Feuille de TP N°02 – Résolution des Systèmes Triangulaires (Solution)

Exercice 01

1. Ecrire une fonction Matlab « est_triangulaire_supérieur » qui prend comme argument une matrice *A* et qui retourne *1* si la matrice *A* est une matrice triangulaire supérieure, et *0* sinon (On dit que la matrice *A* est triangulaire supérieure, si et seulement si tous les éléments en dessous de sa diagonale sont nuls). Une matrice diagonale est aussi considérée comme matrice triangulaire supérieure.

```
% Solution 01 - Boucles
function b = est_triangulaire_superieur(A)
    [m n] = size(A);
    if m == n
        b = 1;
        for i = 2:m
            for j = 1:i-1
                if A(i, j) ~= 0
                    b = 0;
                end
            end
        end
    else
        b = 0;
        display('La matrice n''est pas carree');
    end
end

% Solution 02 - Opérations Vectorielles
function b = est_triangulaire_superieur(A)
    [m n] = size(A);
    if m == n
        b = isequal(triu(A),A)
    else
        b = 0;
        display('La matrice n''est pas carree');
    end
end
```

2. Ecrire une fonction Matlab « est_triangulaire_inférieur » qui prend comme argument une matrice *A* et qui retourne *1* si la matrice *A* est une matrice triangulaire inférieure, et *0* sinon (On dit que la matrice *A* est triangulaire inférieure, si et seulement si tous les

éléments en dessus de sa diagonale sont nuls). Une matrice diagonale est aussi considérée comme matrice triangulaire inférieure.

```
% Solution 01 - Boucles
function b = est_triangulaire_inferieur(A)
    [m n] = size(A);
    if m == n
        b = 1;
        for i = 1:m
            for j = i+1:n
                if A(i, j) ~= 0
                    b = 0;
                end
            end
        end
    else
        b = 0;
        display('La matrice n''est pas carree');
    end
end

% Solution 02 - Opérations Vectorielles
function b = est_triangulaire_inferieur(A)
    [m n] = size(A);
    if m == n
        b = isequal(tril(A), A)
    else
        b = 0;
        display('La matrice n''est pas carree');
    end
end
```

3. Ecrire une fonction Matlab « type_matrice » qui prend comme argument une matrice *A* et qui retourne l'un des résultats suivants :
   - 0 si la matrice *A* n'est pas triangulaire.
   - 1 si la matrice *A* est triangulaire supérieure.
   - 2 si la matrice *A* est triangulaire inférieure.
   - 3 si la matrice *A* est une matrice diagonale.

   Tester la fonction « type_matrice » sur une matrice non triangulaire, une matrice triangulaire supérieure, une matrice triangulaire inférieure, et une matrice diagonale de votre choix et vérifier si elle retourne le bon résultat dans chaque situation.
   Indication : pour vérifier qu'une matrice est diagonale, il faut tester si elle est triangulaire supérieure et inférieure au même temps.

```
function t = type_matrice(M)
    if est_triangulaire_inferieur(M) &&
                est_triangulaire_superieur(M)
        t = 3;
    elseif est_triangulaire_inferieur(M)
        t = 2;
    elseif est_triangulaire_superieur(M)
        t = 1;
    else
        t = 0;
    end
end
```

Exercice 02

1. Ecrire une fonction Matlab « determinant_triangulaire » qui prend comme argument une matrice *A* et qui calcule et retourne le déterminant de *A* si elle est triangulaire (ou diagonale), sinon la fonction doit afficher le message suivant : "La matrice A n'est pas triangulaire".
    - Tester votre fonction sur une matrice triangulaire de votre choix, et comparer le résultat obtenu par votre fonction à celui obtenu par la fonction Matlab prédéfinie « det » pour vérifier si elle retourne le bon résultat.

```
Solution 01 - Boucle
function d = determinant_triangulaire(A)
    [m n] = size(A);
    if m == n
        if type_matrice(A) > 0
            d = 1;
            for i = 1:m
                d = d * A(i, i);
            end
        else
            display('La matrice non triangulaire');
        end
    else
        display('La matrice n''est pas carree');
    end
end

Solution 02 - Opérations Vectorielles
function d = determinant_triangulaire(A)
    [m n] = size(A);
    if m == n
        if type_matrice(A) > 0
```

```
            d = prod(diag(A));
        else
            display('La matrice non triangulaire');
        end
    else
        display('La matrice n''est pas carree');
    end
end
```

2. Ecrire une fonction Matlab « inversible » qui prend comme argument une matrice *A* et qui retourne *1* si *A* est inversible, et *0* sinon.
   Indication : utiliser les fonctions que déclarées dans l'exercice 01.

```
function inv = triangulaire_inversible(A)
    [m n] = size(A);
    if m == n
        if type_matrice(A) > 0
            inv = determinant_triangulaire(A) == 0;
        else
            inv = 0;
            display('La matrice non triangulaire');
        end
    else
        inv = 0;
        display('La matrice n''est pas carree');
    end
end
function x = resoudre_triangulaire_superieur(A, b)
    n = size(A, 1);
    x(n) = b(n) / A(n, n);
    for i = n-1:-1:1
        s = b(i);
        for j = i+1:n
            s = s - A(i, j) * x(j);
        end
        x(i) = s / A(i, i);
    end
end
```

Exercice 03

1. Ecrire une fonction Matlab « résoudre_triangulaire_supérieur » qui prend comme argument une matrice triangulaire supérieure de coefficients *A* et un vecteur de second membre *b* et qui résout le système triangulaire supérieur $Ax = b$.

```
function x = resoudre_triangulaire_superieur(A, b)
    n = size(A, 1);
    x = zeros(n, 1);
    x(n) = b(n) / A(n, n);
    for i = n-1:-1:1
        % Opération vectorielle (commentée)
        % x(i) = (b(i) - x(i+1:n)*A(i,i+1:n)) / A(i, i);
        % Ou utiliser les boucles
        s = b(i);
        for j = i+1:n
            s = s - A(i, j) * x(j);
        end
        x(i) = s / A(i, i);
    end
end
```

2. Ecrire une fonction Matlab « résoudre_triangulaire_inférieur » qui prend comme argument une matrice triangulaire inférieure de coefficients A et un vecteur de second membre b et qui résout le système triangulaire inférieur $Ax = b$.

```
function x = resoudre_triangulaire_inferieur(A, b)
    n = size(A, 1);
    x = zeros(n, 1);
    x(1) = b(1) / A(1, 1);
    for i = 2:n
        % Opération vectorielle (commentée)
        % x(i) = (b(i) - x(1:i-1)*A(i,1:i-1)) / A(i, i);
        % Ou utiliser les boucles
        s = b(i);
        for j = 1:i-1
            s = s - A(i, j) * x(j);
        end
        x(i) = s / A(i, i);
    end
end
```

3. Ecrire un script Matlab permettant à l'utilisateur de :
    - Saisir n la taille d'un système linéaire à résoudre.
    - Saisir les éléments de la matrice A et du second membre b.
    - Si la matrice A est triangulaire (ou diagonale) et inversible, résoudre le système triangulaire $Ax = b$ selon la nature de la matrice A (triangulaire supérieure ou inférieure) et afficher le résultat. Sinon, le script doit afficher un des messages suivants :

i. « La matrice *A* n'est pas triangulaire » : si la matrice *A* n'est pas triangulaire.
ii. « La matrice *A* n'est pas inversible » : si la matrice *A* est triangulaire mais pas inversible.

- Tester le script sur une matrice *A* et un vecteur *b* de votre choix, et comparer le résultat à celui obtenu par l'instruction Matlab « x = A\b » pour vérifier que votre code est juste.

Indication : utiliser les fonctions définies dans les exercices 01 et 02.

```
n = input('Taille du systeme lineaire a resoudre : ');
display('Matrice de coefficients A');
A = zeros(n, n);
for i = 1:n
    for j = 1:n
        A(i, j) = input(['A(', num2str(i), ', ', num2str(j),
') : ']);
    end
end
if type_matrice(A) == 0
    display('Matrice non triangulaire');
elseif inversible(A) == 0
    display('Matrice non inversible');
else
    display('Second membre b');
    b = zeros(n, 1);
    for i = 1:n
        b(i) = input(['b(', num2str(i), ') : ']);
    end
    if est_triangulaire_superieur(A)
        x = resoudre_triangulaire_superieure(A, b);
    else % ça marche même si la matrice est diagonale
        x = resoudre_triangulaire_inferieure(A, b);
    end
    display(x);
end
```