

**Discrétisation d’un problème aux dérivées partielles par la méthode des différences finies**

**Réalisé par :**  
Eloued Yosra   
Hmaied Oumaima

**Groupe INDP1D**

**Année Universitaire**

**2016/2017**

**Discrétisation d’un problème aux dérivées partielles par la méthode des différences finies**

**Objectifs :**

1. Donner la matrice A et le vecteur b du système
2. Décrire la méthode LU, Cholesky et QR pour la résolution des systèmes linéaire
3. Programmer la méthode de LU pour la résolution du système linéaire relatif au probléme (1)
4. **La détermination de la matrice A et le vecteur b du système**

Les calculs théoriques aboutissent au système AU+b.

On pose l’équation matricielle AUh=b avec A la matrice donnée par la fonction suivante :

function [A]=matriceA(n)

N=(n-1)^2;

h=pi/n;

D1=-1\* diag(ones((n-1)^2-(n-1),1),(n-1))

D2=-1\* diag(ones((n-1)^2-(n-1),1),(1-n))

A=4\*diag(ones((n-1)^2,1))-diag(ones((n-1)^2-1,1),1)-diag(ones((n-1)^2-1,1),-1)+D1+D2

La matrice b est donné par :

function [b]=matriceb(n)

h=pi/n;

X=2:n-2;

b(1)=2\*sin(2\*h)+2\*sin(h)/(h^2);

b(X)=2\*sin((X+1)\*h)+sin(h\*X)/(h^2);

b(n-1)=2\*sin(n\*h)+(sin((n-1)\*h)-sin(h))/(h^2);

k=n;

for j=2:n-2

b(k)=2\*sin((1+j)\*h)+sin(j\*h)/(h^2);

b(k+X-1)=2\*sin((X+j)\*h);

b(k+n-2)=2\*sin((n-1+j)\*h)-sin(j\*h)/(h^2);

k=k+n-1;

end

b(k)=2\*sin(n\*h)+(sin((n-1)\*h)-sin(h))/(h^2);

b(k+X-1)=2\*sin((X+n-1)\*h)-sin(h\*X)/(h^2);

b(k+n-2)=2\*sin(2\*(n-1)\*h)-(2\*sin((n-1)\*h))/(h^2);

1. **Description de la méthode LU, Cholesky et QR pour la résolution des systèmes linéaire :**

**Methode LU :**

**Principe :**

C’est une méthode de décomposition d'une [matrice](https://fr.wikipedia.org/wiki/Matrice_(math%C3%A9matiques)) comme produit de 2 matrices L et U. L est une matrice triangulaire inférieure ayant des 1 sur la diagonale et U une matrice triangulaire supérieure.

Comment construire Let U ?

**Idée** :

La décomposition LU est une forme particulière d'élimination de Gauss Jordan. On transforme la matrice *A* en une matrice triangulaire supérieure *U* en éliminant les éléments sous la [diagonale](http://www.techno-science.net/glossaire-definition/Diagonale.html). Les éliminations se font colonne après colonne, en commençant par la gauche, en multipliant *A* par la gauche avec une matrice triangulaire inférieure.

### Algorithme

Étant donné une matrice de [dimension](http://www.techno-science.net/glossaire-definition/Dimension.html) N \times N

A= (a_{n,n})\;

on définit

A^{(0)} := A\;

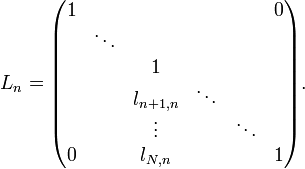
et les itérations se font pour *n* = 1,...,*N-1* de la manière suivante.

Sur *n*ième colonne de *A*(*n-1*), on élimine les éléments sous la diagonale en ajoutant à la *i*ème ligne de cette matrice, la *n*ième ligne multipliée par

l_{i,n} := -\frac{a_{i,n}^{(n-1)}}{a_{n,n}^{(n-1)}}

pour i = n+1,\ldots,N.

Ceci peut être fait en multipliant par la gauche *A*(*n-1*) avec la matrice triangulaire inférieure



A^{(n)} := L_n A^{(n-1)}.\;

Après *N-1* itérations, nous avons éliminé tous les éléments sous la diagonale, par conséquent, nous avons maintenant une matrice triangulaire supérieure *A*(*N-1*).

A = L_{1}^{-1} L_{1} A^{(0)} = L_{1}^{-1} A^{(1)} = L_{1}^{-1} L_{2}^{-1} L_{2} A^{(1)} =  L_{1}^{-1}L_{2}^{-1} A^{(2)} =\ldots = L_{1}^{-1} \ldots L_{N-1}^{-1} A^{(N-1)}.Nous obtenons la décomposition

Notons *U* la matrice triangulaire supérieure *A*(*N-1*) et L=L_{1}^{-1} \ldots L_{N-1}^{-1}. Sachant que l'inverse d'une matrice triangulaire inférieure est aussi une matrice triangulaire inférieure et que le produit de 2 matrices triangulaires inférieures est encore une matrice triangulaire inférieure, *L* est donc une matrice triangulaire inférieure. On obtient *A* = *LU*.

Au vu de l'algorithme, il est nécessaire que a_{n,n}^{(n-1)}\not=0 à chaque itération (voir la définition de *li*,*n*). Si, au cours du calcul, ce cas de figure venait à se produire, il faut intervertir la *n*ième ligne avec une autre pour pouvoir continuer (il est toujours possible de trouver un élément non nul sur la colonne qui pose problème car la matrice est inversible). C'est la raison pour laquelle la décomposition LU s'écrit généralement *P* − 1*A* = *LU*.

### Le cas symétrique

* Si la matrice *A* est une [matrice symétrique](http://www.techno-science.net/?onglet=glossaire&definition=5200), il existe une décomposition dite *factorisation de Crout*

*A* = *L*.*D*.*tL*

où *L* est une matrice triangulaire inférieure dont la diagonale ne comprend que des 1, *tL* est la transposée de *L*, et *D* est une [matrice diagonale](http://www.techno-science.net/?onglet=glossaire&definition=5183).

* Si la matrice *A* est symétrique définie positive, il existe une décomposition plus simple, [donnée](http://www.techno-science.net/glossaire-definition/Donnee.html) par la méthode de Cholesky

*A* = *L*.*tL*

où *L* est une matrice triangulaire inférieure à diagonale positive et *tL* est la transposée de *L*.

**Méthode de Cholesky :**

La méthode de Cholesky ne s’applique qu’aux matrices réelles symétriques dé- finies positives. Elle consiste en une factorisation A = BBt , où B est une matrice triangumaire inférieure, dans le but de ramener la résolution de l’équation linéaire Ax = b à la résolution de deux équations By = b et Bt x = y. Ceci est intéressant lorsqu’on a à résoudre plusieurs systèmes avec la même matrice et différents second membres car la décomposition est effectuée une fois pour toutes.

**Methode QR**

L’idée de la méthode de factorisation QR est encore de se ramener à la résolution d’un système linéaire dont la matrice est triangulaire. Cependant, on ne va pas écrire une matrice A comme le produit de deux matrices triangulaires mais comme le produit d’une matrice triangulaire R et d’une matrice orthogonale unitaire Q, qui est aussi facile à inverser puisque Q−1 = Qt . Pour résoudre le système linéaire

Ax = b. , on effectue : y = Qt b  
 puis, Rx = y.

1. **Programmer la méthode de LU pour la résolution du système linéaire relatif au probléme (1)**

function [ P, L, U ] = LUdecomposition(A)

% Ensures A is n by n

sz = size(A);

if sz(1)~=sz(2)

fprintf('A is not n by n\n');

clear x;

return;

end

n = sz(1);

L = eye(n);

P = eye(n);

U = A;

for i=1:sz(1)

% Row reducing

if U(i,i)==0

maximum = max(abs(U(i:end,1)));

for k=1:n

if maximum == abs(U(k,i))

temp = U(1,:);

U(1,:) = U(k,:);

U(k,:) = temp;

temp = P(:,1);

P(1,:) = P(k,:);

P(k,:) = temp;

end

end

end

if U(i,i)~=1

temp = eye(n);

temp(i,i)=U(i,i);

L = L \* temp;

U(i,:) = U(i,:)/U(i,i); %Ensures the pivotsare 1.

end

if i~=sz(1)

for j=i+1:length(U)

temp = eye(n);

temp(j,i) = U(j,i);

L = L \* temp;

U(j,:) = U(j,:)-U(j,i)\*U(i,:);

end

end

end

P = P';

end