Haamdi Alaa / Channouf Hamdi
Info Groupe C
TP Analyse Numérique Linéaire

Objectives :

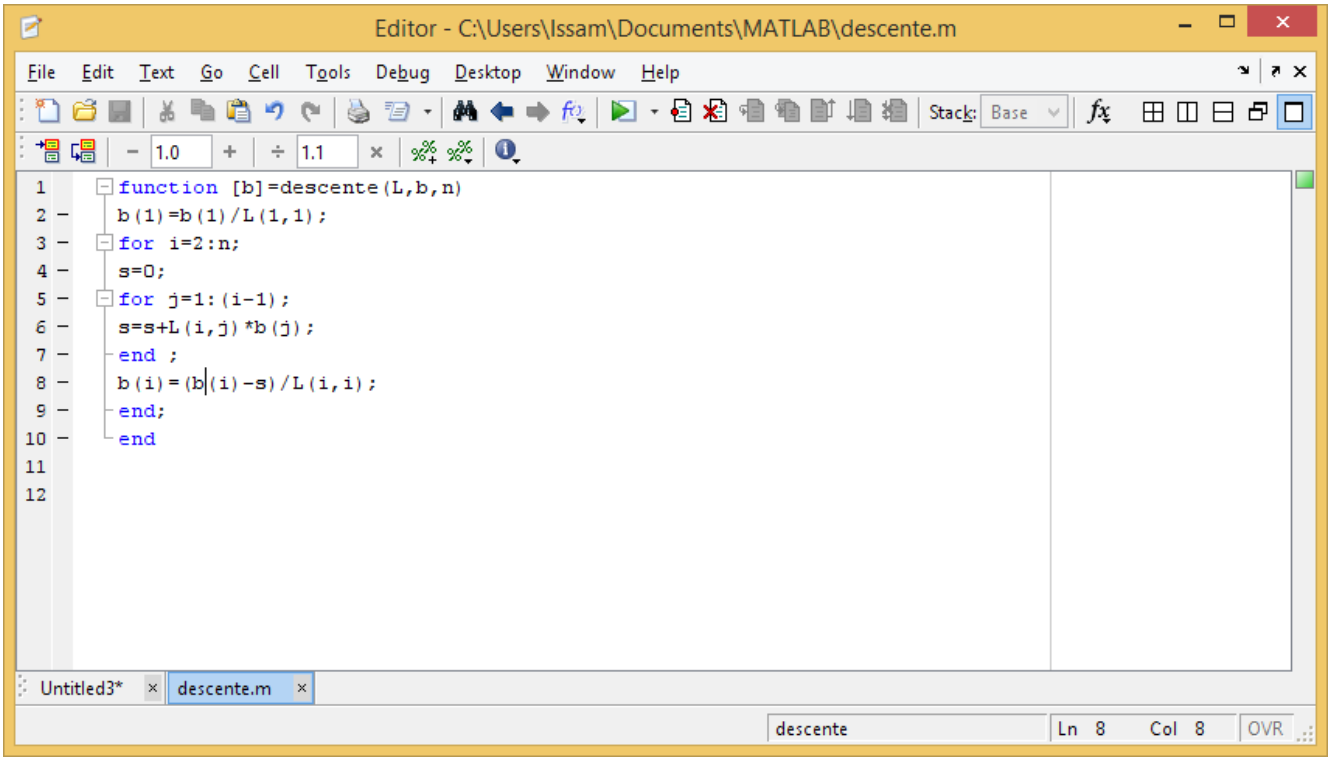
Factorisation LU à l'aide des applications .

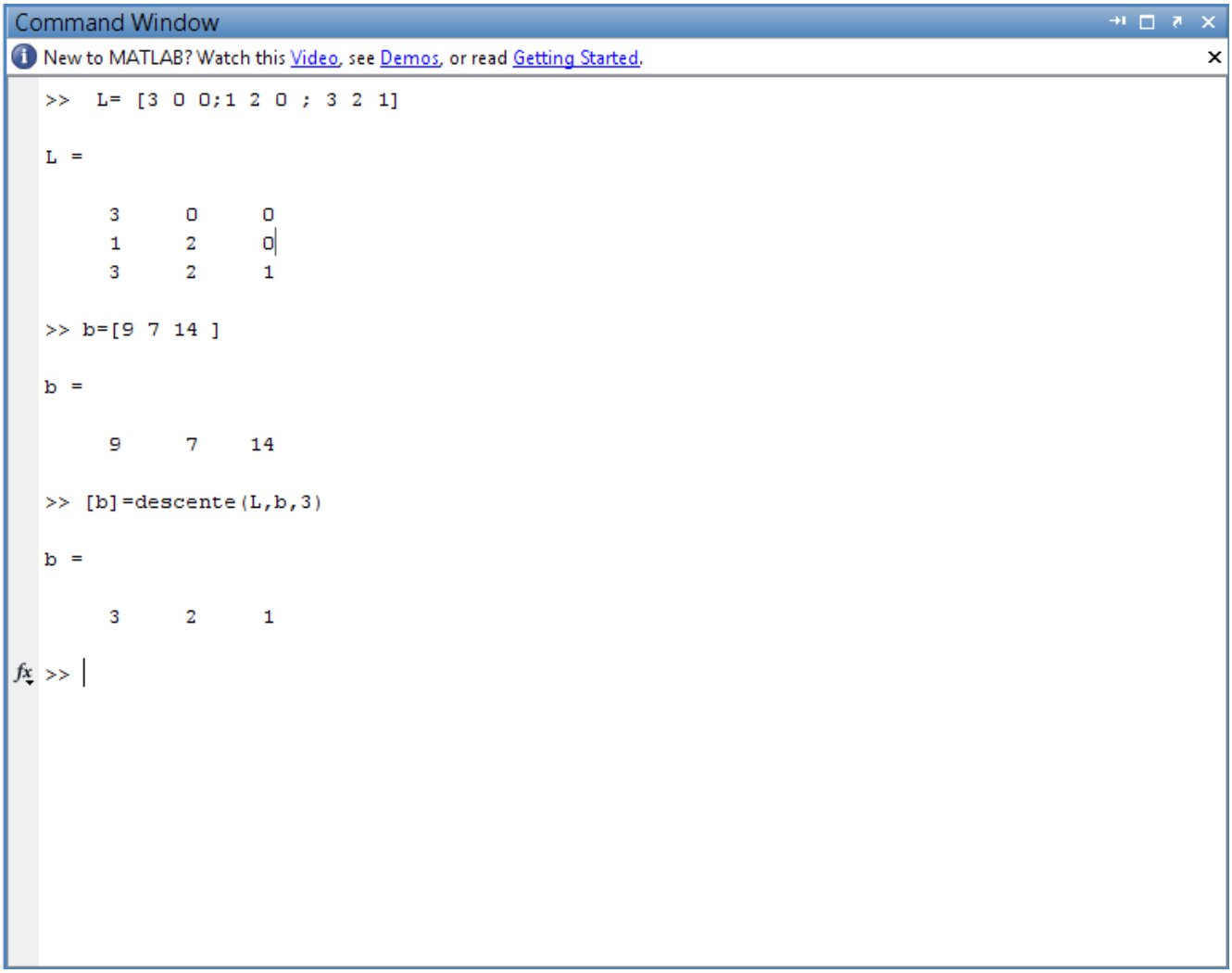
I-Factorisation LU :

1) La fonction **descente** résout un système triangulaire inférieure (Lx = b) :

Algorithme de la fonction *descente(L,b,n)* :

```
>> function [b]=descente(L,b,n)
b(1)=b(1)/L(1,1);
for i=2:n;
s=0;
for j=1:(i-1);
s=s+L(i,j)*b(j);
end ;
b(i)=(b(i)-s)/L(i,i);
end;
end
```

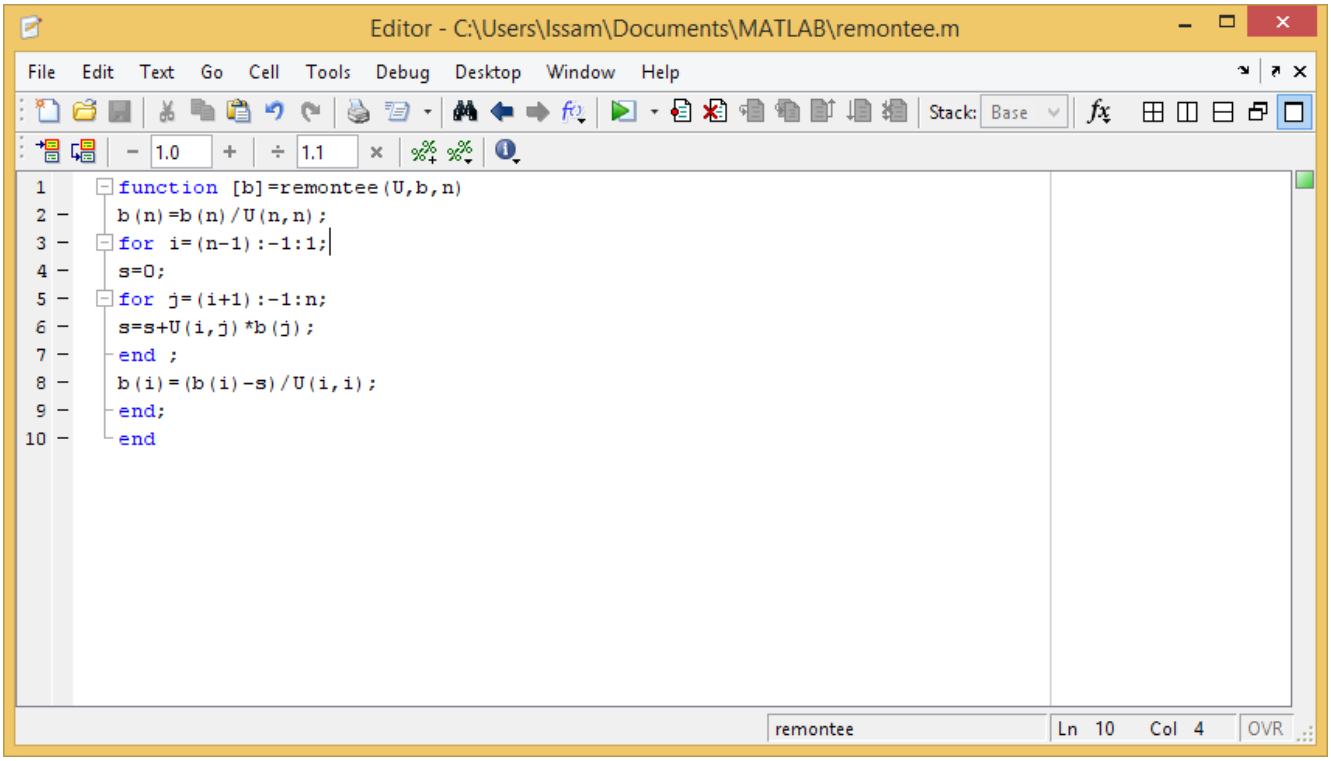


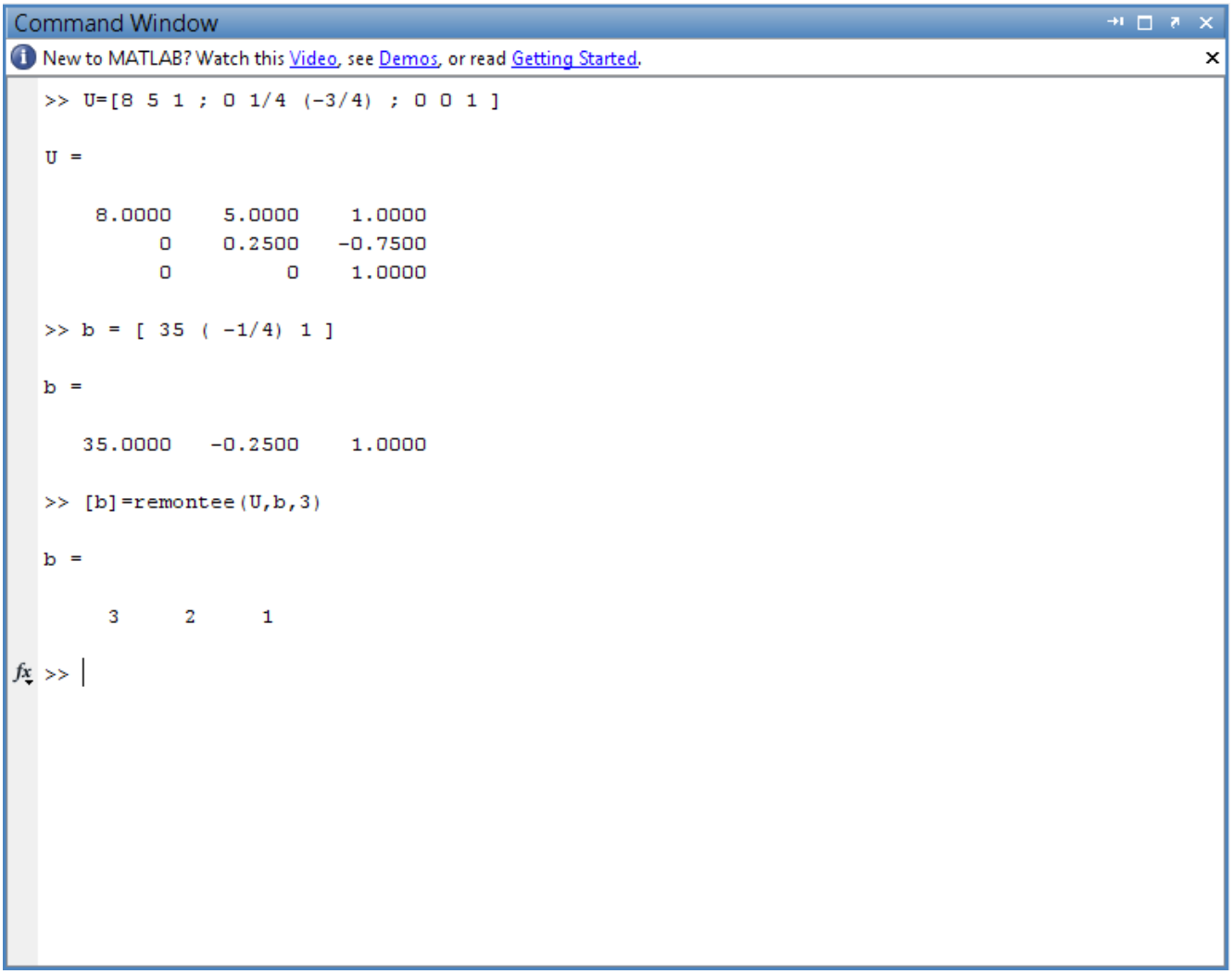

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> L= [3 0 0;1 2 0 ; 3 2 1]
L =
3 0 0
1 2 0
3 2 1
>> b=[9 7 14]
b =
9 7 14
>> [b]=descente(L,b,3)
b =
3 2 1
>>

2) La fonction **remontee** résout un système triangulaire inférieure (Ux = b) :

Algorithme de la fonction *remontee(U,b,n)* :

```
>> function [b]=remontee(U,b,n)
b(n)=b(n)/U(n,n);
for i=(n-1):(-1):1;
s=0;
for j=(i+1):n;
s=s+U(i,j)*b(j);
end ;
b(i)=(b(i)-s)/U(i,i);
end;
end
```




Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> U=[8 5 1 ; 0 1/4 (-3/4) ; 0 0 1]
U =
8.0000 5.0000 1.0000
0 0.2500 -0.7500
0 0 1.0000
>> b = [35 (-1/4) 1]
b =
35.0000 -0.2500 1.0000
>> [b]=remontee(U,b,3)
b =
3 2 1
>>

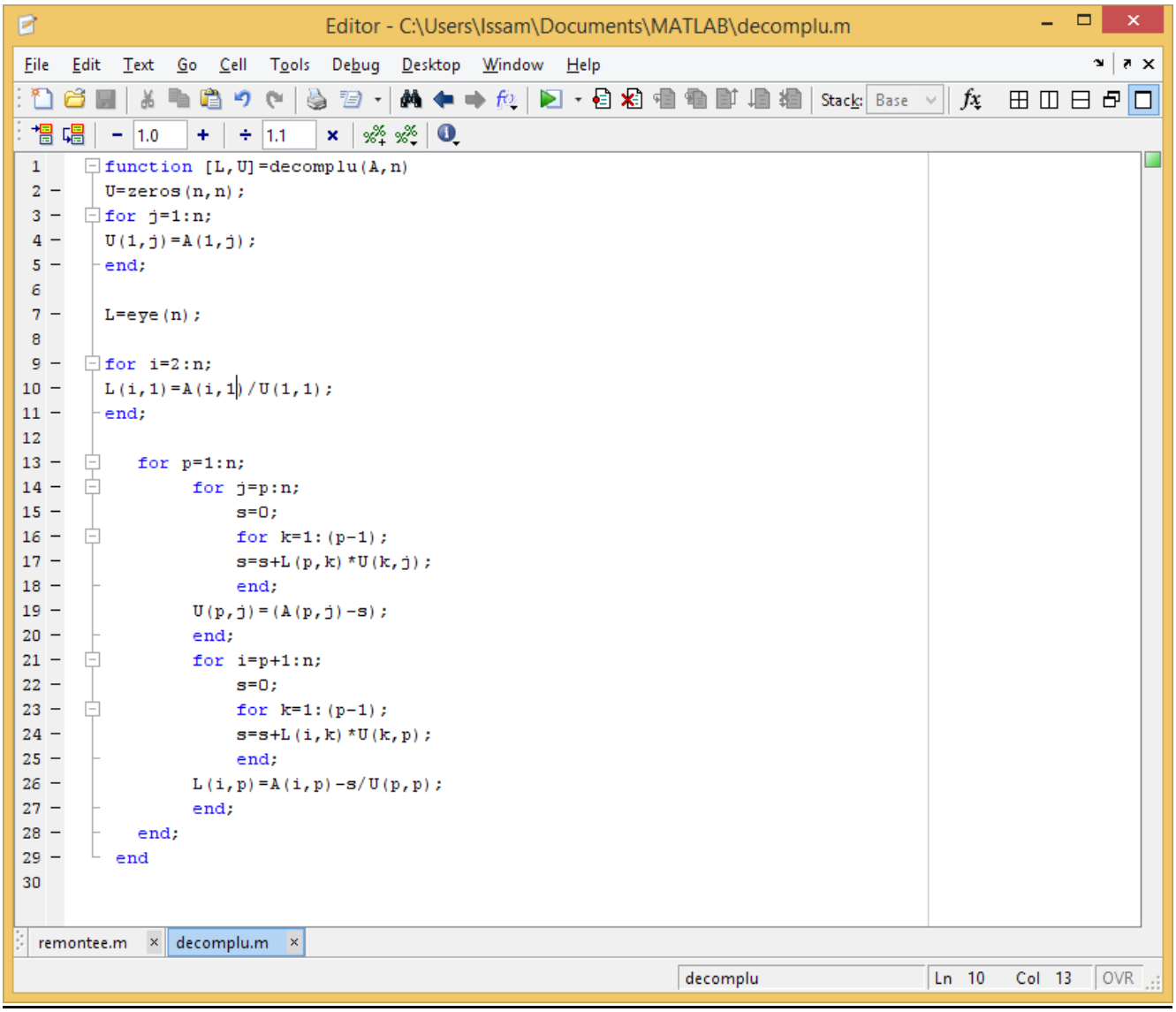
3) Effectuer la factorisation LU d'une matrice A réelle inversible d'ordre n, avec :

L : matrice triangulaire inférieure à diagonale unité.

U : matrice triangulaire supérieure.

Algorithme de la fonction *decomplu(A,n)* :

```
>> function [L,U]=decomplu(A,n)
U=zeros(n,n);
for j=1:n;
U(1,j)=A(1,j);
end;

L=eye(n);

for i=2:n;
L(i,1)=A(i,1)/U(1,1);
end;

   for p=1:n;
        for j=p:n;
            s=0;
            for k=1:(p-1);
            s=s+L(p,k)*U(k,j);
            end;
        U(p,j)=(A(p,j)-s);
        end;
        for i=(p+1):n;
            s=0;
            for k=1:(p-1);
            s=s+L(i,k)*U(k,p);
            end;
        L(i,p)=(A(i,p)-s)/U(p,p);
        end;
   end;
 end
```


Editor - C:\Users\Issam\Documents\MATLAB\decomplu.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
function [L,U]=decomplu(A,n)
U=zeros(n,n);
for j=1:n;
U(1,j)=A(1,j);
end;
L=eye(n);
for i=2:n;
L(i,1)=A(i,1)/U(1,1);
end;
for p=1:n;
for j=p:n;
s=0;
for k=1:(p-1);
s=s+L(p,k)*U(k,j);
end;
U(p,j)=(A(p,j)-s);
end;
for i=p+1:n;
s=0;
for k=1:(p-1);
s=s+L(i,k)*U(k,p);
end;
L(i,p)=A(i,p)-s/U(p,p);
end;
end;
end
remontee.m
decomplu.m
decomplu
Ln 10 Col 13
OVR

Command Window

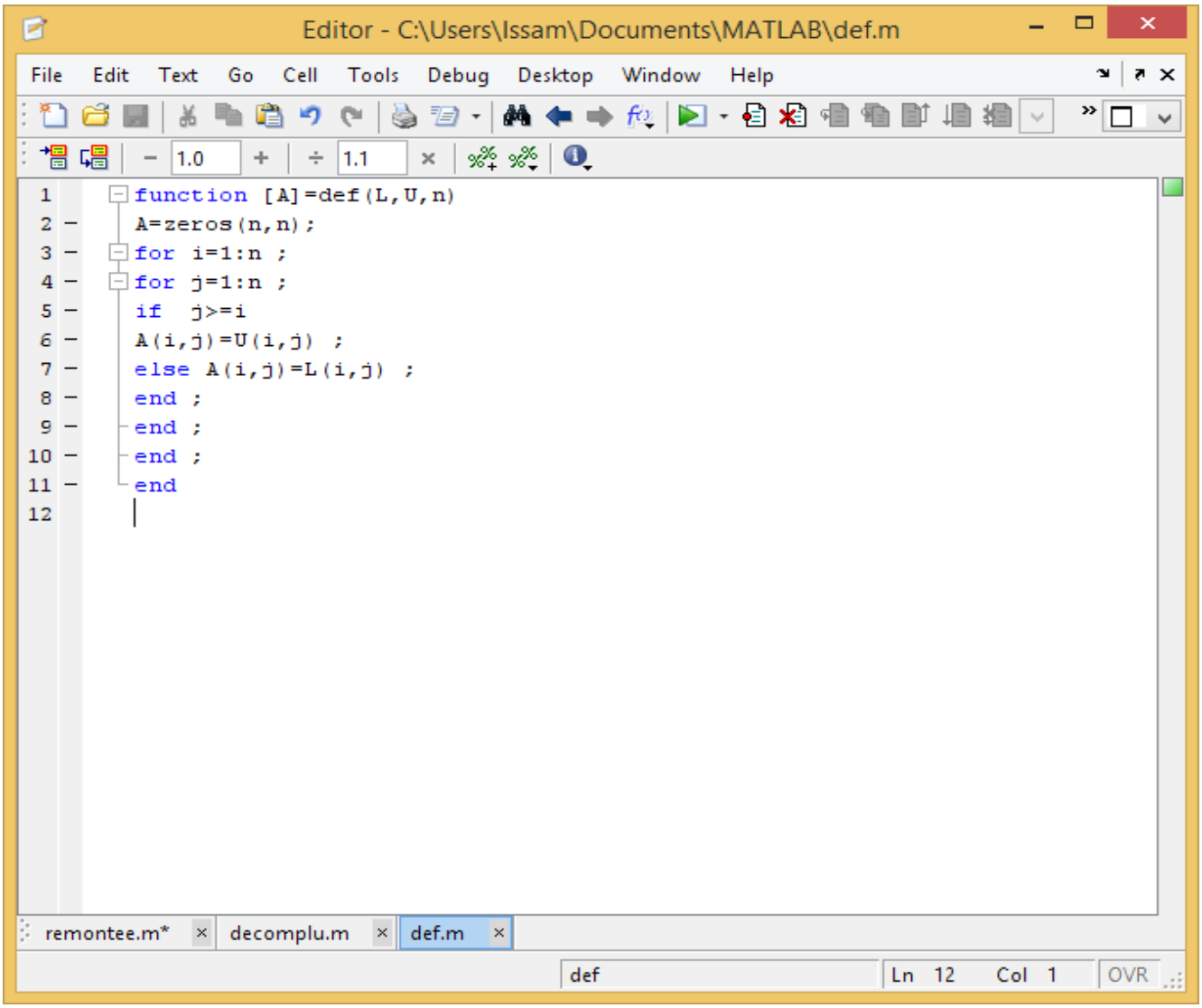
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

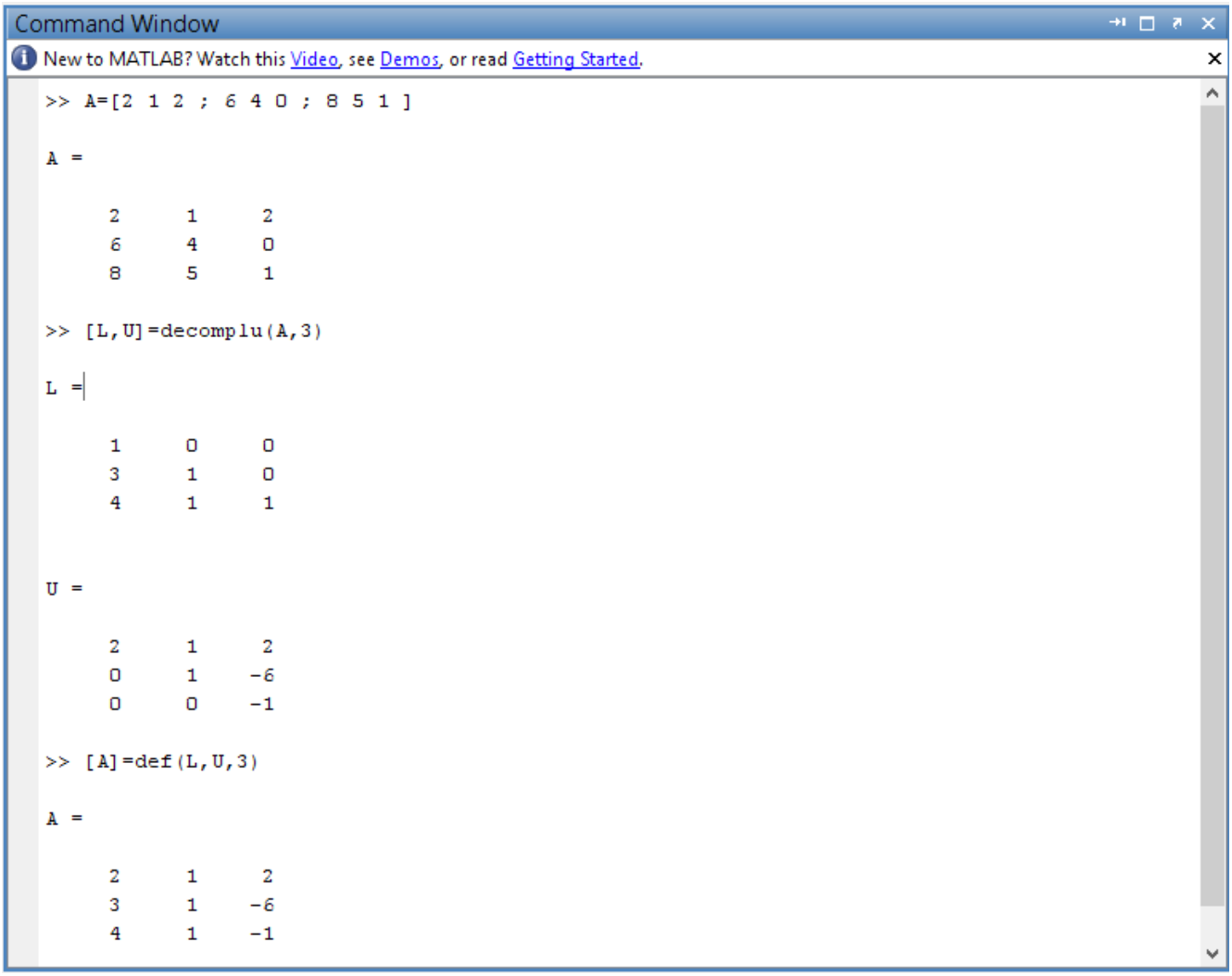
```
>> A = [ 2 1 2 ; 6 4 0 ; 8 5 1]

A =

     2     1     2
     6     4     0
     8     5     1

>> [L,U]=decomplu(A,3)

L =

     1     0     0
     3     1     0
     4     1     1


U =

     2     1     2
     0     1    -6
     0     0    -1

>> L*U

ans =

     2     1     2
     6     4     0
     8     5     1
```

°) La Matrice A Contiendra Les coefficients de U et L :

Algorithme de la fonction *def(L,U,n)* :

```
>> function [A]=def(L,U,n)
A=zeros(n,n);
for i=1:n ;
for j=1:n ;
if  j>=i
A(i,j)=U(i,j) ;
else A(i,j)=L(i,j) ;
end ;
end ;
end ;
end
```



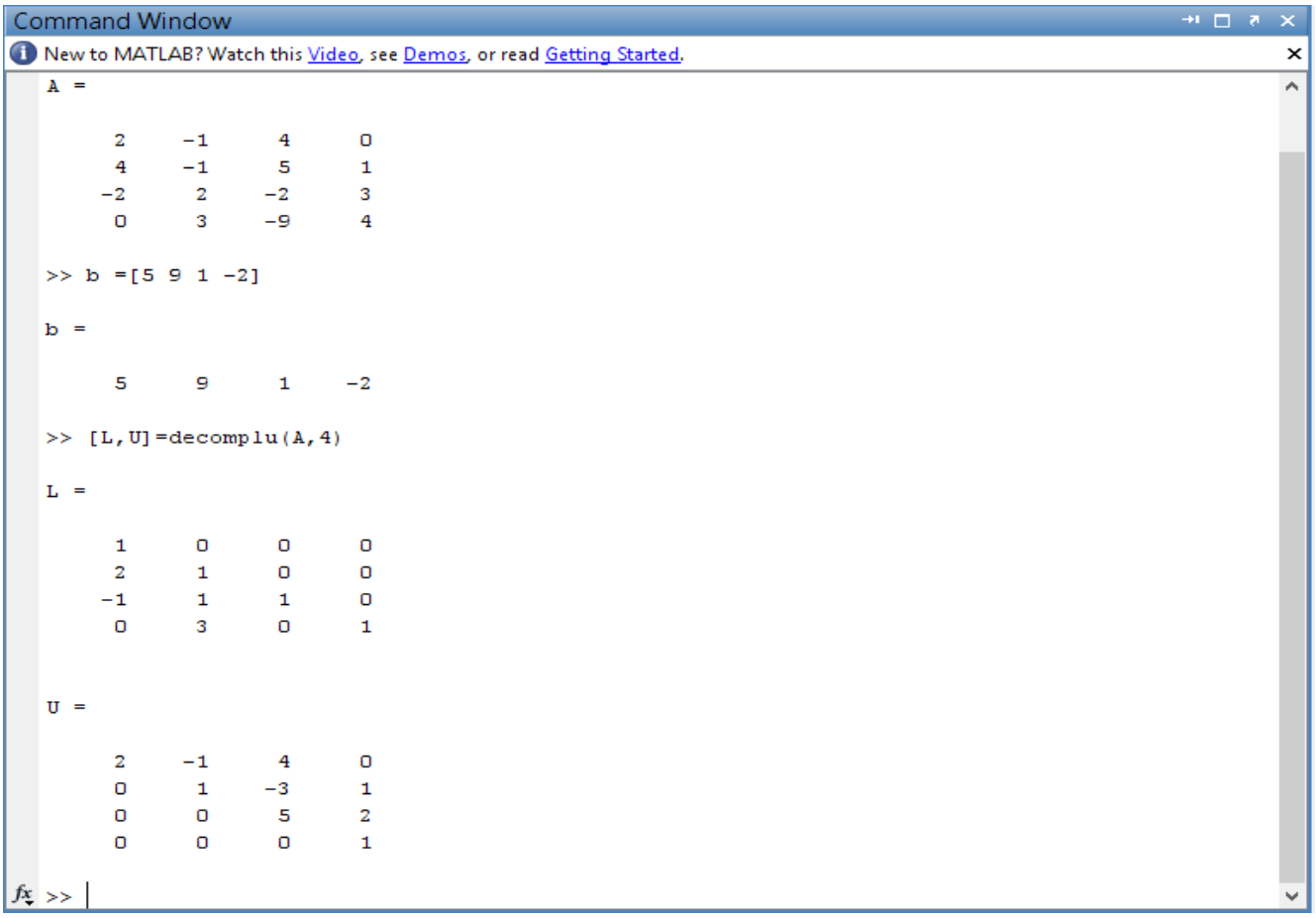

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=[2 1 2 ; 6 4 0 ; 8 5 1]
A =
2 1 2
6 4 0
8 5 1
>> [L,U]=decomplu(A,3)
L =
1 0 0
3 1 0
4 1 1
U =
2 1 2
0 1 -6
0 0 -1
>> [A]=def(L,U,3)
A =
2 1 2
3 1 -6
4 1 -1

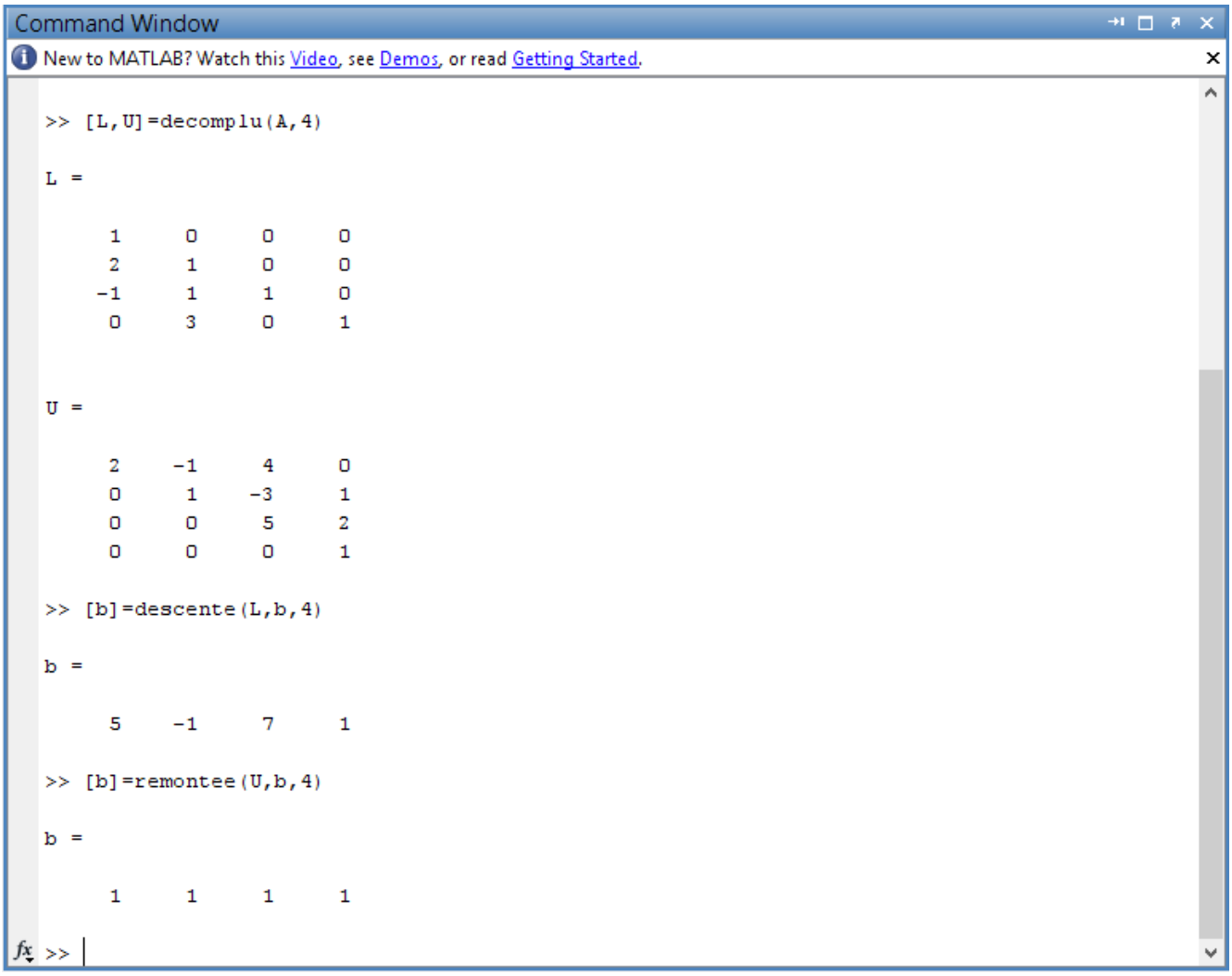
4) Résoudre le système linéaire Ax=b par la méthode de factorisation LU : déjà la fonction de calcul de L et U existe dans 3) `([L,U]=decomplu(A,n))`

Algorithme de la fonction *resollu(A,b)* :

```
>> [L,U]=decomplu(A,n)
>> [X]=descente(L,b,n)
>> [b]=remontee(U,b,n)
```

Exemple Ex 5 TD n°2 :



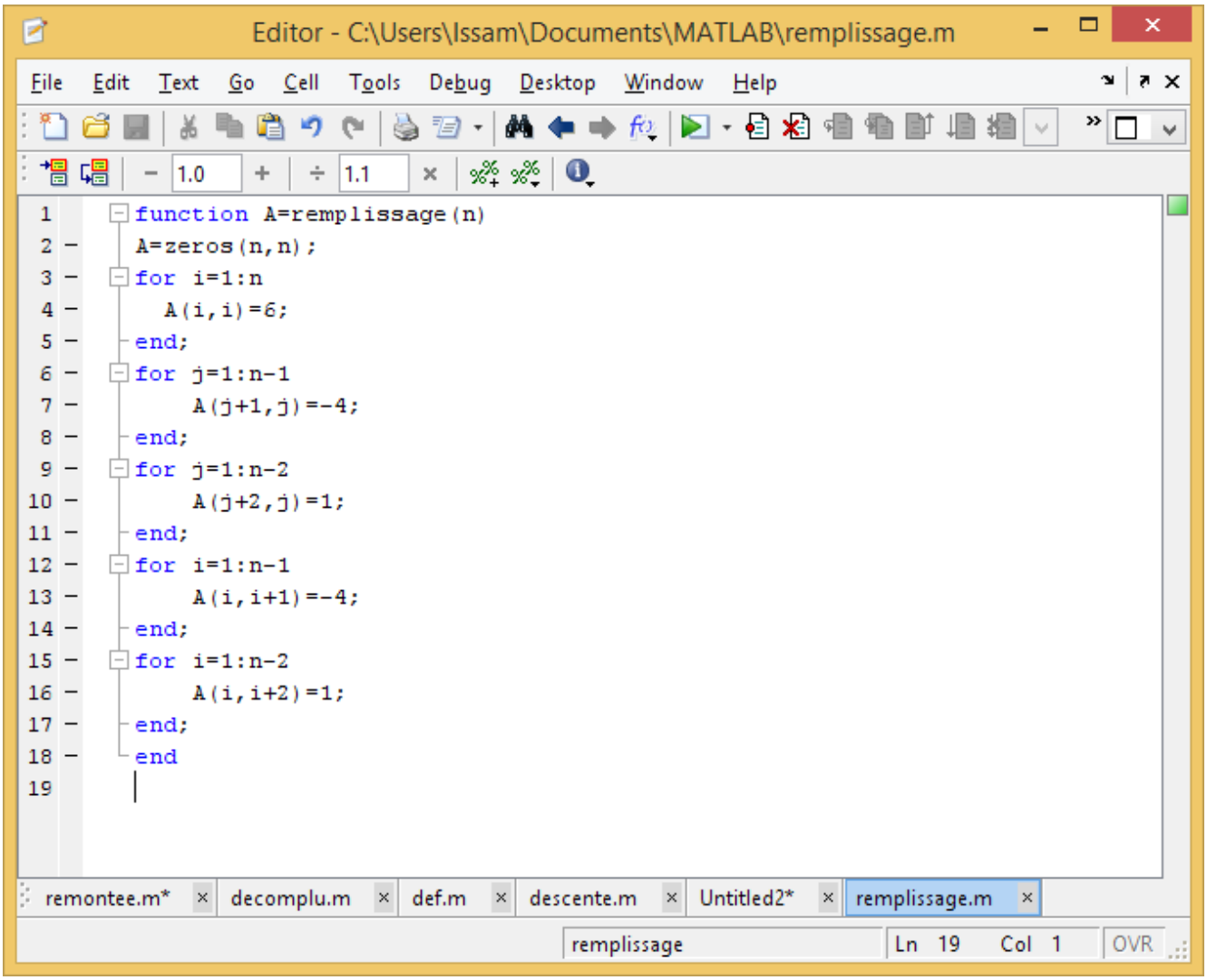

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> [L,U]=decomplu(A,4)
L =
1 0 0 0
2 1 0 0
-1 1 1 0
0 3 0 1
U =
2 -1 4 0
0 1 -3 1
0 0 5 2
0 0 0 1
>> [b]=descente(L,b,4)
b =
5 -1 7 1
>> [b]=remontee(U,b,4)
b =
1 1 1 1
>>

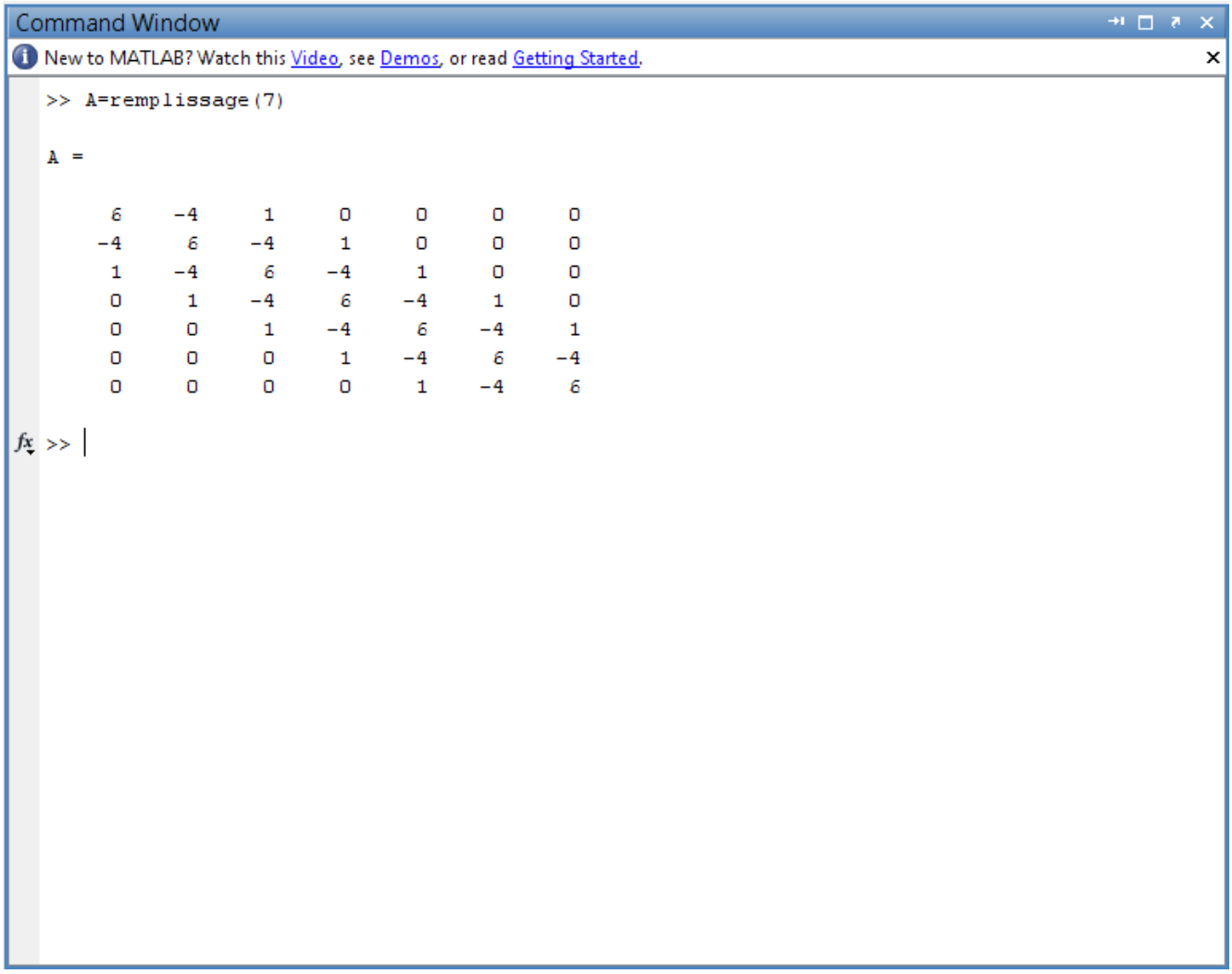
II- Application : Flexion d'une poutre encastrée :

1) La fonction **remplissage** permet de construire une matrice selon la forme demandée :

Algorithme de la fonction *remplissage(n)* :

```
function A=remplissage(n)
A=zeros(n,n);
for i=1:n
  A(i,i)=6;
end;
for j=1:n-1
    A(j+1,j)=-4;
end;
for j=1:n-2
    A(j+2,j)=1;
end;
for i=1:n-1
    A(i,i+1)=-4;
end;
for i=1:n-2
    A(i,i+2)=1;
end;
end
```

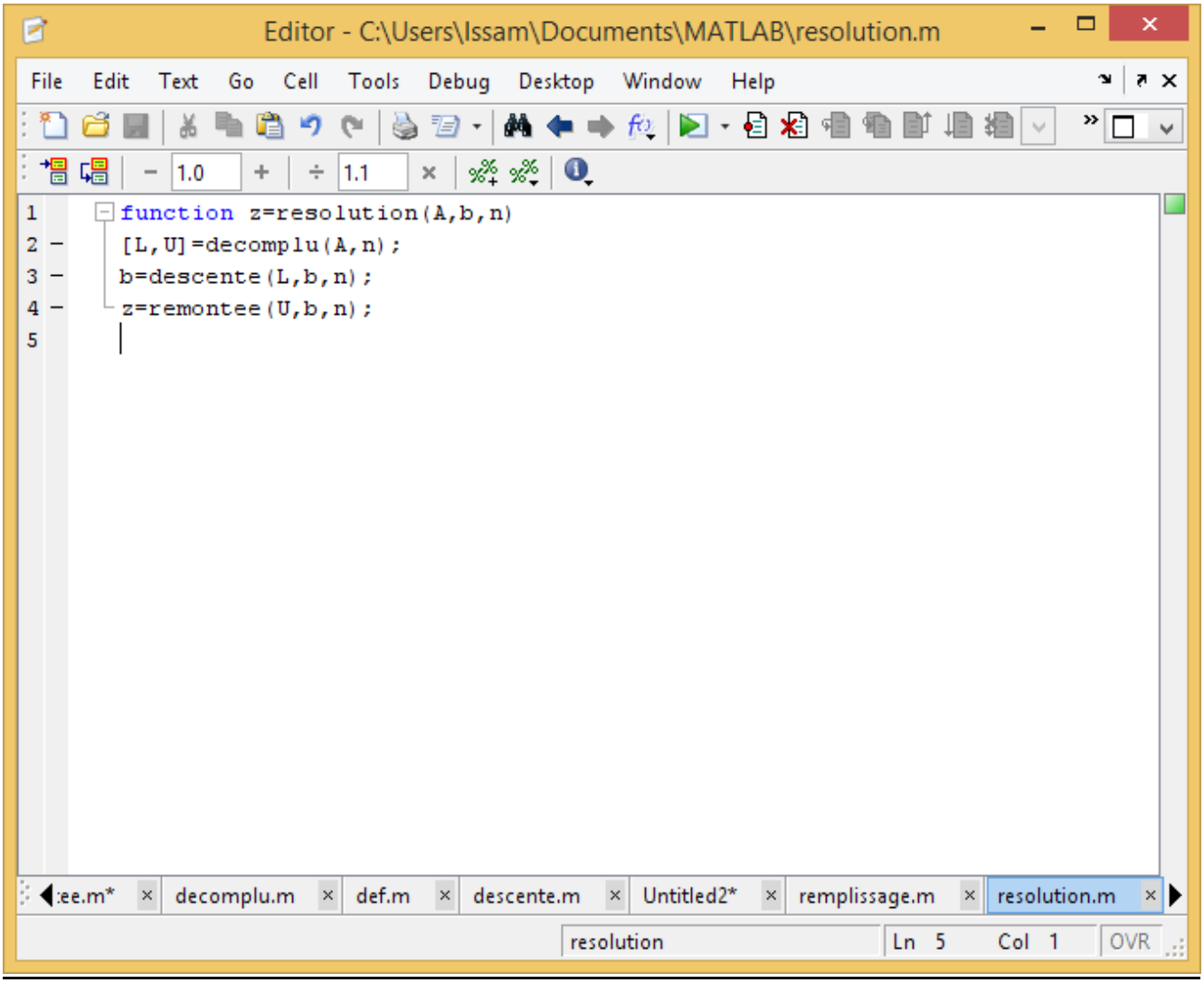

Editor - C:\Users\Issam\Documents\MATLAB\remplissage.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1.0
1.1
function A=remplissage(n)
A=zeros(n,n);
for i=1:n
A(i,i)=6;
end;
for j=1:n-1
A(j+1,j)=-4;
end;
for j=1:n-2
A(j+2,j)=1;
end;
for i=1:n-1
A(i,i+1)=-4;
end;
for i=1:n-2
A(i,i+2)=1;
end;
end
remontee.m*
decomplu.m
def.m
descente.m
Untitled2*
remplissage.m
remplissage
Ln 19 Col 1
OVR


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(7)
A =
6 -4 1 0 0 0 0
-4 6 -4 1 0 0 0
1 -4 6 -4 1 0 0
0 1 -4 6 -4 1 0
0 0 1 -4 6 -4 1
0 0 0 1 -4 6 -4
0 0 0 0 1 -4 6
>>

2) La fonction **resolution** donne la resolution du système (3) :

Algorithme de la fonction resolution (A , b , n) :

```
function z=resolution(A,b,n)
[L,U]=decomplu(A,n);
b=descente(L,b,n);
z=remontee(U,b,n);
```



a) Charge équirépartie :

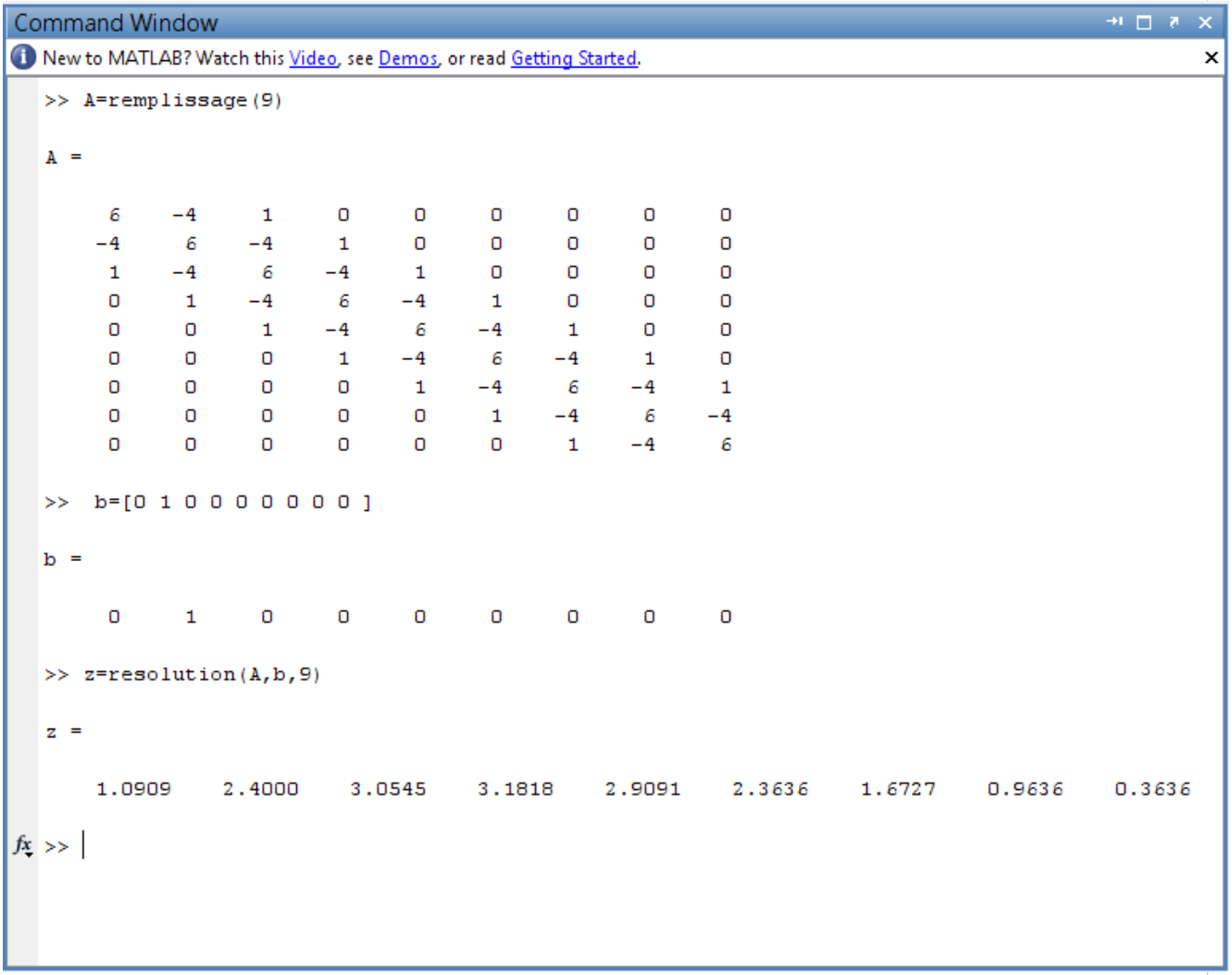
```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> A=remplissage(9)

A =

     6    -4     1     0     0     0     0     0     0
    -4     6    -4     1     0     0     0     0     0
     1    -4     6    -4     1     0     0     0     0
     0     1    -4     6    -4     1     0     0     0
     0     0     1    -4     6    -4     1     0     0
     0     0     0     1    -4     6    -4     1     0
     0     0     0     0     1    -4     6    -4     1
     0     0     0     0     0     1    -4     6    -4
     0     0     0     0     0     0     1    -4     6

>> b=[1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9 1/9 ]

b =

    0.1111    0.1111    0.1111    0.1111    0.1111    0.1111    0.1111    0.1111    0.1111

>> z=resolution(A,b,9)

z =

    0.8333    2.0000    3.1111    3.8889    4.1667    3.8889    3.1111    2.0000    0.8333

fx >>
```

b) **Charge localisé en un point :**

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

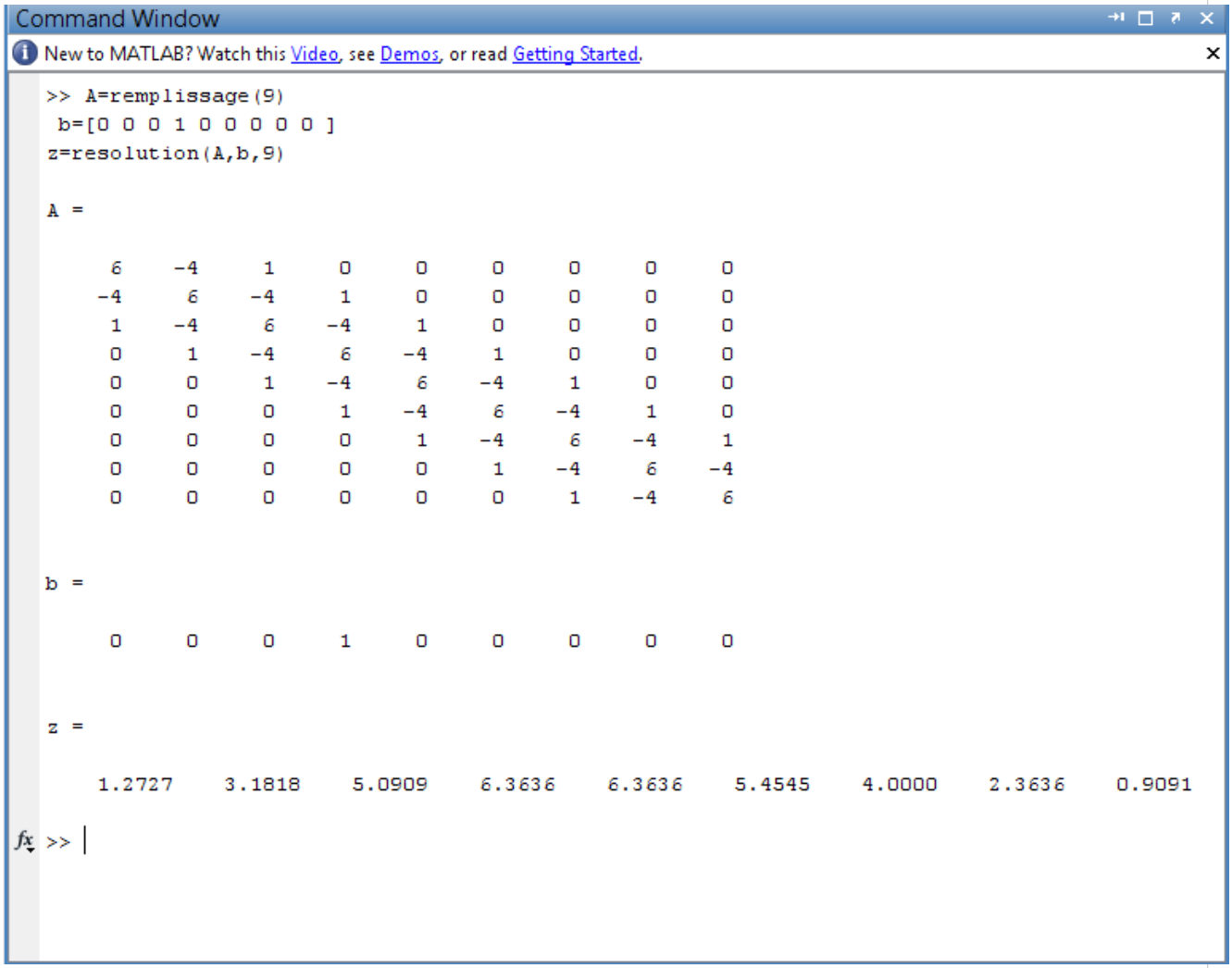
```
>> A=remplissage(9)

A =

     6    -4     1     0     0     0     0     0     0
    -4     6    -4     1     0     0     0     0     0
     1    -4     6    -4     1     0     0     0     0
     0     1    -4     6    -4     1     0     0     0
     0     0     1    -4     6    -4     1     0     0
     0     0     0     1    -4     6    -4     1     0
     0     0     0     0     1    -4     6    -4     1
     0     0     0     0     0     1    -4     6    -4
     0     0     0     0     0     0     1    -4     6

>> b=[1 0 0 0 0 0 0 0 0 ]

b =

     1     0     0     0     0     0     0     0     0

>> z=resolution(A,b,9)

z =

    0.6818    1.0909    1.2727    1.2727    1.1364    0.9091    0.6364    0.3636    0.1364

fx >>
```

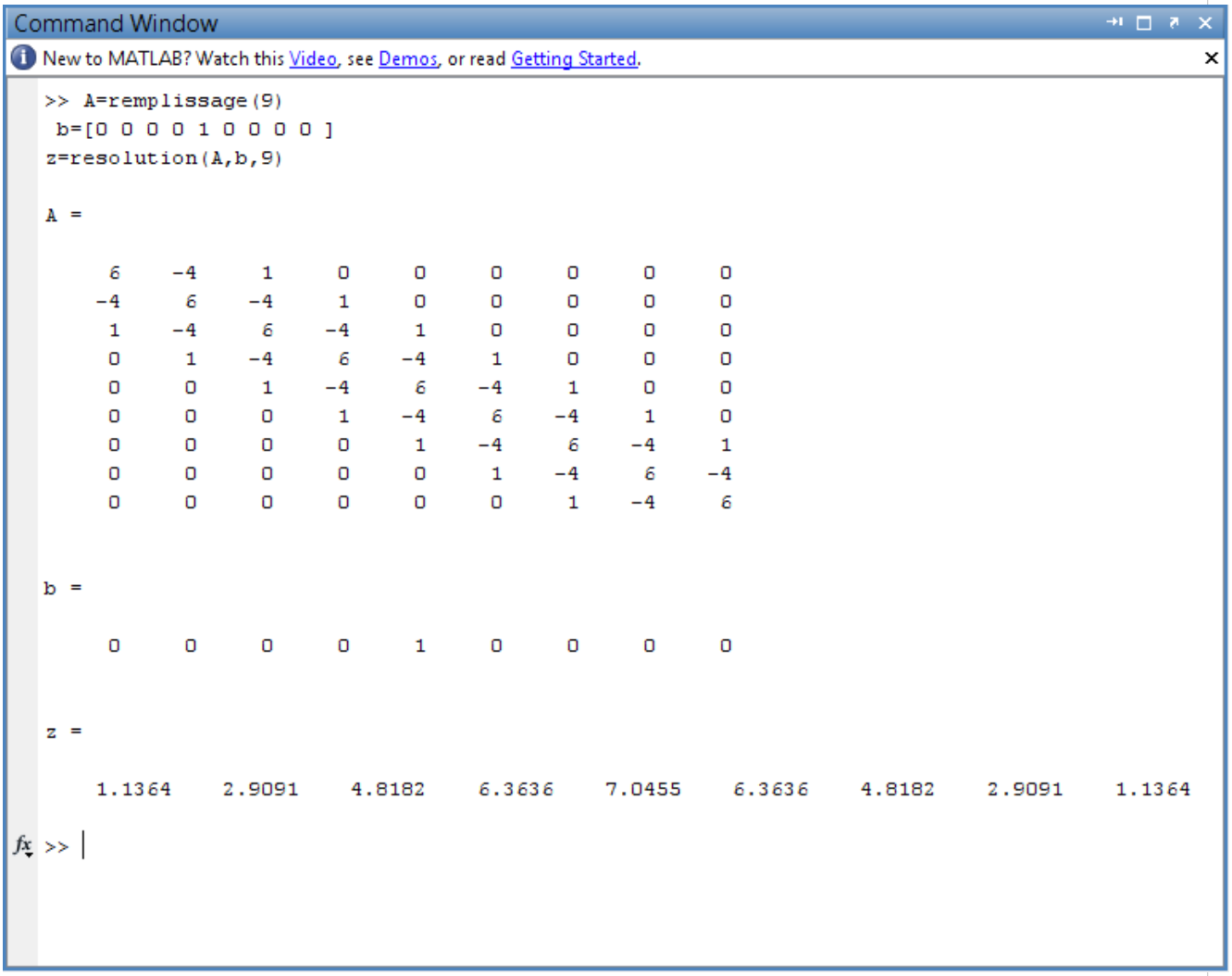

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
>> b=[0 1 0 0 0 0 0 0 0]
b =
0 1 0 0 0 0 0 0 0
>> z=resolution(A,b,9)
z =
1.0909 2.4000 3.0545 3.1818 2.9091 2.3636 1.6727 0.9636 0.3636
>>

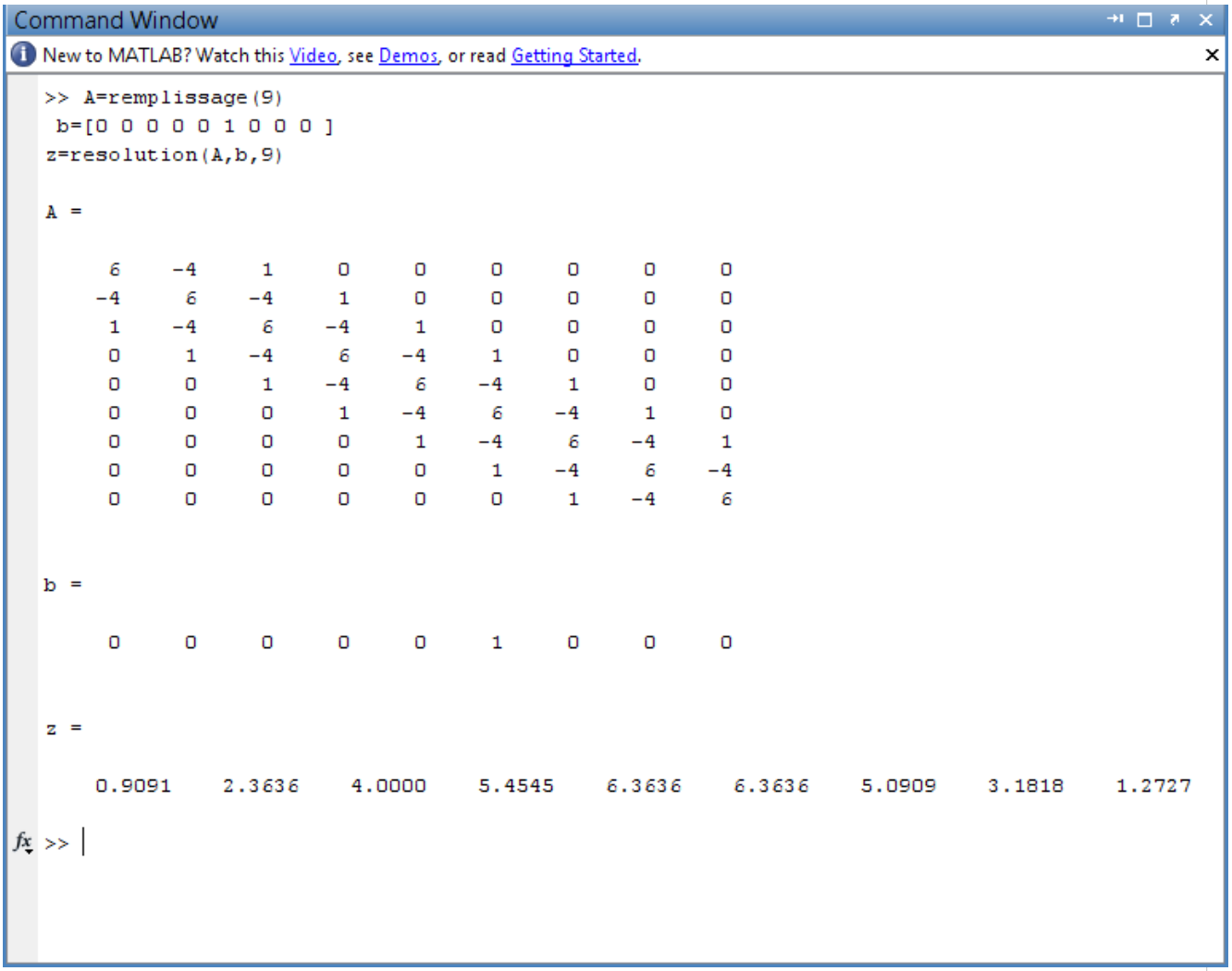
Command Window

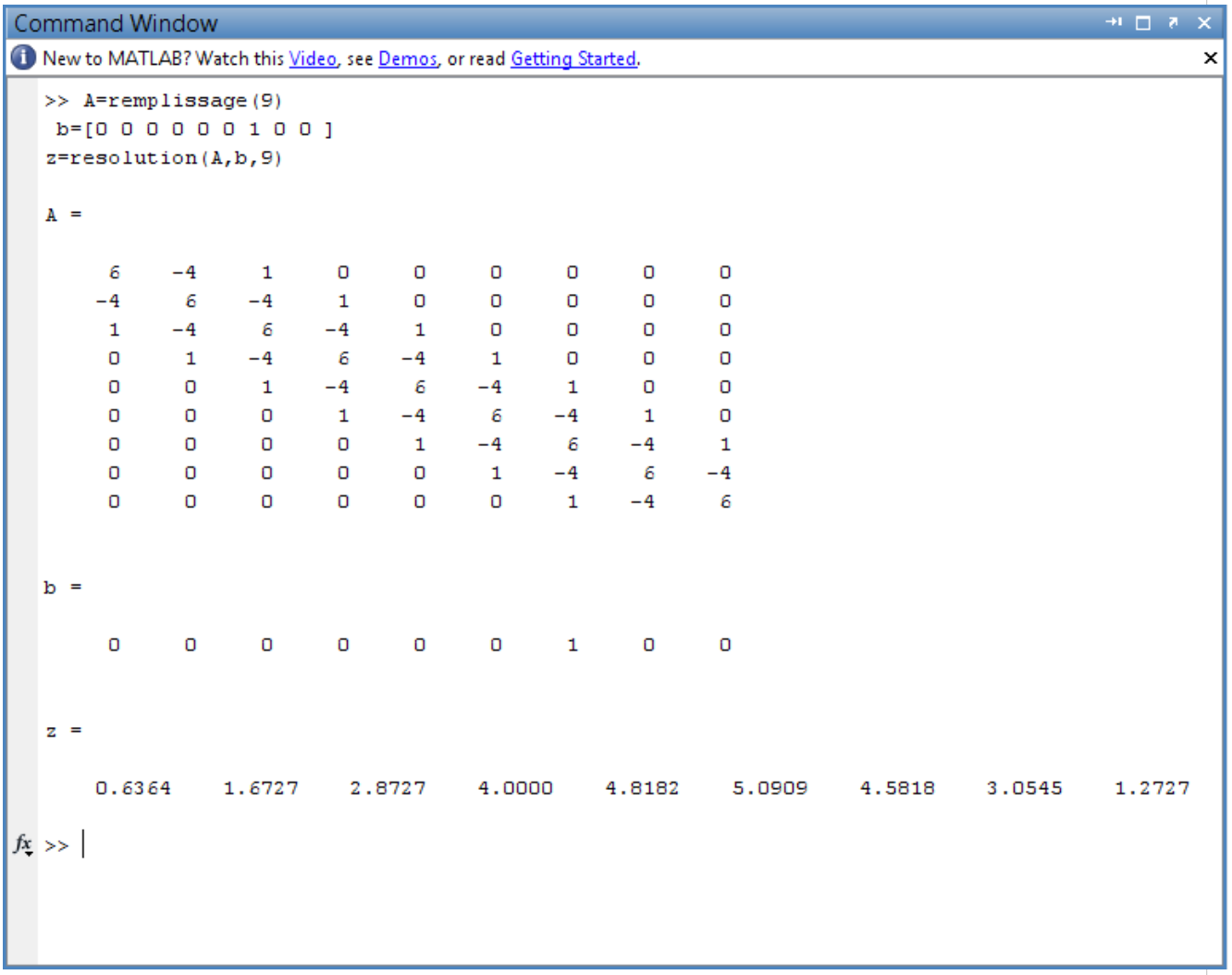
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

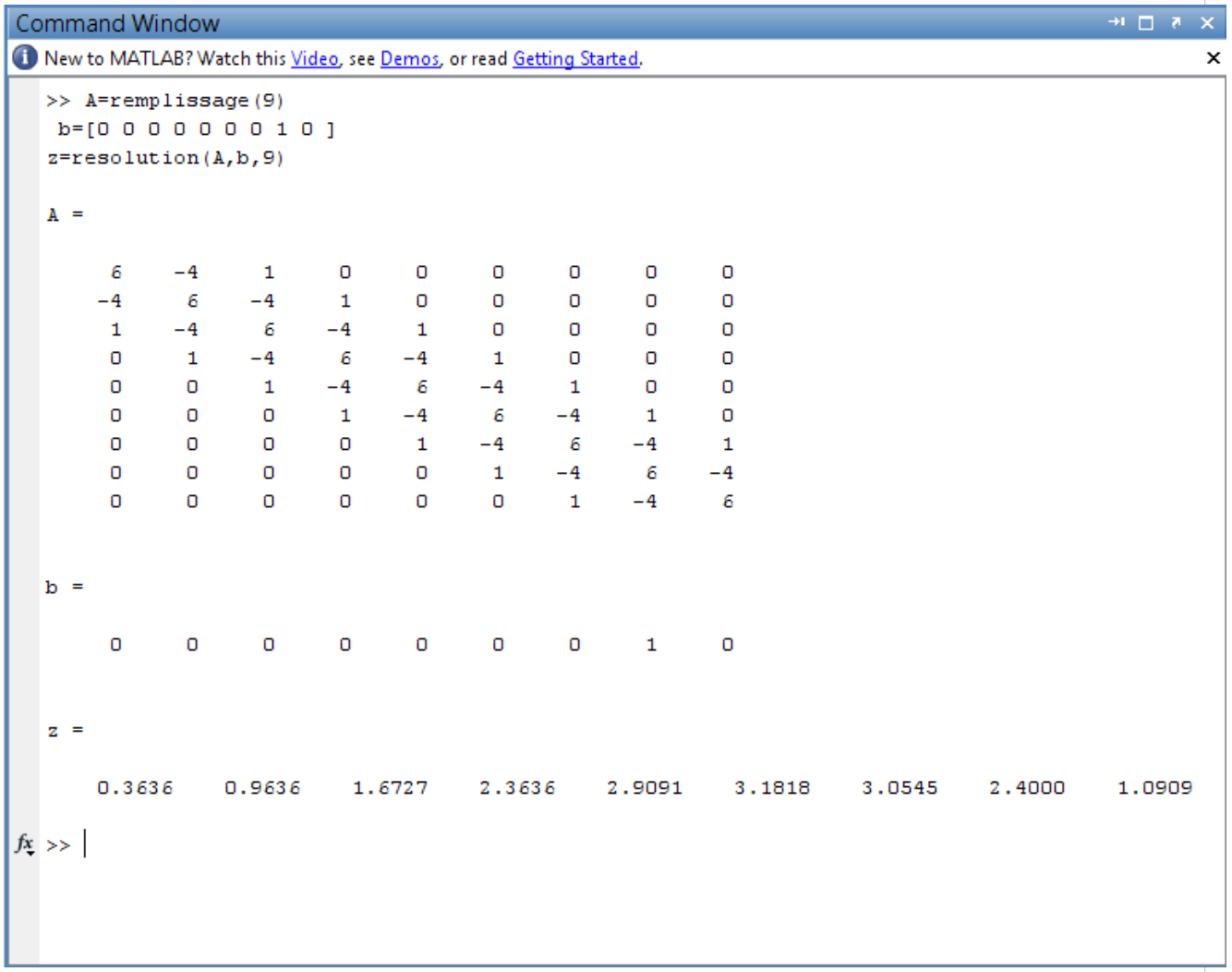
```
>> A=remplissage(9)
 b=[0 0 1 0 0 0 0 0 0 ]
z=resolution(A,b,9)

A =

     6    -4     1     0     0     0     0     0     0
    -4     6    -4     1     0     0     0     0     0
     1    -4     6    -4     1     0     0     0     0
     0     1    -4     6    -4     1     0     0     0
     0     0     1    -4     6    -4     1     0     0
     0     0     0     1    -4     6    -4     1     0
     0     0     0     0     1    -4     6    -4     1
     0     0     0     0     0     1    -4     6    -4
     0     0     0     0     0     0     1    -4     6


b =

     0     0     1     0     0     0     0     0     0


z =

    1.2727    3.0545    4.5818    5.0909    4.8182    4.0000    2.8727    1.6727    0.6364

fx >>
```

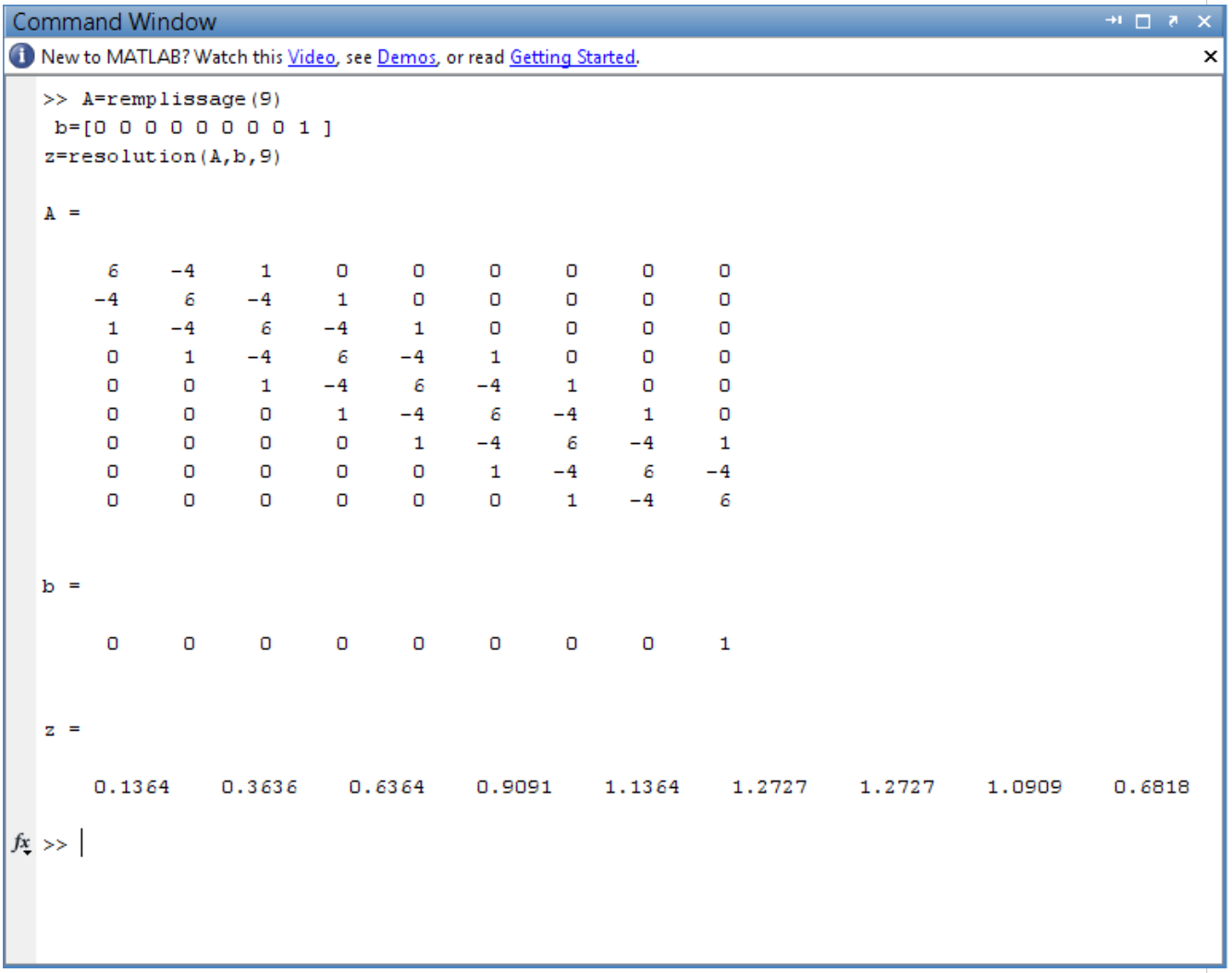

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 1 0 0 0 0 0]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 1 0 0 0 0 0
z =
1.2727 3.1818 5.0909 6.3636 6.3636 5.4545 4.0000 2.3636 0.9091
>>


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 0 1 0 0 0 0]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 0 1 0 0 0 0
z =
1.1364 2.9091 4.8182 6.3636 7.0455 6.3636 4.8182 2.9091 1.1364
>>


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 0 0 1 0 0 0]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 0 0 1 0 0 0
z =
0.9091 2.3636 4.0000 5.4545 6.3636 6.3636 5.0909 3.1818 1.2727
>>


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 0 0 0 1 0 0]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 0 0 0 1 0 0
z =
0.6364 1.6727 2.8727 4.0000 4.8182 5.0909 4.5818 3.0545 1.2727
>>

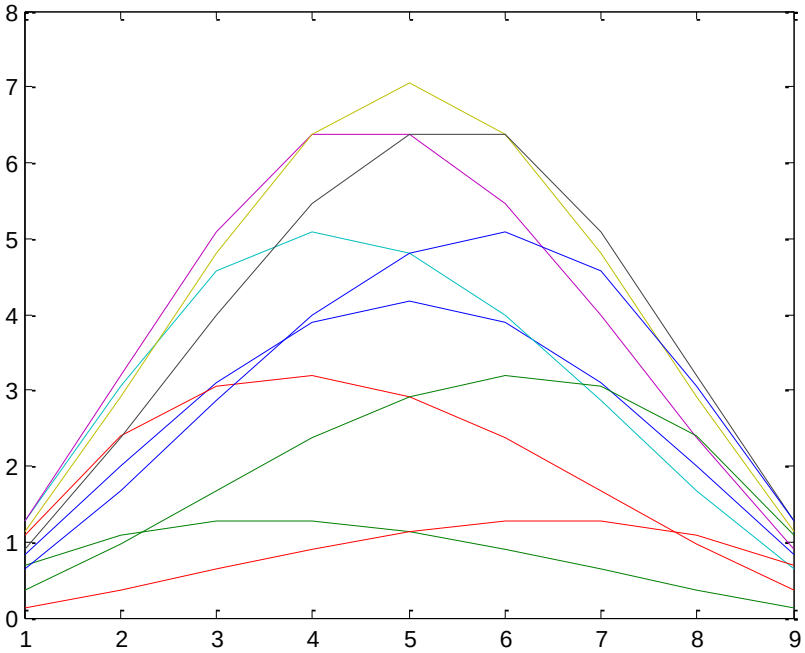

Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 0 0 0 0 1 0]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 0 0 0 0 1 0
z =
0.3636 0.9636 1.6727 2.3636 2.9091 3.1818 3.0545 2.4000 1.0909
>>


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
>> A=remplissage(9)
b=[0 0 0 0 0 0 0 0 1]
z=resolution(A,b,9)
A =
6 -4 1 0 0 0 0 0 0
-4 6 -4 1 0 0 0 0 0
1 -4 6 -4 1 0 0 0 0
0 1 -4 6 -4 1 0 0 0
0 0 1 -4 6 -4 1 0 0
0 0 0 1 -4 6 -4 1 0
0 0 0 0 1 -4 6 -4 1
0 0 0 0 0 1 -4 6 -4
0 0 0 0 0 0 1 -4 6
b =
0 0 0 0 0 0 0 0 1
z =
0.1364 0.3636 0.6364 0.9091 1.1364 1.2727 1.2727 1.0909 0.6818
>>

3) Traçage des flexions de la poutre :

```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

A =

     6    -4     1     0     0     0     0     0     0
    -4     6    -4     1     0     0     0     0     0
     1    -4     6    -4     1     0     0     0     0
     0     1    -4     6    -4     1     0     0     0
     0     0     1    -4     6    -4     1     0     0
     0     0     0     1    -4     6    -4     1     0
     0     0     0     0     1    -4     6    -4     1
     0     0     0     0     0     1    -4     6    -4
     0     0     0     0     0     0     1    -4     6

>> b1=[1 0 0 0 0 0 0 0 0 ];
>> z1=resolution(A,b,9);
>> b2=[0 1 0 0 0 0 0 0 0 ];
>> z2=resolution(A,b,9);
>> b3=[0 0 1 0 0 0 0 0 0 ];
>> z3=resolution(A,b,9);
>> b4=[0 0 0 1 0 0 0 0 0 ];
>> z4=resolution(A,b,9);
>> b5=[0 0 0 0 1 0 0 0 0 ];
>> z5=resolution(A,b,9);
>> b6=[0 0 0 0 0 1 0 0 0 ];
>> z6=resolution(A,b,9);
>> b7=[0 0 0 0 0 0 1 0 0 ];
>> z7=resolution(A,b,9);
>> b8=[0 0 0 0 0 0 0 1 0 ];
>> z8=resolution(A,b,9);
>> b9=[0 0 0 0 0 0 0 0 1 ];
>> z9=resolution(A,b,9);
fx >>
```

```
plot(z1)
hold all
plot(z2)
hold all
plot(z3)
hold all
.
.
.
.
plot(z9)
hold all
```


8
7
6
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9

4) La fonction **temps(n)** calcule le nombre d'opérations élémentaires pour la résolution du système linéaire (3) .

Algorithme de la fonction temps(n) :

```
function [add,div,mul]=temps(n)
add=0;
for p=1:n;

    for j=p:n;
        add = (p-1);
    end;
end;
for p=1:n
    for i=(p+1):n
        add=add+(p-1);
    end;
end;
mul=0;
for p=1:n;

    for j=p:n;
        mul=mul+(p-1);
    end;
end;
for p=1:n;
    for i=p+1:n;
        mul=mul+(p-1);
    end;
end;
div=0;
for p=1:n
    div=div+(n-p);
end;
end
```

****Pour n=9 :**

```
>> [add,div,mul]=cout_calcul(9)
add =
    92
div =
    36
mul =
   204
>>
```

****Pour n=10 :**

```
>> [add,div,mul]=cout_calcul(10)
add =
   129
div =
    45
mul =
   285
>>
```

****Pour n=11 :**

```
>> [add,div,mul]=cout_calcul(11)
add =
   175
div =
    55
mul =
   385
>>
```