—— Méthodes des Éléments Finis en 1D ——

par
MAAMIR, Mohamed
ZHANG, Xunjie

Compte-rendu du TP 1 de l'UE Méthodes des Éléments Finis

fait le 22 octobre 2016

Table des matières

Table des figures

Chapitre 1

Introduction et Modélisation par Éléments Finis

1.1 Introduction

Dans le cadre de ce premier TP de MEF, nous avons étudié le déplacement des éléments d'une pale (de section variable)en rotation constante et soumise seulement à la force centrifuge. Dans un premier temps, nous avons fait l'étude mathématique du modèle et trouvé une solution analytique des déplacements $u(r)$.Par la suite, grâce aux méthodes des éléments finis en 1D, nous avons construit un système linéaire à résoudre avec les outils numériques. Nous avons programmé sur Python un algorithme permettant de construire les éléments du système à résoudre pour trouver les valeurs approchées de $u(r)$ sur différents points du maillage. La résolution de ce système de manière numérique nous a permis de construire cette approximation. Par la suite, nous avons étudié les erreurs relatives et l'effet du changement de la densité du maillage dans la solution trouvée. Ce compte-rendu décrit précisément les étapes de résolutions analytiques et numériques. De même, nous y analysons les résultats trouvés.

1.2 Problème Physique

1.2.1 Équation d'Équilibre

Un petit élément de la pale de largeur δr et de section $S(r)$ est soumis aux actions suivantes :

1. Force centrifuge : $(\rho * S(r) * \omega^2 * r) * \Delta r$
2. Efforts internes : $ES(r+\Delta r)\frac{du}{dr}|_{r+\Delta r} - ES(r)\frac{du}{dr}|_r = \frac{d}{dr} * [ES(r)\frac{du}{dr}(r)]$

On retrouve donc à l'état d'équilibre l'équation suivante :

$$\frac{d}{dr}(ES(r)\frac{du}{dr}) + (\rho S(r)\omega^2)r = 0 \tag{1.1}$$

1.3 Formulation Faible

Pour une fonction test $v(r)$, on retrouve une formulation faible de la forme :

$$\text{Trouver } u(r) \text{ tel que } u(0) = 0 \text{ et } \frac{du}{dr}\Big|_{r=L} = 0 :$$

$$\int_0^L ES(r)\frac{du}{dr}\frac{dv}{dr}\ dr = \int_0^L \big(\rho S(r)\omega^2 r\big)v(r)\ dr \tag{1.2}$$

$$\forall v(0) = 0 \text{ et } \frac{dv}{dr}\big|_L = 0$$

1.3.1 Solution Analytique

L'équation d'équilibre est une **équation différentielle linéaire d'ordre deux**. Sa résolution est possible et est de manière relativement simple. De plus, nous avons **deux conditions limites** qui vont pouvoir nous aider à trouver les valeurs des constantes d'intégration.

1. Dirichlet : $u(0) = 0$
2. Neumann : $\frac{du}{dr}\Big|_{r=L} = 0$

Cas d'une Section Constante

Si on considère la section S_{const} comme étant constante, en intégrant l'équation d'équilibre (1.1) et en utilisant les conditions limites de Dirichlet et de Neumann, on retrouve une solution analytique de la forme :

$$u(r)_{S_{const}} = \frac{\rho\omega^2}{2 * E}\big(L^2 r - \frac{r^3}{3}\big) \tag{1.3}$$

Cas d'une Section non-constante

Dans notre TP, nous avons considéré la section comme étant variables linéairement et étant de la forme :

$$S = S(r) = a * r + b$$

avec

$$a = \frac{S(L) - S(0)}{L}, b = S(0)$$

En intégrant l'équation d'équilibre (1.1)avec les mêmes conditions limites que précédemment, on retrouve une solution analytique de la forme :

$$u(r)_{S(r)} = \frac{\omega^2\rho}{36a^3E}\Big(ar(-4a^2r^2 - 3abr + 6b^2) - 6(b - 2aL)(aL + b)^2\ln(ar + b) + 6\ln(b)(b - 2aL)(aL + b)^2\Big) \tag{1.4}$$

Résolu avec Wolfram Alpha® : This link

1.4 la forme d'approximation par interpolation

1.5 Expressions des matrices A et B

1.5.1 Fonctions de Bases

Pour un maillage de N éléments : $M^h = \bigcup_{i=1}^{N} e_k$. On peux considérer l'approximation recherchée comme une sommes de fonctions de base tel que :

$$u^h(x) = \sum_{i=0}^{n} u_i * \Phi_i(x) \tag{1.5}$$

On peut considérer les fonctions tests de la forme d'un polynôme de degré n. C'est fonction de base servent a définir l'allure de la courbe entre deux valeurs u_i. Ainsi, si on pose la fonction test $v^h(x) = \delta u^h(x) = \frac{\delta u^h}{\delta u_i}$, on retrouve

$$v^h(x) = \delta(\sum_{i=0}^{n} u_i * \Phi_i(x)) = \sum_{i=0}^{n} \Phi_i(x)$$

.

1.5.2 Fonctions de Formes

Les fonctions de forme donnent la forme d'un polynôme de degré n sur un élément de référence $\hat{e}$. Un changement de variable nous permet de passer des fonctions de base aux fonctions de forme. Ces fonctions de forme vont nous permettre de définir la courbe sur chaque élément de manière plus simple.

Le passage entre les deux notations se fait comme suit :

$$x = x_{k-1}\frac{1-\xi}{2} + x_k\frac{1+\xi}{2} \Leftrightarrow \xi = 2\frac{x - x_{k-1}}{x_k - x_{k-1}} - 1$$

Ordre 2

Comme vue dans le cours, l'approximation u^h sur un élément du maillage s'écrit :

$$u^h(x) = \sum_{i=0}^{nn} u_i \Phi_i(x)$$

Pour une approximation d'ordre 2, on découpe l'élément de référence $\hat{e}$ en 3 points d'interpolations $[-1,,0,1]$ qui représentent respectivement sur e_k les points $((n_1, n_2, n_3))$ tel que :

$$u_n^h(x) = u_{n_1}\Phi_{n_1}(x) + u_{n_2}\Phi_{n_2}(x) + u_{n_3}\Phi_{n_3}(x)$$

On retrouve la relation entre les fonctions de base $\Phi_{n_q}(x)$ et les fonctions de forme $N_q(\xi)$:

$$\Phi_{n_1}(x) = N_1(\xi), \Phi_{n_2}(x) = N_2(\xi), \Phi_{n_3}(x) = N_3(\xi)$$

avec

$$N_1(\xi) = \frac{\xi(\xi-1)}{2}, N_2(\xi) = \frac{\xi(\xi+1)}{2}, N_3(\xi) = 1 - \xi^2$$

Ordre 1

On retrouve la même démarche pour le cas d'ordre 1. Cette fois-ci, nous n'avons besoin que de deux nœuds ((n_1, n_2) sur sur e_k. Ainsi nous avons les fonctions de formes suivantes :

$$N_1(\xi) = \frac{1-\xi}{2} \text{ et } N_2(\xi) = \frac{1+\xi}{2}$$

On retrouve donc :

$$u_n^h(x) = u_{n_1}\Phi_{n_1}(x) + u_{n_2}\Phi_{n_2}(x)$$

1.5.3 Interpolation Polynomiale

Pour un polynôme d'ordre n tel que :

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0.$$

Et vérifiant

$$p(x_i) = y_i, \qquad \forall i \in \{0, 1, \ldots, n\}$$

On obtient un système linéaire à résoudre de la forme suivante :

$$\begin{pmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \ldots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \ldots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{1.6}$$

Interpolation d'ordre 1

Dans le cas d'une interpolation d'ordre 1, chaque nœud d'un maillage est relié au suivant par une fonction affine de la forme : $f_k(x) = \alpha * x + \beta$.

Pour un maillage M^h avec N éléments e_k et $N+1$ nœuds r_k, avec $k \in [0, N], k \in \mathbb{Z}$, nous pouvons facilement déduire la forme de l'interpolation sur chaque élément e_k :

$$u_k^h(x) = \Big(\frac{U^h(r_k) - U^h(r_{k-1})}{r_k - r_{k-1}}\Big)x + \big(\frac{r_k U^h(r_{k-1}) - r_{k-1}U^h(r_k)}{r_k - r_{k-1}}\Big) \tag{1.7}$$

Ainsi pour un élément

Interpolation d'ordre 2

Dans le cas d'une interpolation d'ordre 2, chaque nœud d'un maillage est relié au suivant par un polynôme d'ordre 2 de la forme : $f_k(x) = \alpha * x^2 + \beta * x + \gamma$. Le système à résoudre pour une élément e_k ($r \in [r_{k-1}, r_k]$) est de cette forme :

$$\begin{pmatrix} r_{k-2}^2 & r_{k-2} & 1 \\ r_{k-1}^2 & r_{k-1} & 1 \\ r_k^2 & r_k & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} U^h(r_{k-2}) \\ U^h(r_{k-1}) \\ U^h(r_k) \end{pmatrix} \tag{1.8}$$

Nous devons ajouter un troisième point (r_{k-1}) à l'intervalle pour résoudre ce système. Ainsi, en augmentant la densité du maillage tel que : $N_2 = 2 * N + 1$,

nous pouvons construire une interpolation d'ordre deux sur chaque élément. Si notre calcul numérique se fait nœud par nœud, le polynôme trouvé tiendra compte de la valeur précédente et ainsi de suite. La courbe finale sera ainsi continue.

1.5.4 Système linéaire pour un élément k

Soit la solution approchée :

$$u^h(x) = \sum_{i=0}^{n} u_i \Phi_i(x)$$

Si on pose pour

$$v_i(x) = \Phi_i(x)$$

Alors

$$\frac{\delta u^h}{\delta x} = \sum_{i=0}^{n} u_i \frac{\delta \Phi_i}{\delta x}$$

Et en partant de la formulation faible, nous trouvons ce système :

$$\Big[\int_0^L ES(r) \frac{d\Phi_i}{dr} \frac{\Phi_j}{dr} \, dr \Big] * u_j = \int_0^L \big(\rho S(r) \omega^2 r\big) \Phi_i \, dr \tag{1.9}$$

On retrouve un système linéaire $A * \vec{U} = \vec{B}$, avec :

$$dim(A) = (N+1) \times (N+1), dim(U) = dim(B) = (N+1)$$

Ainsi, connaissant l'équation d'équilibre sur un élément e_k (1.1) et sa formulation faible (1.2), nous pouvons écrire, que pour un élément e_k, la matrice A^k est de la forme 2×2 et s'écrit de cette façon :

$$A_{ij}^k = \int_{e_k} ES(r) \frac{d\Phi_{ki}}{dr} \frac{\Phi_{kj}}{dr} \, dr \tag{1.10}$$

Si on considère $S(r)|_{e_k} = \frac{S(r_k)+S(r_k+1)}{2}$ constante sur l'élément, alors :

$$A_{ij}^k = ES_{e_k} \int_{e_k} \frac{d\Phi_{ki}}{dr} \frac{\Phi_{kj}}{dr} \, dr \tag{1.11}$$

Cas d'ordre 1

Dans le cas d'ordre 1, nous avons :

$$\frac{d\Phi_{k1}}{dr} = \frac{1}{N_2 - N_1} = \frac{1}{h_k}$$

et

$$\frac{d\Phi_{k2}}{dr} = \frac{-1}{N_2 - N_1} = \frac{-1}{h_k}$$

Soit :

$$A^k = \frac{ES_{e_k}}{h_k} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{1.12}$$

On procède de la même façon pour le membre de droite de l'équation 1.9 pour trouver B^k

$$B^k = S_{e_k} h_k \rho \omega^2 \begin{bmatrix} 1/3 & 1/6 \\ 1/6 & 1/3 \end{bmatrix} * \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \tag{1.13}$$

Cas d'ordre 2

Chapitre 2

Résolution numérique

L'objectif du programme est de construire puis de résoudre un système linéaire fermé. Pour ce faire, nous avons préalablement trouvé la forme du système ainsi que ses conditions limites. Par la suite, nous étudierons l'influence du maillage dans l'approximation des valeurs analytiques. Faute de temps, nous n'avons pas pu étudier le cas l'interpolation est d'ordre 2. Néanmoins, notre programme peut être facilement utilisé dans ce cas, les seuls éléments qui changeront seront le nombre de nœuds et la forme analytique de A et B. Le Programme a été codé en *Python 2.7*.

2.1 Algorithme

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 B, en incrémentant la valeur de k.

En 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 requière pas d'addition avec des valeurs voisines. Une fois le système implémenté, sa résolution et la comparaison avec les valeurs analytique seront relativement simples.

2.2 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 donnée 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

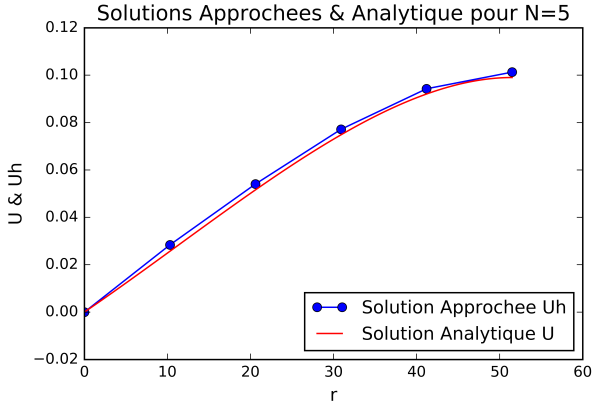

FIGURE 3.1 – Solution approchée pour $N = 5$

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

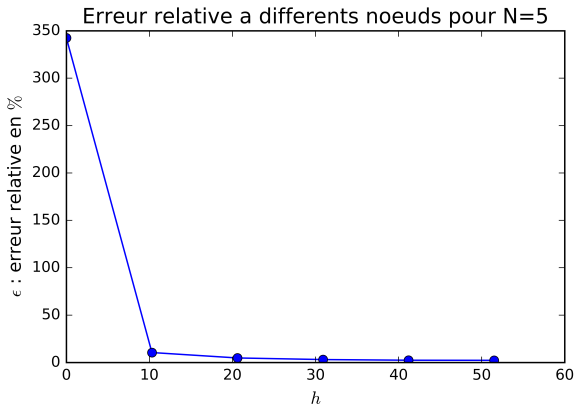

FIGURE 3.2 – Erreur relative en chaque nœud pour $N = 5$

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

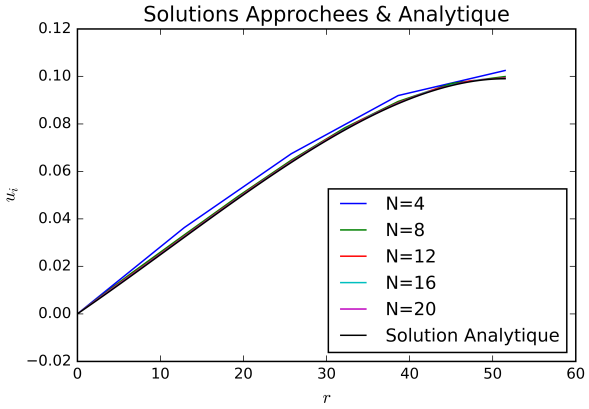

FIGURE 3.3 – Solutions Approchées pour un nombre de nœud croissant

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

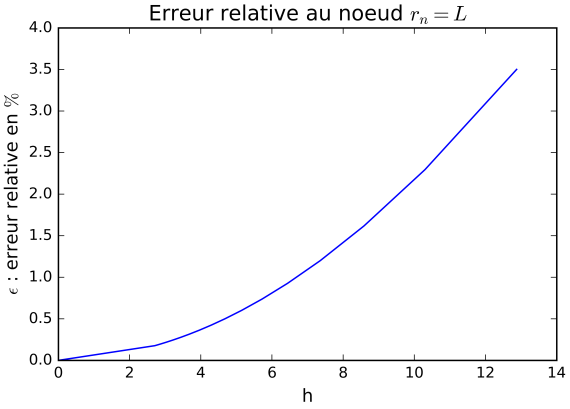

FIGURE 3.4 – Évolution de l'erreur relative en fonction de la finesse du maillage h

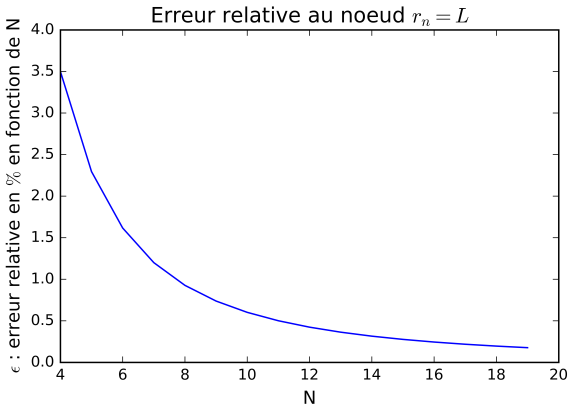

FIGURE 3.5 – Évolution de l'erreur relative en fonction du nombre de nœuds h

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 était 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): #Foction 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=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

```
#Parametre 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,uh,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
    differents 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

```
#Parametre 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