MAGHOUS Abdellah et Ludovic Planchon

Master 1 : Statistiques et données du vivants

Rapport de TP :Optimisation

Année universitaire : 2021/2022

1 Partie théorique

1.0.1 Question 1

On a A_h une matrice de taille J × J

$$A_h = 1/h \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \vdots & \vdots \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & -0 & -1 & 2 \end{pmatrix}$$

$$\begin{aligned} u_h^t A_h u_h = &= (u_1, u_2, ...u_j) A_h (u_1, u_2, ...u_j)^t \\ &= \frac{1}{h}((2u_1 - u_2, -u_1 + 2u_2 - u_3, \cdots, -u_{J-2} + 2u_{J-1} - uj, -u_{J-1} + 2u_J)u_h) \\ &= \frac{1}{h}(u_1(2u_1 - u_2) + ... + u_{J-1}(-u_{J-2} + 2u_{J-1} - uj) + u_J(-u_{J-1} + 2u_J)) \\ &= \frac{1}{h}(2u_1^2 - 2u_1u_2 + 2u_2^2 - 2u_3u_2 + 2u_3^2... - 2u_{J-2}u_{J-1} + 2u_{j-1}^2 - 2u_Ju_{J-1} + 2u_J^2) \\ &= \frac{1}{h}(u_1^2 + (u_2 - u_1)^2 + (u_3 - u_2)^2....(u_{J-1} - u_{J-2})^2 + (u_J - u_{J-1})^2 + u_J^2) \\ &= \frac{1}{h}(u_1^2 + u_J^2 + \sum_{j=1}^{J-1}(u_{j+1} - u_j)^2) \end{aligned} \tag{1}$$

1.0.2 Question 2

Soit la fonction d'énergie $E_h(u_h) = \frac{\epsilon}{2h}(u_1^2 + u_J^2 + \sum_{j=1}^{J-1}(u_{j+1} - u_j)^2) + \sum_{j=1}^{J} F(u_j)$ d'après (1) on en déduit que $E_h(u_h) = \epsilon u_h^t A_h u_h + \sum_{j=1}^{J} F(u_j)$ alors :

$$\begin{aligned} \nabla E_h(u_h) = &= \nabla(\epsilon u_h^t A_h u_h + \sum_{j=1}^{J} F(u_j)) \\ &= \epsilon A_h u_h + h(F'(u_1), F'(u_2), \cdots, F'(u_J))^t \\ &= \epsilon A_h u_h + hG_h(u_h) \end{aligned} \tag{2}$$

1.0.3 Question 3

on a $F'(s) = s^3 - s$ donc $F'(0) = 0$ et $(F'(0), F'(0), \cdots, F'(0))^t = (0, 0, \cdots, 0)^t$ d'après (2) on a $\nabla E_h(u_h) = \epsilon A_h u_h + hG_h(u_h)$ si u_h un vecteur nul alors $\nabla E_h(u_h) = 0$ c'est à dire u_h un point critique de E_h.

1.0.4 Question 4

on a $\nabla G_h(u_h) = T_h = \begin{pmatrix} F''((u_1) & 0 & \cdots & 0 \\ 0 & F''((u_2) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & F''((u_J) \end{pmatrix}$ d'après (2) la hessienne de E_h est $\nabla^2 E_h(u_h) = \nabla(\epsilon A_h u_h + hG_h(u_h)) = \epsilon A_h + hT_h$

2 Partie code

Dans ce projet on va utiliser 3 bibliothèques très importantes la première pour définir les matrices et faire les opérations basiques, la deuxième bibliothèque pour tracer les 3 graphes demandés et la dernière pour calculer la norme matricielle.

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy import linalg as LA
```

On commence par définir la fonction F, sa dérivée première et deuxième.

```python
def F(s):
    return ((s**2 - 1)**2)/4

def Fprime(s):
    return (s**3) - s

def fseconde(s):
    return 3*(s**2) - 1
```

Après on définit la subdivision x_j, le pas d'espace et la matrice A_h de taille $J * J$

```python

J=200

xj=np.zeros((J+2,1))
for i in range(J+2):
    xj[i,0]=i*h

h = 1/(J + 1)

A_h=(1/h)*(2*np.eye(200)-np.eye(200,k=-1)-np.eye(200,k=1))
```

D'après la première partie on a $\nabla E_h(u_h) = \epsilon A_h u_h + hG_h(u_h)$,donc on définit la matrice colonne $G_h(u_h)$ de taille 200*1 puis le gradient de la fonction E_h.

```python

def G_h(u1_h):
    t=np.zeros((200,1))
    for j in range(200):
```

```
        t[j,0]=Fprime(u1_h[j,0])
    return t

def grad1(u1_h):
    return ep*np.dot(A_h,u1_h)+h*G_h(u1_h)
```

D'après la question 4 on a montré que l'hessienne $\nabla^2 E_h(u_h)$ de E_h égale à $\epsilon A_h + hT_h$ où

$$T_h = \begin{pmatrix} F''((u_1) & 0 & \cdots & 0 \\ 0 & F''((u_2) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & F''((u_J) \end{pmatrix}$$

Donc on définit la matrice T_h puis on calcule la matrice hessienne $\nabla^2 E_h(u_h)$

```

def T_h(u1_h):
    t=np.zeros((200,200))
    for j in range(200):
        t[j,j]=fseconde(u1_h[j,0])
    return t

def hess1(u1_h):
    return ep*A_h + h*T_h(u1_h)
```

Pour calculer numériquement les points critiques de $E_h(u_h)$ solution de $\nabla E_h(u_h) = 0$ on utilise l'algorithme de Newton pour trois conditions initiales différentes, premièrement on commence par $u_h^{(0)} = (sin(2\pi x_j))_{1\leq j\leq J}$

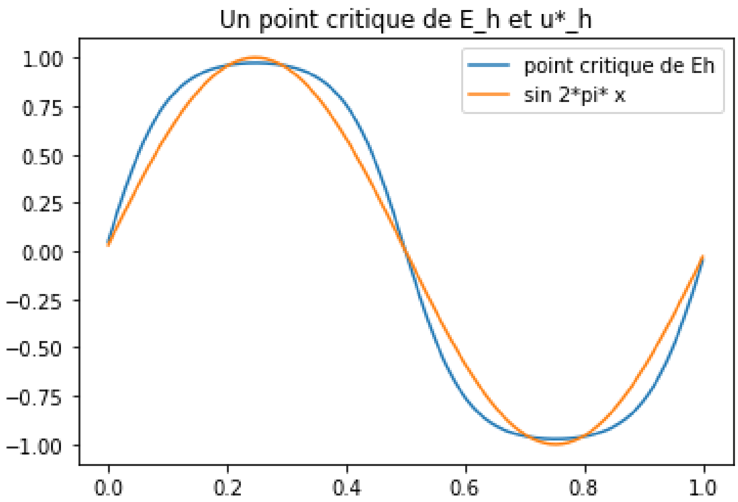
```
uh=np.zeros((J,1))
for i in range(J):
    uh[i,0]=np.sin(2*np.pi*xj[i+1,0])
```

puis on choisit deux tests d'arrêts pour l'algorithme de Newton avec une tollérance de $1e-5$ et pour un nombre d'itération maximual égale à 500.

```
def Newton(x):
    tol=1e-5
    z=x
    k=0
    w_h1=np.linalg.solve(hess1(z),grad1(z))
    while LA.norm(grad1(x),np.inf) >tol and k<500 and LA.norm(w_h1)/LA.
   norm(x)>tol:
        w_h1=np.linalg.solve(hess1(z),grad1(z))
        z=z-w_h1
    return z
```

On représente sur une figure la donnée initiale et le point critique u_h^*

```
plt.plot(np.linspace(0,1,200),Newton(uh),label='point critique de Eh')
plt.plot(np.linspace(0,1,200),uh,label="sin 2*pi* x")
plt.title("Un point critique de E_h et u0_h")
plt.legend();
```



La valeur maximale pour le point critique trouvé u_h^* est 0.9726040667747572

```
np.max(Newton(uh))
```

Pour calculer $E_h(u_h)$ on définit les deux sommations : $\sum_{i=1}^{J-1}(u_{j+1}-u_j)^2$ et $\sum_{i=1}^{J} F(u_j)$ puis la fonction E_h

```

def somme1(u):
    a=[]
    for j in range(0,J-1):
        a.append((u[j+1,0] -u[j,0])**2)
        aa=np.sum(a)
    return aa

def somme2(u):
    b=[]
    for j in range(0,J):
        b.append(F(uh[j,0]))
        bb=np.sum(b)
    return bb

def def Eh(u):
    return (ep/(2*h))* ((u[0,0])**2 +  (u[J-1,0])**2 + somme1(u) ) + h*
   somme2(u)
```

Donc pour $u_h^{(0)} = (sin(2\pi x_j))_{1\leq j\leq J}$ $E_h(uh) = 0.14185022261441413$

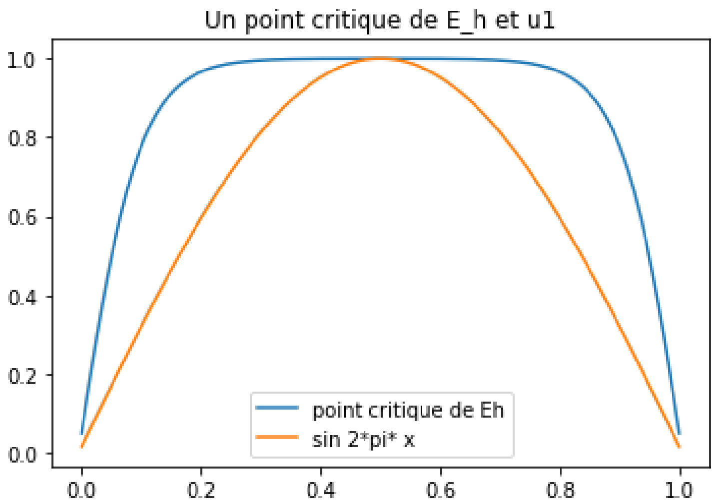
2.1 Cas n°2

Pour le deuxième cas on prend $u_h^{(0)} = (sin(\pi x_j))_{1\leq j\leq J}$ comme une donnée initiale et on débute par le définir.

```
xh=np.zeros((J+2,1))
for i in range(J+2):
    xh[i,0]=i*h
u1=np.zeros((J,1))
for i in range(J):
    u1[i,0]=np.sin(np.pi*xh[i+1,0])
print(u1)
```

Puis on applique l'algorithme de Newton sur le vecteur u1,on utilise les tests d'arrêt qui sont proposés comme dans le cas précédent et on trace sur la même figure la donnée initiale et le point critique $Newton(u1)$

```
plt.plot(np.linspace(0,1,200),Newton(u1),label='point critique de Eh')
plt.plot(np.linspace(0,1,200),u1,label="sin 2*pi* x")
plt.title("Un point critique de E_h et u1")
plt.legend();
```



La valeur maximale pour le point critique trouvé u_h^* est 0.9998177089335369 et $E_h(u1) = 0.10484297325716746$

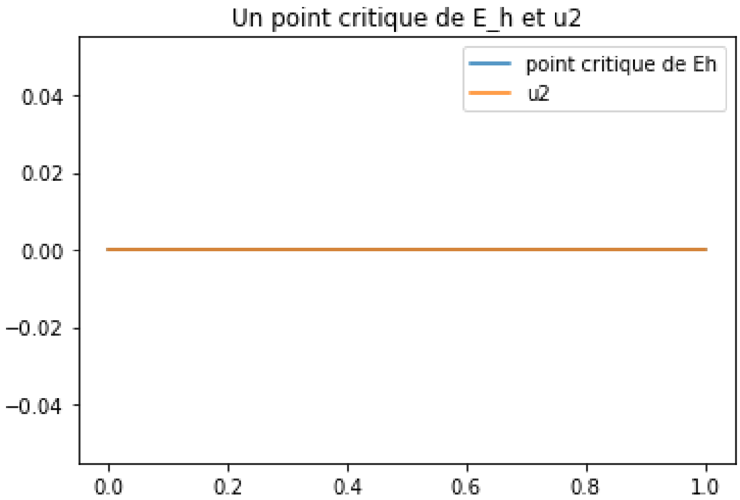
2.2 Cas n°3

Pour le dernier cas on définit un vecteur colonne u_2 de taille 200*1

```
tt=np.zeros(200)
u2=tt.reshape(200,1)
```

Puis on applique l'algorithme de Newton sur le vecteur u_2,avec les mêmes tests d'arrêt proposés dans le cas précédent et on trace sur la même figure la donnée initiale et le point critique $Newton(u_2)$.

```
plt.plot(np.linspace(0,1,200),Newton(u2),label='point critique de Eh')
plt.plot(np.linspace(0,1,200),u2,label='u2')
plt.title("Un point critique de E_h et u2")
plt.legend();
```



Ce point initiale u_2 est déja un point critique de E_h, c'est pour cette raison l'algorithme de Newton nous donne un vecteur nul comme résultat Pour cette méthode l'énergie E_h égale à 0.09250621890547261.
On remarque que le point critique lié au dernier cas qui a l'énergie E_h la plus faible.