Université Toulouse III Paul Sabatier

Méthode numérique pour les EDP

M1 MApI3

Problème de surface libre

Sujet 4

Auteurs :
Morgan Perret
Antoine Perrot
Tanguy Samelor
Wendy Revailler
Adèle Georgeot

Sous la direction de :
Pr. Jacek Narski

April 22, 2020

Remerciements

Nous souhaitons remercier nos responsables de cours, Monsieur Jacek Narski et Monsieur Stefan Le Coz, pour le temps qu'ils nous ont consacré lors de leurs différentes interventions au sein du parcours afin de nous apporter les outils méthodologiques indispensables à la conduite de cette recherche.

Contents

1 Introduction

L'équation de Laplace définie par $\Delta\psi = 0$, est une équation extrêmement présente dans le monde scientifique, notamment en Physique. Nous la retrouvons dans le cas de l'évolution de la température en régime stationnaire d'un milieu de conductivité thermique uniforme ou encore pour le potentiel gravitationnel entre les planètes.

Dans le cadre de ce projet nous nous intéresserons à résoudre de manière numérique le problème de Laplace suivant :

$$\text{Chercher } \psi :]0,L[\times]0,1[\times]0,T[\to \mathbb{R} \text{ tel que :}$$

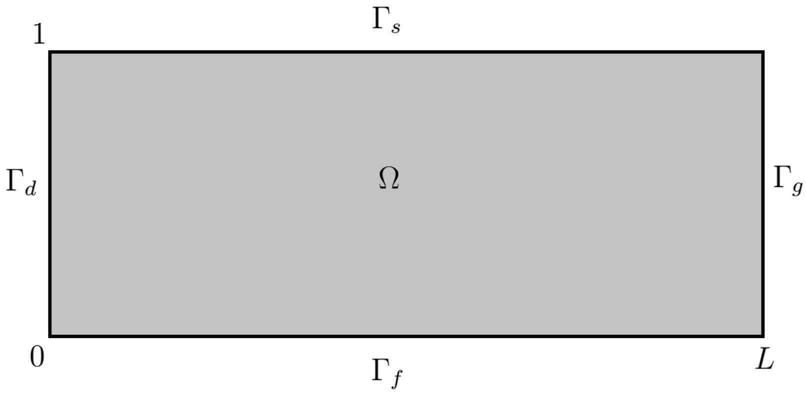
$$\Delta\psi = 0 \quad \text{dans } \Omega \times (0,T)$$

$$\text{Sous les conditions :}$$

$$\begin{gathered}
\frac{\partial^2\psi}{\partial t^2} + g\frac{\partial\psi}{\partial n} = 0 \quad \text{sur } \Gamma_s \times (0,T) \\
\frac{\partial\psi}{\partial n} = 0 \quad \text{sur } \Gamma_f \times (0,T) \\
\frac{\partial\psi}{\partial n} = v_d(t) \quad \text{sur } \Gamma_d \times (0,T) \\
\frac{\partial\psi}{\partial n} = v_g(t) \quad \text{sur } \Gamma_g \times (0,T) \\
\psi(x,0) = \phi_0 \quad \text{et} \quad \frac{\partial\psi}{\partial t}(x,0) = \phi_1 \quad \text{sur } \Gamma_s
\end{gathered}$$

$$\text{Avec :}$$

$$\begin{gathered}
\Omega =]0,L[\times]0,1[, \quad \Gamma_f =]0,L[\times\{0\}, \quad \Gamma_s =]0,L[\times\{1\} \\
\Gamma_d = \{0\}\times]0,1[, \quad \Gamma_g = \{L\}\times]0,1[
\end{gathered}$$



Espace de travail

2 Problème variationnel

Pour résoudre notre système nous souhaitons utiliser la méthode des éléments finis. Pour ce faire, nous devons déterminer le problème variationnel associé à notre problème de Laplace.

Soit $v \in H^1(\Omega)$,

$$\begin{aligned}
-\Delta\psi v = 0 &\iff -\int_{\Omega} \Delta\psi v dx = 0 \\
&\iff \int_{\Omega} \nabla\psi\nabla v dx - \int_{\partial\Omega} \frac{\partial\psi}{\partial n} v dx = 0 \\
&\iff \int_{\Omega} \nabla\psi\nabla v dx + \frac{1}{g}\int_{\Gamma_s} \frac{\partial^2\psi}{\partial t^2} v dx - \int_{\Gamma_d} v_d(t) v dx - \int_{\Gamma_g} v_g(t) v dx = 0
\end{aligned}$$

Nous avons par conséquent notre problème variationnel. Nous pouvons à présent le discrétiser afin de l'utiliser dans un programme des éléments finis. Ainsi, nous discrétisons ψ en temps afin d'appliquer un schéma implicite.

$$\begin{aligned}
-\Delta\psi v = 0 &\iff \int_{\Omega} \nabla\psi\nabla v dx + \frac{1}{g}\int_{\Gamma_s} \frac{\partial^2\psi}{\partial t^2} v dx - \int_{\Gamma_d} v_d(t) v dx - \int_{\Gamma_g} v_g(t) v dx = 0 \\
&\quad \textit{Discretisation} \\
&\iff \int_{\Omega} \nabla\psi^{n+2}\nabla v dx + \frac{1}{g}\int_{\Gamma_s} \frac{\psi^{n+2} - 2\psi^{n+1} + \psi^n}{\Delta t^2} v dx = \int_{\Gamma_d} v_d(t) v dx + \int_{\Gamma_g} v_g(t) v dx \\
&\iff \int_{\Omega} \nabla\psi^{n+2}\nabla v dx + \frac{1}{g\Delta t^2}\int_{\Gamma_s} \psi^{n+2} v dx = \frac{1}{g\Delta t^2}\int_{\Gamma_s} (2\psi^{n+1} - \psi^n) v dx + \int_{\Gamma_d} v_d(t) v dx \\
&\qquad + \int_{\Gamma_g} v_g(t) v dx
\end{aligned}$$

Maintenant que le problème variationnel est discrétisé, nous pouvons entamer l'implémentation de la résolution numérique.

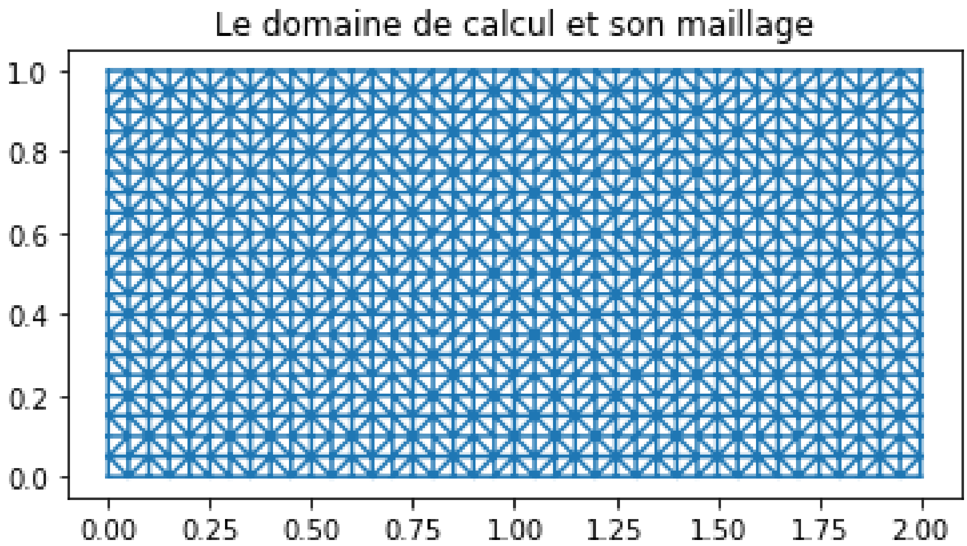
3 Résolution numérique

Nous choisissons d'utiliser les éléments finis $\mathbb{P}_1$ pour la résolution de notre problème.

3.1 Création du maillage

Dans un premier temps, nous devons découper notre espace afin de pouvoir approximer la solution au sein de chacune des unités de ce maillage. Nous choisissons dans notre projet de réaliser un maillage triangulaire comme suit :

```
n = 20
x,y = meshgrid(linspace(0,2,2*n+1),linspace(0,1,n+1))

X = x.flatten()
Y = y.flatten()

Tria = matplotlib.tri.Triangulation(X,Y)
nbElem = len(Tria.triangles)
nbNoeuds = len(Tria.x)
```



3.2 Implémentation de Γ_f, Γ_s, Γ_d, Γ_g et des fonctions de bords

Afin de pouvoir avoir accès aux différents points des frontières de Ω, nous devons réaliser les opérations suivantes :

```
FRs = []
FRg = []
FRd = []
L= 2

for t in Tria.triangles:
    for i in range(3):
        if X[t[i]] == 0 and X[t[(i+1)%3]] == 0 :
            FRg.append([t[i],t[(i+1)%3]])

        if X[t[i]] == L and X[t[(i+1)%3]] == L :
            FRd.append([t[i],t[(i+1)%3]])


        if Y[t[i]] == 1 and Y[t[(i+1)%3]] == 1 :
            FRs.append([t[i],t[(i+1)%3]])
```

Comme nous travaillons actuellement sur les conditions de notre problème, nous pouvons implémenter $v_d(t)$ et $v_g(t)$. Dans notre projet, nous choisissons les valeurs suivantes pour ces fonctions :

$$v_d(t) = cos(t^2) \ et \ v_g(t) = sin(t)$$

Nous pouvons par conséquent les implémenter de la manière suivante :

```
def vg(x,y,t):
    return sin(t)
def vd(x,y,t):
    return cos(t*2)
```

3.3 Éléments finis $\mathbb{P}_1$

Dans notre projet, nous choisissons une valeur de $L = 2$ afin de définir Ω comme étant $]0,2[\times]0,1[$.

3.3.1 Création des éléments principaux

Nous pouvons débuter notre algorithme des éléments finis par la création des variables qui seront essentielles au cours de la résolution de notre problème.

```
Nt = 50        #discrétisation en temps
Tf = 10        #temps final
g = 9.81       #constante de gravité
dt = Tf / Nt   #pas de temps

#conditions initiales :
phi0 = 0
phi1 = .6

#Intégrale sur Omega
A = zeros((nbNoeuds,nbNoeuds))
Aelt = zeros((3,3))

#Matrice de masse qui servira pour les intégrales sur FRs
M = zeros((nbNoeuds,nbNoeuds))
```

3.3.2 Parcourir le maillage

Nous allons dans cette partie parcourir le maillage afin de calculer la surface de chaque élément de la triangulation dans le but de pouvoir créer la matrice de rigidité en assemblant les matrices élémentaires. Ainsi nous réalisons les opérations suivantes :

```python
# Boucle sur les triangles
for el in range(nbElem):

    # Numeros globaux des sommets du triangle
    i1 = Tria.triangles[el,0]
    i2 = Tria.triangles[el,1]
    i3 = Tria.triangles[el,2]
    I = [i1, i2, i3]

    # Coordonnees des trois sommets
    X1 = X[i1]; Y1 = Y[i1];
    X2 = X[i2]; Y2 = Y[i2];
    X3 = X[i3]; Y3 = Y[i3];

    jacobien = (X2-X1)*(Y3-Y1) - (X3-X1)*(Y2-Y1)
    dsdx =   (Y3-Y1)/jacobien
    dtdx =  -(Y2-Y1)/jacobien
    dudx =  -dsdx-dtdx
    dsdy =  -(X3-X1)/jacobien
    dtdy =   (X2-X1)/jacobien
    dudy =  -dsdy-dtdy

    grad = zeros((3,2))

    grad[0][0] = dudx  # dphi_1/dx
    grad[0][1] = dudy  # dphi_1/dy
    grad[1][0] = dsdx  # dphi_2/dx
    grad[1][1] = dsdy  # dphi_2/dy
    grad[2][0] = dtdx  # dphi_3/dx
    grad[2][1] = dtdy  # dphi_3/dy

    # Calcul de la surface de l'element
    aire = abs(jacobien)*0.5

    Aelt[0,0] = sum(grad[0]*grad[0])
    Aelt[0,1] = sum(grad[0]*grad[1])
    Aelt[0,2] = sum(grad[0]*grad[2])
    Aelt[1,0] = sum(grad[1]*grad[0])
    Aelt[1,1] = sum(grad[1]*grad[1])
    Aelt[1,2] = sum(grad[1]*grad[2])
    Aelt[2,0] = sum(grad[2]*grad[0])
    Aelt[2,1] = sum(grad[2]*grad[1])
    Aelt[2,2] = sum(grad[2]*grad[2])

    Aelt *= aire

    # Assemblage des matrices elementaires

    A[ix_(I,I)] += Aelt
```

3.3.3 Calcul de la solution sur Γ_s

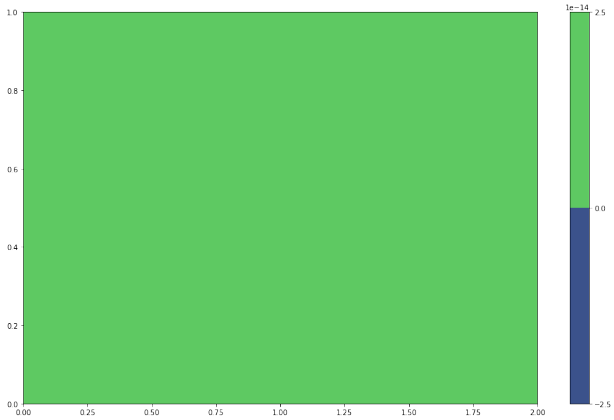
Afin d'avoir la solution sur Γ_s, il nous faut résoudre l'équation : $\frac{\partial^2\psi}{\partial t^2} + g\frac{\partial\psi}{\partial n} = 0$. Cette équation peut être résolue numériquement de la manière suivante :

```
Aelt = zeros((2,2))
for el in FRs:

    #Numéros globaux des extrémitées du segment
    i1 = el[0]
    i2 = el[1]
    I = [i1,i2]

    #Coordonnées des extrémitées :
    X1 = X[i1] ; Y1 = Y[i1] ;
    X2 = X[i2] ; Y2 = Y[i2] ;

    #Longueur du segment :
    h = sqrt((X1-X2)**2 + (Y1-Y2)**2)

    #Matrice élémentaire :
    Aelt[0,0] =  h/3
    Aelt[0,1] =  h/6
    Aelt[1,0] =  h/6
    Aelt[1,1] =  h/3


    # Assemblage matrice principale
    A[ix_(I,I)] += Aelt *  1/g/dt**2

    #Assemblage matrice pour second membre lors des itérations
    M[ix_(I,I)] += Aelt
```

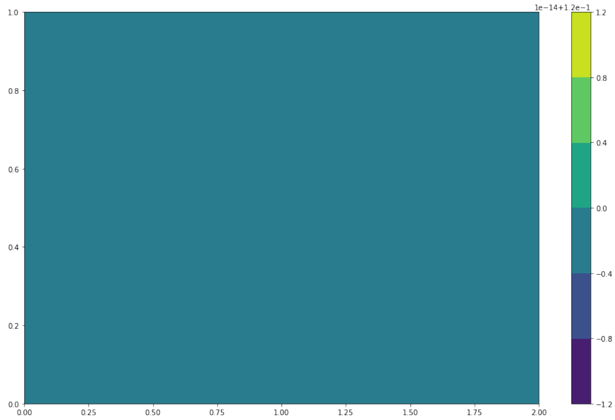
3.3.4 Initialisation

Nous commençons par initialiser notre solution et réalisons les opérations suivantes :

```
psi0 = phi0 * ones(nbNoeuds)  #calcul psi0
j = 0 #temps 0

psi1 = psi0 + dt * phi1      #calcul psi1 par différence finie
j = 1 #temps Delta*t
```



solution au temps 0



solution au temps Δt

3.3.5 Calcul du second membre et résolution du système

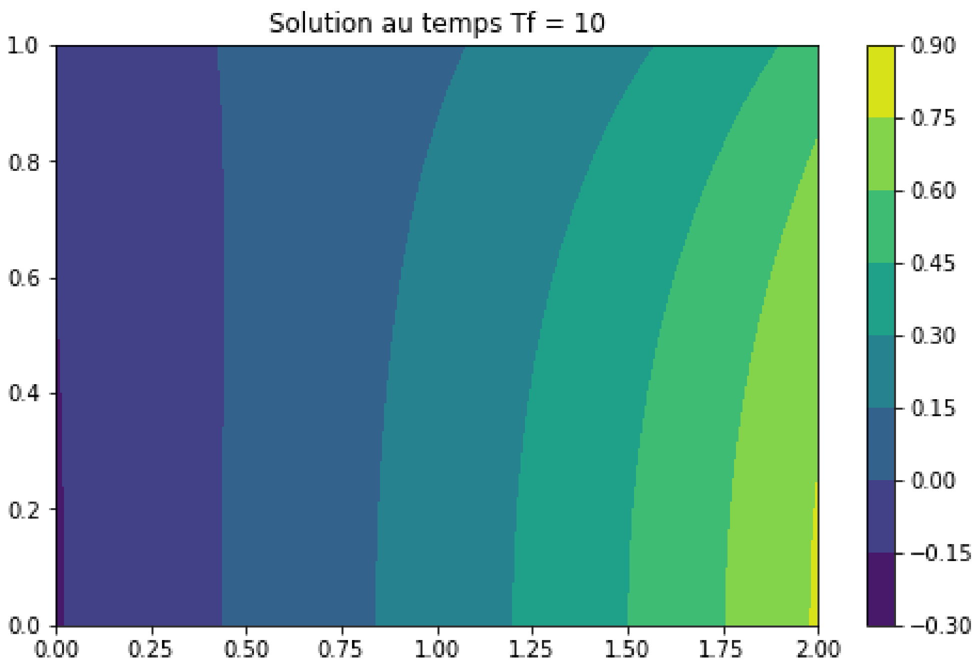
Afin de pouvoir réaliser le calcul du second membre et résoudre le système, nous effectuons les opérations suivantes :

```
#Itérations dans le temps :
for j in range(2,Nt):

    t = j * dt  #temps pour l'itération j

    #############################
    ### Calcul second membre ###
    #############################

    F = dot(M,2*psi1-psi0)  ## terme sur frontière s


    ### terme sur frontière droite ###
    for el in FRd:
        i1 = el[0]
        i2 = el[1]
        I = [i1,i2]

        X1 = X[i1]; Y1 = Y[i1];
        X2 = X[i2]; Y2 = Y[i2];

        h = sqrt((X1-X2)**2 + (Y1-Y2)**2)  #longueur segment

        Xm = (X1+X2)/2 ; Ym = (Y1+Y2)/2

        # Méthode Simpson sur le bord :
        F[I] += [(vd(X1,Y1,t) + 2 * vd(Xm,Ym,t))* h/ 6 ,\
                 (vd(X2,Y2,t) + 2 * vd(Xm,Ym,t))* h/ 6 ]


    ### terme sur frontière gauche ###
    for el in FRg:
        i1 = el[0]
        i2 = el[1]
        I = [i1,i2]

        X1 = X[i1]; Y1 = Y[i1];
```

```
        X2 = X[i2]; Y2 = Y[i2];

        h = sqrt((X1-X2)**2 + (Y1-Y2)**2)  #longueur segment

        Xm = (X1+X2)/2 ; Ym = (Y1+Y2)/2

        # Méthode Simpson sur le bord :
        F[I] += [(vg(X1,Y1,t) + 2 * vg(Xm,Ym,t))* h/ 6 ,\
                 (vg(X2,Y2,t) + 2 * vg(Xm,Ym,t))* h/ 6 ]



    #########################
    ### résolution système ###
    #########################

    psi2 = linalg.solve(A,F)

    ###ps0 devient psi1, psi1 devient psi2 et on avance ...
    psi0 = psi1
    psi1 = psi2

U = psi2    #la solution
```

3.3.6 Affichage de la solution

Nous allons dans cette partie uniquement afficher la solution au temps final. Nous avons joint en annexe une succession d'images représentant l'évolution dans le temps de notre solution. Nous avons également joint à ce projet une vidéo intitulée "*Solution_ numérique_ Problème_ de_ Laplace.avi*" réalisant une animation de cette solution dans le temps.



4 Conclusion

Au cours de ce projet, nous avons étudié un problème de Laplace sur une surface rectangulaire. Nous avions à disposition des conditions particulières pour chacun des bords de l'espace. Nous avons, dans un premier temps, explicité le problème variationnel discrétisé. Dans un second temps, nous avons utilisé la méthode des éléments finis $\mathbb{P}_1$ afin de pouvoir résoudre ce problème.

Une question reste encore en suspens, question que nous ne traiterons pas dans ce projet : nous pouvons nous demander dans quel cadre il serait envisageable de généraliser ce procédé à des espaces quelconques de dimension quelconque.

5 Bibliographie

NARSKI Jacek, *Cours de méthode numérique pour les EDP*, 2020.

LE COZ Stefan, *Cours de théorie des distributions et espaces de Sobolev*, 2020.

6 Annexe