

NOM :
Prénom :

Date :

Faculté des sciences et ingénierie (Toulouse III)
Département de mathématiques – L3 MMESI
Analyse numérique I

Année scolaire
2011-2012

TP n° 3 – Pivot de Gauss et décomposition LU

Aller dans le dossier L3_analyseNumerique et créer le dossier TP03, dans lequel seront placés les fichiers relatifs à ce TP.

1 Découverte de MATLAB (3) : les fonctions

Nous voyons maintenant une nouvelle fonctionnalité : créer de nouvelles commandes, qui s'utiliseront de la même manière que les autres commandes prédéfinies de MATLAB vues dans les deux précédents TP.

Une fonction MATLAB, c'est "comme un script mais en mieux" : on peut donner des entrées et on peut affecter le résultat renvoyé dans une variable (ce qui n'est pas possible avec un script). Une fonction se comporte comme une vraie commande MATLAB. Pour comprendre, il suffit de faire l'analogie avec une fonction mathématique :

- dans $y = f(x)$, x est la variable d'entrée, y est la sortie et f est le nom de la fonction ;
- dans $A = \text{eye}(5)$, 5 est l'entrée, A est la sortie¹ et **eye** est le nom de la fonction.

Une fonction peut s'utiliser partout : en ligne de commande, dans un script et même dans une autre fonction ; à la condition bien sûr que le répertoire de travail soit le dossier où est enregistrée cette fonction !

Exemple 1. Cet exemple décrit deux manières de calculer la somme des éléments d'un vecteur $v \in \mathbb{R}^n$: tel qu'on sait le faire depuis le TP2, avec un script ; tel qu'on sait le faire maintenant, avec une fonction.

1. Avec un script :

```
Définir v
n= longueur de v
S=0
Pour i=1 à n, faire
    S=S+vi
Fin
Afficher S
```

```
v= ...
n=length(v);
S=0;
for i=1:n
    S=S+v(i);
end
S
```

1. Plus précisément : la valeur de la sortie est stockée dans la variable A.

Pour l'utiliser, il faut alors définir un vecteur dans la variable globale v (à l'intérieur du fichier `.m` par exemple) puis revenir en ligne de commande pour lancer le script ; à noter qu'il n'est pas possible d'affecter le résultat du script dans une variable.

2. Avec une fonction :

Nom : somme
Entrée : v (vecteur)
Sortie : S (scalaire)
--> somme des coeff de v

 n = longueur de v
Initialisation de S à 0
Pour $i=1$ à n , faire
 $S=S+v_i$
Fin

```
function S=somme(v)
%Calcule la somme des coeff de v

n=length(v);
S=0;
for i=1:n
    S=S+v(i);
end
```

L'utilisation est intégralement en ligne de commande (inutile de modifier le fichier `.m` lorsque le vecteur change) :

```
somme([1,2,3,4,5])
```

ou

```
v=[1,2,3,4,5];
somme(v)
```

Il est bien sûr possible d'affecter le résultat d'une fonction dans une variable :

```
v=[1,2,3,4,5];
nombre=somme(v)
```

Exercice 1. Créer une fonction `MATLAB` (nommée `produit`) prenant en entrée un vecteur v et renvoyant en sortie le produit P des éléments de v .

--	--

Exercice 2. Nous allons créer, de deux manières, une fonction renvoyant la somme *et* le produit des éléments d'un vecteur $v \in \mathbb{R}^n$.

1. Méthode 1 : avec les fonctions somme et produit précédemment vues. Donner les entrées et les sorties de la fonction suivante :

	<pre>function [S,P]=somProd1(v) %Calcule somme et produit des coeff S=somme(v); P=produit(v);</pre>
--	---

2. Méthode 2 : avec *une seule* boucle parcourant les éléments du vecteur v . Programmer une telle fonction (nommée somProd2) :

--	--

3. Laquelle de ces deux méthodes vous semble la plus efficace ; pourquoi ?

--

4. Indiquer en face de chaque ligne ce que donne les commandes suivantes :

```
1 v = 1:1:5;
2 somProd2(v)
3 [a,b]=somProd2(v)
4 c=somProd2(v);
5 c
```

--

Attention! Les arguments de sortie sont écrits entre crochets dans l'entête de la fonction mais c'est une syntaxe `MATLAB` qui n'a *rien à voir* avec l'usage habituel des crochets (*i.e.* définition de vecteur). Autrement dit, `[S,P]` ne signifie *en aucun cas* "renvoyer le vecteur composé de S et P"; c'est simplement : "renvoyer les deux objets S et P" (ce qui autorise ces objets à avoir des tailles différentes ou même à être de type différent).

2 Pivot de Gauss

Rappels L'algorithme du pivot de Gauss consiste à transformer un système linéaire en un système triangulaire supérieur qui lui est équivalent ; une fois cette transformation faite, il est aisé de le résoudre : il suffit d'utiliser l'algorithme de remontée vu au TP2.

Autrement dit (vision matricielle), l'algorithme du pivot transforme le système $Ax = b$ avec A quelconque, en le système équivalent $Ux = v$ avec U triangulaire supérieure.

Rappelons pour finir l'algorithme, tel qu'il a été présenté dans le cours :

Algorithme du pivot de Gauss

```
n= taille de A
Initialiser U à A
Initialiser v à b
Pour j allant de 1 à n-1, faire
  Pour i allant de j+1 à n, faire
     $v_i = v_i - (u_{ij}/u_{jj})v_j$ 
    Pour k allant de j+1 à n, faire
       $u_{ik} = u_{ik} - (u_{ij}/u_{jj})u_{jk}$ 
    Fin k
     $u_{ij} = 0$ 
  Fin i
Fin j
```

Exercice 3. Le but est de créer une fonction MATLAB réalisant l'algorithme de Gauss.

1. Donner les entrées et les sorties d'une telle fonction (leur nom, leur type, à quoi elles correspondent).

2. Écrire une fonction MATLAB correspondant à cet algorithme.

Exercice 4 (Application). Considérons les valeurs

$$A = \begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 2 \\ 2 & 4 & 6 \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

Que renvoie l'algorithme de Gauss sur ces valeurs ? Que retrouve-t-on (cf. TP2) ?

3 Autour de la décomposition LU

Il s'agit de décomposer une matrice A (qui a les bonnes hypothèses, cf. cours) sous la forme LU , avec L triangulaire inférieure possédant des 1 sur la diagonale et U triangulaire supérieure².

3.1 Décomposition LU par identification des coefficients

Rappels Cette méthode consiste à écrire les coefficients (inconnus) de la matrice LU et de les identifier avec les coefficients (connus) de A ; cela fonctionne bien à condition de procéder dans l'ordre : 1^{re} ligne, puis 1^{re} colonne, puis 2^e ligne, puis 2^e colonne, etc. Les identifications sur les lignes permettent de déduire les coefficients de U ; celles sur les colonnes permettent de trouver L .

L'algorithme correspondant a été vu en TD ; nous le redonnons ici :

Algorithme de décomposition LU n° 1

```
n= taille de A
Initialiser U
Initialiser L
Pour i allant de 1 à n, faire
  Pour j allant de i à n, faire
     $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$ 
  Fin j
  Pour j allant de i+1 à n, faire
     $l_{ji} = (a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki}) / u_{ii}$ 
  Fin j
   $l_{ii} = 1$ 
Fin i
```

2. L pour *lower* et U pour *upper*, en anglais.

Exercice 5 (Décomposition LU n° 1).

1. Donner les entrées et les sorties de cet algorithme.

2. Programmer l'algorithme (sous le nom `decomplU1`).

3.2 Décomposition LU par la méthode du pivot de Gauss

Rappels L'algorithme du pivot de Gauss effectue les manipulations suivantes sur les lignes :

$$L_i \leftarrow L_i + \lambda L_j. \tag{1}$$

En notant $T_{ij}(\lambda)$ la matrice

$$T_{ij}(\lambda) = \begin{matrix} & & j & & \\ i & \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & \lambda & & \ddots & \\ & & & & 1 \end{pmatrix} & & & \end{matrix},$$

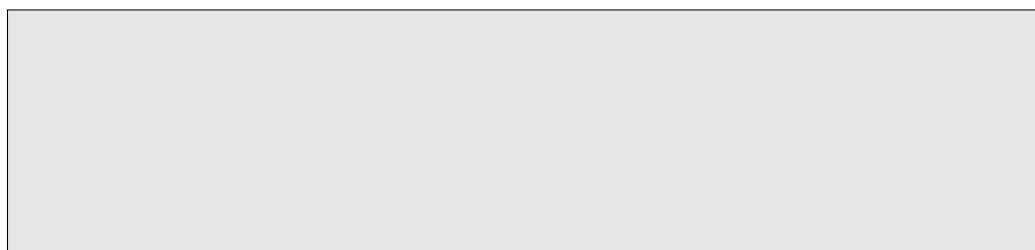
alors (1) revient à remplacer A par la matrice $T_{ij}(\lambda)A$ (produit matriciel à gauche par une matrice de transvection).

3.2.1 Manipulation sur les lignes et les colonnes

Exercice 6 (Mise en place d'algorithmes intermédiaires).

1. Programmer une fonction qui, étant donné une matrice M , effectue l'opération (1) sur M :

```
function M=transvecLigne(M,i,j,lambda)
```



2. Programmer son analogue sur les colonnes

```
function M=transvecCol(M,i,j,lambda)
```

qui réalise l'opération

$$C_i \leftarrow C_i + \lambda C_j. \quad (2)$$

(Prendre garde au fait que multiplier à droite par $T_{ij}(\lambda)$ équivaut à la manipulation $C_j \leftarrow C_j + \lambda C_i$.)

3.2.2 Le pivot de Gauss contre-attaque

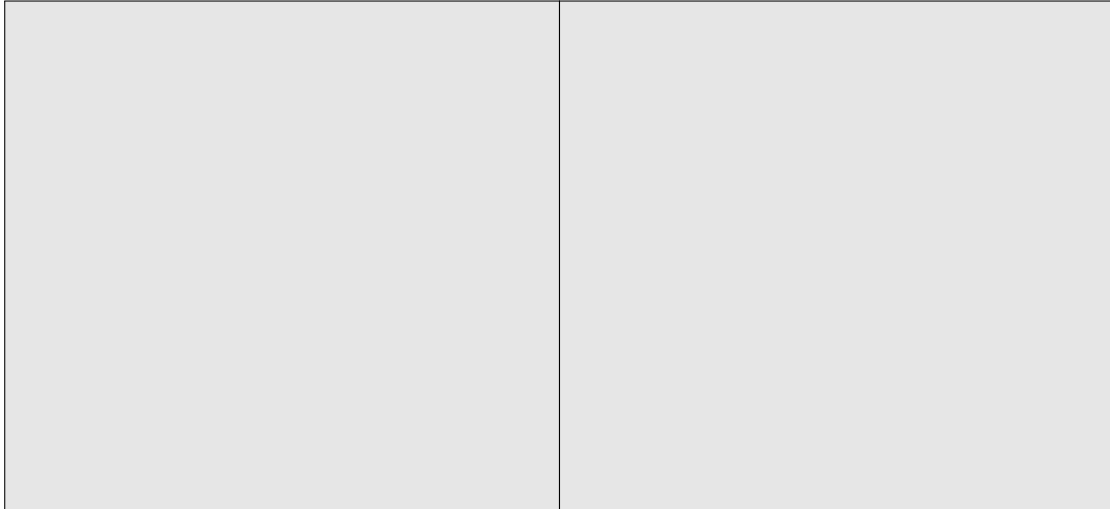
Il s'agit de programmer l'algorithme du pivot de Gauss, sous une autre version que celle vue en section 2 et en ne se préoccupant que de la matrice A .

Exercice 7. Programmer l'algorithme suivant

```
Pivot de Gauss (version matricielle)
-----
n=taille de A
Initialiser U à A
Pour j allant de 1 à n-1, faire
  pivot=ujj
  Pour i allant de j+1 à n, faire
    λ=uij/pivot
    pour U : Li ← Li - λLj
  Fin i
Fin j
```

avec l'entête

function U=GaussMatriciel(A)



3.2.3 Le retour de la décomposition LU

Nous allons mettre en œuvre une autre méthode pour obtenir la décomposition LU, qui s'inspire de la version matricielle du pivot de Gauss.

Rappels Dans l'algorithme de Gauss matriciel ci-dessus, on effectue une succession de multiplications à gauche sur A , par des matrices de transvections. Notons Q_1, \dots, Q_p ces matrices successives ; alors l'algorithme de Gauss matriciel consiste à remplacer A par

$$\underbrace{Q_p Q_{p-1} \dots Q_1 A}_{=Q}$$

que l'on note U . Ainsi, comme Q est inversible (cf. cours), on obtient

$$A = Q^{-1}U.$$

Or Q^{-1} est triangulaire inférieure à diagonale unité (cf. cours) : c'est donc la matrice L cherchée. Au final, on a donc trouvé la décomposition LU de A :

$$U = QA,$$

$$L = Q^{-1}.$$

Remarques complémentaires :



- les Q_k sont de la forme $T_{ij}(-\lambda)$;
- $Q^{-1} = Q_1^{-1} Q_2^{-1} \dots Q_p^{-1}$;
- les Q_k^{-1} sont de la forme $T_{ij}(\lambda)$ car $(T_{ij}(-\lambda))^{-1} = T_{ij}(\lambda)$;
- multiplier à droite par $T_{ij}(\lambda)$ revient à faire $C_j \leftarrow C_j + \lambda C_i$.

Exercice 8 (Décomposition LU n° 2). En observant que

- $Q^{-1} = Q_1^{-1} Q_2^{-1} \dots Q_p^{-1}$ est égal à $I_n Q_1^{-1} Q_2^{-1} \dots Q_p^{-1}$ (où I_n désigne la matrice identité de taille n);
- l'algorithme de Gauss matriciel consiste à faire une succession de multiplications à gauche sur A , pour la transformer en une matrice triangulaire inférieure, et en utilisant les rappels ci-dessus, proposer un algorithme qui effectue la décomposition LU de A en faisant les deux choses suivantes :
- une succession de multiplications à gauche sur A , pour la transformer en une matrice triangulaire supérieure (notée U);
- une succession de multiplications à droite sur l'identité, pour la transformer en une matrice triangulaire inférieure à diagonale unité (notée L).

function [L,U]=decompLU2(A)

--	--

 *N'oubliez pas de rendre le TP,* 
avec vos nom et prénom