***TP Analyse Numérique Linéaire***

***Préparé par :*** Raoudha Hellal (groupe C)

Amal Dakhli (groupe D)

1.Factorisation LU :

1.1. La fonction *fact\_Lu(A*) effectue la factorisation LU d’une matrice réelle inversible d’ordre n, avec :

L : matrice triangulaire inférieure à diagonale unité.

U : matrice triangulaire supérieure.

**Algorithme de la fonction *fact\_Lu(A)* :**

function A=fact\_lu(n,A)

for i=2:n;

A(i,1)=A(i,1)/A(1,1);

end;

for i=2:n;

for j=i:n;

s=0;

for k=1:(i-1);

s=s+A(i,k)\*A(k,j);

end;

A(i,j)=A(i,j)-s;

end;

for j=i+1:n;

s=0;

for k=1:(i-1);

s=s+A(j,k)\*A(k,i);

end;

A(j,i)=(A(j,i)-s)/A(i,i);

end;

end;

end

2.2. La fontion *sol\_Lu(A,b)* résout le système linéaire Ax=b en utilisant la factorisation LU de A.

**Algorithme de la fonction *sol\_Lu(A,b):***

function [x,y]=sol\_Lu(n,A,b)

y(1)=b(1);

for i=2:n;

s=0;

for j=1:(i-1);

s=s+A(i,j)\*y(j);

end;

y(i)=b(i)-s;

end;

x(n)=y(n)/A(n,n);

for i=(n-1):(-1):1;

s=0;

for j=(i+1):n;

s=s+A(i,j)\*x(j);

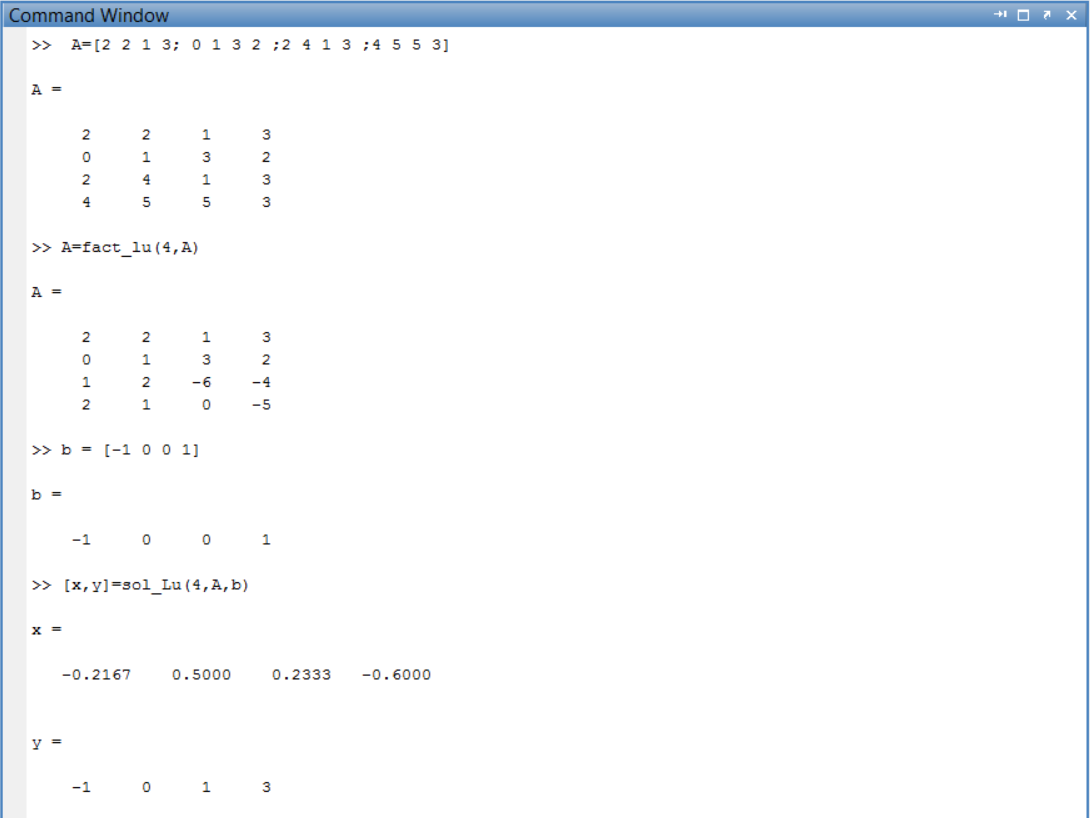
end;

x(i)=(y(i)-s)/A(i,i);

end;

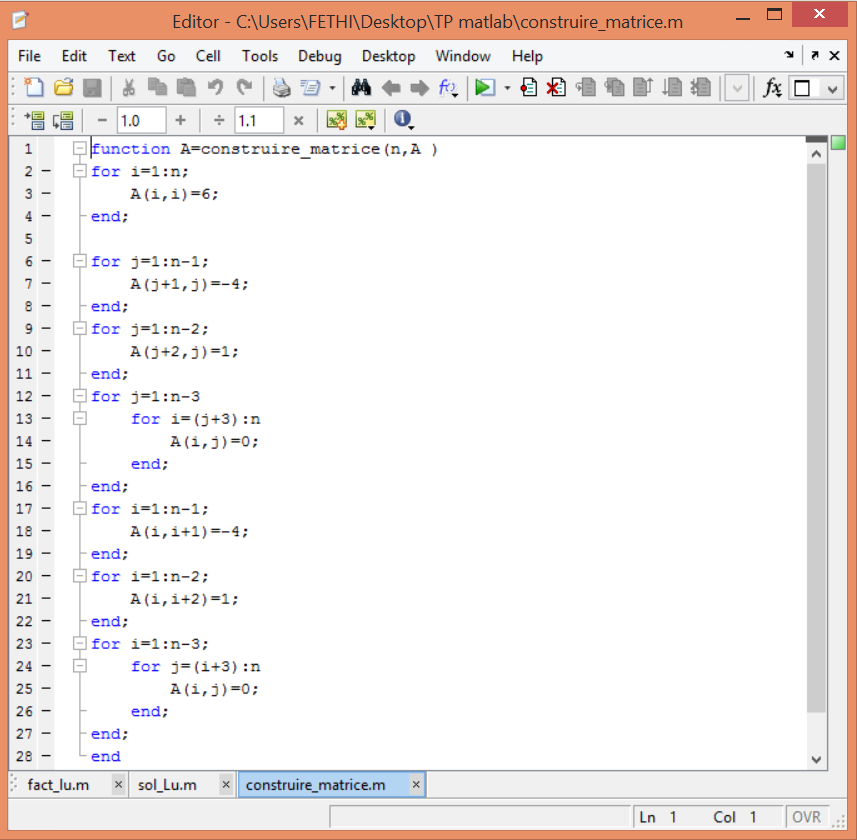
end

**3.3. Application :**

****

**2.Application : Flexion d’une poutre encastrée**

2.1. La fonction *construire\_matrice(A)* permet de construire une matrice selon la forme demandée.



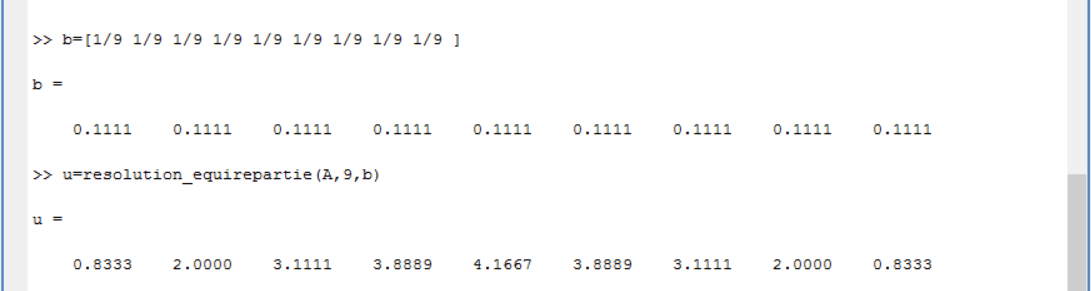
2.2. La fonction resolution\_equirepartie(A) donne la resolution du système (3) :

function u=resolution\_equirepartie(A,n,b)

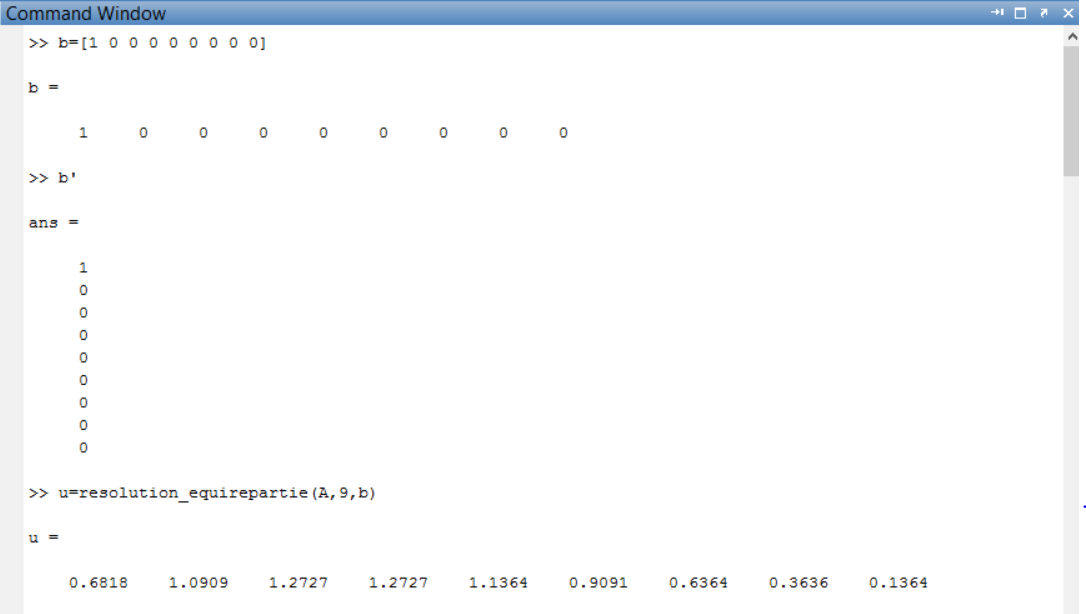
A=fact\_lu(n,A);

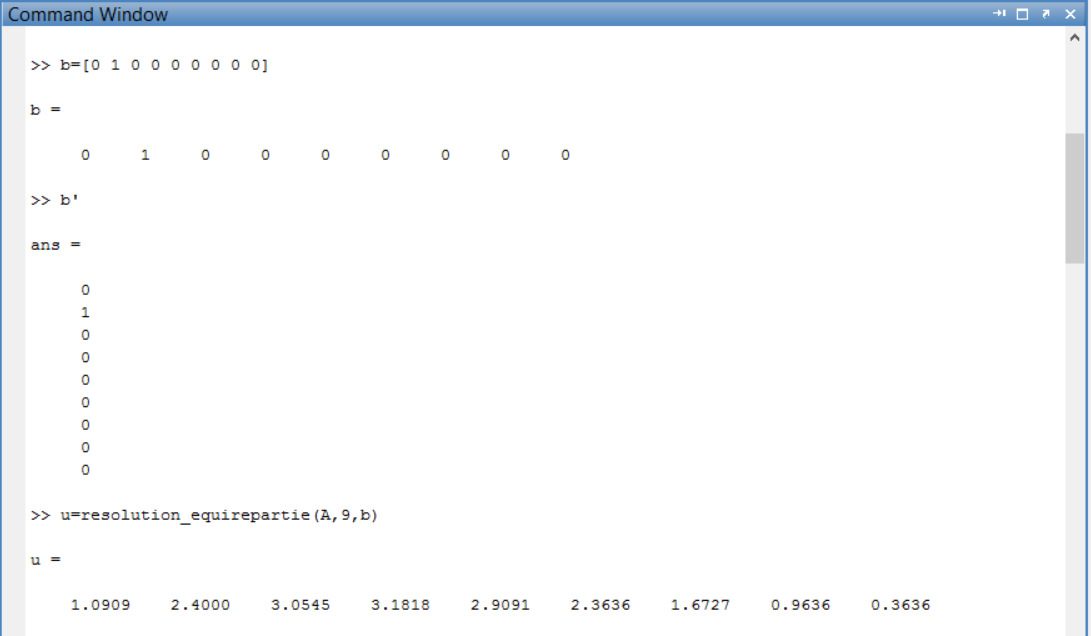
u=sol\_Lu(n,A,b);

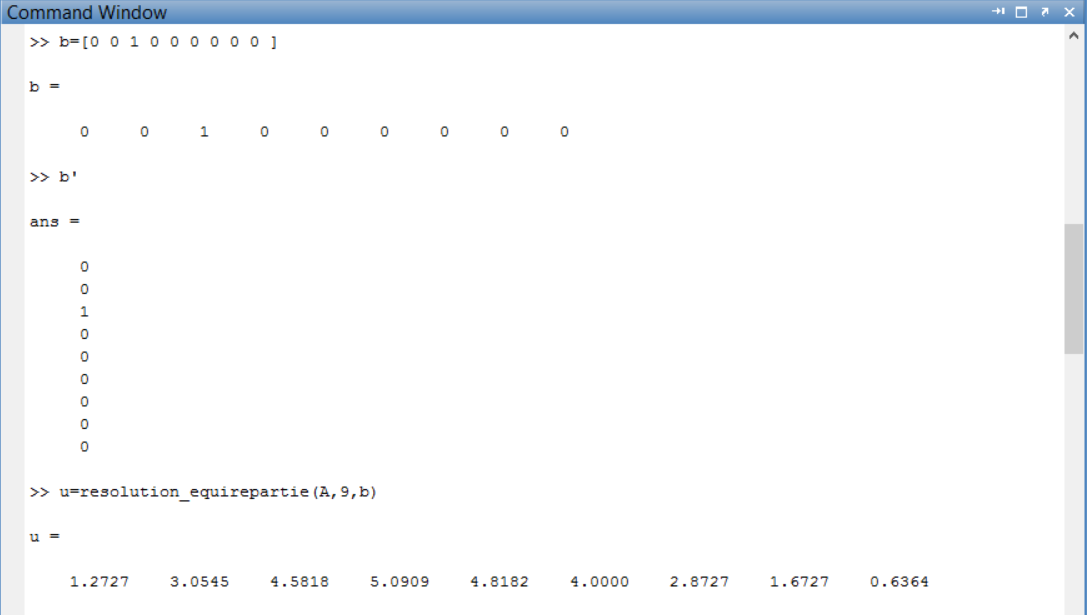
end

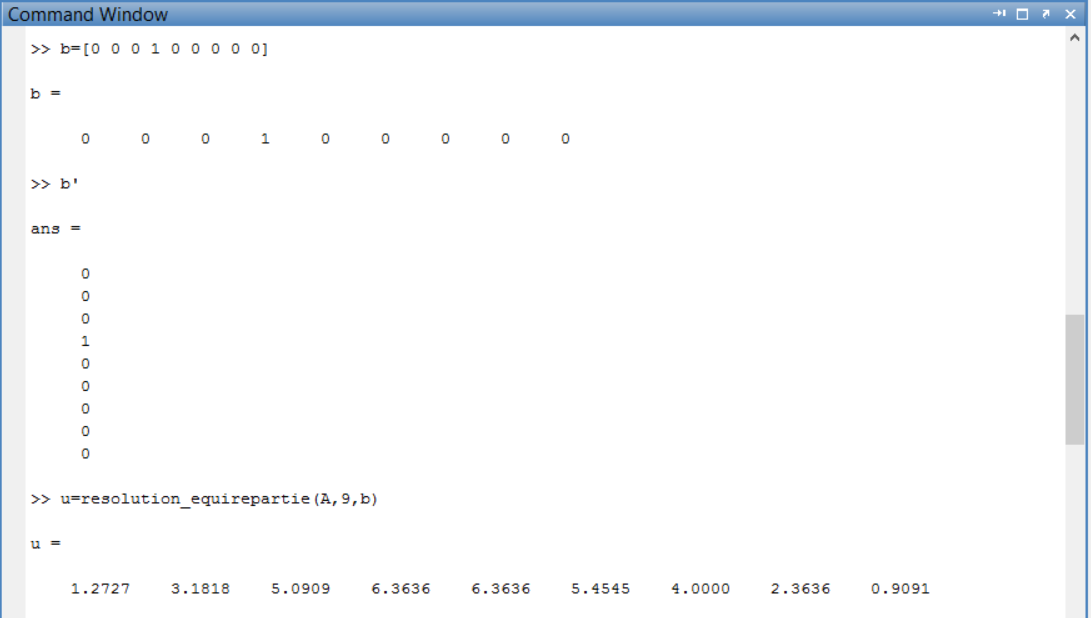
2.2.a. Charge équirépartie :

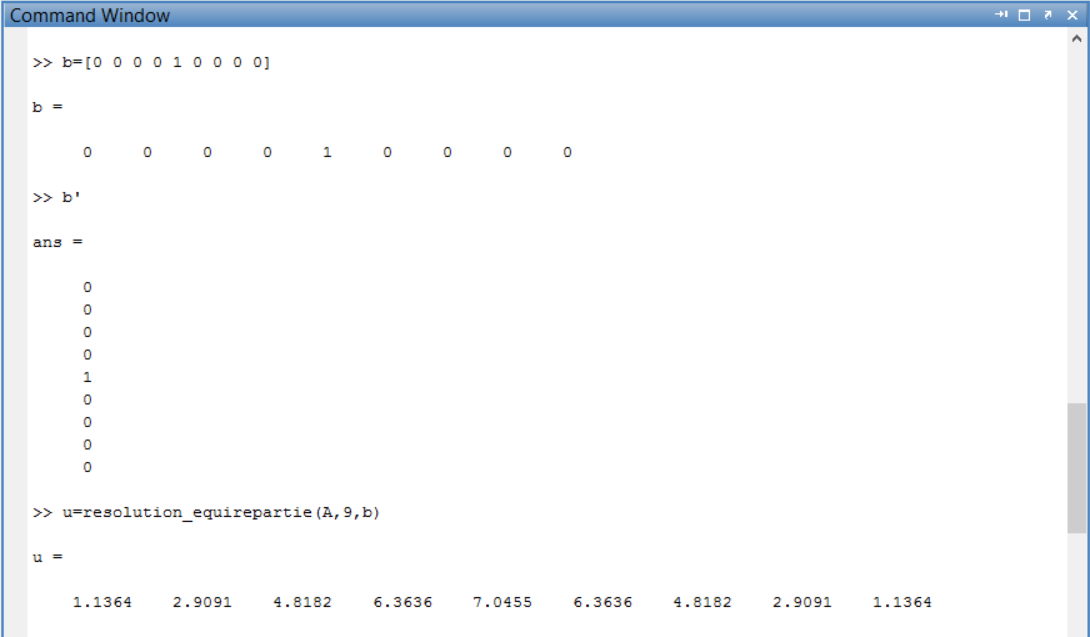
2.2.b. Charge localisée en un point :

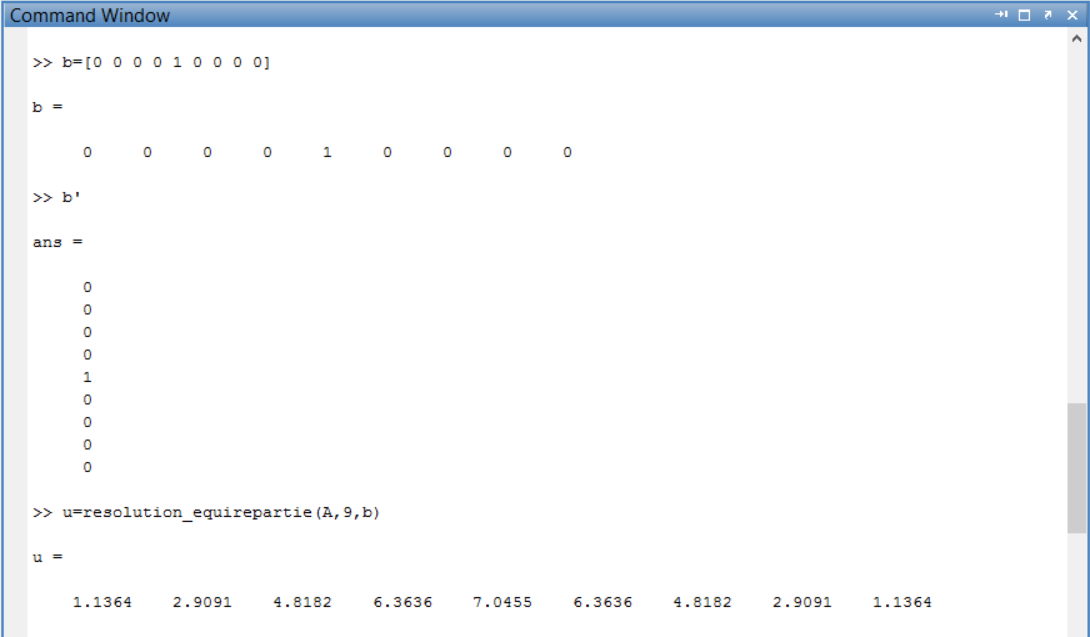


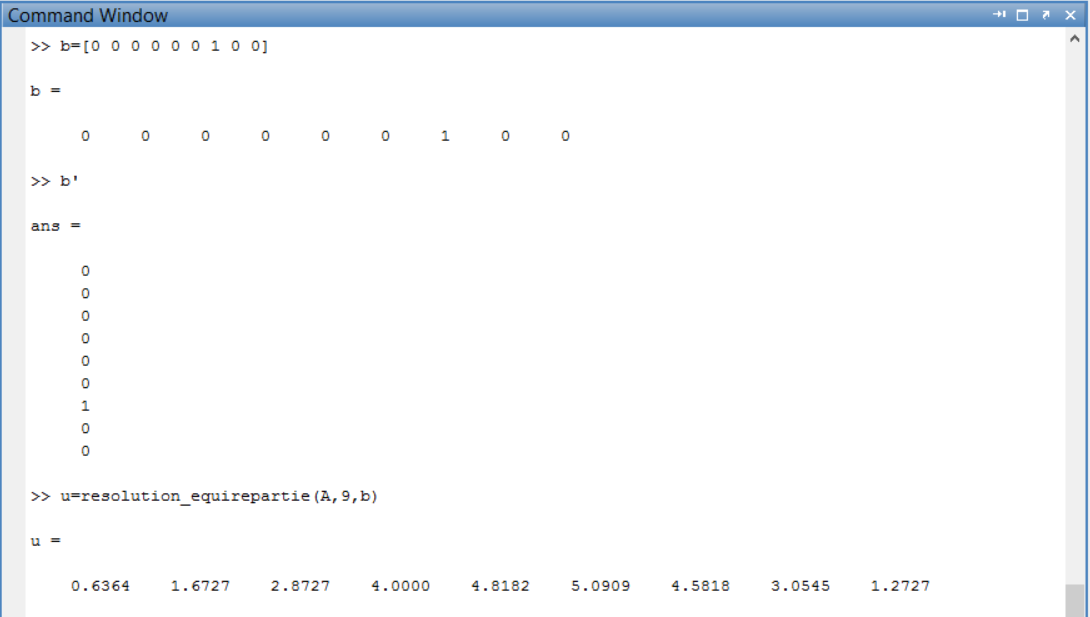


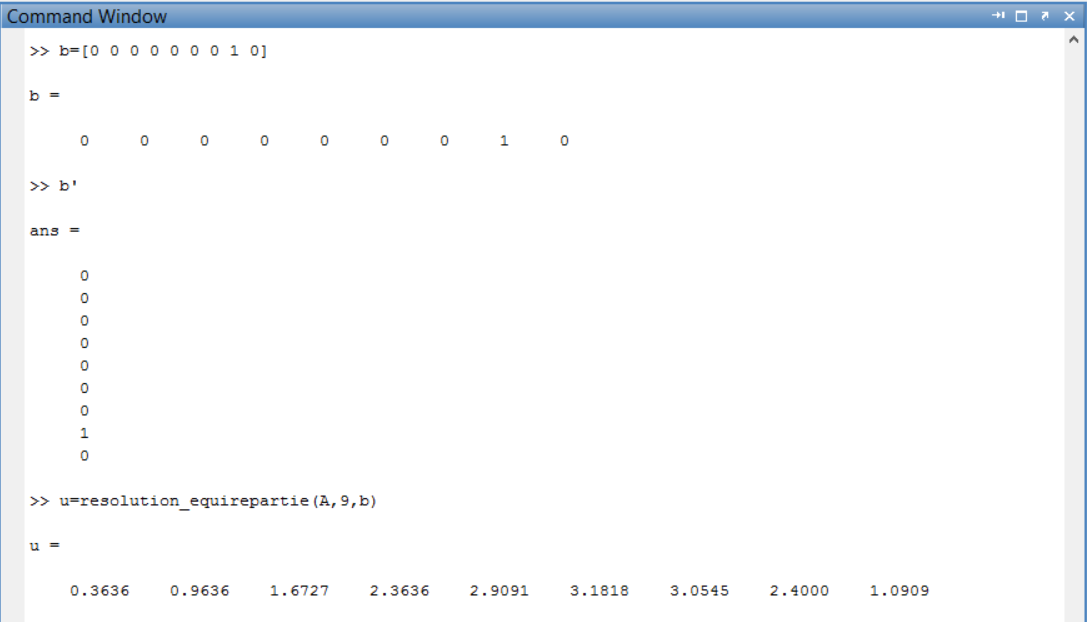


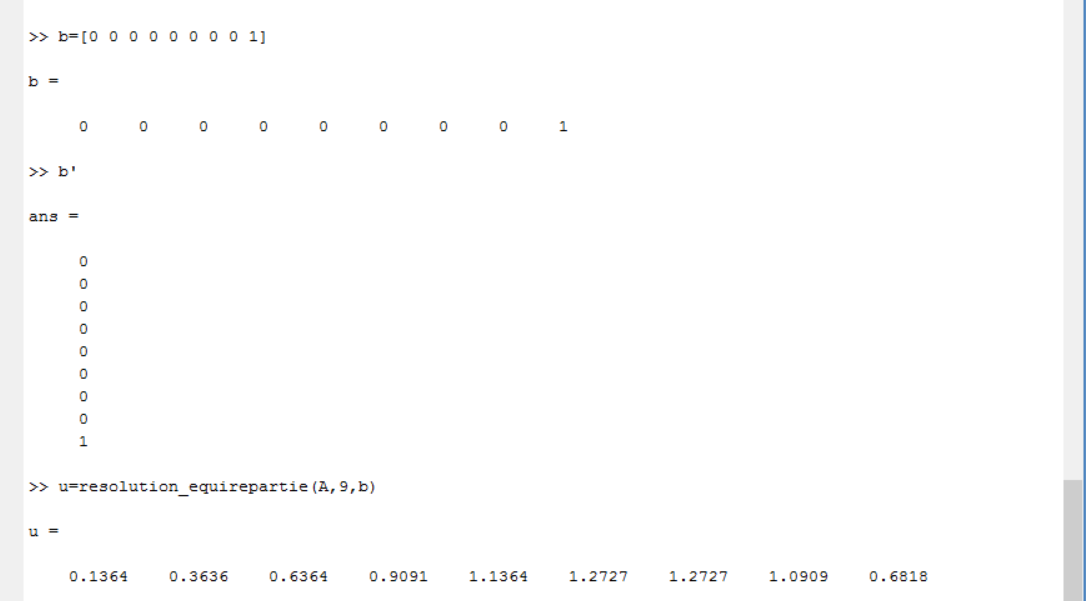




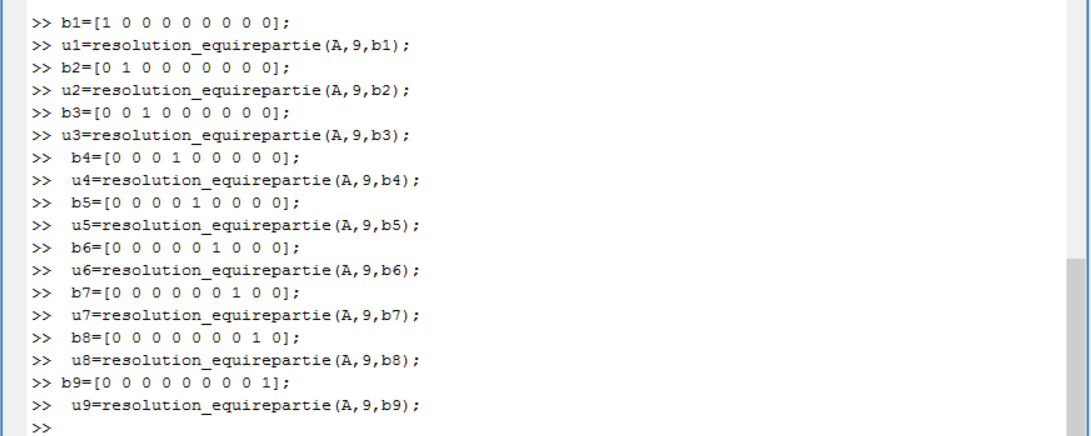


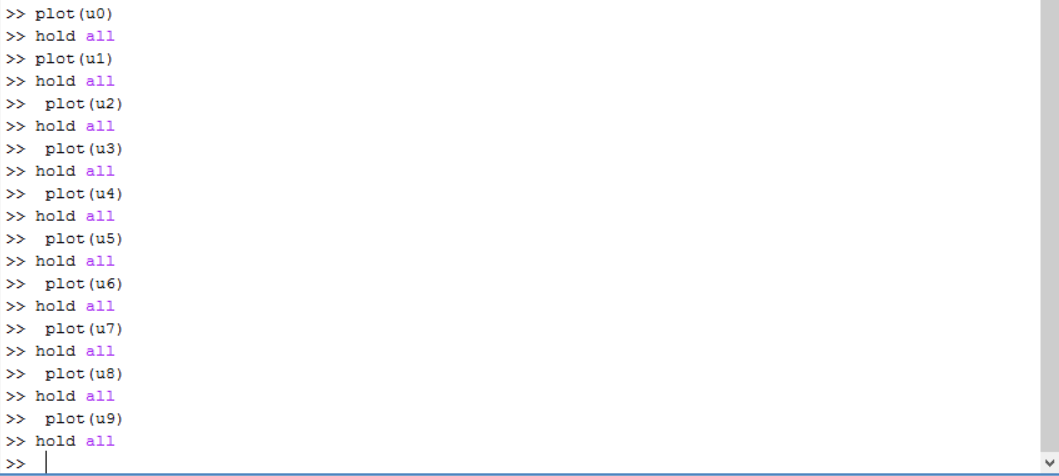






2.3. Traçage des flexions de la poutre :







2.4. La fonction cout\_calcul(n) calcule le nombre d’opérations élémentaires pour la résolution du système linéaire (3) .

**Algorithme de la fonction :**

function [add,div,mul]=cout\_calcul(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;

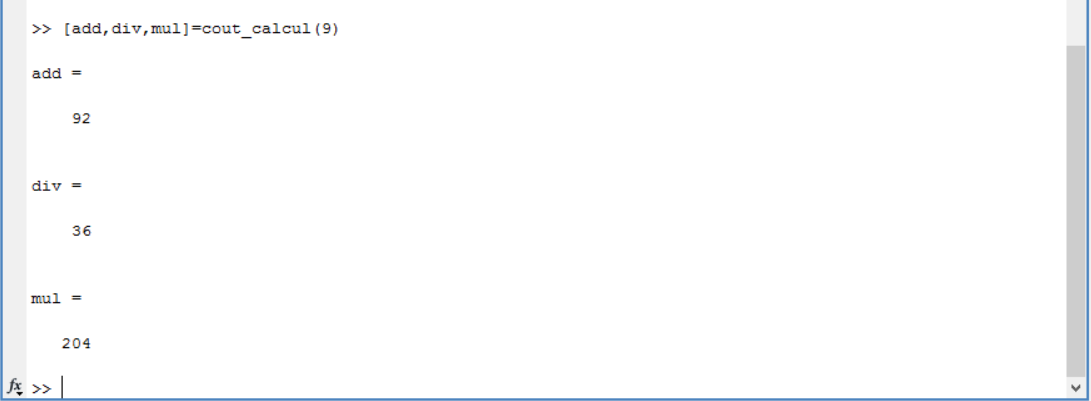
for p=1:n

div=div+(n-p);

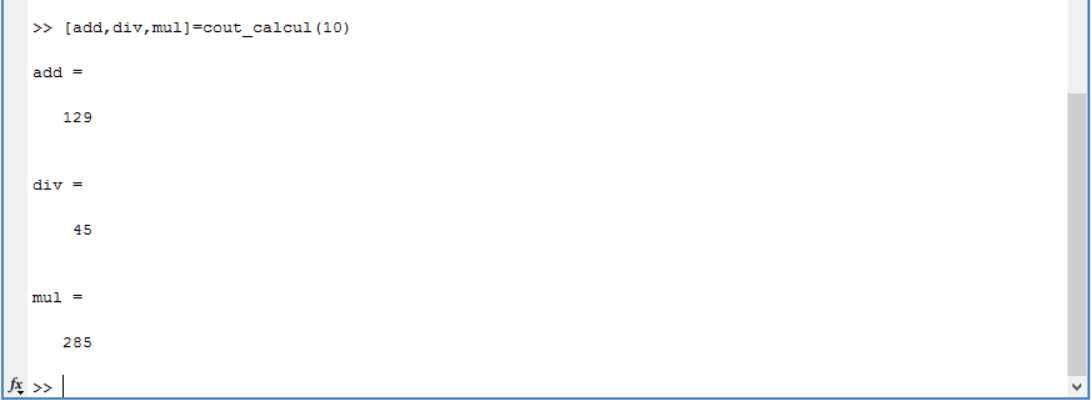
end;

end

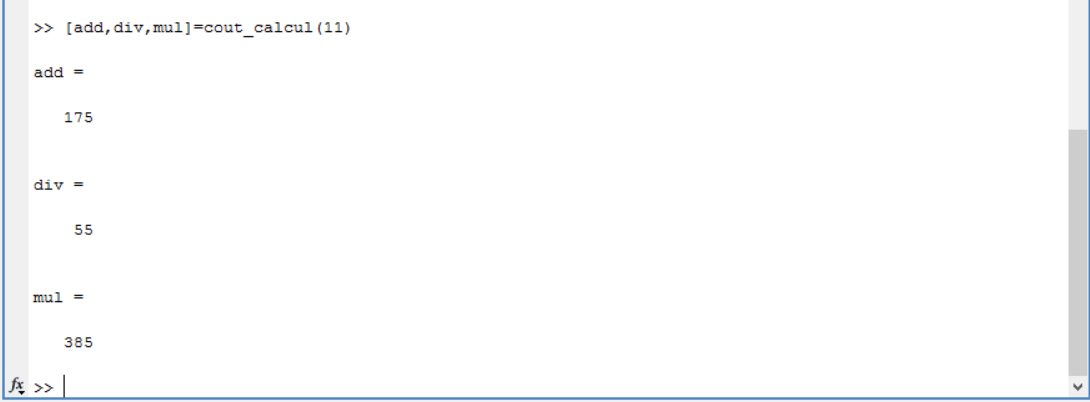
**\*\*Pour n=9 :**

****

**\*\*Pour n=10 :**

****

**\*\*Pour n=11 :**

****