DOSSIER MINF0402

TP n°1

TONNELLE NATHAN & MACZYTA NICOLAS
2019-2020

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Table des matières

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Avant-propos

Avant chaque début de script, nous effaçons la table de variable de scilab, puis chargeons les fonctions.
Pour cela, nous avons la commande **clear** qui permet de vider la table de variable et la commande **exec()** qui permet d'exécuter un script, des fonctions, etc. ici nous utilisons **exec()** sous la forme : (variable + nom-du-fichier,-1) , car nous devons récupérer dans un premier temps le chemin absolue où se situe le fichier du script puis rajouter à se chemin le fichier que l'on cherche à exécuter. Le **-1** signifiant l'option -1 fait que rien n'est affiché dans la console (source : https://help.scilab.org/docs/5.5.2/fr_FR/exec.html).

```
3 clear
4 pathname=get_absolute_file_path("exo4.sce")
5 exec(pathname+'\fonctions.sci',-1);
```

Nous utilisons également l'instruction **disp()**, qui permet l'affichage dans la console de textes et de variables, nous l'utilisons sous la forme : (variable, "texte"), ce qui permet d'avoir l'affichage suivant et non le second affichage sans le nom de la variable (qui se fait avec la forme (variable)).

```
Asup=

  32.   27.   44.  -11.   7.
  0.    20.   29.  -28.  -24.
  0.    0.   -11.  -1.   -10.
  0.    0.    0.    17.  -48.
  0.    0.    0.    0.   -37.

 -46.   0.    0.    0.    0.
  23.  -7.    0.    0.    0.
 -21.  -20.  -27.   0.    0.
 -30.   43.  -46.   45.   0.
  22.   6.   -16.  -19.  -27.
```

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Exercice 4

Présentation du script

Nous commençons le script par une initialisation de chaque variable et l'affichage de celles-ci.

```
//variables :
//Asup= matrice triangulaire supérieure inversible (n,n)
//Ainf= matrice triangulaire inférieure inversible (n,n)
//b= vecteur colonne à n composantes
n=5;
b=round(10*rand(n,1));
Asup=triu(round(100*rand(n,n)-50));
Ainf=tril(round(100*rand(n,n)-50));

//affichage des variables :
disp(n,"n=");
disp(Asup,"Asup=");
disp(Ainf,"Ainf=");
disp(b,"b=");
```

Comme vous pouvez le voir, le script est bien commenté pour présenter ce que sont chaque variable. Ainsi nous trouvons que Asup est une matrice triangulaire supérieure de type (n,n), tout comme Ainf qui est une matrice triangulaire inférieure, b est un vecteur de n éléments.
Ces matrices, et vecteur sont générés de manière aléatoire avec les instructions **round()**, **rand()**, pour les nombres et **triu()** pour la matrice triangulaire supérieure et **tril()** pour la matrice triangulaire inférieure.
L'instruction **rand()** permet de générer un nombre de chiffre suivant les arguments qu'il contient.
L'instruction **round()** permet qu'en à lui de générer des matrices avec des nombres aléatoires.
Les instructions **tril** et **triu** permettent de faire des matrices générées dans leurs arguments, des matrices triangulaires, ou inférieures ou supérieures.

```
/* TEST fonction
//FONCTION RESOUINF:
//Ainf=matrice triangulaire inférieure inversible
//b=vecteur colonne à n composantes
//Ax=b => lorsque A est triangulaire inférieure
function X=RESOUINF(Ainf,b,n)
    X=zeros(b);
    X(1,1)=b(1,1)/Ainf(1,1);
    for i=2:n
        aux=0;
        for k=1:i-1
            aux=aux+Ainf(i,k)*X(k);
        end
        X(i)=(b(i)-aux)/Ainf(i,i);
    end
endfunction


//FONCTION RESOUSUP:
//Asup=matrice triangulaire supérieure inversible
//b=vecteur colonne à n composantes
//Ax=b => lorsque A triangulaire supérieur
function Y=RESOUSUP(Asup,b,n)
    Y=zeros(b);

    Y(n,1)=b(n,1)/Asup(n,n);
    for i=n-1:-1:1
        aux=0;
        for k=i+1:n
            aux=aux+Asup(i,k)*Y(k);
        end
        Y(i)=(b(i)-aux)/Asup(i,i);
    end
endfunction
*/
```

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Nous avons fait le choix de laisser les fonctions apparentes et donc ne les effacer dans le script car nous avons trouver que cela était très pratique pour en faire les modifications nécessaires, comme nous travaillons en binôme.
Dans ce script, il y a 2 fonctions qui ont chacune leurs utilités propres, la première, RESOUINF doit etre utilisé avec une matrice triangulaire inférieure et la seconde RESOUSUP qu'avec des matrices triangulaires supérieures.
Pour faire une fonction, nous utilisons la syntaxe suivante :
Les fonctions sont délimitées par les instructions **function** et **endfunction**.

function [variable-de-sortie]=nomFonction(arguments)
Instructions
endFonction

Pour ce script, nous avons utiliser les instructions de calcules basique ('+','-','=','*','/'), comme dans les fonctions suivantes, dans les autres exercices.
La fonction RESOUINF permet de résoudre, comme son nom l'indique, une matrice inférieure, via 2 boucles pour parcourir la matrice en entière.
La fonction RESOUSUP permet de résoudre, comme son nom l'indique également, une matrice supérieure, via le même système que pour la fonction RESOUINF.
Nous obtenons ainsi, les résultats suivants, avec les instructions :

```
58 //affichage des résultats :
59 res=RESOUSUP(Asup,b,n);
60 disp(res,"res Asup =");
61 res=RESOUINF(Ainf,b,n);
62 disp(res,"res Ainf =");
```

```
 res Asup =

  -3.2017438
  -0.6137985
  -0.9530864
   1.8989899
   1.3333333

 res Ainf =

   0.
  -0.14
   0.2963636
   0.1536364
  -0.6349697
```

Avec les données :

```
Asup=

   9.  -21.   15.   11.    7.
   0.   36.   49.   35.    7.
   0.    0.  -45.  -44.   32.
   0.    0.    0.   33.  -44.
   0.    0.    0.    0.    6.

Ainf=

 -38.    0.    0.    0.    0.
  23.  -50.    0.    0.    0.
 -23.    9.   11.    0.    0.
   5.  -19.   18.  -26.    0.
  49.  -24.  -17.    1.  -15.

b=

   0.
   7.
   2.
   4.
   8.
```

Nous avons vérifié plusieurs fois les résultats de nos script et des fonctions pour produire de bons résultats à chaque fois.

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Exercice 5

Présentation du script

Nous commençons le script par une initialisation de chaque variable et l'affichage de chacune d'elles.

```
7  //variables :
8  //A= matrice triangulaire supérieure inversible (n,n)
9  //b= vecteur colonne à n composantes
10 n=4;
11 b=round(10*rand(n,1));
12 A=triu(round(100*rand(n,n)-50));
13
14 //affichage des variables :
15 disp(n,"n=");
16 disp(A,"A=");
17 disp(b,"b=");
```

Comme montré sur l'image, sur le script il y a des commentaires montrant à quoi correspond chaque variable.
Le n correspond au nombre de lignes et de colonnes de la matrice A, n correspond également au nombre de chiffre du vecteur colonne b.
En début de programme nous affichons les variables dans la console pour savoir avec quels matrice et vecteur colonne nous utilisons, car, hormis n, A et b sont générées aléatoirement, avec les instructions **round()** permet de générer des chiffres aléatoires, **rand()** permet de générer une matrice ou vecteur selon les 2 arguments qu'il prend, ainsi que **triu()** qui permet de faire de la matrice qu'il prend en argument d'en faire une matrice triangulaire supérieure.

```
19 /* TEST fonction
20 //FONCTION REDUC:
21 //A=matrice triangulaire suppérieure
22 //b=vecteur colonne à n composantes
23 //Ax=b => systeme triangulaire supérieur obtenu après la réduction de Gauss
1  function [A,b]=REDUC(A,b,n)
2      for k=1:n-1
3          for i=k+1:n
4              Aux=A(i,k)/A(k,k);
5              for j=k+1:n
6                  A(i,j)=A(i,j)-Aux*A(k,j);
7              end
8              A(i,k)=0;
9              b(i)=b(i)-Aux*b(k);
10         end
11     end
12 endfunction
36
37 //FONCTION RESOUSUP:
38 //A=matrice triangulaire suppérieure de composant (n,n)
39 //b=vecteur colonne à n composantes
40 //Ax=b => systeme triangulaire supérieur obtenu après la réduction de Gauss
1  function Y=RESOUSUP(A,b,n)
2      Y=zeros(b);
3
4      Y(n,1)=b(n,1)/A(n,n);
5      for i=n-1:-1:1
6          aux=0;
7          for k=i+1:n
8              aux=aux+A(i,k)*Y(k);
9          end
10         Y(i)=(b(i)-aux)/A(i,i);
11     end
12 endfunction
54 //FONCTION GAUSS
55 //A = matrice triangulaire supérieure de composant (n,n)
56 //b = vecteur colonne de n composant
57 //résolution de la matrice A par la méthode de GAUSS
1  function X=GAUSS(A,b,n)
2      X=zeros(n,1);
3      [A,b]=REDUC(A,b,n);
4      X=RESOUSUP(A,b,n);
5  endfunction
63 */
```

TONNELLE Nathan
MACZYTA Nicolas MINF0402

Voici les fonctions que nous avons utiliser pour construire le script de cet exercice, nous avons choisi de les laisser délibérément car cela montre ce quelle font meme si elles ne sont pas utilisées car commentées. Nous les avons laissés également pour pouvoir les modifier le plus facilement possible comme nous sommes 2 à faire ce rapport, ainsi que ces exercices.

Ainsi nous avons combiner les opérations et les instructions, comme **for**, **zeros**, pour construire nos fonctions et donc le script qui en découle.

La fonction REDUC permet d'avoir le résultat de la réduction de Gauss, il faut donc parcourir les éléments de la matrice un par un en effectuant un calcul sur ceux-ci.

La fonction RESOUSUP est la même que **l'exercice 4** donc nous n'allons ici reprendre son explication.

La fonction GAUSS fait fonctionner les deux fonctions citées juste au-dessus, pour obtenir ainsi la résolution de la matrice par réduction de Gauss.

Nous obtenons ainsi le résultat, via le même système d'affichage que les données, au début du script, suivant :

```
n=

   4.

A=

   17.    38.    23.   -27.
   0.    -43.   -30.   -28.
   0.     0.     4.     38.
   0.     0.     0.     15.

b=

   2.
   8.
   0.
   3.

res =

   0.7497948
   1.0093023
  -1.9
   0.2
```

Au final, dans la console, nous voyons seulement les données de base ainsi que le résultat de la réduction de Gauss et sa résolution.

TONNELLE Nathan
MACZYTA Nicolas

Exercice 6

Présentation du script

Comme pour les exercices précédents, nous commençons le script par l'initialisation des variables de manière aléatoire, avec les mêmes instructions, à savoir **roud()** et **rand()**, dont leurs fonctionnement a été décrit plus haut.

```
7  //variables :
8  //A= vecteur des variables A des equations 2 a n
9  //B= vecteur des variables B des equations 1 a n
10 //C= vecteur des variables C des equations 1 a n-1
11 //D= Vecteur contenant les membres droits des equations
12 //n= nombre d'équations
13 n=6;
14 A=round(10*rand(n,1));
15 B=round(10*rand(n,1));
16 C=round(10*rand(n,1));
17 D=round(10*rand(n,1));
18
19 //affichage des variables :
20 disp(n,"n=");
21 disp(A,"A=");
22 disp(B,"B=");
23 disp(C,"C=");
24 disp(D,"D=");
```

Nous n'avons pas changé nos habitudes pour cet exercice, donc nous retrouvons les variables, leurs significations, ainsi que leurs affichages.
Les variables A, B, C et D sont des vecteurs, ils représentent les équations 2 à n, 1 à n, 1 à n-1 et les membres droits des équations respectivement.

```
26 /* TEST fonction
27 //FONCTION RESOUTRI:
28 //A= vecteur des variables A des equations 2 a n
29 //B= vecteur des variables B des equations 1 a n
30 //C= vecteur des variables C des equations 1 a n-1
31 //D= Vecteur contenant les membres droits des equations
32 //n= nombre d'équations
33 //x= vecteur des solutions
 1 function x=RESOUTRI(A,B,C,D,n)
 2     e(1)=C(1)/B(1);
 3     f(1)=D(1)/B(1);
 4     for i=2:n-1
 5         aux=A(i)*e(i-1)+B(i);
 6         e(i)= -C(i)/aux;
 7         f(i)=D(i)-A(i)*f(i-1)/aux;
 8     end
 9     f(n)=D(n)-A(n)*f(n-1);
10     x(n)=f(n);
11     for i=n-1:1
12         x(i)=e(i)*x(i+1)+f(i);
13     end
14     
15 endfunction
49 
50 //FONCTION PRODUITMAT:
51 //A= vecteur des variables A des equations 2 a n
52 //B= vecteur des variables B des equations 1 a n
53 //C= vecteur des variables C des equations 1 a n-1
54 //D= Vecteur contenant les membres droits des equations
55 //n= nombre d'équations
56 //R= résultat du produit matriciel
 1 function R=PRODUITMAT(A,B,C,D,n)
 2     R(1)=B(1)*D(1)+C(1)*D(2);
 3     for i=2:n-1
 4         R(i)=A(i)*D(i-1)+B(i)*D(i)+C(i)*D(i+1);
 5     end
 6     R(n)=A(n)*D(n-1)+B(n)*D(n);
 7 endfunction
```

Cette image vous montre les fonctions que nous avons écrit dans le script. Comme dit précédemment, nous avons choisi de laisser les fonctions dans les scripts pour une raison de praticité.
Dans les fonctions de ce script, nous avons utilisé les instructions basics de calculs et d'attribution.
La fonction RESOUTRI permet de résoudre le système d'équation, pour commencer, nous avons le traitement de la première ligne, puis le traitement des lignes 2 à n-1 pour échelonner la matrice et enfin nous remontons depuis la dernière ligne en résolvant au fur et à mesure le système échelonné.
La fonction PRODUITMAT permet quant à elle d'avoir le résultat du produit matriciel avec 3 phases de traitement : la première ligne, puis les lignes 2 à n-1 et pour finir, la ligne n.

Comme avec les autres scripts, voici les données et les résultats de ce script :

```
n=

   6.

A=

   7.
   5.
   3.
   7.
   1.
   4.

B=

   7.
   9.
   2.
   4.
   10.
   5.

C=

   5.
   6.
   6.
   5.
   8.
   8.

D=

   10.
   8.
   4.
   2.
   9.
   1.
```

```
x=

   0.
   0.
   0.
   0.
   0.
 -35.606873

res=

   0.
   0.
   0.
   0.
 -284.85498
 -178.03437
```

Et voici ce que nous affiche le script dans la console.

Conclusion

Nous avons essayé de faire au mieux pour faire ce TP ainsi que ce Rapport dans les conditions et avec les instructions que vous nous avez envoyé.
Nous vous remercions de nous avoir inculqué les bases de SCILAB et son fonctionnement.