Table des matières

Table des figures

0.1 Introduction

Ce projet porte sur la résolution d'un problème aux limites dans le cadre des équations différentielles. Nous étudions un problème défini sur l'intervalle ouvert $]0,1[$ qui consiste à trouver une fonction u vérifiant une équation différentielle spécifique et des conditions aux limites données.

L'objectif principal est de démontrer l'existence et l'unicité d'une solution pour ce problème. Dans la première partie du projet, nous nous concentrons sur la preuve de l'existence et de l'unicité de la solution.
Dans la deuxième partie, nous abordons la discrétisation du problème en utilisant la méthode des éléments finis avec une discrétisation uniforme de l'intervalle $[0,1]$.
Dans la troisième partie, nous étudions un maillage non-uniforme de l'intervalle $[0,1]$ défini par des points de discrétisation particuliers. Nous démontrons que la fonction $u(x) = \sin(\frac{\pi x}{2})$ est une solution exacte du système discretisé correspondant, pour des conditions aux limites spécifiques, une fonction source donnée et des valeurs particulières pour a et$b$.

0.2 Existence et unicité de solution

Le but de ce projet est la résolution du problème ci-dessous

$$\begin{cases} u''(x) = f(x), & \text{dans }]0,1[, \\ u(0) = a, \qquad u(1) = b. \end{cases} \tag{1}$$

pour celà nous commençons d'abord par l'étude de l'existence et de l'unicité du problème donné

0.2.1 Relèvement

On cherche une fonction w tel que

$$\begin{cases} w''(x) = f(x), & \text{dans }]0,1[, \\ w(0) = w(1) = 0. \end{cases} \tag{2}$$

Posons

$$w(x) = u(x) + (a-b)x - a \tag{3}$$

On a bien :$w''(x) = u''(x)$ et $w(0) = w(1) = 0$.
Afin de résoudre le système 1, il suffit de résoudre le système 2
On multiple l'équation par une fonction test v très régulière , et on intègrant sur $]0,1[$:

$$\int_0^1 w''(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx$$
$$\iff [w'(x)v(x)]_0^1 - \int_0^1 w'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx.$$

— afin d'éliminer le terme au bord, on vas prendre $v \in H_0^1(0,1)$

le problème (2) se récrit sous la forme variationnelle suivante :

$$\begin{cases} \text{Trouver } w \in H_0^1(0,1) \text{ tel que :} \\ \quad -\int_0^1 w'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx, \quad \forall\, v \in H_0^1(0,1) \end{cases} \tag{4}$$

On note $a(w,u) = \int_0^1 w'(x)v'(x)\,dx$
Et $l(v) = -\int_0^1 f(x)v(x)\,dx$
Vérifions que le problème variationnelle (4) admet une unique solution, pour cela on utilise le théorème de Lax-Milgram

- On a $H_0^1(0,1)$ est un espace de hilbert.

- **Bilinéarité de a**
  Soit w_1, w_2 et $v \in H_0^1(0,1)$, et $\alpha \in \mathbb{R}$

$$\begin{aligned} a(w_1 + \alpha w_2, v) &= \int_0^1 (w_1 + \alpha w_2)' v'\,dx \\ &= \int_0^1 w_1' v'\,dx + \alpha \int_0^1 w_2' v'\,dx \\ &= a(w_1, v) + \alpha a(w_2, v). \end{aligned}$$

Puisque a est symétrique $(a(w,v) = \int_0^1 w'(x)\,v'(x)\,dx = \int_0^1 v'(x)\,w'(x)\,dx = a(v,w))$, d'où la bilinéarité de a à droite.

- **Continuité de a**

$$
\begin{aligned}
|a(w,v)| &= \left| \int_0^1 w'(x)v'(x)\,dx \right| \\
&\leq \int_0^1 |w'(x)|\,|v'(x)|\,dx \\
&\leq \left(\int_0^1 |w'(x)^2\,dx| \right)^{\frac{1}{2}} \left(\int_0^1 |v'(x)^2\,dx| \right)^{\frac{1}{2}} \\
&\leq \|w'\|_{L^2(0,1)} \|v'\|_{L^2(0,1)} \\
&\leq \|w\|_{H^1(0,1)} \|v\|_{H^1(0,1)}
\end{aligned}
$$

Donc a est continue.

- **Coercivité de a**

$$a\,(w,w) = \int_0^1 |\,(w)'\,(x)\,|^2 dx = ||w||^2_{L^2(0,1)}$$

Par le théorème de poincaré, $\forall w \in H_0^1(0,1)$ on a :

$$
\begin{aligned}
\|w'\|_{L^2(0,1)} \geq \|w\|_{L_0^2(0,1)} &\Leftrightarrow ||w'||^2_{L^2(0,1)} \geq ||w||^2_{L_0^2(0,1)} \\
&\Leftrightarrow ||w'||^2_{L^2(0,1)} + ||w'||^2_{L^2(0,1)} \geq ||w||^2_{L_0^2(0,1)} + ||w'||^2_{L^2(0,1)} \\
&\Leftrightarrow (1+c)||w'||^2_{L^2(0,1)} \geq ||w||^2_{H_0^1(0,1)}
\end{aligned}
$$

donc :

$$||w||^2_{L^2(0,1)} \geq \frac{1}{(1+c)} ||w||^2_{H_0^1(0,1)}$$

D'où a est coercive sur $H_0^1(0,1)$

- **Linéarité de l**

pour tout $\alpha \in \mathbb{R}, w, v \in V$

$$
\begin{aligned}
l\,(w+\alpha v) &= \int_0^1 f\,(v+\alpha v)\,dx \\
&= \int_0^1 f\,(x)\,w\,(x) + \alpha f\,(x)\,v\,(x)\,dx \\
&= \int_0^1 f\,(x)\,w\,(x)\,dx + \int_0^1 \alpha f\,(x)\,v\,(x)\,dx \\
&= l\,(w) + \alpha l\,(v)
\end{aligned}
$$

‘

- **Continuité de l**
  On a

$$|l(v)| \leq \int_0^1 |f(x)|\,|v(x)|\,dx \leq \left(\int_0^1 |f(x)^2\,dx|\right)^{\frac{1}{2}} \left(\int_0^1 |v(x)^2\,dx|\right)^{\frac{1}{2}}$$
$$\leq \|f\|_{L^2(0,1)}\|v\|_{L^2(0,1)} \leq \|f\|_{L^2(0,1)}\|v\|_{H^1(0,1)}$$

Donc l est continue.
Les hypothèses du théorème de Lax-Milgram sont satisfaites on peut conclure qu'il existe une unique solution $w \in H_0^1(0,1)$ pour le problème variatoinnelle (4)

0.2.2 Equivalence avec le problème limite

Cette étape consiste à interpréter la formulation variationnelle et de retourner à l'équation. Pour cela on procède aux même intégrations par parties, mais au sens inverse.

— Soit $w \in H_0^1(0,1)$ solution de (4)

— On se restreint à $w \in D(0,1)$ et $v \in D(0,1)$:

$$a(w,v) = l(v)$$
$$\Leftrightarrow \int_0^1 w'(x)\,v'(x)\,dx = -\int_0^1 f(x)\,v(x)\,dx$$

On fait une intégration par partie à gauche :

$$\Rightarrow [w'(x)\,v(x)]_0^1 - \int_0^1 w''(x)\,v(x)\,dx = -\int_0^1 f(x)\,v(x)\,dx$$

$$\int_0^1 -w''(x) + f(x)\,dx = 0 \quad \forall\, v \in D(0,1)$$

$$w''(x) = f(x) \quad satisfaite\ \ presque\ \ partout\ \ dans]0,1[$$

— Puisque $w \in H_0^1(0,1)$, donc $w(0) = w(1) = 0$, donc $w \in H_0^1(0,1) \cap C^2(0,1)$ est la solution unique classique du problème (2).

— Puisque $u(x) = w(x) - (a-b)x + a$, alors $u \in H^1(0,1)$ est l'unique solution du problème

0.3 Élément finis $\mathbb{P}_1$-Lagrange

Dans cette partie, nous introduisons une discrétisassions de l'intervalle $[0,1]$ en N sous-intervalles. Nous fixons un entier $N \geq 1$, posons

$$h = \frac{1}{N+1}$$

Nous définissons l'espace V_h comme sous espace de V formé de fonctions continues sur $[0,1]$ et affines sur chaque segments $T_i =]x_i, x_{i+1}[$:

$$V_h = \left\{ v : [0,1] \to \mathbb{R} \;\middle|\; v \text{ continue sur } [0,1], \ \forall i = 0..N, \ v_{h/T_i} \text{ affine}, \right\}$$

Rappelons que chaque fonction $v \in V_h$ est déterminée de manière unique par la donnée de ses valeurs aux points $x_i pour i = 1, .., N$. L'espace V_h est de dimension $N+2$ et il est engendré par la base qui est formée des N fonctions $\varphi_i \in V_h$ définies par :

$$\varphi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{si } x \in [x_{i-1}, x_i] \\ \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{si } x \in [x_i, x_{i+1}] \\ \\ 0 & \text{sinon} \end{cases}$$

pour $i = 0$

$$\varphi_0(x) = \begin{cases} \frac{x_1 - x}{x_1 - x_0} & \text{si } x \in [x_0, x_1] \\ \\ 0 & \text{sinon} \end{cases}$$

pour $i = N+1$

$$\varphi_{N+1}(x) = \begin{cases} \frac{x - x_N}{x_{N+1} - x_N} & \text{si } x \in [x_N, x_{N+1}] \\ \\ 0 & \text{sinon} \end{cases}$$

- Soit $u_h \in V_h$, tel que $\int_0^1 u_h v_h \, dx = -\int_0^1 f v_h \, dx$ pour toute fonction $v_h \in V_{0h} = V_h \cap H_0^1(0,1)$,
  et $u_h(0) = a, u_h(1) = b$.

- En utilisant φ_j comme fonction test, on obtient à l'aide de la formulation variationnelle que pour tout $0 < j < N+1$,

$$\sum_{i=0}^{N+1}(u_h)_i \int_0^1 \varphi_i'\varphi_j'\,dx = -\int_0^1 f\varphi_j\,dx,$$

où $(u_h)_i$ sont les coordonnées de u_h dans la base (φ_i).
Les conditions aux limites impliquent que $(u_h)_0 = a$ et $(u_h)_{N+1} = b$, ainsi

$$\int_0^1 u_h'(x)\varphi_i'(x)\,dx = -\int_0^1 f(x)\varphi_i(x)\,dx \Leftrightarrow \int_0^1 \sum_{j=0}^{N+1} u_h(x_j)\varphi_j'(x)\varphi_i'(x)\,dx = -\int_0^1 f(x)\varphi_i(x)\,dx$$
$$\Leftrightarrow \sum_{j=1}^{N} u_j \int_0^1 \varphi_j'\varphi_i'\,dx + \int_0^1 u_0\varphi_0'\varphi_i'\,dx + \int_0^1 u_{N+1}\varphi_{N+1}'\varphi_i'\,dx = -\int_0^1 f\varphi_i\,dx$$
$$\Leftrightarrow \sum_{j=1}^{N} u_j \int_0^1 \varphi_j'\varphi_i'\,dx = -\left(\int_0^1 f\varphi_i\,dx + a\int_0^1 \varphi_0'\varphi_i'\,dx + b\int_0^1 \varphi_{N+1}'\varphi_i'\,dx\right).$$

Déterminer $U_h = ((u)_i)_{1\le i\le n}$ consiste donc à résoudre le système linéaire

$$A_hU_h = F_h,$$

où

$$A_h = \begin{pmatrix} \int_0^1 \varphi_1'\varphi_1'dx & \cdots & \int_0^1 \varphi_1'\varphi_N'dx \\ \vdots & \ddots & \vdots \\ \int_0^1 \varphi_N'\varphi_1'dx & \cdots & \int_0^1 \varphi_N'\varphi_N'dx \end{pmatrix} \qquad \text{et} \quad U_h = \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix}$$

Tandis que le second membre est défini par

$$(F_h)_i = -\int_{x_{i-1}}^{x_{i+1}} f\varphi_i\,dx, \quad \text{pour tout } 1 < i < n,$$

$$(F_h)_1 = \frac{a}{h} - \int_0^{x_2} f(x)\varphi_1(x)\,dx$$

$$(F_h)_N = \frac{b}{h} - \int_{x_{N-1}}^1 f(x)\varphi_N(x)\,dx$$

0.3.1 Propriété de la matrice A_h

Soit $x \in \mathbb{R}^N$ non nul ($x = (x_1, \ldots, x_N)^T$ avec les x_i non tous nuls). On a :

$$x^T A_h x = \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \int_0^1 \varphi_j'\varphi_i'\,dx = \int_0^1 \left(\sum_{i=1}^{N} x_i\varphi_i'\right)\left(\sum_{j=1}^{N} x_j\varphi_j'\right) dx$$

Posons $X_h = \sum_{i=1}^{N} x_i\varphi_i$, on a alors $x^T A_h x = \int_0^1 |X_h'|^2 dX = \|X_h'\|^2_{L^2(0,1)} \geq 0$ car les dérivées des φ_i peuvent éventuellement être nulles pour certains $i \in [1, N]$. Donc A_h est semi-définie positive.

0.3.2 Calculs des éléments A_h

Pour $j = i$

$$a(\varphi_i, \varphi_i) = \int_0^1 \varphi_i'(x)^2\,dx = \int_{x_{i-1}}^{x_{i+1}} \varphi_i'(x)^2\,dx = \frac{2}{h}$$

Pour $j = i + 1$

$$a(\varphi_i, \varphi_{i+1}) = \int_0^1 \varphi_i'(x)\varphi_{i+1}'(x)\,dx = \int_0^1 \varphi_i'(x)\varphi_{i+1}'(x)\,dx = \frac{-1}{h}$$

Comme a est symétrique alors on a : $a(\varphi_i, \varphi_j) = \frac{-1}{h}$ pour $j = i - 1$ et $j = i + 1$

D'où :
$$A_h = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

0.3.3 Résolution numérique par un maillage uniforme

Dans cette partie, nous allons résoudre numériquement le sustème linéaire $A_h U_h = F_h$ par la méthode des éléments finis comme le montre le code 1.

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.legendre import leggauss

N = 50
a_val = 0
b_val = 0
h_val = 1 / (N + 1)
```

```
def laplacian_matrix(n):
    h = 1 / (n + 1)
    return (2 * np.eye(n) - np.diag(np.ones(n - 1), -1) - np.diag(np.
        ones(n - 1), 1)) / h

x_vals = np.linspace(0, 1, N + 2)

def hat_function(x, left, mid, right):
    if left <= x <= mid:
        return (x - left) / (mid - left)
    elif mid <= x <= right:
        return (right - x) / (right - mid)
    else:
        return 0

def f(x):
    if 0.4 < x < 0.6:
        return -1
    else:
        return 0

def calculate_integral(f, left, mid, right, method):
    if method == 'trapeze':
        h = right - left
        return (h / 2) * (f(left) + f(right))
    elif method == 'quadrature':
        nodes, weights = leggauss(N)
        integral = 0
        for i in range(len(nodes)):
            xi = 0.5 * (left + right) + 0.5 * (right - left) * nodes[i
                ]
            integral += weights[i] * f(xi) * hat_function(xi, left,
                mid, right)
        return integral * 0.5 * (right - left)

F_interior_trapeze = np.zeros(N - 2)
F_left_trapeze = a_val / h_val - calculate_integral(f, 0, x_vals[0],
    x_vals[1], method='trapeze')
F_right_trapeze = b_val / h_val - calculate_integral(f, x_vals[-2],
    x_vals[-1], 1, method='trapeze')

for i in range(1, N - 1):
    F_interior_trapeze[i - 1] = -calculate_integral(f, x_vals[i - 1],
        x_vals[i], x_vals[i + 1], method='trapeze')
```

```
F_trapeze = np.concatenate(([F_left_trapeze], F_interior_trapeze, [
    F_right_trapeze]))

F_interior_quadrature = np.zeros(N - 2)
F_left_quadrature = a_val / h_val - calculate_integral(f, 0, x_vals
    [0], x_vals[1], method='quadrature')
F_right_quadrature = b_val / h_val - calculate_integral(f, x_vals[-2],
     x_vals[-1], 1, method='quadrature')

for i in range(1, N - 1):
    F_interior_quadrature[i - 1] = -calculate_integral(f, x_vals[i -
        1], x_vals[i], x_vals[i + 1], method='quadrature')

F_quadrature = np.concatenate(([F_left_quadrature],
    F_interior_quadrature, [F_right_quadrature]))

U_trapeze = np.linalg.solve(laplacian_matrix(N), F_trapeze)
U_quadrature = np.linalg.solve(laplacian_matrix(N), F_quadrature)

U_trapeze = np.concatenate(([a_val], U_trapeze, [b_val]))
U_quadrature = np.concatenate(([a_val], U_quadrature, [b_val]))

plt.plot(x_vals, U_trapeze[:N + 2], label='Approximation (Trap ze)')
plt.plot(x_vals, U_quadrature[:N + 2], label='Approximation (
    Quadrature)')

plt.xlabel('x')
plt.ylabel('u(x)')
plt.legend()
plt.title("solution approch e")
plt.show()

err_trapeze_quadrature = np.sqrt(h_val) * np.linalg.norm(U_trapeze -
    U_quadrature[:N + 2])

print("Erreur entre U_trapeze et U_quadrature =",
    err_trapeze_quadrature)
```

Listing 1 – Solution approchée du problème

— **Cas CL Dirichlit homogène**

La figure 1 représente la solution approchée par deux méthodes de calculs d'intégrals different pour le second membre.

— **Cas CL Dirichlit non homogène**

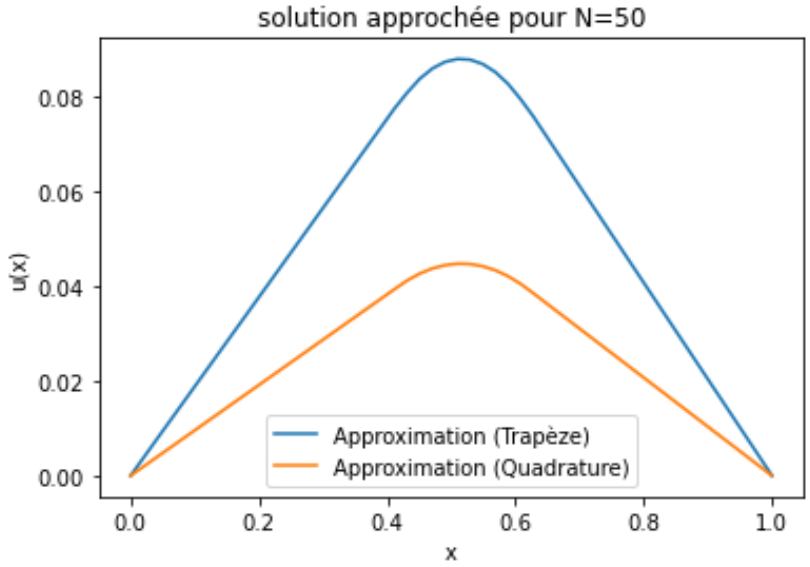

FIGURE 1 – Solution approchée pour N=50.

La figure 2 représente la solution approchée par deux méthodes de calculs d'intégrals different pour le second membre.

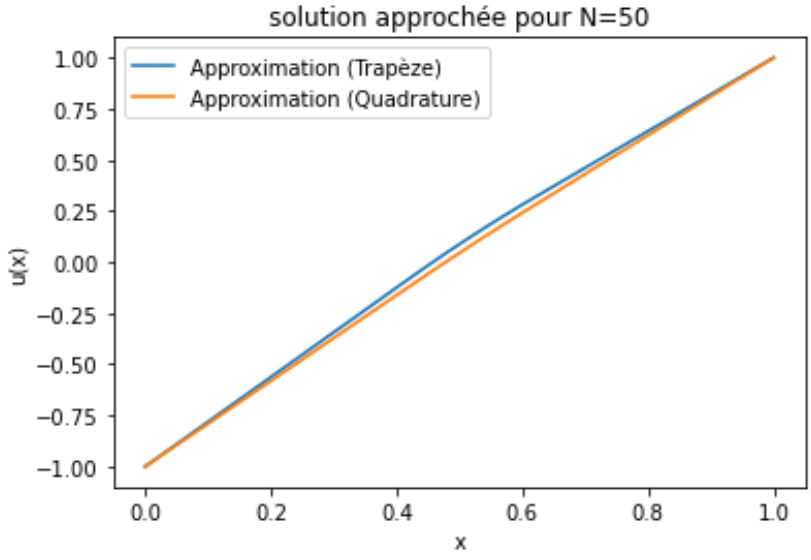

FIGURE 2 – Solution approchée pour N=50.

0.4 Résolution numérique pour un maillage non uniforme

Pour montrer que $u(x)$ est une solution du système, nous devons vérifier l'équation différentielle Nous avons $u(x) = \sin\left(\frac{\pi}{2}x\right)$ et $f(x) = -\frac{\pi}{2}\left(\frac{1}{4}\sin\left(\frac{\pi}{2}x\right)\right)$.
Remplaçons ces valeurs dans l'équation différentielle.
On a

$$u(x) = \sin\left(\frac{\pi x}{2}\right)$$

et

$$u'(x) = \frac{\pi}{2}\cos(\frac{\pi x}{2})$$

$$u''(x) = -\frac{\pi^2}{4}\sin(\frac{\pi x}{2}) = f(x)$$

et, on a $u(0) = 0 = a$, $u(1) = 1 = b$
D'où $u(x)$ est une solution de systéme.
Dans cette partie, nous allons résoudre numériquement le sustème linéaire $A_h U_h = F_h$ pour un maillage non uniforme par la méthode des éléments finis comme le montre le code 2.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate

N = 50
i = np.arange(N)
Points = np.cos((N + 1 - i) * np.pi / (2 * (N + 1)))
print(i)
print(Points)

def createMesh(Points, deg):
    N = len(Points)
    if deg == 1:
        Nint = N
        Next = 2
        Nodes = np.zeros((N + 2, 2))
        for i in range(0, N):
            Nodes[i, 0] = Points[i]
        Nodes[N, 0] = 0
        Nodes[N, 1] = 1
        Nodes[N + 1, 0] = 1
        Nodes[N + 1, 1] = 1
```

```
        Cells = np.zeros((N + 1, 2))
        for i in range(1, N + 1):
            Cells[i, 0] = i - 1
            Cells[i, 1] = i
        Cells[0, 0] = Nint
        Cells[N, 1] = Nint + 1
        Cells = Cells.astype(int)
    return Nint, Next, Nodes, Cells

Nint, Next, Nodes, Cells = createMesh(Points, 1)
print('Verification', Nint, Next)
print('Tableau de noeuds')
print(Nodes)
print('Tableau de cellules')
print(Cells)

def RefBasisFct(xhat, deg):
    res = np.zeros(deg + 1)
    if deg == 1:
        res[0] = 1 - xhat
        res[1] = xhat
    if deg == 2:
        res[0] = 2 * (xhat ** 2) - 3 * xhat + 1
        res[1] = -4 * (xhat ** 2) + 4 * xhat
        res[2] = 2 * (xhat ** 2) - xhat
    return res

def DerRefBasisFct(xhat, deg):
    res = np.zeros(deg + 1)
    if deg == 1:
        res[0] = -1
        res[1] = 1
    if deg == 2:
        res[0] = 4 * xhat - 3
        res[1] = -8 * xhat + 4
        res[2] = 4 * xhat - 1
    return res

print('Verification')
print('Valeur au point x_hat=0 :', RefBasisFct(0.4, 1), DerRefBasisFct
    (0.4, 1))

def FK2(xhat, noeud1, noeud2):
    h = noeud2 - noeud1
    x = xhat * h + noeud1
```

```
    return x

def jacFK2(xhat, noeud1, noeud2):
    return abs(noeud2 - noeud1)

def BasisFct(xhat, j, noeud1, noeud2, deg):
    vals = RefBasisFct(xhat, deg)
    return vals

def DerBasisFct(xhat, j, noeud1, noeud2, deg):
    h = abs(noeud2 - noeud1)
    vals = DerRefBasisFct(xhat, deg) / h
    return vals

def main(f, deg, Points, ex):
    print("Creation du maillage")
    Nint, Next, Nodes, Cells = createMesh(Points, deg)
    print("Assemblage de A")
    A = np.zeros((Nint, Nint))
    for j in range(0, np.size(Cells, 0)):
        for mu in range(0, np.size(Cells, 1)):
            for lamb in range(0, np.size(Cells, 1)):
                i = Cells[j, lamb]
                k = Cells[j, mu]
                if (i < Nint) and (k < Nint):
                    x_j = Nodes[Cells[j, 0], 0]
                    x_jp1 = Nodes[Cells[j, 1], 0]
                    g = lambda xhat: DerRefBasisFct(xhat, deg)[lamb] *
                        DerRefBasisFct(xhat, deg)[mu] / jacFK2(xhat,
                       x_j, x_jp1)
                    A[i, k] = A[i, k] + integrate.quad(g, 0, 1)[0]

    print("Assemblage du second membre")
    F = np.zeros((Nint, 1))
    for j in range(0, np.size(Cells, 0)):
        for lamb in range(0, np.size(Cells, 1)):
            i = Cells[j, lamb]
            if (i < Nint):
                x_j = Nodes[Cells[j, 0], 0]
                x_jp1 = Nodes[Cells[j, 1], 0]
                g = lambda xhat: -RefBasisFct(xhat, deg)[lamb] * f(FK2
                   (xhat, x_j, x_jp1)) * jacFK2(xhat, x_j, x_jp1)
                F[i] = F[i] + integrate.quad(g, 0, 1)[0]

    h = 1 - Points[len(Points) - 1]
```

```
    F[Nint - 1] = F[Nint - 1] + 1 / h

    print(A)
    print(F)

    print("Resolution du systeme lineaire")
    U = np.linalg.solve(A, F)

    print("Affichage...")
    X = np.zeros(Nint + Next)
    X[0] = 0
    X[Nint + Next - 1] = 1
    X[1:Nint + 1] = Nodes[0:Nint, 0]
    UU = np.zeros(Nint + Next)
    UU[1:Nint + 1] = U[0:Nint, 0]
    UU[0] = 0
    UU[Nint + 1] = 1

    plt.figure()
    plt.plot(X, UU, 'b', label='P1-Lag')
    plt.plot(X, ex(X), 'k', label='exacte')
    plt.legend()
    plt.title("solution approch e d'un maillage non uniforme pour N
       =50")
    plt.show()
    print("That's all, folks!!")

def f(x):
    return -((np.pi ** 2) / 4) * np.sin((np.pi * x) / 2)

def g(x):
    return np.sin((np.pi * x) / 2)

main(f, 1, Points, g)
```

Listing 2 – Solution exacte et approchée du problème Pour N=50

La figure 3 représente la solution exacte et approchée pour un maillage non uniforme.

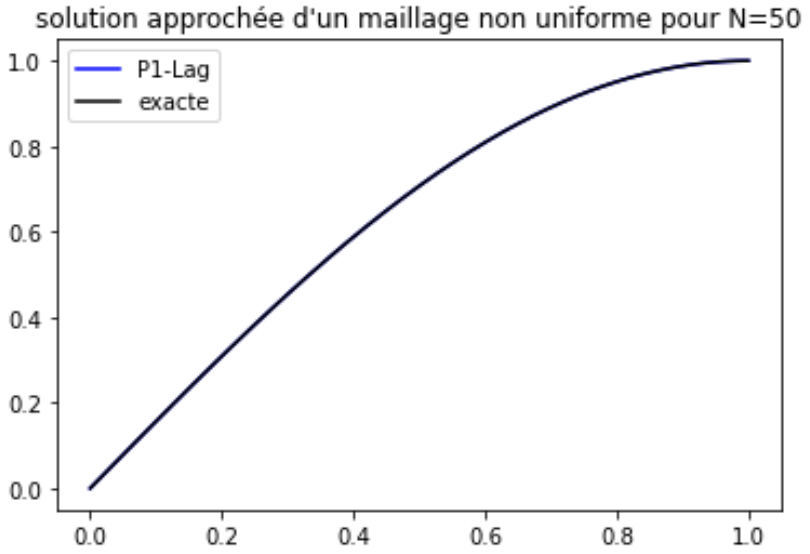

FIGURE 3 – Solution exacte et approchée pour N=50.

0.5 Conclusion

En conclusion, ce projet nous a permis de mettre en pratique les concepts théoriques de la méthode des éléments finis et de les appliquer à la résolution numérique d'un problème aux limites. Nous avons acquis une compréhension approfondie de la formulation et de la résolution du système linéaire associé, ainsi que des propriétés de la matrice A_h. De plus, nous avons exploré l'influence du maillage sur la précision de l'approximation. Ce projet a renforcé nos compétences en programmation et en analyse numérique, et nous a permis de mieux appréhender la résolution des problèmes aux limites en utilisant la méthode des éléments finis.