans une rivière mesurée une fois par an.

On veut savoir si l'évolution est linéaire !

On cherche donc à résoudre

$$\min_x \|Ax - b\|$$

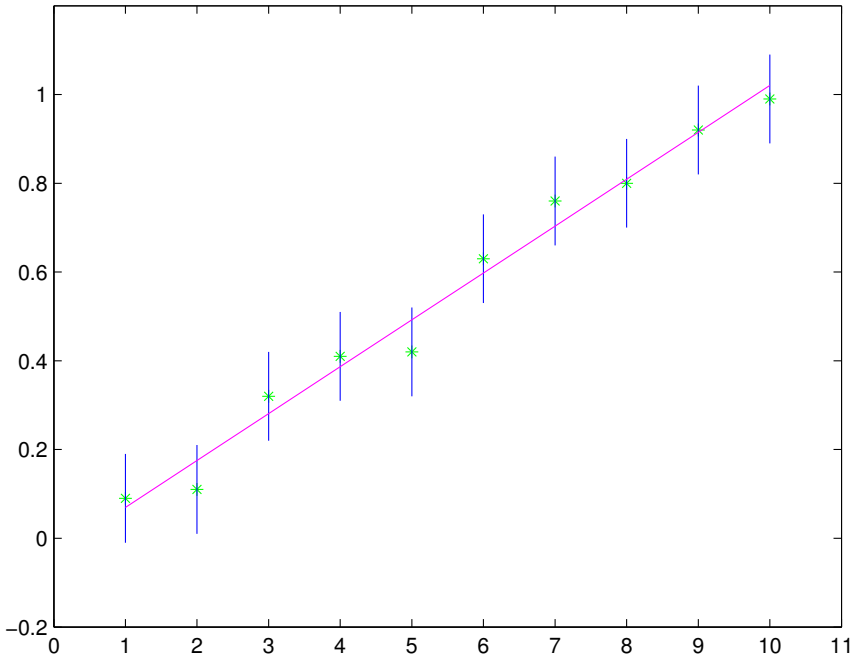
avec

$$A = \begin{pmatrix} 1 & t_1 \\ . & . \\ . & . \\ . & . \\ 1 & t_{10} \end{pmatrix}, \qquad b = \begin{pmatrix} f_1 \\ . \\ . \\ . \\ f_{10} \end{pmatrix}$$

Problème des moindres carrés (suite)

```
sigma=.05;
t = [1:10];
b = [0.09 0.11 0.32 0.41 0.42 0.63 0.76 0.8 0.92 0.99];
plot(t,b,'g*');
hold on;
for i=1:10
   plot([t(i),t(i)],[b(i)+2*sigma,b(i)-2*sigma]);
end
axis([0 11 -.2 1.2]);
A = [ones(10,1),t'];
x = A \ b';
plot(t,A*x,'m');
```

Problème des moindres carrés (suite)



Décomposition en éléments propres

Définition 2

Une matrice A de taille $n \times n$ est diagonalisable si $A = U\Lambda U^{-1}$ où Λ est une matrice diagonale dont les éléments sont les valeurs propres λ_i. Les colonnes de U sont appelés les vecteurs propres :

$$Au_i = \lambda_i u_i.$$

On est sur que la décomposition existe si

- A est symétrique réelle ou bien hermitienne, ou
- A est normale ($AA^* = A^*A$)
- les valeurs propres de A sont distinctes

Dans les autres cas, la décomposition peut très bien ne pas exister. Même si elle existe pour une matrice proche de A

Calcul de la réduction

Algorithme en 2 étapes :
Étape 1 : réduire la matrice A sous une forme compacte facilement manipulable.
Trouver une matrice unitaire V telle que

$$V^* A V = H$$

où H est

- tridiagonale sur A est hermitienne (ou symétrique réelle)
- Hessenberg supérieure dans les autres cas

On peut faire cela en $\mathcal{O}(n^3)$

Les matrices A et H étant semblables, si on trouve une décomposition de H en

$$H = U \Lambda U^{-1}$$

alors on a une décomposition

$$A = (VU) \Lambda (VU)^{-1}$$

pour A

Calcul de la réduction (suite)

Étape 2 : Trouver la décomposition de H par une itération QR

- calculer $H = QR$
- remplacer H par RQ

Puisque $Q^* Q = I$ et $H = QR$, on a

$$RQ = (Q^* Q) RQ = Q^* H Q$$

Par conséquent le nouveau H a les mêmes valeurs propres que l'ancien H et si on a une décomposition de RQ, on a aussi une décomposition de H

On répète l'étape 2 plusieurs fois (environ $5n$ fois) et très souvent, les éléments sous la diagonale s'annulent. On lit alors les valeurs propres sur la diagonale

En MATLAB, `[U,Lambda]=eig(A)`

Coût de la réduction

- le coût de l'étape 1 est en $\mathcal{O}(n^3)$
- le coût de $\mathcal{O}(n)$ itération de l'étape 2 est en $\mathcal{O}(n^3)$
- le coût total est donc en $\mathcal{O}(n^3)$

Utilisation de la réduction

- stabilité de systèmes
- convergence de méthodes itératives
- calcul de sous-espaces stables
- convergence de A^p quand $p \to +\infty$

La décomposition en Valeurs Singulières (SVD)

Définition 3 (SVD)

Toute matrice A de dimension $m \times n$ (avec $m \geq n$) peut se décomposer de la manière suivante

$$A = U\Sigma V^*$$

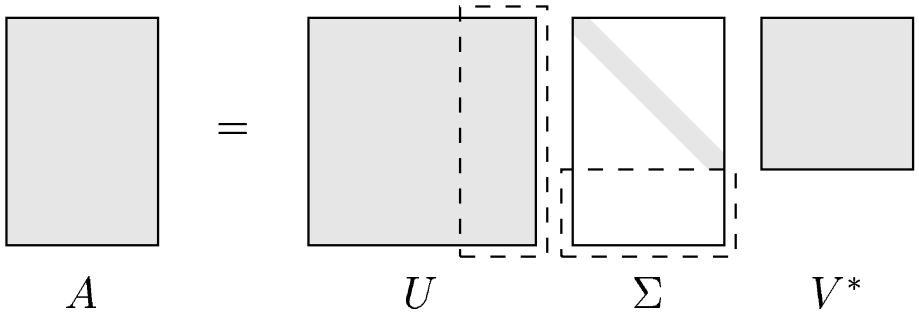
avec

- *U de dimension $m \times m$ et $U^*U = I$*
- *Σ de dimension $m \times n$, les seuls éléments non nuls étant sur la diagonale et ce sont des nombres positifs $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$*
- *V de dimension $m \times m$ et $V^*V = I$*

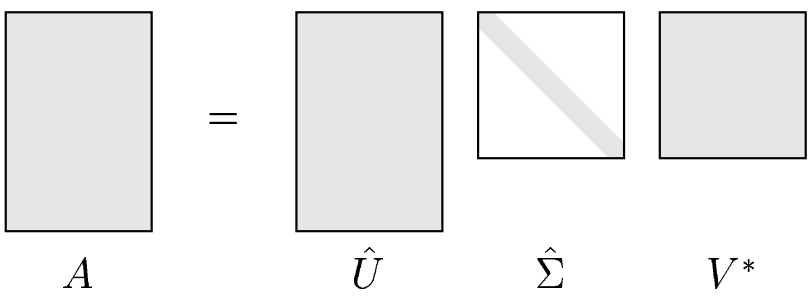
Les σ_i sont appelés les valeurs singulières de la matrice A

La décomposition en Valeurs Singulières (SVD)

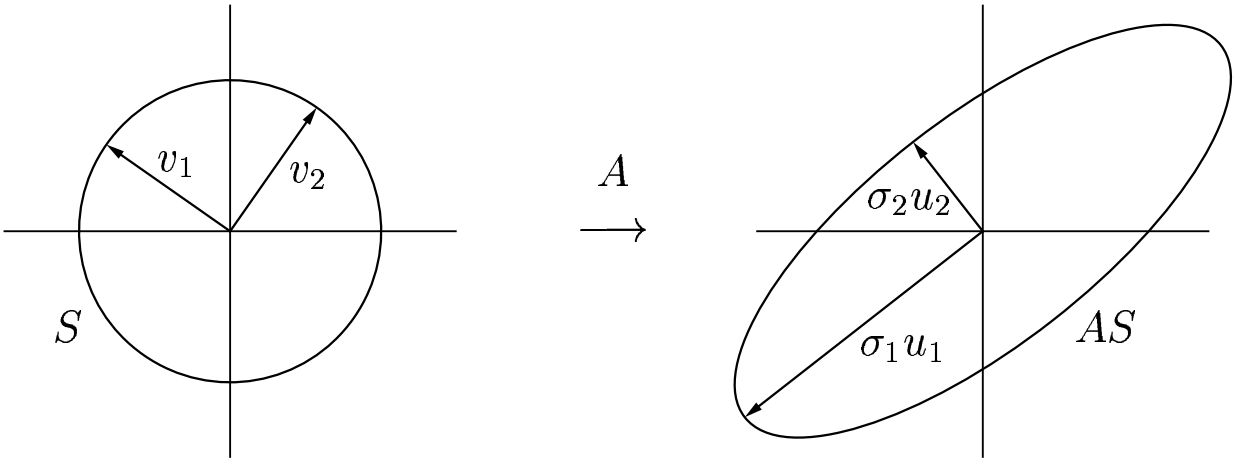
- Décomposition pleine :



- Décomposition réduite :



Interprétation géométrique de la SVD



Remarque

Si $A = U\Sigma V^*$ alors

$$A^*A = (U\Sigma V^*)U\Sigma V^* = V\Sigma^* U^* U\Sigma V^* = V\Sigma^*\Sigma V^*$$

Par conséquent

- Les valeurs singulières σ_i de A sont les racines carrées des valeurs propres de A^*A
- Les colonnes de V (qui sont les vecteurs singuliers droits de A) sont les vecteurs propres de A^*A
- Les colonnes de U (qui sont les vecteurs singuliers gauches de A) sont les vecteurs propres de AA^*

Calcul de la SVD

On peut calculer la SVD de la matrice A de la façon suivante :

1. Calculer A^*A
2. Calculer une décomposition en éléments propres $A^*A = V\Lambda V^*$
3. Soit Σ la matrice de taille $m \times n$ ayant dans la diagonale la racines carrée de Λ
4. Résoudre $U\Sigma = AV$ avec U unitaire

En fait, cet algorithme est instable ! Il existe néanmoins des algorithmes stables et efficaces pour le calcul de SVD

En MATLAB, `[U,S,V]=svd(A)`

Coût : $\mathcal{O}(mn^2)$ avec une constante de l'ordre de 10

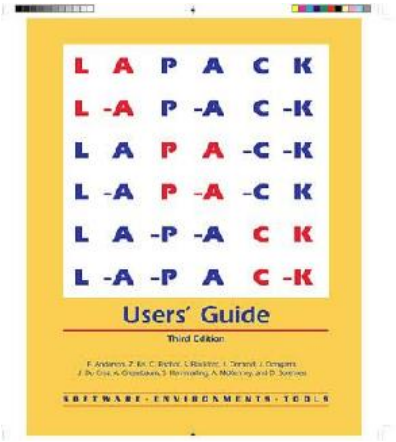
Applications : résoudre des problèmes de moindre carré mal conditionné, calcul de l'image ou du noyau d'une matrice, compression d'images, défloutage d'images, etc.

À éviter ...

- Calculer l'inverse d'une matrice !
  On peut résoudre $Ax = b$ en multipliant par A^{-1} de chaque coté
  $$A^{-1}Ax = x = A^{-1}b$$
  C'est une TRÈS MAUVAISE IDÉE. C'est plus couteux que la décomposition LU et en général, cela génère des erreurs d'arrondi plus importantes. Quand vous voyez l'inverse d'une matrice dans une formule à calculer, pensez « décomposition LU » !
- Calculer une décomposition de Jordan !
  Toutes les matrices ne sont pas diagonalisables
  $$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$
  mais toutes les matrices peuvent se décomposer sous forme de Jordan,
  $$A = WJW^{-1}$$
  avec W inversible et J presque diagonale (les valeurs propres sur la diagonale et des 1 sur la sur-diagonale definissant des blocs)

Logiciels

Pour calculer des décompositions de matrice et résoudre des problèmes matriciels en Fortran ou en C, il faut utiliser LAPACK (`http://www.netlib.org/lapack/`).



- algorithmes numériquement stables
- implantation orienté ligne ou colonne
- basé sur les BLAS et donc efficace (pour des tailles de matrices supérieures à 100)

En Java, on peut utiliser JavaNumerics (`http://math.nist.gov/javanumerics/`)

Et bien sur, MATLAB qui est basé sur LAPACK

Récapitulatif

Décomposition	Coût	Utilisation
LU	$n^3/3$	• résolution de systèmes linéaires • calcul de déterminant
QR	$mn^2 - 1/3n^3$	• résolution de problèmes de moindres carrés • résolution de systèmes linéaires
Réduction	$\mathcal{O}(n^3)$	• détermination de la convergence • détermination de la stabilité • calcul d'espaces invariants
SVD	$\mathcal{O}(mn^2)$	• résolution de systèmes linéaires • résolution de problèmes de moindres carrés • calcul de l'image et du noyau d'une matrice

Conclusion

- Nous n'avons pas parlé des matrices creuses (celles avec beaucoup de zéros). Il existe des algorithmes spécifiques pour les matrices creuses. On essaie souvent d'utiliser, lorsque cela est possible, des décompositions qui préservent le caractère creux
- Il existe des algorithmes spécialisés pour les matrices structurées (par exemple symétrique, tridiagonale, etc)