

—— Vabiration Libres d'une Poutre en Flexion
——

par
GAYE, Ibrahima
ZHANG, Xunjie
Compte-rendu du Ex.31 de l'UE Outil de Mathématique

fait le 28 octobre 2016

Table des matières

1	Introduction et Séparation des variables	3
1.1	Introduction	3
1.2	Séparation des variables	3
1.2.1	Équation d'Équilibre	3
1.2.2	Séparations	3
1.3	Solution des fonctions	4
1.3.1	équation différentielles ordinaires	4
1.3.2	recherche la solution	4
1.4	Solution finalement	5
2	Résolution dans le cas de la poutre encastrée-appuie	6
2.1	conditons limites	6
2.1.1	extrémité en appui simple à l'abscisse $x=L$	6
2.1.2	Encastrement à l'abscisse $x=0$	6
2.1.3	Résultat par les conditions limites	7
2.2	Infinité de solutions	7
2.3	Programme	8
3	Résultats et Conclusions	9
3.1	Résultats	9
3.1.1	Solution approchée pour $N=5$	9
3.1.2	Étude pour N croissant	11
3.1.3	Influence du polynôme d'interpolation	13
3.2	Conclusion	13
A	Code Python	15
A.1	Listes des fonctions	15
A.1.1	Paramètres du problème	16
A.2	Étude pour $N=5$	16
A.2.1	Solution approchée pour $N=5$	16
A.2.2	Erreur relative pour $N=5$	17
A.3	Étude pour différents N	17
A.3.1	Solutions Approchées pour différents N	17
A.3.2	Erreur relative en fonction de h	18
A.3.3	Erreur relative en fonction de N	18
A.4	Étude de l'erreur relative en fonction du polynôme interpolation	19

Table des figures

2.1	infinité points	7
3.1	Solution approchée pour $N = 5$	9
3.2	Erreur relative en chaque nœud pour $N = 5$	10
3.3	Solutions Approchées pour un nombre de nœud croissant	11
3.4	Évolution de l'erreur relative en fonction de la finesse du maillage h	12
3.5	Évolution de l'erreur relative en fonction du nombre de nœuds h	13

Chapitre 1

Introduction et Séparation des variables

1.1 Introduction

On ne traitera que ici que le problème d'une poutre droite , de section uniforme , en flexion simple dans un plan principal . Ce pendant , le plus souvent , la poutre n'est pas contrainte à rester dans ce plan , et il existe une possibilité de déplacement dans la direction perpendiculaire . Si les excitations ne sont pas contenues dans un plan principal , il faut étudier les vibrations de flexion , dans les 2 plans principaux définis chacun par l'axe de la poutre et l'une des directions principales de la section .

1.2 Séparation des variables

1.2.1 Équation d'Équilibre

Pour une poutre droite de section constante en flexion , et considérons un effort extérieur t_{ext} nul , on a l'équation :

$$\frac{\partial^2 v}{\partial t^2} = -\frac{El}{\rho S} \frac{\partial^4 v}{\partial x^4} \quad (1.1)$$

Posons : $v(x, t) = \phi(x)q(t)$ et reportons cette expression dans l'équation du mouvement :

$$\phi \frac{d^2 q}{dt^2} = -\frac{El}{\rho S} \frac{d^4 \phi}{dx^4} q \quad (1.2)$$

1.2.2 Séparations

Séparons les termes qui dépendent de t et de x :

$$\frac{1}{q} \frac{d^2 q}{dt^2} = -\frac{El}{\rho S} \frac{d^4 \phi}{dx^4} \frac{1}{\phi} \quad (1.3)$$

1.3 Solution des fonctions

Les 2 membres de cette équation sont indépendants l'un de x , l'autre de t ; ils sont donc constants. Pour que la solution $q(t)$ reste bornée quand le temps tend vers l'infini, cette valeur constante doit être négative.

Trouver $q(t)$ et $\phi(x)$ quand $-\omega^2$ est constant :

$$\frac{1}{q} \frac{d^2 q}{dt^2} = -\frac{El}{\rho S} \frac{d^4 \phi}{dx^4} \frac{1}{\phi} = -\omega^2 \quad (1.4)$$

1.3.1 équation différentielles ordinaires

On en tire 2 équations différentielles ordinaires, l'une en x , l'autre en t , que l'on résout séparément :

$$\begin{cases} \frac{d^4 \phi}{dx^4} - \omega^4 \frac{\rho S}{EI} \phi = 0 \\ \frac{d^2 q}{dt^2} + \omega^2 q = 0 \end{cases} \quad (1.5)$$

1.3.2 recherche la solution

On recherche la solution de l'équation sous la forme harmitienne suivante : $\phi(x) = ae^{sx}$, d'où l'équation caractéristique, de degré 4 en S :

$$s^4 - \omega^4 \frac{\rho S}{EI} = 0 \quad (1.6)$$

qui a solutions :

$$s = \pm \sqrt[4]{\frac{\rho S}{EI} \omega^2}, \quad s = \pm i \sqrt[4]{\frac{\rho S}{EI} \omega^2} \quad (1.7)$$

que l'on peut aussi écrire en fonction d'un paramètre γ sans dimension :

$$s = \pm \frac{\gamma}{L}, \quad s = \pm i \frac{\gamma}{L}, \quad \gamma = L \sqrt[4]{\frac{\rho S}{EI} \omega^2} \quad (1.8)$$

On en déduit l'expression de la forme modale :

$$\phi(x) = ae^{\gamma \frac{x}{L}} + be^{-\gamma \frac{x}{L}} + ce^{i\gamma \frac{x}{L}} + de^{-i\gamma \frac{x}{L}} \quad (1.9)$$

ou encore :

$$\phi(x) = A \sinh\left(\gamma \frac{x}{L}\right) + B \cosh\left(\gamma \frac{x}{L}\right) + C \sin\left(\gamma \frac{x}{L}\right) + D \cos\left(\gamma \frac{x}{L}\right) \quad (1.10)$$

On recherche l'autre solution de l'équation sous la forme harmitienne suivante : $q(t) = ae^{bt}$, d'où l'équation caractéristique, de degré 2 en t :

$$b^2 + \omega^2 = 0 \quad (1.11)$$

qui a 2 solutions :

$$b = \pm i\omega, \quad \omega = \frac{\gamma^2}{L^2} \sqrt{\frac{EI}{\rho S}} \quad (1.12)$$

Donc on en déduit l'expression de la forme modale :

$$q(t) = \hat{e}e^{i\omega t} + \hat{f}e^{-i\omega t} \quad (1.13)$$

Ou encore :

$$q(t) = E \sin \omega t + F \cos \omega t, \quad \omega = \frac{\gamma^2}{L^2} \sqrt{\frac{EI}{\rho S}} \quad (1.14)$$

1.4 Solution finalement

On obtient donc finalement :

$$v(x, t) = \phi(x)q(t) \quad (1.15)$$

avec :

$$\begin{cases} \phi(x) &= A \sinh(\gamma \frac{x}{L}) + B \cosh(\gamma \frac{x}{L}) + C \sin(\gamma \frac{x}{L}) + D \cos(\gamma \frac{x}{L}) \\ q(t) &= E \sin \omega t + F \cos \omega t \\ \omega &= \frac{\gamma^2}{L^2} \sqrt{\frac{EI}{\rho S}} \end{cases} \quad (1.16)$$

- $\phi(x)$ définit la forme de la poutre pendant sa vibration . Elle contient les 5 constantes A, B, C et γ (ou ω) , qui sont fixées par les conditions aux limites .
- $q(t)$ définit le mouvement de la poutre . Elle contient , outre la constante ω déjà évoquée , les 2 constantes E et F , qui sont fixées par les conditions initiales .

Chapitre 2

Résolution dans le cas de la poutre encastree-appuie

2.1 conditions limites

Il existe 2 conditions aux limites usuelles pour une poutre en flexion :

2.1.1 extrémité en appui simple à l'abscisse $x=L$

La flèche est nulle en tout instant , ainsi que le moment fléchissant :

$$\forall t \begin{cases} v(L, t) &= 0 \\ \frac{\partial^2 v}{\partial x^2}(L, t) &= 0 \end{cases} \quad (2.1)$$

On a :

$$\begin{cases} \phi(L) &= A \sinh \gamma + B \cosh \gamma + C \sin \gamma + D \cos \gamma = 0 \\ \frac{d^2 \phi}{dx^2}(L) &= \frac{\gamma^2}{L^2} A \sinh \gamma + B \frac{\gamma^2}{L^2} \cosh \gamma - C \frac{\gamma^2}{L^2} \sin \gamma - D \frac{\gamma^2}{L^2} \cos \gamma = 0 \end{cases} \quad (2.2)$$

Et on en déduit :

$$\begin{cases} A \sinh \gamma + B \cosh \gamma &= 0 \\ C \sin \gamma + D \cos \gamma &= 0 \end{cases} \quad (2.3)$$

2.1.2 Encastrement à l'abscisse $x=0$

La flèche est nulle en tout instant , ainsi que la pente de la poutre :

$$\forall t \begin{cases} v(0, t) &= 0 \\ \frac{\partial v}{\partial x}(0, t) &= 0 \end{cases} \quad (2.4)$$

d'où , on a :

$$\begin{cases} \phi(0) &= B + D = 0 \\ \frac{d\phi}{dx}(0) &= A \frac{\gamma}{L} + C \frac{\gamma}{L} = 0 \end{cases} \quad (2.5)$$

On en déduit :

$$\begin{cases} A + C = 0 \\ B + D = 0 \end{cases} \quad (2.6)$$

2.1.3 Résultat par les conditions limites

On cherche solution pour les 4 équations :

$$\begin{cases} A \sinh \gamma + B \cosh \gamma = 0 \\ C \sin \gamma + D \cos \gamma = 0 \\ A + C = 0 \\ B + D = 0 \end{cases} \quad (2.7)$$

On trouve simplement :

$$\begin{cases} \tanh \gamma - \tan \gamma = 0 \\ B = -A \tanh \gamma = -A \tan \gamma \end{cases} \quad (2.8)$$

2.2 Infinité de solutions

On écrit une programme pour trouver les n valeurs , ce que on noté γ_n , et un tracé graphique simple permet d'apprécher les solutions . En effet , l'équation peut s'écrire :

$$\tanh \gamma = \tan \gamma \quad (2.9)$$

Les solutions sont données par les intersections des courbes $\tanh \gamma$ et $\tan \gamma$:

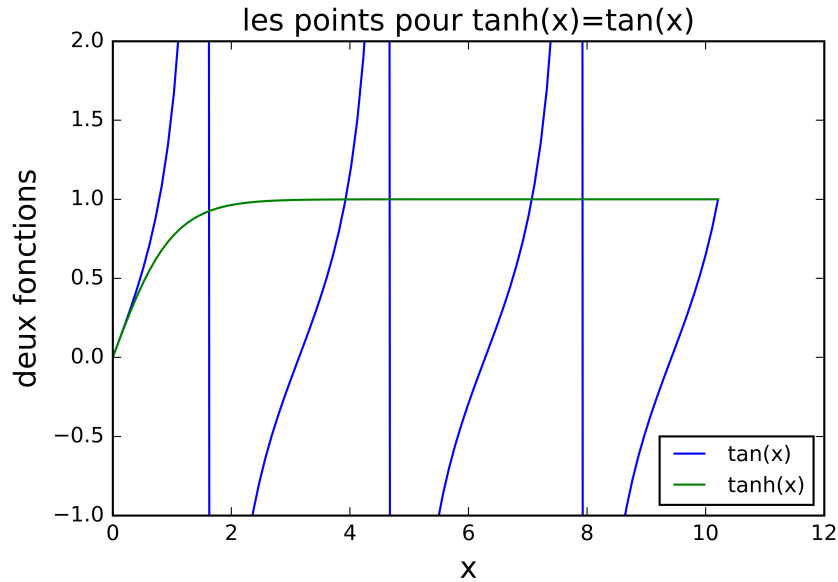


FIGURE 2.1 – infinité points

On constate qu'il existe une infinité de solutions, voisines de $\pi/4$

La construction de la matrice A et du vecteur B se font de manière relativement simple. En effet, nous devons calculer A^k et B^k pour chaque élément, puis les assembler, avant de résoudre le système. La façon la plus simple est de remplir A et B , en incrémentant la valeur de k .

On retrouve cette forme :

$$A^k = \begin{bmatrix} A_1^k 1 + A_2^{k-1} 2 & A_1^k 2 \\ A_1^k 2 & A_2^k 2 + A_1^{k+1} 1 \end{bmatrix}, k \neq 0, k \neq N$$

On tiendra compte des valeurs limites et de la particularité des valeurs de A_{11}^1, B_{11}^1 et B_{12}^N qui ne requièrent pas d'addition avec des valeurs voisines. Une fois le système implémenté, sa résolution et la comparaison avec les valeurs analytiques seront relativement simples.

2.3 Programme

Le programme se décompose en trois parties. Nous avons d'abord défini les fonctions qui nous permettront de construire notre système, ainsi que la solution analytique. Puis, nous avons étudié la solution approchée pour des paramètres donnés et avec un maillage avec 5 éléments ($N = 5$). Nous avons terminé le programme avec une étude de l'erreur relative en un point donné en fonction de la variation du nombre de nœuds.

Le programme fonctionne très bien et est joint en annexe. Il sauvegarde automatiquement les courbes tracées.

Chapitre 3

Résultats et Conclusions

3.1 Résultats

3.1.1 Solution approchée pour $N=5$

Courbe approchée

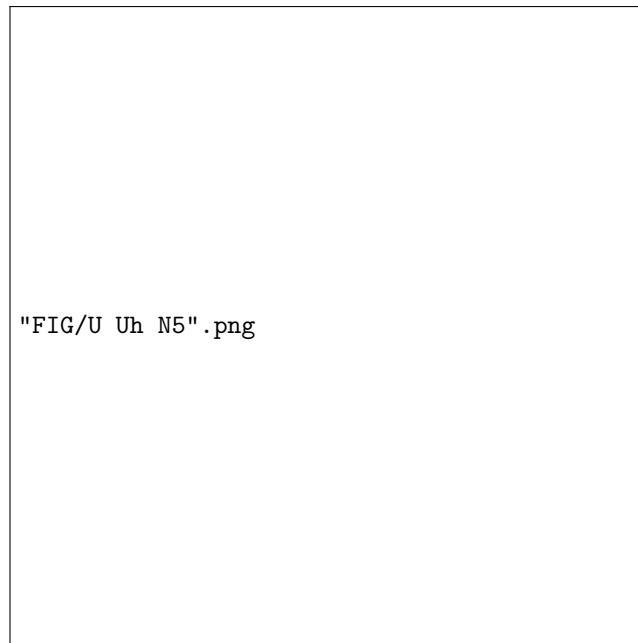


FIGURE 3.1 – Solution approchée pour $N = 5$

Les valeurs sont de l'ordre de grandeur des déplacements possibles dans nos modèles physiques connues. On remarque très bien que les valeurs trouvées numériquement sont très proches de celle de la courbe analytique en rouge. On peut aussi voir que si on fait une interpolation polynomiale d'ordre 1, c'est-à-

dire relier tous les points bleus par des droites, la forme de la solution approchée sur des valeurs autre que celle des nœuds reste inexacte et ne correspond pas à la forme de la solution analytique.

Erreur relative en chaque nœud

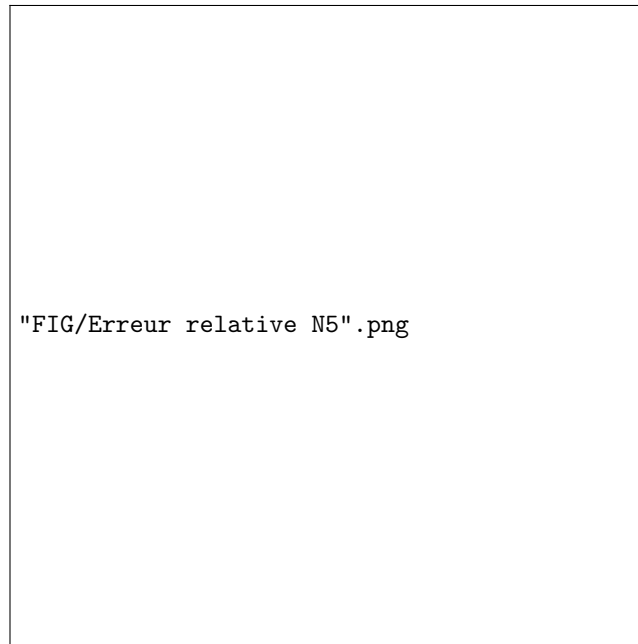


FIGURE 3.2 – Erreur relative en chaque nœud pour $N = 5$

Sur chaque nœud, nous avons calculé l'erreur relative de la solution approchée

$$Err = \frac{\|u - u_h\|}{\|u\|}$$

On remarque qu'elle est très faible et qu'elle diminue légèrement en fonction de r . On voit aussi que pour le premier nœud, ou $r = 0$, nous avons un écart important, même si les valeurs des solutions analytique et approchée tendent vers 0. Cette erreur numérique est due au fait que l'erreur relative ne peut être calculée en ce point. Ce que nous avons sont des valeurs très proches de 0, mais non nulles que l'ordinateur calcul quand même. On peut ne pas tenir compte de cet effet de bord.

3.1.2 Étude pour N croissant

Solutions Approchées

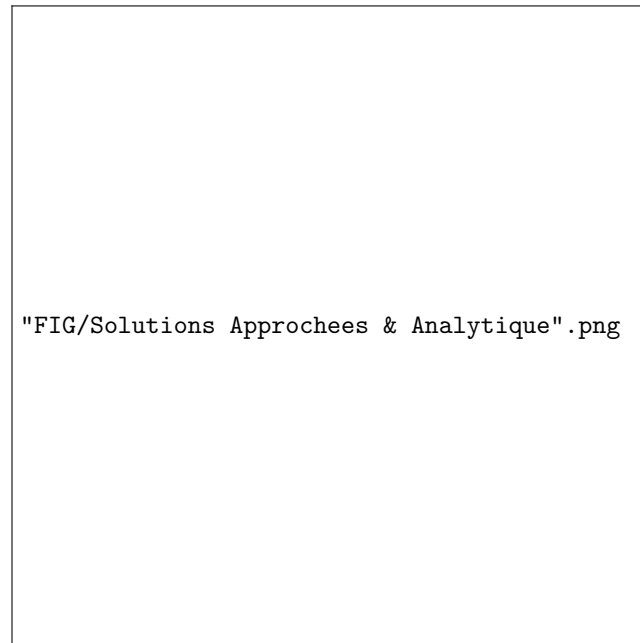


FIGURE 3.3 – Solutions Approchées pour un nombre de nœud croissant

On remarque que très rapidement, les courbes des solutions approchées se confondent avec la courbe de la solution analytique. On peut déjà prétendre que la solution approchée converge vers la solution exacte quand le nombre de nœuds augmente.

Erreur relative

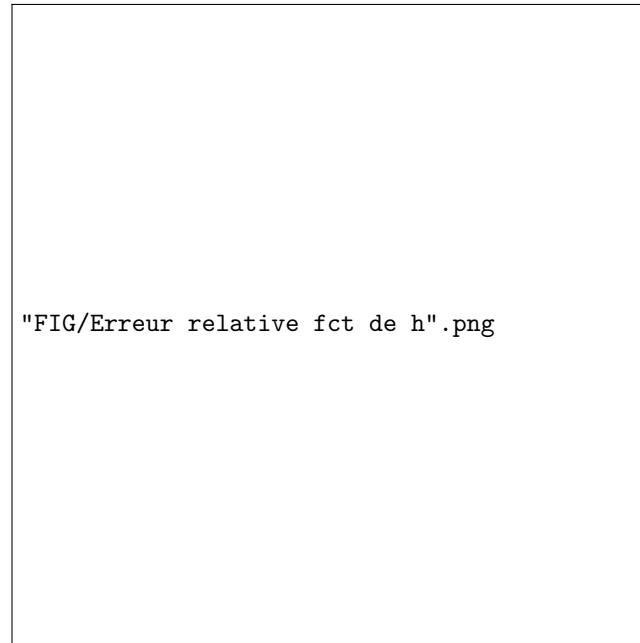
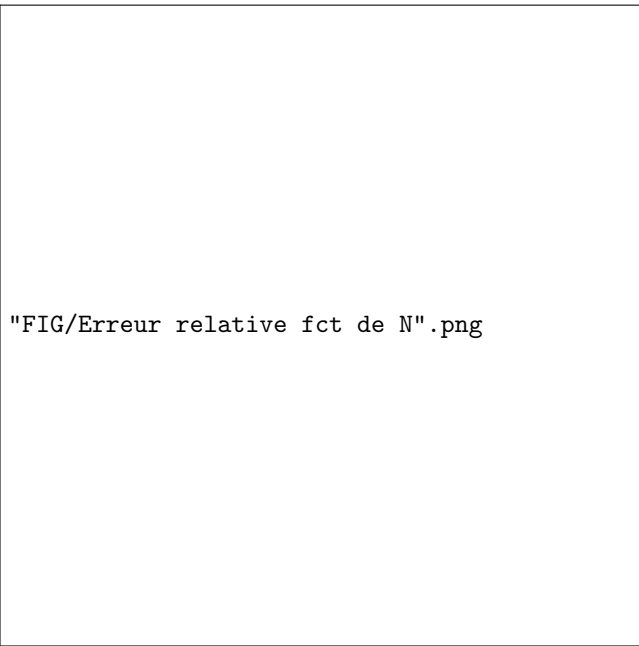


FIGURE 3.4 – Évolution de l'erreur relative en fonction de la finesse du maillage h



"FIG/Erreur relative fct de N".png

FIGURE 3.5 – Évolution de l'erreur relative en fonction du nombre de nœuds h

Dans cette partie nous avons calculé et tracé l'erreur relative en fonction de l'évolution de h . L'erreur relative a été calculée au nœud $r_n = L$, à l'extrémité de la pale. On remarque que l'erreur diminue si la finesse du maillage augmente. Cette diminution n'est pas linéaire. On peut en déduire qu'à partir d'un certain seuil, l'augmentation de la finesse ne sera plus intéressante d'un point de vue numérique. En effet, en augmentant le maillage, nous augmentons les efforts de calcul de la machine. À partir d'un certain seuil, le gain en précision devient négligeable, comme la courbe tend vers zéro. Se pose alors la question de la capacité et la durée du calcul fait par la machine. Cette durée est aussi dépendante de la complexité du calcul, dans les cas en $2D$ et $3D$.

3.1.3 Influence du polynôme d'interpolation

Malheureusement, par manque de temps, nous n'avons pas pu tester avec une interpolation polynomiale de degré 2. La solution approchée aurait été plus proche de la solution analytique, notamment par le fait que le nombre de nœuds augmente dans ce cadre. Nous n'avons pas non-plus étudié l'erreur relative en tout point x . Cette erreur est plus importante entre deux nœud. En l'étudiant, nous aurions pu comparer l'influence du polynôme d'interpolation dans l'approximation de la solution.

3.2 Conclusion

Dans ce TP, nous avons pu programmer une méthode simple et rapide d'approximation d'équations différentielles linéaires d'ordre deux, grâce aux

méthodes d'éléments finis et notamment la formulation faible. La solution approchée converge bien vers la solution analytique quand la finesse du maillage augmente. Nous avons pu comparer les résultats avec la solution analytique qui était connue. Dans un cas où la solution n'est pas connue, nous n'avions plus de moyen analytique pour comparer les résultats. En expérimentant physiquement, les données pourront être comparées et un modèle peut en être déduit. Dans les deux cas, le modèle en éléments finis doit tenir compte de la précision recherchée et surtout de la puissance de calcul nécessaire. Les méthodes des éléments finis sont des outils de calculs très intéressants tant que l'on garde en tête les limites des modèles et des calculs numériques.

Annexe A

Code Python

A.1 Listes des fonctions

```
def section(L,N,k,S0,SL): #Focion S(r)
    a=(SL-S0)/L
    b=S0
    return ((a*(L/N)*(k-1)+b)+a*(L/N)*k+b)/2

def matrixA(E,L,N,S0,SL): #Algorithme de construction de la Matrice A
    A=np.zeros((N+1,N+1))
    h=L/N
    S1=section(L,N,1,S0,SL)
    SN=section(L,N,N,S0,SL)
    A[0,0]= (E*S1)/h
    A[0,1]= -(E*S1)/h
    A[1,0]= -(E*S1)/h
    A[1,1]= (E*S1)/h
    i=1
    while i<N:
        S=section(L,N,i,S0,SL)
        A[i,i]=A[i,i]+(E*S)/h
        A[i,i+1]=-(E*S)/h
        A[i+1,i]=-(E*S)/h
        A[i+1,i+1]=(E*S)/h
        i=i+1
    return A

def vecteurB(rho,omega,N,S0,SL): #Algorithme de construction du vecteur B
    B=np.zeros(N+1)
    h=L/N
    S1=section(L,N,1,S0,SL)
    SN=section(L,N,N,S0,SL)
    R=1*L/N #k=0 donc r1
```



```

B[0]=rho*omega*omega*S1*h*(R/3)
B[1]=rho*omega*omega*S1*h*(R/6)
i=1
while i<N:
    R1=i*L/N
    R2=(i+1)*L/N
    S=section(L,N,i,S0,SL)
    B[i]=B[i]+rho*omega*omega*S*h*(R1/3+R2/6)
    B[i+1]=rho*omega*omega*S*h*(R1/3+R2/6)
    i=i+1
return B

def u2(E,L,rho,omega,N,S0,SL,r): #Solution analytique pour S variable.
a=(SL-S0)/L
b=S0
y=((omega**2*rho)/(36*E*a**3))*(a*r*(-4*a**2*r**2-3*a*b*r+6*b**2)
    -6*(b-2*a*L)*((a*L+b)**2)*np.log(a*r+b)+6*np.log(b)*(b-2*a*L)*(
    a*L+b)**2)
return y

```

A.1.1 Paramètres du problème

```

#Paramettre du probleme:
S0=16.2
SL=6.7
L=51.5
err=0.076
rho=1600
E=21300*10**6
omega=2*np.pi
N=5 # Premiere etude avec N=5

```

A.2 Étude pour N=5

A.2.1 Solution approchée pour N=5

```

A=matrixA(E,L,N,S0,SL)
B=vecteurB(rho,omega,N,S0,SL)
#Condition limite:
A[0,0]=1
A[0,1]=0
B[0]=0
uh=np.linalg.solve(A,B) #Resolution du syteme : solution approchee pour
    N=5

x=np.linspace(0,L,100) #Axe horizontale pour la solution analytique
r=np.linspace(0,L,N+1) #Array avec tout les noeuds en fonction de N

```

```

u5=u2(E,L,rho,omega,N,S0,SL,x)

p1=plt.plot(r,u5,marker='o',label='Solution Approchee Uh')
plt.plot(x,u2(E,L,rho,omega,N,S0,SL,x),'r',label='Solution Analytique U
')
plt.legend(loc='lower right')
plt.xlabel('r',fontsize=12)
plt.ylabel('U & Uh',fontsize=12)
plt.title("Solutions Approchees & Analytique pour N="+str(N), fontsize
=14)
plt.savefig('D:\Google Drive — Mohamed\Cours\S1\Element Fini\TP 1\U Uh
N5.png',dpi=800)

```

A.2.2 Erreur relative pour N=5

```

ur=u2(E,L,rho,omega,N,S0,SL,r) #On calcul les valeurs theoriques pour
différents noeuds (array "r")
error=np.absolute(ur-uh)/ur #formule pour chaque point

plt.plot(r,error*100,marker='o') #pourcentage
#plt.axhline(y=err*100, hold=None)
plt.ylabel('$\epsilon$ : erreur relative en %',fontsize=12)
plt.xlabel('$h$',fontsize=12)
plt.title("Erreur relative a differents noeuds pour N="+str(N),
fontsize=14)
plt.savefig('D:\Google Drive — Mohamed\Cours\S1\Element Fini\TP 1\
Erreur relative N5.png',dpi=800)

```

A.3 Étude pour différents N

A.3.1 Solutions Approchées pour différents N

#Paramettre du probleme:

```

S0=16.2
SL=6.7
L=51.5
err=0.076
rho=1600
E=21300*10**6
omega=2*np.pi
Nmin=4 #nombre de noeud minimal
Nmax=20 #nombre de noeud maximal

for i in range (Nmin,Nmax+1,4):
    N=i
    x=np.linspace(0,L,100)
    r=np.linspace(0,L,N+1)

```

```

A=matrixA(E,L,N,S0,SL)
B=vecteurB(rho,omega,N,S0,SL)
#Condition limite:
A[0,0]=1
A[0,1]=0
B[0]=0
uh=np.linalg.solve(A,B)
plt.plot(r,uh,label='N='+str(N),)
plt.plot(x,u2(E,L,rho,omega,N,S0,SL,x),'k',label='Solution Analytique')
plt.legend(loc='lower right')
plt.xlabel('$r$', fontsize=12)
plt.ylabel('$u_{i}$', fontsize=12)
plt.title("Solutions Approchees & Analytique", fontsize=14)
plt.savefig('D:\Google Drive — Mohamed\Cours\S1\Element Fini\TP 1\
Solutions Approchees & Analytique.png',dpi=800)

```

A.3.2 Erreur relative en fonction de h

```

error=np.zeros(Nmax)
H=np.zeros(Nmax)
for i in range (Nmin,Nmax):
    N=i
    h=L/N
    x=np.linspace(0,L,100)
    r=np.linspace(0,L,N+1)
    A=matrixA(E,L,N,S0,SL)
    B=vecteurB(rho,omega,N,S0,SL)
    #Condition limite:
    A[0,0]=1
    A[0,1]=0
    B[0]=0
    uh=np.linalg.solve(A,B)
    U=u2(E,L,rho,omega,N,S0,SL,r)
    error[i-Nmin]=(np.absolute(U[N]-uh[N])/U[N])
    H[i-Nmin]=h
plt.plot(H,error,label='$Err(h)$')
#plt.axhline(y=err, hold=None) #tracer ligne pour errmax
plt.ylabel('$\epsilon$ : erreur relative en $\%$ en fonction de N',
    fontsize=12)
plt.xlabel('h', fontsize=12)
plt.title("Erreur relative au noeud $r_{n}=L$", fontsize=14)
plt.savefig('D:\Google Drive — Mohamed\Cours\S1\Element Fini\TP 1\
Erreur relative fct de h.png',dpi=800)

```

A.3.3 Erreur relative en fonction de N

```

plt.plot(L/H,error*100,label='$Err(N)$')

```

```
plt.ylabel('$\epsilon$ : erreur relative en $\%$ en fonction de N',  
          fontsize=12)  
plt.xlabel('N', fontsize=12)  
plt.title("Erreur relative au noeud  $r_n=L$ ", fontsize=14)  
plt.savefig('D:\Google Drive — Mohamed\Cours\S1\Element Fini\TP 1\  
          Erreur relative fct de N.png', dpi=800)
```

A.4 Étude de l'erreur relative en fonction du polynôme interpolation
