Rapport TP :
TP I : Elements Finis

Dupont Ronan
Croguennec Guillaume

I. Introduction

Lors de ce TP, nous nous intéresserons à la résolution du problème suivant :

$$\begin{cases} -u'' = f \\ u(0) = 0 \; et \; u(1) = 1 \end{cases} \quad sur \; \Omega = [0,1] \qquad avec \qquad f = \begin{cases} 1 & sur \;]\frac{1}{3}, \frac{2}{3}[\\ 0 & sur \;]0, \frac{1}{3}[\cup]\frac{2}{3}, 1[\end{cases}$$

Pour se faire, nous utiliserons la méthode des éléments finis.

II. Résolution du problème

Pour résoudre le problème suivant :

$$\begin{cases} -u'' = f \\ u(0) = 0 \; et \; u(1) = 1 \end{cases} \quad sur \; \Omega = [0,1] \qquad avec \qquad f = \begin{cases} 1 & sur \;]\frac{1}{3}, \frac{2}{3}[\\ 0 & sur \;]0, \frac{1}{3}[\cup]\frac{2}{3}, 1[\end{cases}$$

Nous le ramènerons à l'aide des espaces d'approximation au système suivant :

$$A\lambda = b$$

Avec $A_{i,j} = \int_\Omega \varphi_i' \varphi_j' \, dx$ où $(\varphi_i')_{i=1,\dots,n}$ est une base avec $\varphi_i \in W_h \; et \; \varphi_i(x_j) = \delta_{i,j} \; \forall i, j \in \{1, \dots, n\}$

Notre intervalle est divisé en n morceaux. On a donc un pas $h = \frac{1}{n-1}$.

On veut une méthode qui marche quel que soit les conditions limites notamment quand les conditions limites sont naturelles.

On a donc $W_h = \{v \in C^0([0,1]), v \text{ linéaire par morceaux}\}$

On trouve donc la matrice suivante ainsi que par la méthode de pénalisation, on pose le second membre b.

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & (0) \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ (0) & & & -1 & 2 \end{pmatrix} \qquad b = \begin{pmatrix} \int_\Omega f\varphi_1 + 10^6.u(0) \\ \int_\Omega f\varphi_2 \\ \vdots \\ \int_\Omega f\varphi_{n-1} \\ \int_\Omega f\varphi_n + 10^6.u(1) \end{pmatrix}$$

1ére ligne : $\frac{1}{h}[(1+10^6)\lambda_1 - \lambda_2] = \int_\Omega f\varphi_1 + \frac{10^6.u(0)}{h} \sim (1+10^6)\lambda_1 = 10^6.u(0)$
$d'o\grave{u}\ \lambda_1 = u(0)$

n-iéme ligne $\frac{1}{h}[(1+10^6)\lambda_n - \lambda_{n-1}] = \int_\Omega f\varphi_n + \frac{10^6.u(1)}{h} \sim (1+10^6)\lambda_n = 10^6.u(1)$
$d'o\grave{u}\ \lambda_n = u(1)$

Calcul de l'intégrale de $f\varphi_i$

En général, $f\varphi_i(x)$ n'a pas une forme analytique, donc on ne peut pas calculer son intégrale de manière exacte. On va utiliser une forme d'approximation. Ici on utilise la formule de Simpson à 3 points.

$$\int_a^b f(x)dx \approx \int_a^b P(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

Une fois ces termes obtenus, il ne nous restera plus qu'à résoudre le système de type Ax=b pour trouver le vecteur solution x. Nous utiliserons la méthode du gradient conjugué que nous avons déjà codé lors d'un autre TP.

III. Partie codage

Pour résoudre ce problème, nous codons dans un premier temps la fonction f définie par : $f = \begin{cases} 1 & sur\ ]\frac{1}{3},\frac{2}{3}[\\ 0 & sur\ ]0,\frac{1}{3}[\cup]\frac{2}{3},1[\end{cases}$

On rappel qu'ici notre intervalle est discrétisé en l morceaux de taille h=1/(l-1).

```
do i=1,l
    xi=real((i-1))/(l-1)
    if (xi.ge.(1./3).and.xi.le.(2./3)) then
        f(i)=1
    else
        f(i)=0
    endif
enddo
```

Nous codons ensuite la matrice A définie par $A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & (0) \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ (0) & & & -1 & 2 \end{pmatrix}$.

```
c *** Matrice A ***

      DO I=1,n
          DO J=1,n
              IF (i.EQ.j) THEN
              A(i,j)=2.d0/h/h
          ELSE IF (i.EQ.j-1) THEN
              A(i,j)=1.d0/h/h
          ELSE IF (i-1.EQ.j) THEN
              A(i,j)=1.d0/h/h
          ELSE
              A(i,j)=0
          END IF
          ENDDO
      ENDDO
```

On ajoute les conditions essentielles pour la pénalisation.

```
c On ajoute les conditions essentielles
      A(1,1)=A(1,1)+1000000/h/h
      A(n,n)=A(1,1)
```

On peut donc calculer le vecteur b en utilisant la méthode d'approximation de la manière suivante :

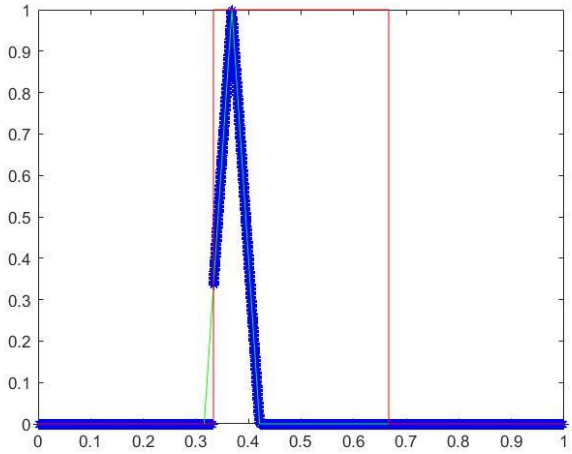
```
do i=2,l-1
        b(i)=(f(i)+4/2*(f(i)+f(i-1))/2)/(6*(l-1))
        b(i)=b(i)+(f(i)+4/2*(f(i)+f(i+1))/2)/(6*(l-1))
enddo
b(1)=(f(1)+4/2*(f(1)+f(2))/2)/(6*(l-1))+1000000*u0
b(l)=(f(l)+4/2*(f(l)+f(l-1))/2)/(6*(l-1))+1000000*u1
```

On obtient donc un vecteur b ayant les valeurs ci-dessous pour n=20.

```
0.0000000000000000
0.0000000000000000
0.0000000000000000
0.0000000000000000
0.0000000000000000
0.0000000000000000
8.7719298245614030E-003
4.3859649122807015E-002
5.2631578947368418E-002
5.2631578947368418E-002
5.2631578947368418E-002
5.2631578947368418E-002
4.3859649122807015E-002
8.7719298245614030E-003
0.0000000000000000
0.0000000000000000
0.0000000000000000
0.0000000000000000
0.0000000000000000
1000000.0000000000
```

Ces valeurs sont bien celles attendues, car nous avons 0 sur les tiers du bord, et la valeur de h au centre du tier du milieu. Et bien sûr, b est symétrique.

Ce vecteur contient les $\int_\Omega f\varphi_i$. Par exemple pour un maillage avec l=20, on peut tracer la courbe de f ainsi que celle d'un φ particulier, ici φ_8. On obtient une courbe comme celle-ci :



Il ne nous reste plus qu'à calculer l'air sous la courbe bleu avec la méthode Simpson comme nous avons fait au-dessus.

Une fois la matrice A et le vecteur b obtenu, nous pouvons donc résoudre le système à l'aide de la méthode du gradient conjugué :
La méthode du gradient conjugué permettant de résoudre Ax=b se base sur le principe mathématique suivant :
On cherche x $x \in R^n \; tel \; que \; J(x) = \frac{1}{2} < Ax, x > - < b, x >$
Il en découle de ce postulat un tat d'équations mathématiques permettant de trouver x.
Ces équations nous amènent à en déduire le pseudo-code suivant :
On initialise :

$v = A.x_0$
$r_0 = b - v = b - A.x_0$
$d_0 = r_0$

On répète k fois la boucle suivante :

$$\alpha_k = \frac{r_k^T.r_k}{d_k^T.A.d_k}$$
$$x_{k+1} = x_k + d_k.\alpha$$
$$r_{k+1} = b - A.x_1 = r_k - \alpha_k.A.d_k \qquad r \; est \; le \; résidus \; indiquant \; la \; précision$$
$$\beta_k = \frac{r_{k+1}^T.r_{k+1}}{r_k^T.r_k}$$
$$d_{k+1} = r_{k+1} + d_k.\beta$$

Fin de la boucle
Sur notre code, nous avons mis une condition pour que l'algorithme continue à boucler tant que le résidu a une valeur supérieure à 10^{-8}.
On aura utilisé différentes variables x0, x1 / r0, r1 afin de pouvoir garder le terme en k et en k+1.
Ceci nous donne donc l'algorithme suivant :

```
            call matvop(l,x0,A,v)

            do i=1,n
               r0(i)=b(i)-v(i)
            enddo

            do i=1,n
               d0(i)=r0(i)
            enddo

Début des itérations

             t=1000
            do while(sqrt(t).ge.0.00000001)
             call matvop(l,d0,A,w)
             call scal(n,w,d0,z)
             call scal(n,r0,r0,y)

alpha=(r.r)/(p.a.p)

             alpha0=y/z

             do i=1,n

c xk+1=xk+alp.p
                  x1(i)=x0(i)+alpha0*d0(i)

             enddo
             do i=1,n
                  call matvop(l,x1,A,g)

rk+1=rk-alpha.a.p

                  r1(i)=b(i)-g(i)
             enddo
```

$$\alpha_k = \frac{r_k^T . r_k}{d_k^T . A . d_k}$$

$$x_{k+1} = x_k + d_0 . \alpha$$

$$r_{k+1} = b - A . x_1 = r_k - \alpha_k . A . d_k$$

```
beta=(rk+1.rk+1)/(rk.rk)

             call scal(n,r1,r1,t)
             call scal(n,r0,r0,u)
             beta=t/u

             do i=1,n
                 d1(i)=r1(i)+beta*d0(i)
             enddo

             do i=1,n
                 x0(i)=x1(i)
                 d0(i)=d1(i)
                 r0(i)=r1(i)
             enddo
            enddo

Affichage des valeurs de xk

             do i=1,n
                  print*,x1(i)
             enddo
```

$$\beta_k = \frac{r_{k+1}^T . r_{k+1}}{r_k^T . r_k}$$

$$d_{k+1} = r_{k+1} + d_k . \beta$$

IV. Calcul de la solution exacte

Afin de comparer nos futurs résultats, nous avons besoin de calculer la solution exacte.
Pour se faire nous intégrons deux fois la fonction f.

Après intégration, nous avons obtenu :

- u1(x)=ax+b sur [0,1/3]
- u2(x)=-x²/2+cx+d sur [1/3,2/3]
- u3(x)=ex+f sur [2/3,1]

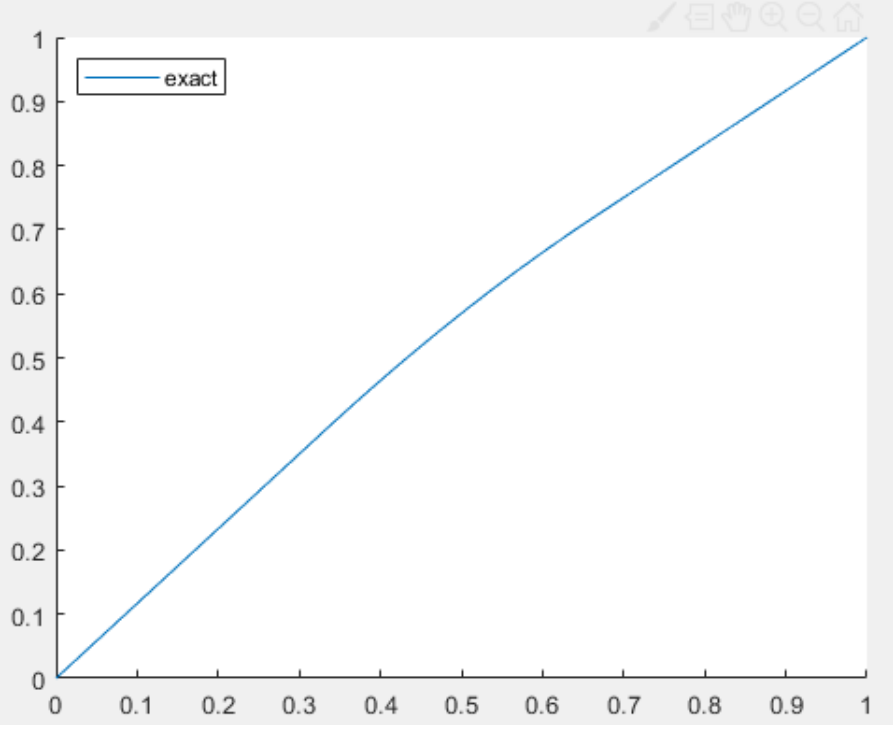
A l'aide des conditions de continuité, conditions initiales :

- u1(0)=0
- u3(1)=1
- u1(1/3)= u2(1/3)
- u2(2/3)= u3(2/3)
- u1'(1/3)= u2'(1/3)
- u2'(2/3)= u3'(2/3)

Nous obtenons un système à 6 équations que nous avons résolu en le mettant sous forme d'équation matricielle. Nous avons obtenu les valeurs suivantes :

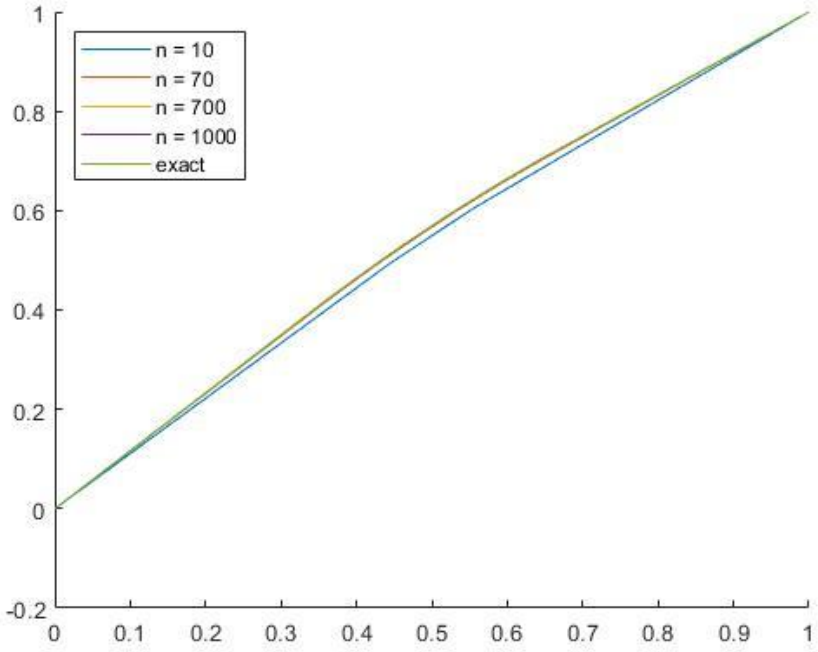
- a= 7/6
- b= 0
- c= 3/2
- d=-1/18
- e=5/6
- f=1/6

En la traçant nous obtenons la courbe suivante :

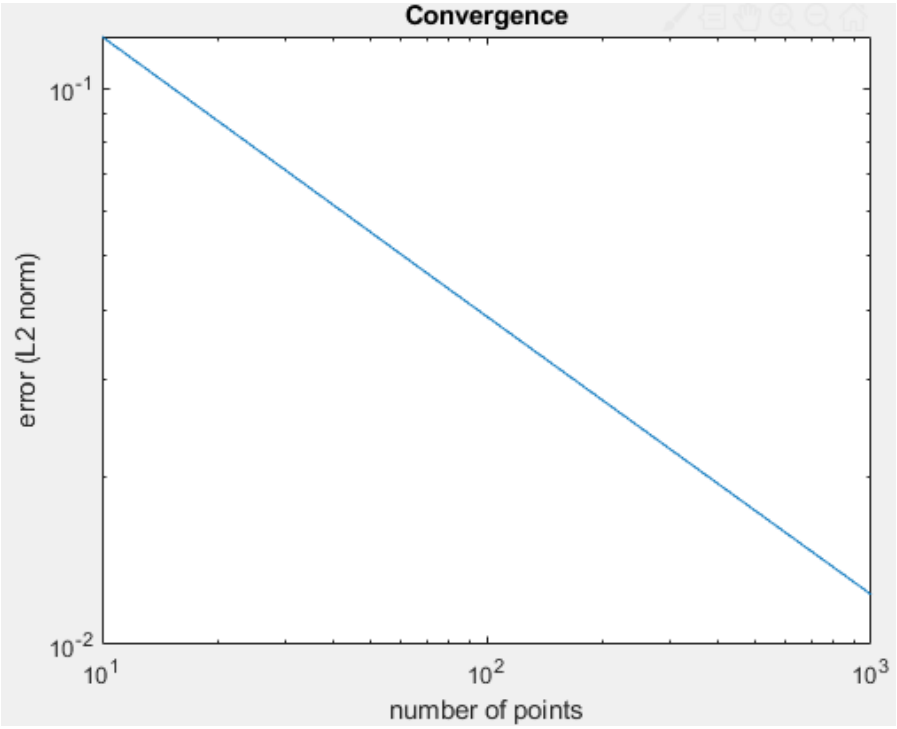


V. Résultats

Une fois le gradient conjugué exécuté, nous obtenons les résultats suivants :



On remarque que la courbe converge très rapidement. Vers la solution exacte. Nous avons tracé la convergence vers la solution exacte :



On remarque que c'est une convergence en O($\sqrt{n}$).

VI. Conclusion

Dans ce TP d'initialisation aux éléments finis en 1D, nous avons pu observer pour un cas simple comment cette méthode fonctionne. En effet, par des méthodes simples d'approximations, nous avons pu calculer la solution de problèmes qui paraissent « complexes ».
Nos résultats ne sont pas très précis mais ceci vient du fait que les approximations ne sont pas parfaites. Plus notre maillage est grand, plus nos résultats sont précis : la solution simulée converge vers la solution exacte en $O(\sqrt{n})$.
Cette méthode que nous avons utilisée fonctionne pour n'importe quelles conditions essentielles, car il suffit de modifier u0 et u1.
En revanche, une petite modification de programme serait nécessaire si le problème contenait des conditions limites naturelles. Cela nous imposerait donc des conditions sur u' aux bords. Pour résoudre ce problème, une solution serait d'imposer u(0)+h*u'(0)=u(1) et u(n-1)+h*u'(n)=u(n).