Séances 4 et 5 : résolution de systèmes linéaires par méthodes directes

I Introduction

* Problème à résoudre : soit A une matrice inversible ($A \in GL_n(\mathbb{R})$) et $b \in \mathbb{R}^n$. On cherche à résoudre le système $Ax = b$ de manière efficace (*) numériquement.

(*) rapide et robuste (par rapport aux erreurs d'arrondis).

* Ce problème se rencontre dans de nombreuses modélisations (voir séance 3 et textes de modélisation).

* La méthode d'inversion de Cramer :

$$A \cdot ({}^t\mathrm{com}(A)) = \det(A) \cdot I_n$$

comatrice de A :

$$A_{ij} = \begin{matrix} & j & \\ i & \begin{pmatrix} * & * \\ * & * \end{pmatrix} \end{matrix}$$

(terme : $(-1)^{i+j} \det(A_{ij})$)

n'est pas adaptée dans ce contexte. Par exemple, si $n = 100$, avec

la formule du déterminant :

$$\det(A) = \sum_{\sigma \in \mathfrak{S}_n} \varepsilon(\sigma)\, a_{1,\sigma(1)} \dots a_{n,\sigma(n)}$$

on doit effectuer

- $\star$ $n!$ additions
- $\star$ $n \cdot n!$ multiplications

soit de l'ordre de $(n+1)!$ opérations. Au final, pour résoudre un système 100×100, il faudrait effectuer $> 3 \cdot 10^{154}$ opérations.

Avec un ordinateur effectuant 10^8 opérations par seconde, (soit 100 Mflops), il faudrait

floating operations per second

plus de $3 \cdot 10^{148}$ secondes, soit plus que la durée de vie de l'univers, par résoudre ce système ....

$\star$ L'objectif est ici de construire de nouvelles méthodes directes, c'est à dire exactes (théoriquement) hors arrondis numériques.

$\star$ La robustesse numérique de la méthode à construire ne doit pas être

Confondre avec le conditionnement d'une matrice. Celui-ci est indépendant de la méthode construite et mesure seulement la capacité à pouvoir résoudre numériquement un système linéaire.

(On ne cherche jamais à résoudre numériquement un système mal conditionné : c'est l'échec assuré)

II) Méthode de Gauss

→ exacte (directe)

→ rapide

→ robuste par rapport aux erreurs d'arrondis commis pendant son exécution.

* Principe général : on cherche à se ramener à la résolution d'un système triangulaire, simple à résoudre, en effectuant des transformations de A et b.

* Résolution d'un système triangulaire :

$Tu = B$ où

$T = \begin{pmatrix} t_{11} & & \\ & \ddots & \\ 0 & & t_{nn} \end{pmatrix}$ avec $t_{ii} \neq 0$

La résolution s'effectue par le principe de remontée :

$$\begin{cases} u_n = \dfrac{\beta_n}{t_{n,n}} \\ u_{n-1} = \dfrac{1}{t_{n-1,n-1}} \left(\beta_{n-1} - t_{n-1,n} u_n \right) \\ \vdots \\ u_1 = \dots \end{cases}$$

en n^2 opérations environ (équivalent)

* On présente tout d'abord un principe de transformation élémentaire de A et b faisant apparaître des "zéros" sous la diagonale de la 1ère colonne de A :

On note $A = [a_{i,j}]$ et $b = [b_j]$

Etape 1 (échange de lignes)

Soit i la ligne de A telle que

$$|a_{i,1}| = \max_{k \in \{1,\dots,n\}} |a_{k,1}| \quad (\neq 0)$$

(plus grand coefficient de la 1ère colonne de $|A|$)

On échange les lignes 1 et i dans A et b

$\rightarrow$ $\tilde{A}$ et $\tilde{b}$

Etape 2 (combinaisons de lignes)

$\tilde{A} = [\tilde{a}_{ij}]$ et $\tilde{b} = [\tilde{b}_i]$.

Pour $2 \leq i \leq n$, on remplace la ligne i de $\tilde{A}$ (et $\tilde{b}$), notée $\tilde{L}_i$ par

$$\tilde{L}_i \longrightarrow \tilde{L}_i - \frac{\tilde{a}_{i1}}{\tilde{a}_{11}} \tilde{L}_1$$

On note $G(A)$ et $G(b)$ les nouvelles matrices et second membre. On a

pivot $\neq 0$

$$G(A) = \begin{pmatrix} \tilde{a}_{11} & * & \cdots & * \\ 0 & * & & * \\ \vdots & \vdots & & \vdots \\ 0 & * & & * \end{pmatrix}$$

A_1 : $\tilde{a}_{11}$; A'_1 : ??

* On présente ici l'algorithme de Gauss de manière encore formelle.

L'idée consiste à itérer le principe précédent sur une sous matrice.

* Initialisation : on part de A et b de taille n

* Itération 1 :

$A, b \longrightarrow G(A), G(b)$. On note $A_1 = G(A)$ et $(b)_1 = G(b)$

* Itération 2 : on note A'_1 (et $(b')_1$) la sous matrice de A_1 formée des lignes et colonnes de 2 à n.

$(A'_1, b'_1) \rightsquigarrow G(A'_1), G(b'_1)$ (à justifier)

On note $A'_2 = G(A'_1)$ (taille $n-1$)
et A_2 la matrice de taille n
(où on rajoute la 1ère ligne et colonne de A_1)

Itération $n-1$: On dispose de A_{n-1} et b_{n-1} telles que

$$A_{n-1} = \begin{pmatrix} a^1_{1,1} & \cdots & a^1_{1,n} \\ 0 & a^2_{2,2} \cdots & a^2_{2,n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a^{n-1}_{n,n} \end{pmatrix}$$

On résout $A_{n-1} x = b_{n-1}$ par remontée.

FIN //

Par justifier cet algorithme, il faut traduire en termes d'opérations algébrique la transformation élémentaire :

* Echange de lignes :
l'échange de la ligne 1 et de la ligne i dans A revient à transformer $\begin{pmatrix} A \\ b \end{pmatrix}$ en $T^{1,i} \begin{pmatrix} A \\ b \end{pmatrix}$ où

multiplication à gauche

$$T^{1,i} = \begin{pmatrix} 0 & & 1 & & \\ & 1 & & & \\ 1 & & 0 & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix}$$

(matrice de transposition)

Combinaison de lignes :

le remplacement de L_i $(2 \leq i \leq n)$ par $L_i + \alpha_i L_1$ revient à transformer A en $E.A$ où

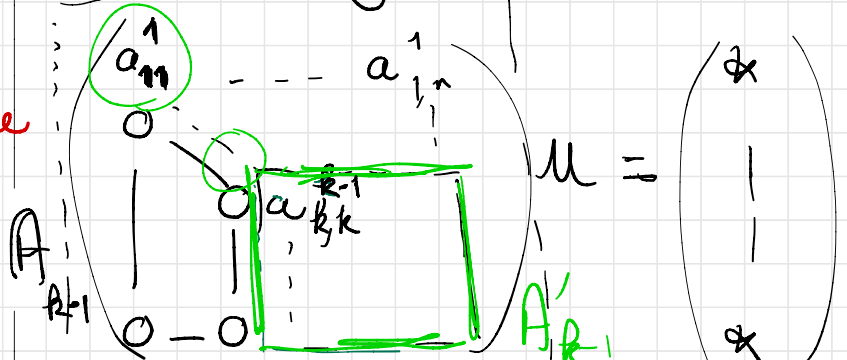
$$E = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \alpha_2 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ \alpha_n & 0 & \cdots & 1 \end{pmatrix}$$

(matrice de transvection)

On remarque que T et E sont inversibles ($\det T = -1$ et $\det E = 1$)

Le système $G(A)x = G(b)$ est donc équivalent au système $Ax = b$.

En se plaçant alors à l'itération k, (ceci en raisonnant par récurrence) de l'algorithme, on a un système équivalent du type



A'_k est inversible car

$$\det(A_{k-1}) = \det(A'_{k-1}) \cdot a^1_{11} \cdots a^{k-1}_{k-1,k-1}$$

pivots $\neq 0$

Ainsi A'_{k-1} est inversible et il est possible de réaliser l'itération k. //

On dispose à ce stade d'un algorithme universel de résolution exacte d'un système linéaire. On parle de l'algorithme de Gauss avec stratégie de pivot partiel.

→ Coût de l'algorithme de Gauss :

* n fois (négligeable)
* A l'itération 1 :

chaque ligne : $\left(L_i \leftarrow L_i - \frac{a_{i,1}}{a_{11}} L_1\right)$

$1 + 2n$ opérations

soit $(n-1)(1+2n) \sim 2n^2$ opérations

* Au final, et avant remontée, il faut effectuer environ

$$2n^2 + 2(n-1)^2 + \dots + 2.1 = \dots = 2\,\frac{n(n+1)(2n+1)}{6} \sim \frac{2n^3}{3}$$

Sur l'exemple d'un système 100×100, et d'un ordinateur à 100 Mflops, il faut moins d'un centième de seconde....

→ Robustesse de Gauss

On peut montrer que la stratégie de pivot permet d'assurer la

robustesse de l'algorithme de Gauss (en évitant les risques de division par un pivot très petit). (Voir TD4). (*)

Remarque :

* L'algorithme de Gauss sert aussi à calculer des inverses (*) et des déterminants (*).

(*) $\det(A_{n-1}) = \pm 1 \det(A)$

A_{n-1} : triangulaire ; ± 1 : nbre d'échanges

(*) Résolution de n systèmes élémentaires //

(*) Référence : Lascaux-Théodor (Tome 1)

III Factorisation LU

* Dans le cas où il n'a pas été nécessaire de faire des échanges de lignes (et qu'aucune stratégie n'est utilisée), la matrice triangulaire supérieure provient de produits de A avec des matrices de transvection :

$$\underbrace{\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & \alpha_n^{n-1} & \end{pmatrix}}_{E_{n-1}} \underbrace{\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & * & 1 \end{pmatrix}}_{E_{n-2}} \cdots \underbrace{\begin{pmatrix} 1 & & & \\ \alpha_2^1 & \ddots & & \\ \vdots & & 1 & \\ \alpha_n^1 & & & 1 \end{pmatrix}}_{E_1} A = A_{n-1}$$

$n-1$; $n-2$; 1

En transformant cette relation on a

$$A = (E_1)^{-1}(\ )\cdots(E_{n-1})^{-1} A_{n-1}$$

On a donc $A = L\,U$

lower (L) upper (U)

On $T.I \uparrow \rightarrow$

$$(E_{n-1})^{-1} = \begin{pmatrix} 1 & & \\ -\alpha_2^1 & \ddots & 0 \\ \vdots & & \\ -\alpha_n^1 & 0 & 1 \end{pmatrix} \text{ et}$$

$$(E_1)^{-1}\cdots(E_{n-1})^{-1} =$$

$$\begin{pmatrix} 1 & & & \\ -\alpha_2^1 & 1 & & \\ \vdots & -\alpha_3^2 & \ddots & \\ -\alpha_n^1 & -\alpha_n^2 & -\alpha_n^{n-1} & 1 \end{pmatrix}$$

Triangulaire supérieure (U)

Triangulaire inférieure déjà calculée (L)

Théorème : soit $A \in GL_n(\mathbb{R})$ telle que tous ses mineurs principaux :

$$\Delta_k = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix}$$

soient inversible. Alors il existe un unique couple (L, U) tel que :

- L triangulaire inférieure avec $l_{ii} = 1$ $(1 \leq i \leq n)$
- U triangulaire supérieure

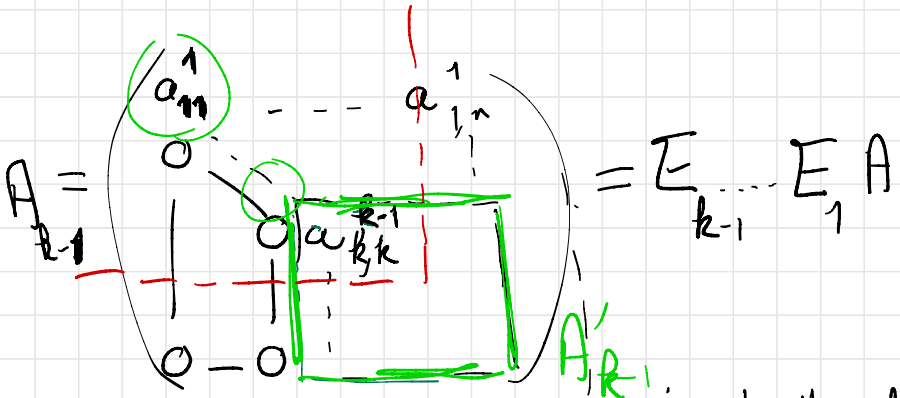
et $A = LU$

preuve :

* existence : il suffit de vérifier que la condition du Théorème autorise de ne pas avoir à utiliser d'échange de ligne dans Gauss.

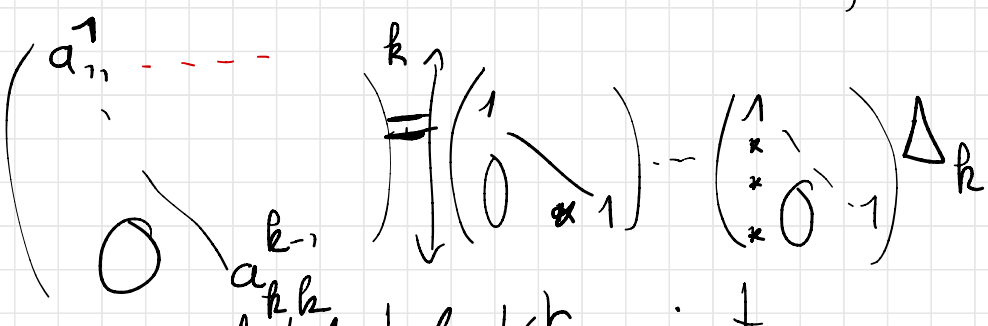
Réurrence sur k :

* Itération 1 de Gauss : on prend $a_{11} \neq 0$ comme 1er pivot.
* Itération k : on se trouve dans la situation suivante :



$= E_{k-1} \dots E_1 A$

En prenant les mineurs de taille k, on a :



puis en calculant le déterminant :

$$a^1_{11} \dots a^{k-1}_{kk} = 1 \cdot \det(\Delta_k)$$

$\neq 0$ $\neq 0$ $\neq 0$

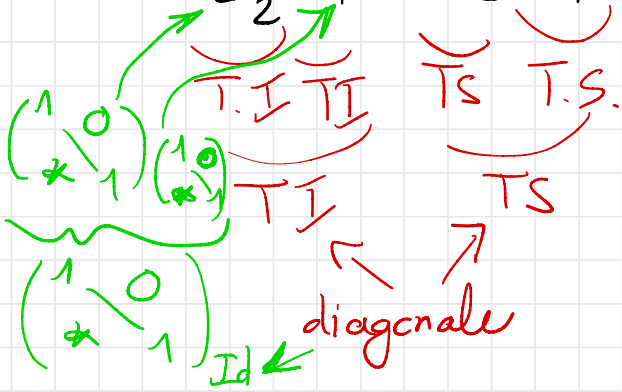
En conclusion $a^{k-1}_{kk} \neq 0$ et aucun échange de ligne n'est nécessaire à cette nouvelle étape.

12

* unicité : on suppose que

$$L_1 U_1 = L_2 U_2$$

$$\Leftrightarrow L_2^{-1} L_1 = U_2 U_1^{-1}$$



Ainsi, $L_1^{-1} L_2$ est diagonale avec des coefficients egaux à 1. D'où

$$L_1^{-1} L_2 = Id \Rightarrow L_1 = L_2 \text{ et } U_1 = U_2$$

Remarques

1). La factorisation LU permet de résoudre ensuite tous les systèmes du type $Ax = b$ en $2n^2$ opérations en résolvant successivement 2 systèmes triangulaires :

$$Ly = b$$

puis

$$Ux = y$$

2) Même si A est bien conditionnée il est possible que L et/ou U ne le soient pas.

3) De manière générale, si $A \in GL_n(\mathbb{R})$ il existe une matrice P de permutation (cf) tq

$$PA = LU$$

($P_\sigma = [p_{ij}]$ où $p_{ij} = \delta_{i,\sigma(j)}$)

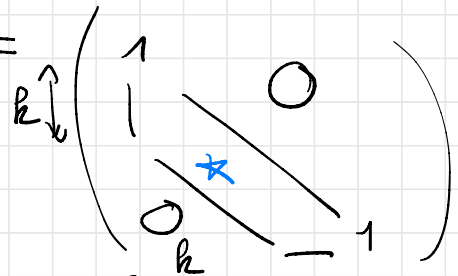
(voir aussi extensions en TD 4).

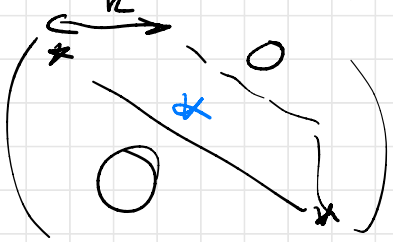
4) la factorisation LU conserve la structure "bande" d'une matrice : si

$$A = \begin{pmatrix} * & \cdots & * & 0 \\ \vdots & \ddots & \ddots & * \\ * & \ddots & \ddots & \vdots \\ 0 & * & \cdots & * \end{pmatrix} \quad (k)$$

$= LU$

alors

$L =$  et

$U =$ 

(voir TD4, preuve à l'aide de Gauss)

IV Factorisation de Cholesky

On suppose ici que A est symétrique définie positive ($A \gg 0$)

Par le critère de sylvester, $\det(\Delta_k) > 0$,

on en déduit que A admet une factorisation LU. Plus précisément, on peut récrire sous une autre forme cette factorisation :

Théorème : soit $A \gg 0$. Il existe une unique matrice $B = [b_{ij}]$ triangulaire inférieure telle que :

* $b_{ii} > 0$ $(1 \leq i \leq n)$
* $A = B({}^tB)$

"lower" "upper"

preuve : * existence : on a déjà $A = LU$. De plus, on

montrer que

$\forall i \in \{1, \dots n\}, u_{ii} > 0$

(par récurrence, en reprenant la preuve de Gauss, en calculant les déterminants mineurs, voir preuve existence LU).

On note $D = \text{diag}(\sqrt{u_{11}}, \sqrt{u_{22}}, \dots \sqrt{u_{nn}})$

et on a

$$A = L(D D^{-1}) U$$
$$= \underbrace{(LD)}_{B}\underbrace{(D^{-1}U)}_{C}$$

B : Tr. inf. $\quad$ C : Tr. supp

Comme A est symétrique ${}^tA = A$

soit ${}^tC\,{}^tB = BC$

puis

$$B^{-1}\,{}^tC = C\,{}^tB^{-1}$$

Tr inf Tr inf Tr sup Tr sup

Tr inf $\quad$ Tr sup

Il s'agit donc d'une matrice diagonale dont les coefficients sont : $\frac{1}{\sqrt{u_{ii}}}\sqrt{u_{ii}} = 1$

On a donc $B^{-1}\,{}^tC = Id$ soit

$C = {}^tB$ et $A = B\,{}^tB$ avec $b_{ii} = \sqrt{u_{ii}} > 0$.

* Unicité : si on a $A = B\,{}^tB$, alors avec $\Delta = \text{diag}(b_{11}, \dots b_{nn})$, on a

$A = (B \cdot \Delta^{-1})(\Delta^r B)$

lower ($l_{ii} = 1$) — upper

Par l'unicité de LU, $\Delta^r B$ est unique.

Or, $\Delta^r B = \begin{pmatrix} b_{11}^2 & & \\ & \ddots & b_{ii} b_{ij} \\ 0 & & b_{nn}^2 \end{pmatrix}$

On en déduit que les b_{ii}^2 sont uniques et de même pour les $b_{ii} (> 0)$ et Δ est donc unique, tout comme B //

En pratique, pour rechercher une factorisation de Cholesky, on n'utilise pas la méthode de Gauss mais une autre méthode, environ 2 fois plus rapide : la méthode des coefficients indéterminés.

Le système $A = B^r B$ (d'inconnues $b_{ij}, j \le i$) s'écrit :

$$a_{ij} = \sum_{k=1}^{n} b_{ik} b_{jk} = \sum_{k=1}^{\min(i,j)} b_{ik} b_{jk}$$

et peut se résoudre "colonne par colonne" :

* $i = 1$:
$b_{11} = \sqrt{a_{11}}$ (cf $a_{11} > 0$)
$b_{21} = \frac{a_{21}}{b_{11}}$
$\vdots$
$b_{n1} = \frac{a_{n1}}{b_{11}}$

puis de manière générale)

pour $i \geq 2$:

$$b_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} b_{i,k}^2}$$ (positivité garantie par l'existence de Cholesky)

$$b_{i+1,i} = \frac{a_{i,i+1} - \sum_{k=1}^{i-1} b_{i,k}\, b_{i+1,k}}{b_{ii}}$$

$$b_{n,i} = \dots$$

Cette méthode nécessite :

- n racines carrés (rapide avec Newton)
- $\frac{n(n-1)}{2}$ divisions
- $\frac{n(n^2-1)}{6}$ ad. et $\frac{n(n^2-1)}{6}$ multiplications.

soit de l'ordre de $n^3/3$ opérations.

V | Factorisation QR

On présente une nouvelle factorisation d'une matrice $A \in GL_n(\mathbb{R})$ sous la forme :

$A = QR$ avec

$Q \in O_n(\mathbb{R})$ (orthogonale)

R triangulaire supérieure.

On peut alors résoudre facilement un système $Ax = b$

en résolvant le système triangulaire

$$Rx = {}^tQb.$$

Théorème : soit $A \in GL_n(\mathbb{R})$. Il existe un unique couple (Q, R) tq

$A = QR$ où :

* $Q \in O_n(\mathbb{R})$
* R triangulaire supérieure et $r_{ii} > 0$ $(1 \leq i \leq n)$

preuve :

* Existence : on utilise le procédé d'orthonormalisation de Gram Schmidt :

$A = \begin{pmatrix} a_1 & \cdots & a_n \\ | & & | \end{pmatrix}$ La famille $(a_1, \dots a_n)$ forme une base de $\mathbb{R}^n$ ($A \in GL_n(\mathbb{R})$). Elle s'orthonormalise en une famille $(q_1, \dots q_n)$ tq

$$q_1 = \frac{a_1}{\|a_1\|}$$

$$q_2 = \frac{a_2 - (\langle a_2, q_1 \rangle) q_1}{\| a_2 - \langle a_2, q_1 \rangle q_1 \|}$$

$$q_n = \frac{a_n - \sum_{i=1}^{n-1} \langle a_n, q_i \rangle q_i}{\| \cdots \|}$$

On a donc :

$$\begin{pmatrix} a_1 & \cdots & a_n \\ | & & | \end{pmatrix}_A = \begin{pmatrix} q_1 & \cdots & q_n \\ | & & | \end{pmatrix}_Q \begin{pmatrix} \|a_1\| & * & * \\ 0 & \ddots & * \\ & \ddots & \\ & 0 & \|a_n\| \end{pmatrix}_R$$

* Unicité :

On suppose $A = Q_1 R_1 = Q_2 R_2$.

On a

$T = {}^tQ_2 Q_1 = R_2 R_1^{-1}$ est triang. sup.

De plus,

$T({}^tT) = {}^tQ_2 Q_1 ({}^tQ_1 Q_2)$

$= Id$

Comme $t_{ii} > 0$ ($(R_2)_{ii} > 0$ et $(R_1)_{ii} > 0$)

On se retrouve dans le cas d'une factorisation de Cholesky de la matrice Id. Par l'unicité de cette factorisation, on a

$T = Id$

donc $Q_2 = Q_1$ et $R_2 = R_1$

(on peut aussi utiliser l'unicité de Gram Schmidt).

Remarques :

1) Dans la pratique, l'algorithme de Gram Schmidt n'est pas robuste numériquement. On utilise l'algorithme de Householder (voir TD3) nécessitant $\sim \frac{4n^3}{3}$ opérations

2) La factorisation QR sert plutôt à rechercher les valeurs propres d'une matrice (méthode QR : $A = Q_1 R_1$,

$A_2 = R_1 Q_1 = Q_2 R_2$

$\downarrow (k \to +\infty)$

$\begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$

avec le langage de votre choix (Python, Matlab, Scilab, ...) et tester celui-ci sur des matrices de grande taille, bien ou mal conditionnées. (voir TP3, Ex1)

→ Me 21/10.

aléatoire 100×100

Hilbert $\left[\frac{1}{i+j-1}\right]$

$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ 1 & & & & \\ \vdots & & & & \\ 1 & & & & \end{pmatrix}$

3) La factorisation QR sert aussi à résoudre des problèmes de type moindres carrés.

A faire : TD3 (par groupes) et TP3 : Implémenter un des algorithmes (Gauss ou Cholesky)

Tester avec $Ax = A \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$