

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

MÉMOIRE DE MASTÈRE

2011 - 2012

ELEMENTS FINIS :
DU CLASSIQUE AU ISOGEOMETRIQUE

Étudiant : NGUYEN Thi Thuy Trang
 TRINH Thi Huyen
Responsable : Pierre DREYFUSS
 Francesca RAPETTI

Table des matières

Introduction	4
1 Eléments finis classique	5
1.1 Eléments finis de degré un pour le problème de Dirichlet homogène en dimension $n = 1$	5
1.1.1 Ecriture du problème approché	6
1.1.2 Calcul des coefficients de la matrice	7
1.1.3 Calcul des composantes du second membre	8
1.1.4 Technique d'assemblage	9
1.2 Eléments finis de degré deux pour le problème de Dirichlet homogène	12
1.2.1 Technique de l'élément de référence	13
1.2.2 Calcul de la matrice de masse élémentaire	13
1.2.3 Calcul de la matrice de raideur élémentaire	14
1.2.4 Calcul du second membre élémentaire	14
1.2.5 Technique d'assemblage	15
1.3 Eléments finis de degré trois pour le problème de Dirichlet homogène	18
1.3.1 Technique de l'élément de référence	19
1.3.2 Calcul de la matrice de masse élémentaire	20
1.3.3 Calcul de la matrice de raideur élémentaire	20
1.3.4 Calcul du second membre élémentaire	20
1.3.5 Technique d'assemblage	21
2 B-splines	28
2.1 Vecteur des noeuds	28
2.2 Les fonctions B-splines	28
2.3 Les dérivées de fonctions B-splines	31
2.4 Courbes B-splines	31
2.5 Interpolation par des courbes B-splines	32

2.5.1	Le problème linéaire	33
2.5.2	Résolution numérique d'un problème d'interpolation . . .	35
2.6	La méthode d'éléments finis par B-splines	39
3	Commentaire	48
RÉFÉRENCES		50

Introduction

La méthode des éléments finis fait partie des outils de mathématiques appliquées. En analyse numérique, la méthode des éléments finis est utilisée pour résoudre numériquement des équations aux dérivées partielles. Celles-ci peuvent par exemple représenter analytiquement le comportement dynamique de certains systèmes physiques (mécaniques, thermodynamiques, acoustiques, etc.). En mathématique, Il s'agit de remplacer un problème compliqué pour lequel a priori on ne connaît pas de solution, par un problème plus simple que l'on sait résoudre.

Dans notre mémoire, dans la première partie, on va regarder la méthode des éléments finis classique pour le problème Dirichlet homogène en dimension un de degré un, degré deux, degré trois. Au début, on va donner les définitions de la matrice de masse, de la matrice de raideur, de la matrice de masse élémentaire, de la matrice de raideur élémentaire. Dans chaque cas, on va regarder l'algorithme d'assemblage des matrices et second membres élémentaire afin de constituer le système global. D'où on va donner les programmes en Matlab pour calculer une solution approchée et donner l'erreur.

Dans la deuxième partie, on va considérer les fonctions B-splines et leurs premières dérivées, donner la définition d'une courbe B-splines. Après on va regarder la résolution numérique d'un problème d'interpolation par des courbes B-splines et la méthode des éléments finis utilisant les fonctions B-splines cubiques pour approcher une solution exacte. Dans cette partie, on aura aussi les programmes en Matlab pour les pratiquer. En fin, on va donner les commentaires entre deux méthodes.

Nous voudrions exprimer notre reconnaissance à Monsieur le Professeur Pierre Dreyfuss et à Madame le Professeur Francesca Rapetti de nous avoir beaucoup aidé à faire ce stage. Nous remercions bien aux professeurs du département des Mathématiques à l'Université de Nice - Sophia Antipolis, surtout Prof. Stéphanie Nivoche et Prof. Jacques Blum qui nous ont aidés et nous ont encouragés. Enfin, il nous faut aussi dire remercier à mes amis avec leur soutien, leur encouragement et leur sympathie à notre égard.

Nice, Juillet 2012

1 Eléments finis classique

Considérons le problème de Dirichlet homogène en dimension $n = 1$:

$$\begin{cases} -\alpha u''(x) + \beta u(x) = f(x) \text{ sur }]a, b[, \\ u(a) = u(b) = 0 \end{cases}$$

où α, β sont des constantes et f est une fonction continue.

La formulation variationnelle de ce problème s'écrit : Trouver la fonction u appartenant à $H_0^1[a, b]$ telle que pour tout $v \in H_0^1[a, b]$ on ait :

$$\int_a^b \alpha u'(x)v'(x)dx + \int_a^b \beta u(x)v(x)dx = \int_a^b f(x)v(x)dx. \quad (1)$$

L'existence et l'unicité d'une solution pour ce problème se démontrent en utilisant le théorème de Lax-Milgram :

Théorème 1 (*Théorème de Lax-Milgram*)

Soient V un espace de Hilbert et a une forme bilinéaire continue sur $V \times V$ et coercive sur V . Supposons que L est une forme linéaire continue sur V . Alors, il existe un unique fonction u de V tel que l'équation $a(u, v) = L(v)$ pour tout v de V .

On suppose maintenant que l'on connaît un sous-espace $V_h \subset V$ de dimension finie, paramétré par h et tel que pour tout $v \in V$, il existe un élément $r_h v \in V_h$ vérifiant : $\lim_{h \rightarrow 0} \|r_h v - v\| = 0$. Considérons alors le problème suivant : Trouver la fonction u_h appartenant à V_h tel que : $a(u_h, v_h) = L(v_h)$, $\forall v_h \in V_h$. Ce problème admet également une solution unique car V_h est un sous-espace fermé de V et donc les hypothèses du théorème de Lax-Milgram sont également vérifiées dans V_h .

Supposons u et u_h les solutions correspondent à les problèmes $a(u, v) = L(v)$ et $a(u_h, v_h) = L(v_h)$. On a alors un résultat général de majoration d'erreur suivant :

Théorème 2 Soit M la constante intervenant dans l'hypothèse de continuité de a : $a(u, v) \leq M\|u\|\|v\|$ et m la constante intervenant dans l'hypothèse de coercive : $a(u, v) \geq m\|v\|^2$, on a la majoration d'erreur suivante :

$$\|u - u_h\| \leq \frac{M}{m} \inf_{v_h \in V_h} \|u - v_h\|.$$

1.1 Eléments finis de degré un pour le problème de Dirichlet homogène en dimension $n = 1$

Introduisons une discrétisation de l'intervalle $[a, b]$ en N sous-intervalles ou éléments $T_i = [x_{i-1}, x_i]$. Les éléments T_i n'ont pas forcément même longueur. $V_{0,h}$ est alors l'espace des fonctions continues affines par morceaux (affines sur les segments T_i) et nulles aux extrémités a et b .

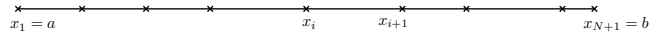


FIGURE 1 – Discrétisation (maillage) du segment $[a, b]$ en éléments finis $P1$

Rappelons que chaque fonction $v_h \in V_{0,h}$ est déterminée de manière unique par la donnée de ses valeurs aux points x_i pour $i = 2, \dots, N$. L'espace $V_{0,h}$ est de dimension $N - 1$ et il est engendré par la base de Lagrange qui est formée des $N - 1$ fonctions $w_i \in V_{0,h}$ définies par :

$$w_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{sinon} \end{cases} \quad \forall i = 2, \dots, N \quad \text{et} \quad \forall j = 2, \dots, N.$$

D'où :

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

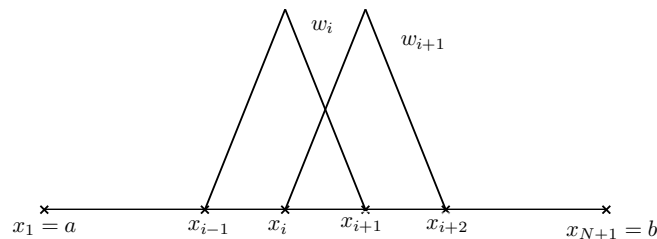


FIGURE 2 – Fonctions de base $P1$

Une fonction v_h quelconque s'écrit dans cette base :

$$v_h(x) = \sum_{i=2}^N v_i w_i(x)$$

avec $v_i = v_h(x_i)$. Les coefficients v_i sont donc les valeurs de v_h aux points x_i .

1.1.1 Ecriture du problème approché

Ecrivons le problème approché dans $V_{0,h}$: Trouver la fonction u_h appartenant à $V_{0,h}$ telle que pour tout $v_h \in V_{0,h}$ on ait :

$$\int_a^b \alpha u_h'(x) v_h'(x) dx + \int_a^b \beta u_h(x) v_h(x) dx = \int_a^b f(x) v_h(x) dx. \quad (2)$$

Le problème étant linéaire, l'égalité est vraie pour tout v_h si et seulement si elle est vraie pour une base de l'espace vectoriel $V_{0,h}$, c'est-à-dire :

$$(2) \text{ est vérifiée } \forall v_h \in V_{0,h} \Leftrightarrow (2) \text{ est vérifiée } \forall w_i \text{ pour } i = 2, \dots, N.$$

D'autre part, si u_h est solution du problème approché dans $V_{0,h}$, on peut l'exprimer dans la base des w_i comme suit :

$$u_h(x) = \sum_{j=2}^N u_j w_j(x),$$

avec $u_j = u_h(x_j)$ est la valeur approchée de la solution exacte au point x_j . Ainsi le problème approché s'écrit : Chercher u_2, u_3, \dots, u_N tels que

$$\sum_{j=2}^N \left(\int_a^b \alpha w_j'(x) w_i'(x) dx + \int_a^b \beta w_j(x) w_i(x) dx \right) u_j = \int_a^b f(x) w_i(x) dx.$$

Posons

$$F_i := \int_a^b f(x) w_i(x) dx$$

$$A_{ij} := \int_a^b \alpha w_j'(x) w_i'(x) dx + \int_a^b \beta w_j(x) w_i(x) dx.$$

On remarque que le problème approché prend la forme d'un système linéaire de $N - 1$ équations à $N - 1$ inconnues, qui peut s'écrire sous la forme matricielle suivante :

$$AU = F.$$

1.1.2 Calcul des coefficients de la matrice

La matrice A apparaît comme la somme de deux matrices K et M . K est appelée matrice de raideur. Elle est donnée par

$$K_{ij} = \alpha \int_a^b w_j'(x) w_i'(x) dx,$$

M est la matrice de masse. Son expression est la suivante :

$$M_{ij} = \beta \int_a^b w_j(x) w_i(x) dx.$$

On obtient sans difficulté les contributions de chaque élément T_i aux matrices de raideur et de masse, dites matrices élémentaires de raideur et matrices élémentaires de masse.

Matrice élémentaire de raideur

On calcule les coefficients K_{ij} en sommant les contributions des différents éléments selon :

$$K_{ij} = \alpha \int_a^b w_j'(x) w_i'(x) dx = \alpha \sum_{k=1}^N \int_{x_k}^{x_{k+1}} w_j'(x) w_i'(x) dx.$$

Considérons par exemple l'élément $T_i = [x_i, x_{i+1}]$. Sur cet élément, il n'y a que deux fonctions de base non nulles : w_i et w_{i+1} . L'élément T_i produira donc effectivement une contribution non nulle aux quatre coefficients $K_{i,i}$, $K_{i,i+1}$, $K_{i+1,i+1}$ et $K_{i+1,i}$ de la matrice globale K . Calculons les contributions élémentaires de T_i et disposons les sous la forme d'une matrice élémentaire 2×2 :

$$ElemK_i = \alpha \begin{pmatrix} e_{1,1}^i & e_{1,2}^i \\ e_{2,1}^i & e_{2,2}^i \end{pmatrix}$$

avec

$$\begin{aligned} e_{1,1}^i &= \int_{x_i}^{x_{i+1}} w_i'^2(x) dx = \frac{1}{x_{i+1} - x_i}, \\ e_{1,2}^i = e_{2,1}^i &= \int_{x_i}^{x_{i+1}} w_i'(x) w_{i+1}'(x) dx = -\frac{1}{x_{i+1} - x_i}, \\ e_{2,2}^i &= \int_{x_i}^{x_{i+1}} w_{i+1}'^2(x) dx = \frac{1}{x_{i+1} - x_i}. \end{aligned}$$

D'où :

$$ElemK_i = \alpha \begin{pmatrix} \frac{1}{x_{i+1} - x_i} & -\frac{1}{x_{i+1} - x_i} \\ -\frac{1}{x_{i+1} - x_i} & \frac{1}{x_{i+1} - x_i} \end{pmatrix} = \alpha \frac{1}{x_{i+1} - x_i} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

Matrice élémentaire de masse

Avec le même raisonnement, on obtient la matrice de masse élémentaire :

$$ElemM_i = \beta \frac{x_{i+1} - x_i}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

1.1.3 Calcul des composantes du second membre

Chaque composante F_i du vecteur second-membre global est calculée également par assemblage de contributions élémentaires :

$$F_i = \int_a^b f(x) w_i(x) dx = \sum_{k=1}^N \int_{x_k}^{x_{k+1}} f(x) w_i(x) dx.$$

On utilise des formules d'intégration numérique, par exemple la formule des trapèzes ou la formule de Simpson :

- La formule des trapèzes :

$$\int_a^b \psi(x) \approx \frac{b-a}{2} (\psi(a) + \psi(b))$$

est exacte pour les fonctions affines.

– La formule de Simpson :

$$\int_a^b \psi(x) \approx \frac{b-a}{6} (\psi(a) + 4\psi(\frac{a+b}{2}) + \psi(b))$$

est exacte pour les polynômes de degré inférieur ou égal à 3.

Avec des fonctions tests $w_i \in P1$, la méthode des trapèzes conduit à une valeur approchée de l'intégrale, qui dans le cas de points équidistribués de pas h redonne le résultat

$$F_i = hf_i$$

obtenu en différences finies. La méthode de Simpson permet aussi un calcul approché et donne dans le même cas

$$F_i = \frac{h}{6} [f_{i-1} + 4f_i + f_{i+1}].$$

1.1.4 Technique d'assemblage

Considérons un maillage à N éléments et notons B la matrice globale à assembler (matrice de raideur ou de masse globale), b_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k et F la matrice globale du second membre, l_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k . L'algorithme d'assemblage est très simple dès lors que l'on dispose d'un tableau associant les points d'un élément T_k et les noeuds du maillage global.

Son schéma en le suivant :

Pour des matrices :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 2$ FAIRE ! boucle sur les numéros locaux

 POUR $j = 1 : 2$ FAIRE ! boucle sur les numéros locaux

$I = K + i - 1$! numéros globaux

$J = K + j - 1$! numéros globaux

$B(I, J) = B(I, J) + b(i, j)$! B : matrice globale, b : matrice élémentaire

FIN DES 3 BOUCLES

Pour second membre :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 2$ FAIRE ! boucle sur les numéros locaux

$I = K + i - 1$! numéros globaux

$F(I) = F(I) + l(i)$! F : matrice globale, l : matrice élémentaire

FIN DES 2 BOUCLES

Dans ce cas très simple d'éléments de degré un en dimension un, chaque élément T_k comprend 2 noeuds x_{k+1}, x_k . On a alors les programmes suivants en matlab :

```
function test=EFdegre1;  
  Comparaison_degre_1();
```

end

```
function test0=Comparaison_degre_1();
    alpha=1;
    beta=1;
    N=16;
    h=1/N;
    a=0;
    b=1;
    K=(alpha/h)*[1,-1;-1,1];
    M=(beta/3*h)*[1,0.5;0.5,1];
    I=linspace(a,b,N+1);
    elFinis=linspace(a,b,N+1);

    function y=f(x)
        y=x^4;
    end
    g=solution_degre_1(N,h,elFinis,alpha,beta,@f,a,b,K,M);
    c2=(37-24*exp(1))/(exp(1)-exp(-1));
    c1=-24-c2;
    s_exact=c1*exp(I)+c2*exp(-I)+I.^4+12*I.^2+24;
    err=g-s_exact';
    em=max(abs(err))

    I=linspace(a,b,200);
    s=c1*exp(I)+c2*exp(-I)+I.^4+12*I.^2+24;
    hold on
    p=plot(I,s);
    q=plot(elFinis,g,'red');
    hold off
end
```

```
function [ms]=matrixA_degre_1(N,h,elFinis,alpha,beta,K,M)
    ms=zeros(N+1,N+1);
    for ie=1:N
        for i=1:2
            ig=ie+i-1;
            for j=1:2
                jg=ie+j-1;
                ms(ig,jg)=ms(ig,jg)+K(i,j)+M(i,j);
            end
        end
    end
end
```

```
function [ap]=aproximationtrapezes(g)
    ap=1/2*(g(0)+g(1));
```

```

end

function y=phi1(x)
    y=1-x;
end

function y=phi2(x)
    y=x;
end

function [l]=member2_s(elFinis ,h,s , f)
    x1=elFinis (s);
    x2=elFinis (s+1);
    l=zeros (2,1);
    function y=r1(x)
        y=f(x1+h*x)*phi1(x);
    end
    function y=r2(x)
        y=f(x1+h*x)*phi2(x);
    end
    l(1)=aproximationtrapezes (@r1);
    l(2)=aproximationtrapezes (@r2);
end

function [F]=vectMember2_degre_1(elFinis ,h,N,f)
    F=zeros (N+1,1);
    for s=1:N
        l=member2_s(elFinis ,h,s , f);
        for i=1:2
            ig=s+i-1;
            F(ig)=F(ig)+h*l(i);
        end
    end
end

function [U]=solution_degre_1(N,h,elFinis ,alpha ,beta , f , a , b ,K,M)
    A=matrixA_degre_1(N,h,elFinis ,alpha ,beta ,K,M);
    A(1,:) =0;A(1,1)=1;
    A(N+1,:) =0;A(N+1,N+1)=1;
    F=vectMember2_degre_1(elFinis ,h,N,f);
    F(1)=0;
    F(N+1)=0;
    U=A\F;
end

```

Pour le degré un, la table de connectivité des éléments, la table de noeuds de discrétisation et le table des noeuds de quadrature sont toutes identiques. Cette propriété n'est plus

valable en degré supérieur à 1. Les détails sont donnés dans la suite avec degré deux et degré trois.

1.2 Éléments finis de degré deux pour le problème de Dirichlet homogène

On part à nouveau d'une discrétisation de l'intervalle $[a, b]$ en N sous-intervalles ou éléments T_i . Les éléments T_i n'ont pas forcément même longueur. L'espace V_h est considéré ici ensemble des fonctions continues sur $[a, b]$ et étant polynomiales de degré deux sur chaque sous-intervalle. Un polynôme de degré deux est fixé par ses valeurs en trois points. On prend les extrémités et le milieu de chaque élément T_i . On est ainsi amené à considérer une discrétisation de $[a, b]$ en N sous-intervalles comportant eux-mêmes trois points, ce qui nous conduit globalement à une discrétisation par $2N + 1$ points où les noeuds x_i sont donnés par

$$x_1 = a < x_2 < x_3 < \dots < x_{2N} < x_{2N+1} = b,$$

avec $x_{2i} = \frac{x_{2i-1} + x_{2i+1}}{2}$ pour $i = 1, \dots, N$ et $T_i = [x_{2i-1}, x_{2i+1}]$ pour $i = 1, \dots, N$.

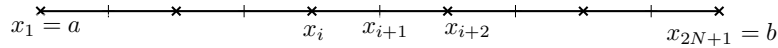


FIGURE 3 – Discrétisation (maillage) du segment $[a, b]$ en éléments finis $P2$

Soit encore la base $\{w_i\}$ avec $i = 1, \dots, 2N + 1$ de V_h données par les conditions :

$$w_i(x_j) = \delta_{ij} \quad \forall i = 1, \dots, 2N + 1 \text{ et } \forall j = 1, \dots, 2N + 1.$$

Après des calculs élémentaires on obtient :

$$w_i(x) = \begin{cases} \frac{(x - x_{i-2})(x - x_{i-1})}{(x_i - x_{i-2})(x_i - x_{i-1})} & \text{si } x \in [x_{i-2}, x_i], \\ \frac{(x - x_{i+1})(x - x_{i+2})}{(x_i - x_{i+1})(x_i - x_{i+2})} & \text{si } x \in [x_i, x_{i+2}], \\ 0 & \text{si } x \notin [x_{i-2}, x_{i+2}], \end{cases}$$

lorsque w_i correspondant à un point x_i extrémité d'un élément (voir Figure 4) et

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

lorsque w_i correspondent à un point milieu d'un élément (voir Figure 5).

La formulation variationnelle du problème de Dirichlet homogène consiste à chercher la fonction u appartenant à $H_0^1[a, b]$ telle que (2) soit satisfaite. Ainsi le problème approché s'écrit : Chercher u_2, \dots, u_{2N} tels que :

$$\sum_{j=2}^{2N} \left(\int_a^b \alpha w_j'(x) w_i'(x) dx + \int_a^b \beta w_j(x) w_i(x) dx \right) u_j = \int_a^b f(x) w_i(x) dx \quad \forall i = 2, \dots, 2N$$

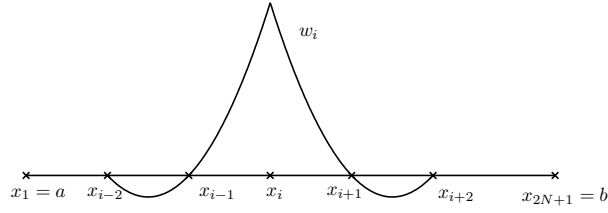


FIGURE 4 – Fonction de base $P2$ associée à une extrémité

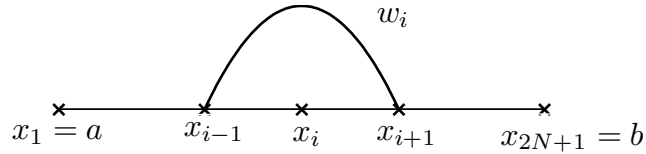


FIGURE 5 – Fonction de base $P2$ associée à un milieu

1.2.1 Technique de l'élément de référence

Par le changement de variable $F_i(t) = x_{i+1} + \frac{x_{i+2} - x_i}{2}t$, on passe de $t \in [-1, 1]$ à $x \in [x_i, x_{i+2}]$. Les fonctions de base dans $[x_i, x_{i+2}]$ s'expriment donc à l'aider des trois fonctions suivantes définies sur $[-1, 1]$:

$$\hat{w}_1(t) = \frac{t(t-1)}{2}, \quad \hat{w}_2(t) = -(t-1)(t+1), \quad \hat{w}_3(t) = \frac{t(t+1)}{2}.$$

L'expression des dérivées est :

$$\frac{d\hat{w}_1}{dt} = t - \frac{1}{2}, \quad \frac{d\hat{w}_2}{dt} = -2t, \quad \frac{d\hat{w}_3}{dt} = t + \frac{1}{2}.$$

1.2.2 Calcul de la matrice de masse élémentaire

Par $w_{2i+k} \circ F_i = \hat{w}_{k+2}$ avec la fonction F_i est définie par le changement de variable précédent et $k = -1, 0, 1$. Les coefficients de la matrice de masse pour l'élément $[x_i, x_{i+2}]$ sont :

$$\beta \frac{x_{i+2} - x_i}{2} \int_{-1}^1 \hat{w}_i(t) \hat{w}_j(t) dt \text{ pour } i, j = 1, 2, 3.$$

On obtient ainsi la matrice de masse élémentaire suivante pour cet élément :

$$M_i = \beta \frac{x_{i+2} - x_i}{2} \begin{pmatrix} \frac{4}{15} & \frac{2}{15} & -\frac{1}{15} \\ \frac{2}{15} & \frac{16}{15} & \frac{2}{15} \\ -\frac{1}{15} & \frac{2}{15} & \frac{4}{15} \end{pmatrix}.$$

1.2.3 Calcul de la matrice de raideur élémentaire

On a $\frac{dw_{2i+k}}{dx} = \frac{d\hat{w}_{k+2}}{dt} \frac{dt}{dx}$ pour $k = -1, 0, 1$ et $i = 1, \dots, N$. Ainsi les coefficients de la matrice de raideur sont :

$$\alpha \frac{2}{x_{i+2} - x_i} \int_{-1}^1 \hat{w}'_i(t) \hat{w}'_j(t) dt \text{ pour } i, j = 1, 2, 3,$$

et on obtient :

$$K_i = \alpha \frac{2}{x_{i+2} - x_i} \begin{pmatrix} \frac{7}{6} & -\frac{4}{3} & \frac{1}{6} \\ \frac{4}{3} & \frac{8}{3} & -\frac{4}{3} \\ \frac{1}{6} & -\frac{4}{3} & \frac{7}{6} \end{pmatrix}.$$

1.2.4 Calcul du second membre élémentaire

Chaque composante F_i du vecteur second-membre global donnée par

$$F_i = \int_a^b f(x) w_i(x) dx,$$

est également calculée par assemblage des contributions élémentaires $F_i^{(k)}$. On a :

$$F_i = \sum_{k=1}^N F_i^{(k)} = \sum_{k=1}^N \int_{x_{2k-1}}^{x_{2k+1}} f(x) w_i(x) dx,$$

où les $F_i^{(k)}$ désignent les contributions des éléments k .

Sur l'élément $T_k = [x_{2k-1}, x_{2k+1}]$, il n'y a que 3 fonctions de base non nulles : $w_{2k-1}, w_{2k}, w_{2k+1}$.

Ainsi les seules contributions non nulles sont $F_{2k-1}^{(k)}, F_{2k}^{(k)}, F_{2k+1}^{(k)}$. On utilise la formule de Simpson pour les calculs,

$$\int_{x_{2k-1}}^{x_{2k+1}} \Phi(x) dx \approx \frac{x_{2k+1} - x_{2k-1}}{6} [\Phi(x_{2k-1}) + 4\Phi(x_{2k}) + \Phi(x_{2k+1})].$$

Au final, le second membre correspondant à l'élément $[x_{2k-1}, x_{2k+1}]$ s'écrit :

$$\begin{pmatrix} F_{2k-1}^{(k)} \\ F_{2k}^{(k)} \\ F_{2k+1}^{(k)} \end{pmatrix} = \frac{x_{2k+1} - x_{2k-1}}{6} \begin{pmatrix} f_{2k-1} \\ 4f_{2k} \\ f_{2k+1} \end{pmatrix}.$$

1.2.5 Technique d'assemblage

Considérons un maillage à N éléments et notons B la matrice globale à assembler (matrice de raideur ou de masse globale), b_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k et F la matrice globale du second membre, l_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k . L'algorithme d'assemblage est très simple dès lors que l'on dispose d'un tableau associant les points d'un élément T_k et les noeuds du maillage global.

Dans ce cas très simple d'éléments de degré deux en dimension un, chaque élément T_k comprend trois noeuds $x_{2k-1}, x_{2k}, x_{2k+1}$.

Son schéma en le suivant :

Pour des matrices :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 3$ FAIRE ! boucle sur les numéros locaux

 POUR $j = 1 : 3$ FAIRE ! boucle sur les numéros locaux

$I = 2 * K + i - 2!$ numéros globaux

$J = 2 * K + j - 2!$ numéros globaux

$B(I, J) = B(I, J) + b(i, j)!$ B : matrice globale, b : matrice élémentaire

FIN DES 3 BOUCLES

Pour second membre :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 3$ FAIRE ! boucle sur les numéros locaux

$I = 2 * K + i - 2!$ numéros globaux

$F(I) = F(I) + l(i)!$ F : matrice globale, l : matrice élémentaire

FIN DES 2 BOUCLES

On a alors les programmes suivants en matlab :

```
function test=EFdegre2;  
    Comparaison_degre_2();  
end
```

```
function test0=Comparaison_degre_2();  
    alpha=1;  
    beta=1;  
    N=16;  
    h=1/N;  
    a=0;  
    b=1;  
    K=(2*alpha/h)*[7/6, -4/3, 1/6; -4/3, 8/3, -4/3; 1/6, -4/3, 7/6];  
    M=(beta/2*h)*[4/15, 2/15, -1/15; 2/15, 16/15, 2/15; -1/15, 2/15, 4/15];  
    I=linspace(a, b, 2*N+1);  
    elFinis=linspace(a, b, 2*N+1);
```

```

function y=f(x)
    y=x ^ 4;
end
g=solution _degre _2(N,h,elFinis ,alpha ,beta ,@f,a,b,K,M);
c2 =(37-24*exp(1))/(exp(1)-exp(-1));
c1 =-24-c2;
s_exact=c1*exp(I) + c2*exp(-I)+ I.^4 + 12*I.^2 + 24 ;
err=g-s_exact';
em=max(abs(err))
I=linspace(a,b,200);
s=c1*exp(I) + c2*exp(-I)+ I.^4 + 12*I.^2 + 24 ;
hold on
p=plot(I,s);
q=plot(elFinis ,g,'red');
hold off
end

function [ms]=matrixA _degre _2(N,h,elFinis ,alpha ,beta ,K,M)
ms=zeros(2*N+1,2*N+1);
for ie=1:N
    for i=1:3
        ig=2*ie+i-2;
        for j=1:3
            jg=2*ie+j-2;
            ms(ig ,jg)=ms(ig ,jg)+K(i ,j)+M(i ,j);
        end
    end
end
end

function [ap]=aproximation2points(g)
w=[1,1];
xpq=[-sqrt(1/3),sqrt(1/3)];
gpq=[g(xpq(1)),g(xpq(2))];
ap=0;
for i=1:2
    ap=ap+w(i)*gpq(i);
end
end

function y=phi1_d2(x)
y=1/2*x*(x-1);
end

function y=phi2_d2(x)
y=(1-x)*(1+x);
end

```



```

function y=phi3_d2(x)
    y=1/2*x*(x+1);
end
function [l]=member2_degre_2_s(elFinis ,h,s ,f)
    x1=elFinis (2*s -1);
    x2=elFinis (2*s );
    x3=elFinis (2*s +1);
    l=zeros (3 ,1);
    function y=r1 (x)
        y=f(x2+h/2*x)*phi1_d2(x);
    end
    function y=r2 (x)
        y=f(x2+h/2*x)*phi2_d2(x);
    end
    function y=r3 (x)
        y=f(x2+h/2*x)*phi3_d2(x);
    end
    l(1)=aproximation2points (@r1);
    l(2)=aproximation2points (@r2);
    l(3)=aproximation2points (@r3);
end

function [F]=vectMember2_degre_2(elFinis ,h,N,f)
    F=zeros (2*N+1 ,1);
    for s=1:N
        l=member2_degre_2_s(elFinis ,h,s ,f);
        for i=1:3
            ig=2*s+i -2;
            F(ig)=F(ig)+h/2*l(i);
        end
    end
end

function [U]=solution_degre_2(N,h,elFinis ,alpha ,beta ,f ,a ,b,K,M)
    A=matrixA_degre_2(N,h,elFinis ,alpha ,beta ,K,M);
    A(1 ,:)=0;A(1 ,1)=1;
    A(2*N+1 ,:)=0;A(2*N+1 ,2*N+1)=1;
    F=vectMember2_degre_2(elFinis ,h,N,f);
    F(1)=0;
    F(2*N+1)=0;
    U=A\F;
end

```

On peut continuer à argumenter l'ordre de la méthode. Nous présentons maintenant le degré trois.

1.3 Éléments finis de degré trois pour le problème de Dirichlet homogène

On conserve toujours les mêmes notations de bases, mais cette fois l'espace V_h est la ensemble des fonctions continues sur $[a, b]$ et étant polynomiales de degré trois sur chaque sous-intervalle. Un polynôme de degré trois est fixé par ses valeurs en quatre points. On considérera donc une discrétisation globale en $3N + 1$ points ou noeuds x_i indexés par $i = 1, \dots, 3N + 1$:

$$x_1 = a < x_2 < x_3 < \dots < x_{3N} < x_{3N+1} = b$$

avec $x_{3i-1} = x_{3i-2} + \frac{x_{3i+1} - x_{3i-1}}{3}$ pour $i = 1, \dots, N$, $x_{3i} = x_{3i-2} + 2\frac{x_{3i+1} - x_{3i-1}}{3}$ pour $i = 1, \dots, N$ et $T_i = [x_{3i-2}, x_{3i+1}]$ pour $i = 1, \dots, N$.

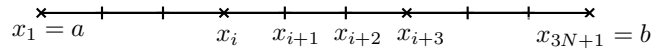


FIGURE 6 – Discrétisation (maillage) du segment $[a, b]$ en éléments finis P_3

Soit encore la base w_i avec $i = 1, \dots, 3N + 1$ de V_h données par :

$$w_i(x_j) = \delta_{ij} \quad \forall i = 1, \dots, 3N + 1 \text{ et } \forall j = 1, \dots, 3N + 1.$$

Ainsi, lorsque les fonctions w_i correspondant à un point x_i qui est extrémité d'un élément, il vient :

$$w_i(x) = \begin{cases} \frac{(x - x_{i-1})(x - x_{i-2})(x - x_{i-3})}{(x_i - x_{i-1})(x_i - x_{i-2})(x_i - x_{i-3})} & \text{si } x \in [x_{i-3}, x_i], \\ \frac{(x - x_{i+1})(x - x_{i+2})(x - x_{i+3})}{(x_i - x_{i+1})(x_i - x_{i+2})(x_i - x_{i+3})} & \text{si } x \in [x_i, x_{i+3}], \\ 0 & \text{si } x \notin [x_{i-3}, x_{i+3}], \end{cases}$$

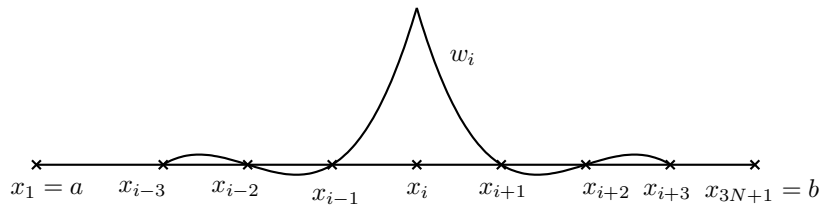


FIGURE 7 – Fonction de base P_3 associée à une extrémité

et lorsque les fonctions w_i correspondant aux deux points intérieurs d'un élément, nous obtenons :

$$w_i(x) = \begin{cases} \frac{(x - x_{i-1})(x - x_{i+1})(x - x_{i+2})}{(x_i - x_{i-1})(x_i - x_{i+1})(x_i - x_{i+2})} & \text{si } x \in [x_{i-1}, x_{i+2}] \\ 0 & \text{si } x \notin [x_{i-1}, x_{i+2}] \end{cases}$$

si x_i est premier point intérieur d'un élément

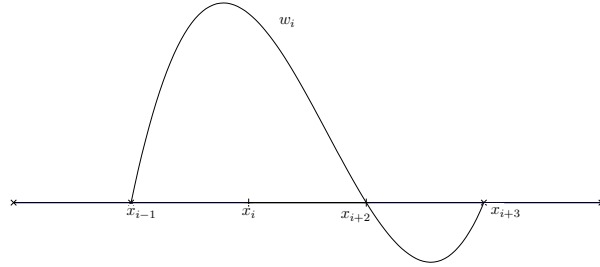


FIGURE 8 – Fonction de base $P3$ associée à premier point intérieur

et

$$w_i(x) = \begin{cases} \frac{(x - x_{i-2})(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-2})(x_i - x_{i-1})(x_i - x_{i+1})} & \text{si } x \in [x_{i-2}, x_{i+1}], \\ 0 & \text{si } x \notin [x_{i-2}, x_{i+1}]. \end{cases}$$

si x_i est deuxième point intérieur d'un élément .

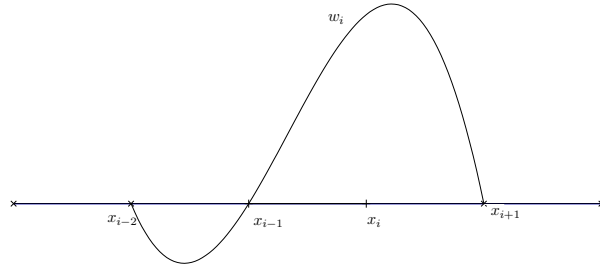


FIGURE 9 – Fonction de base $P3$ associée à deuxième point intérieur

A partir de la formulation variationnelle (2), nous obtenons maintenant un problème approché qui s'écrit : Trouver u_2, \dots, u_{3N} tels que :

$$\sum_{j=2}^{3N} \left(\int_a^b \alpha w'_j(x) w'_i(x) dx + \int_a^b \beta w_j(x) w_i(x) dx \right) u_j = \int_a^b f(x) w_i(x) dx \quad \forall i = 2, \dots, 3N.$$

1.3.1 Technique de l'élément de référence

Par le changement de variable $F_i(t) = \frac{x_i + x_{i+3}}{2} + \frac{x_{i+3} - x_i}{2}t$, on passe de $t \in [-1, 1]$ à $x \in [x_i, x_{i+3}]$. Les fonctions de base dans $[x_i, x_{i+3}]$ s'expriment donc à l'aider des quatre

fonctions suivantes définies sur $[-1, 1]$:

$$\begin{aligned}\hat{w}_1(t) &= \frac{1}{16}(1-t)(3t-1)(3t+1), & \hat{w}_2(t) &= \frac{9}{16}(t+1)(t-1)(3t-1), \\ \hat{w}_4(t) &= \frac{1}{16}(t+1)(3t-1)(3t+1), & \hat{w}_3(t) &= \frac{9}{16}(t+1)(1-t)(3t+1),\end{aligned}$$

dont les dérivées sont respectivement égales à

$$\begin{aligned}\frac{d\hat{w}_1}{dt} &= \frac{1}{16}(-27t^2 + 18t + 1), & \frac{d\hat{w}_2}{dt} &= \frac{9}{16}(9t^2 - 2t - 3), \\ \frac{d\hat{w}_4}{dt} &= \frac{1}{16}(27t^2 + 18t - 1), & \frac{d\hat{w}_3}{dt} &= \frac{9}{16}(-9t^2 - 2t + 3).\end{aligned}$$

1.3.2 Calcul de la matrice de masse élémentaire

Par $w_{3i+k} \circ F_i = \hat{w}_{k+3}$ avec la fonction F_i est définie par le changement de variable précédent et $k = -2, -1, 0, 1$. Le calcul des coefficients de la matrice de masse se ramène à l'évaluation des intégrales :

$$\beta \frac{x_{i+3} - x_i}{2} \int_{-1}^1 \hat{w}_i(t) \hat{w}_j(t) dt \text{ pour } i, j = 1, 2, 3, 4.$$

1.3.3 Calcul de la matrice de raideur élémentaire

On a $\frac{dw_{3i+k}}{dx} = \frac{d\hat{w}_{k+3}}{dt} \frac{dt}{dx}$ pour $k = -2, -1, 0, 1$ et $i = 1, \dots, N$. Donc, les coefficients de la matrice de raideur dans élément $[x_i, x_{i+3}]$ sont

$$\alpha \frac{2}{x_{i+3} - x_i} \int_{-1}^1 \phi'_i(t) \phi'_j(t) dt \text{ pour } i, j = 1, 2, 3, 4.$$

1.3.4 Calcul du second membre élémentaire

Chaque composante F_i du vecteur second-membre global

$$F_i = \int_a^b f(x) w_i(x) dx$$

est calculée également par assemblage de contributions élémentaires $F_i^{(k)}$ selon

$$F_i = \sum_{k=1}^N F_i^{(k)} = \sum_{k=1}^N \int_{x_{3k-2}}^{x_{3k+1}} f(x) w_i(x) dx,$$

où les $F_i^{(k)}$ désignent les contributions des éléments k .

Sur élément $T_k = [x_{3k-2}, x_{3k+1}]$, il n'y a que 4 fonctions de base non nulles : $w_{3k-2}, w_{3k-1}, w_{3k}, w_{3k+1}$. Donc, sur cet élément, il n'y a que les contributions non nulles. Sur l'élément $T_k = [x_{2k-1}, x_{2k+1}]$, il n'y a que 4 fonctions de base non nulles : $w_{3k-2}, w_{3k-1}, w_{3k}, w_{3k+1}$. Ainsi les seules contributions non nulles sont $F_{3k-2}^{(k)}, F_{3k-1}^{(k)}, F_{3k}^{(k)}, F_{3k+1}^{(k)}$.

1.3.5 Technique d'assemblage

Considérons un maillage à N éléments et notons B la matrice globale à assembler (matrice de raideur ou de masse globale), b_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k et F la matrice globale du second membre, l_k les matrices d'élémentaires correspondantes relatives à chaque élément T_k . L'algorithme d'assemblage est très simple dès lors que l'on dispose d'un tableau associant les points d'un élément T_k et les noeuds du maillage global.

Dans ce cas très simple d'éléments de degré deux en dimension un, chaque élément T_k comprend trois noeuds $x_{3k-2}, x_{3k-1}, x_{3k}, x_{3k+1}$.

Son schéma en la suivante :

Pour des matrices :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 4$ FAIRE ! boucle sur les numéros locaux

 POUR $j = 1 : 4$ FAIRE ! boucle sur les numéros locaux

$I = 3 * K + i - 3!$ numéros globaux

$J = 3 * K + j - 3!$ numéros globaux

$B(I, J) = B(I, J) + b(i, j)!$ B : matrice globale, b : matrice élémentaire

FIN DES 3 BOUCLES

Pour second membre :

POUR $K = 1 : N$ FAIRE ! boucle sur les éléments

 POUR $i = 1 : 4$ FAIRE ! boucle sur les numéros locaux

$I = 3 * K + i - 3!$ numéros globaux

$F(I) = F(I) + l(i)!$ F : matrice globale, l : matrice élémentaire

FIN DES 2 BOUCLES

On a alors les programmes suivants en matlab :

```
function test=EFdegre3;  
    Comparaison_degre_3();  
end
```

```
function test0=Comparaison_degre_3();  
    alpha=1;  
    beta=1;  
    N=16;  
    h=1/N;  
    a=0;  
    b=1;  
    K=matrix_raideur_local_d3(alpha, beta, h);  
    M=matrix_masse_local_d3(alpha, beta, h);  
    I=linspace(a, b, 3*N+1);  
    elFinis=linspace(a, b, 3*N+1);  
  
    function y=f(x)
```

```

        y=x ^ 4;
    end

    g=solution _ degre _ 3 (N,h,elFinis ,alpha ,beta ,@f,a ,b,K,M);

    c2 = (37-24*exp(1))/(exp(1)-exp(-1));
    c1 =-24-c2;
    s_exact=c1*exp(I) + c2*exp(-I)+ I.^4 + 12*I.^2 + 24 ;
    err=g-s_exact';
    em=max(abs(err))

    I=linspace(a,b,200);
    s=c1*exp(I) + c2*exp(-I)+ I.^4 + 12*I.^2 + 24 ;
    hold on
    p=plot(I,s);
    q=plot(elFinis ,g,'red');
    hold off

end

function y=phi1_d3(x)
    y=1/16*(1-x)*(3*x-1)*(3*x+1);
end

function y=phi2_d3(x)
    y=9/16*(x+1)*(x-1)*(3*x-1);
end

function y=phi3_d3(x)
    y=9/16*(x+1)*(1-x)*(3*x+1);
end

function y=phi4_d3(x)
    y=1/16*(x+1)*(3*x+1)*(3*x-1);
end

function y=dphi1_d3(x)
    y=1/16*(-27*x^2+18*x+1);
end

function y=dphi2_d3(x)
    y=9/16*(9*x^2-2*x-3);
end

function y=dphi3_d3(x)
    y=9/16*(-9*x^2-2*x+3);
end

```

```

function y=dphi4_d3(x)
    y=1/16*(27*x^2+18*x-1);
end

function [K]=matrix_raideur_local_d3(alpha,beta,h)
    K=zeros(4,4);
    xpq=[-sqrt(3/5),0,sqrt(3/5)];
    w=[5/9,8/9,5/9];
    xx1=[dphi1_d3(xpq(1)),dphi2_d3(xpq(1)),dphi3_d3(xpq(1)),dphi4_d3(xpq(1))];
    xx2=[dphi1_d3(xpq(2)),dphi2_d3(xpq(2)),dphi3_d3(xpq(2)),dphi4_d3(xpq(2))];
    xx3=[dphi1_d3(xpq(3)),dphi2_d3(xpq(3)),dphi3_d3(xpq(3)),dphi4_d3(xpq(3))];
    for i=1:4
        for j=1:4
            K(i,j)=w(1)*xx1(i)*xx1(j)+w(2)*xx2(i)*xx2(j)+w(3)*xx3(i)*xx3(j);
        end
    end
end
K=2*alpha/h*K;
end

function [M]=matrix_masse_local_d3(alpha,beta,h)
    M=zeros(4,4);
    x1=(3+2*sqrt(6/5))/7;
    x2=(3-2*sqrt(6/5))/7;
    xpq=[-sqrt(x1),-sqrt(x2),sqrt(x2),sqrt(x1)];
    w=[(18+sqrt(30))/36,(18-sqrt(30))/36,(18-sqrt(30))/36,(18+sqrt(30))/36];
    xx1=[phi1_d3(xpq(1)),phi2_d3(xpq(1)),phi3_d3(xpq(1)),phi4_d3(xpq(1))];
    xx2=[phi1_d3(xpq(2)),phi2_d3(xpq(2)),phi3_d3(xpq(2)),phi4_d3(xpq(2))];
    xx3=[phi1_d3(xpq(3)),phi2_d3(xpq(3)),phi3_d3(xpq(3)),phi4_d3(xpq(3))];
    xx4=[phi1_d3(xpq(4)),phi2_d3(xpq(4)),phi3_d3(xpq(4)),phi4_d3(xpq(4))];
    xx=[xx1;xx2;xx3;xx4];
    for i=1:4
        for j=1:4
            M(i,j)=w*(xx(1:4,i).*xx(1:4,j));
        end
    end
    M=beta*h/2*M;
end

function [ms]=matrixA_degre_3(N,h,elFinis,alpha,beta,K,M)
    ms=zeros(3*N+1,3*N+1);
    for ie=1:N
        for i=1:4
            ig=3*ie+i-3;
            for j=1:4
                jg=3*ie+j-3;
                ms(ig,jg)=ms(ig,jg)+K(i,j)+M(i,j);
            end
        end
    end
end

```

```

                end
            end
        end
    end

function [ap]=aproximation4points(g)
w=[(18-sqrt(30))/36,(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36];
x1=(3+2*sqrt(6/5))/7
x2=(3-2*sqrt(6/5))/7
xpq=[-sqrt(x1),-sqrt(x2),sqrt(x2),sqrt(x1)];
gpq=[g(xpq(1)),g(xpq(2)),g(xpq(3)),g(xpq(4))];
ap=0;
for i=1:4
    ap=ap+w(i)*gpq(i);
end
end

function [l]=member2_degre_3_s(elFinis ,h ,s , f)
x1=elFinis(3*s-2);
x2=elFinis(3*s-1);
x3=elFinis(3*s);
x4=elFinis(3*s+1);
l=zeros(4,1);
function y=r1(x)
    y=f((x1+x4)/2+h/2*x)*phi1_d3(x);
end
function y=r2(x)
    y=f((x1+x4)/2+h/2*x)*phi2_d3(x);
end
function y=r3(x)
    y=f((x1+x4)/2+h/2*x)*phi3_d3(x);
end
function y=r4(x)
    y=f((x1+x4)/2+h/2*x)*phi4_d3(x);
end
l(1)=aproximation4points(@r1);
l(2)=aproximation4points(@r2);
l(3)=aproximation4points(@r3);
l(4)=aproximation4points(@r4);
end

function [F]=vectMember2_degre_3(elFinis ,h ,N , f)
F=zeros(3*N+1,1);
for s=1:N
    l=member2_degre_3_s(elFinis ,h ,s , f);
    for i=1:4
        ig=3*s+i-3;
    end
end

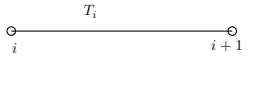
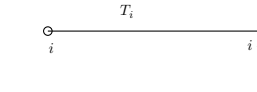
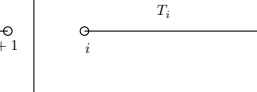



```



```
        F(i g)=F(i g)+h/2*l(i);
    end
end
end

function [U]=solution_degre_3(N,h,elFinis , alpha , beta , f , a , b ,K,M)
    A=matrixA_degre_3(N,h,elFinis , alpha , beta ,K,M);
    A(1 ,:)=0;A(1,1)=1;
    A(3*N+1 ,:)=0;A(3*N+1,3*N+1)=1;
    F=vectMember2_degre_3(elFinis ,h,N,f);
    F(1)=0;
    F(3*N+1)=0;
    U=A\F;
end
```

Comme les programmes en Matlab que nous avons fait, on a la tableau suivante :

1D	P1	P2	P3
Continuité sur élément	C^0	C^0	C^0
Connectivité des éléments			
Degré de liberté $u(x_k)$			
Point de quadrature	Trapèze	Gauss 2 points	Gauss 4 points

Remarquons que la méthode de quadrature de Gauss- Legendre est utilisé dans le calculs approximations $\int_{-1}^1 f(x)dx$, par exemple

– Pour 2 points :

$$\int_{-1}^1 g(x)dx \approx \sum_{i=1}^2 w_i g(x_i)$$

avec

$$x_1 = \frac{1}{\sqrt{3}}, x_2 = -\frac{1}{\sqrt{3}} \text{ et } w_1 = w_2 = 1.$$

– Pour 3 points :

$$\int_{-1}^1 g(x)dx \approx \sum_{i=1}^3 w_i g(x_i)$$

avec

$$x_1 = 0, x_{2,3} = \pm\sqrt{\frac{3}{5}} \text{ et } w_1 = \frac{8}{9}, w_{2,3} = \frac{5}{9}.$$

– Pour 4 points :

$$\int_{-1}^1 g(x)dx \approx \sum_{i=1}^4 w_i g(x_i)$$

avec

$$x_{1,2} = \pm\sqrt{\frac{(3 - 2\sqrt{6/5})}{7}}, x_{3,4} = \pm\sqrt{\frac{(3 + 2\sqrt{6/5})}{7}} \text{ et } w_{1,2} = \frac{18 + \sqrt{30}}{36}, w_{3,4} = \frac{18 - \sqrt{30}}{36}.$$

On reçoit les résultats des programmes en Matlab pour résolution numérique de la solution pour le problème :

$$\begin{cases} -u''(x) + u(x) = x^4 \text{ sur }]0, 1[, \\ u(0) = u(1) = 0 \end{cases}$$

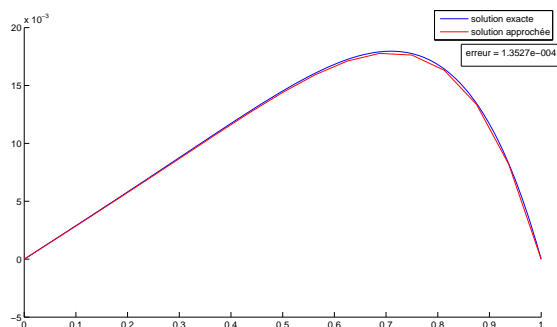


FIGURE 10 – Approcher la solution avec la méthode d’éléments finis de degré 1

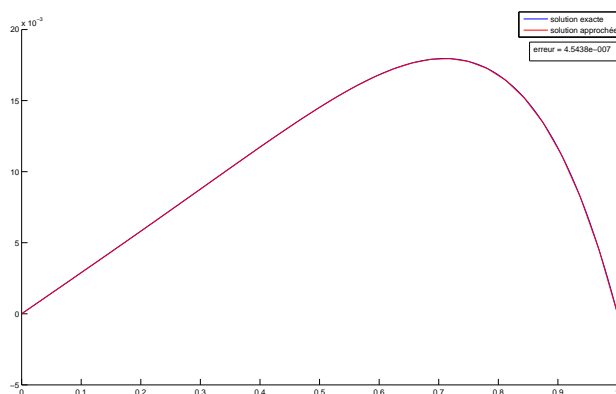


FIGURE 11 – Approcher la solution avec la méthode d’éléments finis de degré 2

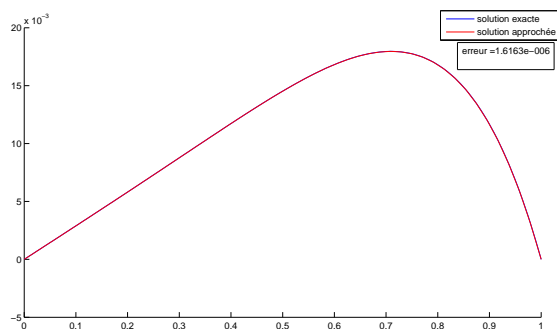


FIGURE 12 – Approcher la solution avec la méthode d’éléments finis de degré 3

2 B-splines

Dans la méthode d'éléments finis classique, le principal défaut de cette méthode est que la solution approchée est une fonction qui n'est que continue. Or, dans de nombreuses applications, par exemple en informatique graphique, il est préférable d'utiliser des fonctions ayant au moins une dérivée continue. Cette propriété sera satisfaite dans la méthode d'éléments finis par B-spline.

Spline est une fonction polynomiale définie par morceaux quelque de degré p sur chaque intervalle.

Une B-spline est une combinaison linéaire de splines non-négatives à support compact minimal.

Pour contruire les fonctions base des B-splines, on doit d'abord introduire la définition de vecteur des noeuds.

2.1 Vecteur des noeuds

Dans l'espace dimension un, un vecteur de noeuds $\{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$ est un ensemble non décroissant de coordonnées dans l'espace des paramètres, où n est le nombre de points de contrôle et p est le degré de la spline. Si les noeuds ξ_i , $i = 1, \dots, n + p + 1$ sont équidistants, on dit que ce vecteur de noeuds est uniforme. Et si les noeuds en première et dernière position sont répétés $p + 1$ fois, on dit que ce vecteur de noeuds est ouvert.

2.2 Les fonctions B-splines

Soit $\{\xi_1, \xi_2, \dots, \xi_k\}$ un vecteur de noeuds. Les fonctions B-splines $N_{i,p}$ sont définies par récurrence sur p par les relations suivantes :

Pour $p = 0$:

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{si } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{sinon} \end{cases} \quad \text{pour } i = 1, \dots, k - 1.$$

Pour $p = l \geq 1$:

$$N_{i,l}(\xi) = \frac{\xi - \xi_i}{\xi_{i+l} - \xi_i} N_{i,l-1}(\xi) + \frac{\xi_{i+l+1} - \xi}{\xi_{i+l+1} - \xi_{i+l}} N_{i+1,l-1}(\xi) \quad \text{pour } i = 1, \dots, k - (l + 1).$$

Remarque :

1. La fonction $N_{i,p}$ est un polynôme de degré inférieur ou égale p sur chaque intervalle $[\xi_j, \xi_{j+1}[$.
2. La fonction $N_{i,p}$ s'annule en dehors de l'intervalle $]\xi_i, \xi_{i+p+1}[$.
3. La fonction $N_{i,p}$ s'annule aussi en ξ_i sauf si $\xi_i = \xi_{i+1} = \dots = \xi_{i+p} < \xi_{i+p+1}$ auquel cas $N_{i,p}(\xi_i) = 1$.
4. $N_{i,p}(\xi) \geq 0$, $\forall \xi$ et $\sum_{i=1}^n N_{i,p}(\xi) = 1$, $\forall \xi$.

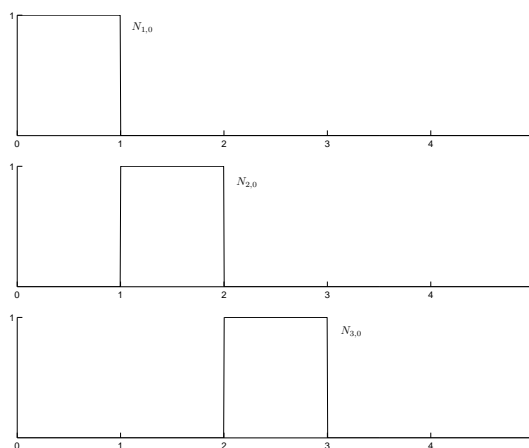


FIGURE 13 – Fonctions B-splines de degré 0

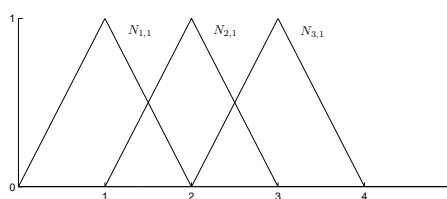


FIGURE 14 – Fonctions B-splines de degré 1

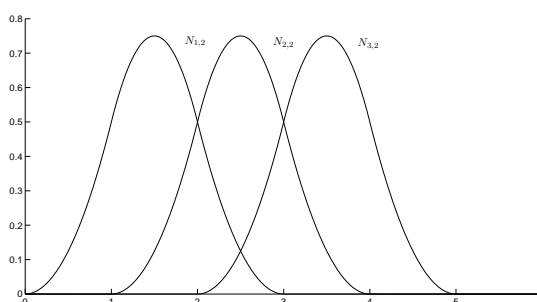


FIGURE 15 – Fonctions B-splines de degré 2

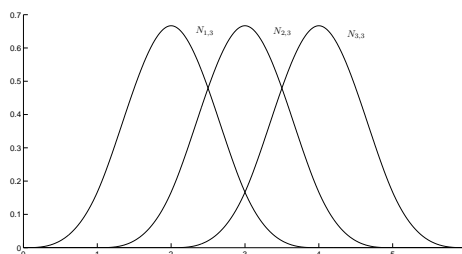


FIGURE 16 – Fonctions B-splines de degré 3

On peut construire une matrice de dimension $(n + p) \times (p + 1)$ qui stocke toutes les fonctions B-splines $N_{i,l}(\xi)$ avec $0 \leq l \leq p$ en forme :

$$\begin{pmatrix} N_{1,0}(\xi) & N_{1,1}(\xi) & \dots & N_{1,p}(\xi) \\ N_{2,0}(\xi) & N_{2,1}(\xi) & \dots & N_{2,p}(\xi) \\ \dots & \dots & \dots & \dots \\ N_{n,0}(\xi) & N_{n,1}(\xi) & \dots & N_{n,p}(\xi) \\ N_{n+1,0}(\xi) & N_{n+1,1}(\xi) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ N_{n+p-1,0}(\xi) & N_{n+p-1,1}(\xi) & \dots & 0 \\ N_{n+p,0}(\xi) & 0 & \dots & 0 \end{pmatrix}$$

D'où on a le programme suivant :

Programme 1 - Bsp : Calcule les fonctions B-splines au point ksi.

```
function [sp]=Bsp(ksiVector,n,p,ksi)
    sp=zeros(n+p,p+1);
    for j=1:p+1
        j0=j-1;
        for i=1:n+p-j0
            ki=ksiVector(i);
            ki1=ksiVector(i+1);
            if(j0==0)
                if ( ksi>=ki && ksi<ki1 )
                    sp(i,j)=1;
                else
                    sp(i,j)=0;
                end
            else
                kip=ksiVector(i+j0);
                kip1=ksiVector(i+j0+1);
                if ( kip==ki )
                    tg=0;
                else
                    tg=(ksi-ki)/(kip-ki);
                end
                if ( kip1==ki1 )
                    td=0;
                else
                    td=(kip1-ksi)/(kip1-ki1);
                end
                sp(i,j)=tg*sp(i,j0)+td*sp(i+1,j0);
            end
        end
    end
end
```

2.3 Les dérivées de fonctions B-splines

La dérivée d'une fonction de base B-spline est donnée par :

$$N'_{i,p}(\xi) = \frac{p}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi).$$

D'où on a le programme suivant :

Programme 2 - deriveBsp : Calcule les dérivées de fonctions splines au point ksi.

```
function [dsp]=deriveBsp(ksiVector,n,p,ksi)
    U=Bsp(ksiVector,n,p,ksi);
    dsp=zeros(n+p,p+1);
    for j=1:p+1
        j0=j-1;
        for i=1:n+p-j0
            ki=ksiVector(i);
            ki1=ksiVector(i+1);
            if(j0==0)
                dsp(i,j)=0;
            else
                kip=ksiVector(i+j0);
                kip1=ksiVector(i+j0+1);
                if ( kip==ki )
                    dtg=0;
                else
                    dtg=j0/(kip-ki);
                end
                if ( kip1==ki1 )
                    dtd=0;
                else
                    dtd=-j0/(kip1-ki1);
                end
                dsp(i,j)=dtg*U(i,j0)+dtd*U(i+1,j0);
            end
        end
    end
end
```

2.4 Courbes B-splines

Une courbe B-splines de degré p définie par n points de contrôle P_1, P_2, \dots, P_n est de la forme :

$$X_p(\xi) = (x(\xi), y(\xi)) = \sum_{i=1}^n N_{i,p}(\xi) P_i,$$

où $P_i = (X_i, Y_i)$ sont les coordonnées de $i^{\text{ème}}$ -point de contrôle.

Remarque :

1. Les composantes de $X_p(\xi)$ sont des polynômes de degré p sur chaque intervalle $[\xi_i, \xi_{i+1}[$.
2. Si $\xi \in [t_i, t_{i+1}[$, $X_p(\xi)$ ne dépend que les points de contrôle P_{i-p}, \dots, P_i et se trouve dans l'enveloppe convexe de ces points.
3. Si ξ_i est un noeud simple et $p \geq 1$, $X_p(\xi_i)$ ne dépend que des points de contrôle P_{i-p}, \dots, P_{i-1} et se trouve dans l'enveloppe convexe de ces points.

On a les programmes suivants :

Programme 3 - iwBsp : Calcule les coordonnées d'un point d'une spline.

```
function [C]=iwBsp(ksiVector,points,p,ksi)
n=size(points,2);
xi=points(1,:);
yi=points(2,:);
N=Bsp(ksiVector,n,p,ksi);
Nip=N(1:n,p+1);
x=xi*Nip;
y=yi*Nip;
C(1)=x;
C(2)=y;
end
```

Programme 4 - ip : Calcule les coordonnées de plusieurs points d'une spline.

```
function[P]=ip(ksiVector,points,p,vect)
l=size(vect,2);
P=zeros(l,2);
for i=1:l
    t=vect(i);
    ki=iwBsp(ksiVector,points,p,t);
    P(i,1)=ki(1);
    P(i,2)=ki(2);
end
end
```

2.5 Interpolation par des courbes B-splines

On se limite aux courbes de degré 3 par morceaux. On cherche à faire passer une courbe B-spline en $N - 1$ points Q_i et en imposant la dérivées aux extrémités. Le problème se divise en deux phases.

Première phase : On se fixe un vecteur de noeuds t et on cherche un polygone de contrôle P tel que la courbe B-spline X_k correspondante passe par les Q_i aux noeuds. L'interpolation se traduit alors par la résolution d'un système linéaire.

Deuxième phase : On cherche à optimiser le choix du vecteur de noeuds. Ce problème est typiquement non linéaire.

2.5.1 Le problème linéaire

On a le théorème suivant pour des B-splines de degré 3.

Théorème 3 Soient Q_0, \dots, Q_N des points de \mathbb{R}^n . Soient v_a, v_b deux vecteurs de \mathbb{R}^n . Soit t un vecteur de noeuds vissé aux extrémités, de la forme

$$t_0 = t_1 = t_2 = t_3 = a < t_4 < \dots < t_{N+2} < b = t_{N+3} = t_{N+4} = t_{N+5} = t_{N+6}.$$

Il existe un unique polygone de contrôle $P = (P_0, \dots, P_{N+2})$ tel que la courbe B-spline de degré 3 associée satisfasse

$$\forall i = 0, \dots, N, \quad X_3(t_{i+3}) = Q_i, \quad X'(a) = v_a, \quad \text{et} \quad X'(b) = v_b.$$

Preuve. Comme chaque coordonnée se traite indépendamment, on peut supposer que $n = 1$. Dans ce cas, on considère l'application linéaire

$$\mathbb{R}^{N+3} \longrightarrow \mathbb{R}^{N+3}, (P_0, \dots, P_{N+2}) \longmapsto (X_3'(a), X_3(t_3), \dots, X_3(t_{N+3}), X_3'(b))$$

Pour prouver qu'elle est bijective, il suffit de prouver qu'elle est injective. Cela résulte du lemme 2 appliqué à la fonction $f = 0$. En effet, si les points et vecteurs interpolés sont tous nuls, le lemme donne X_3'' . Avec la condition initiale $X_3(a) = X_3'(a) = 0$, cela entraîne que X_3 , et donc que les P_i sont tous nuls.

□

Lemme 4 Soient $f, x : [a, b] \longrightarrow \mathbb{R}$ deux fonctions de classe C^2 . On suppose que i , x est polynomiale de degré 3 sur chaque intervalle.

ii, $f(t_i) = x(t_i)$ pour $i = 3, \dots, N + 3$ et $f'(a) = x'(a)$, $f'(b) = x'(b)$.

Alors,

$$\int_a^b (f''(t) - x''(t))^2 dt = \int_a^b f''(t)^2 dt - \int_a^b x''(t)^2 dt.$$

Preuve. On vérifie immédiatement que

$$\int_a^b (f''(t) - x''(t))^2 dt - \int_a^b f''(t)^2 dt + \int_a^b x''(t)^2 dt = -2R, \text{ où } R = \int_a^b (f''(t) - x''(t))x''(t) dt.$$

On intègre par parties sur chaque intervalle

$$\begin{aligned} \int_{t_i}^{t_{i+1}} (f''(t) - x''(t))x''(t) dt &= (f'(t_{i+1}) - x'(t_{i+1}))x''(t_{i+1}) - (f'(t_i) - x'(t_i))x''(t_i) \\ &\quad - \int_{t_i}^{t_{i+1}} (f'(t) - x'(t))x'''(t) dt \\ &= (f'(t_{i+1}) - x'(t_{i+1}))x''(t_{i+1}) - (f'(t_i) - x'(t_i))x''(t_i) \\ &\quad - (f(t_{i+1}) - x(t_{i+1}))x'''(t_{i+1}) + (f(t_i) - x(t_i))x'''(t_i) \\ &\quad + \int_{t_i}^{t_{i+1}} (f(t) - x(t))x^{(4)}(t) dt \\ &= (f'(t_{i+1}) - x'(t_{i+1}))x''(t_{i+1}) - (f'(t_i) - x'(t_i))x''(t_i), \end{aligned}$$

car $x^{(4)} \equiv 0$ et $f(t_{i+1}) - x(t_{i+1}) = f(t_i) - x(t_i) = 0$. En additionnant, il vient

$$R = (f'(b) - x'(b))x''(b) - (f'(a) - x'(a))x''(a) = 0,$$

car $f'(a) - x'(a) = f'(b) - x'(b) = 0$.

□

Théorème 5 Soit $f : [a, b] \rightarrow \mathbb{R}$ une fonction de classe C^2 . Soit X_3 la fonction B-spline de degré 3 qui l'interpole en $N + 1$ point plus les dérivées aux bornes, selon le théorème 1. Alors

$$\|f - X_3\|_\infty \leq \frac{h^{3/2}}{2} \|f''\|_2 \quad \text{et} \quad \|f' - X_3'\|_\infty \leq h^{1/2} \|f''\|_2,$$

où $h = \max|t_{i+1} - t_i|$.

Preuve. Posons $g = f - X_3$. Le lemme 2 donne

$$\begin{aligned} \|g''\|_2^2 &= \int_a^b g''(t)^2 dt \\ &= \int_a^b f''(t)^2 dt - \int_a^b X_3''(t)^2 dt \\ &\leq \|f''\|_2^2. \end{aligned}$$

Par construction, g s'annule aux t_i , donc, d'après le théorème de Rolle, g' s'annule au moins une fois dans chaque intervalle $[t_i, t_{i+1}]$. Tout point $t \in [a, b]$ est donc à distance au plus h d'un point t' tel que $g'(t') = 0$. On écrit :

$$\begin{aligned} |g'(t)| &= |g'(t) - g'(t')| \\ &= \left| \int_{t'}^t g''(s) ds \right| \\ &\leq \left(\int_{t'}^t ds \right)^{1/2} \left(\int_{t'}^t g''(s)^2 ds \right)^{1/2} \\ &\leq h^{1/2} \|g''\|_2 \\ &\leq h^{1/2} \|f''\|_2, \end{aligned}$$

d'après l'inégalité de Cauchy- Schwarz. Ceci montre que

$$\|g'\|_\infty \leq h^{1/2} \|f''\|_2.$$

Tout point $t \in [a, b]$ est à distance au plus $\frac{h}{2}$ d'un point t'' tel que $g(t'') = 0$. Par conséquent,

$$\begin{aligned} |g(t)| &= |g(t) - g(t'')| \\ &= \left| \int_{t''}^t g'(s) ds \right| \\ &\leq |t - t''| \|g'\|_\infty \\ &\leq \frac{h^{1/2}}{2} \|f''\|_2. \end{aligned}$$

□

Remarque Le choix de l'espacement uniforme n'est pas la meilleure solution.

2.5.2 Résolution numérique d'un problème d'interpolation

On se donne le vecteur de noeuds

$$t_0 = t_1 = t_2 = t_3 = 0 < t_4 = 1 < \dots < t_{N+2} = N - 1 < N = t_{N+3} = t_{N+4} = t_{N+5} = t_{N+6}.$$

dans l'intervalle $[0, N]$. Il s'agit de trouver le polygone de contrôle (à $N + 3$ sommets) de la B-spline qui passe par le point Q_i en t_{i+3} et pour dérivées v_0 (resp. v_N) aux extrémités. On peut supposer que $n = 1$.

Il s'agit de résoudre le système $AP = Q$ pour $Q = (Q_0, v_0, Q_1, \dots, Q_{N-1}, v_N, Q_N)$ et $P \in \mathbb{R}^{N+3}$ et

$$A = \begin{pmatrix} N_{1,p}(a) & 0 & \dots & \dots & \dots & 0 \\ N'_{1,p}(a) & N'_{2,p}(a) & 0 & \dots & \dots & 0 \\ 0 & N_{2,p}(a+h) & N_{3,p}(a+h) & N_{4,p}(a+h) & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & N'_{N,p}(b) \\ 0 & 0 & \dots & \dots & \dots & N_{N,p}(b) \end{pmatrix}$$

avec $p = 3, h = t_i - t_{i-1} = 1$. On utilise les valeurs des fonctions B-splines relatives au vecteur de noeuds ci-dessus :

$$\begin{aligned} X_3(0) &= 0, \\ X'_3(0) &= 3(P_1 - P_0), \\ X_3(t_4) &= \frac{1}{4}P_1 + \frac{7}{12}P_2 + \frac{1}{6}P_3, \\ X_3(t_{i+3}) &= \frac{1}{6}P_i + \frac{2}{3}P_{i+1} + \frac{1}{6}P_{i+2}, \end{aligned}$$

pour $i \geq 2$. On a la matrice suivante :

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ -3 & 3 & 0 & 0 & 0 & \dots & \dots & \dots & \dots \\ 0 & \frac{1}{4} & \frac{7}{12} & \frac{1}{6} & 0 & \dots & \dots & \dots & \dots \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \frac{1}{6} & \frac{7}{12} & \frac{1}{4} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & -3 & 3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 1 \end{pmatrix}$$

On peut montrer directement que le système linéaire $AP = Q$ a exactement une solution en montrant que la matrice A est inversible. On remarque que $\det A = 3\det B$ où la matrice

B est :

$$B = \begin{pmatrix} \frac{7}{12} & \frac{1}{6} & 0 & \cdots & \cdots \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \cdots & \cdots \\ 0 & \ddots & \ddots & \cdots & \cdots \\ \vdots & \vdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \vdots & \vdots & \vdots & \frac{1}{6} & \frac{7}{12} \end{pmatrix}$$

Pour montrer que B est inversible, on prouve que $(Bv, v) \neq 0 \quad \forall v \in \mathbb{R}^{N-1} \setminus \{0\}, v = (v_1, \dots, v_{N-1})$. On a

$$\begin{aligned} (Bv, v) &= \left(\frac{7}{12}v_1 + \frac{1}{6}v_2\right)v_1 + \sum_{i=1}^{N-3} \left(\frac{1}{6}v_i + \frac{2}{3}v_{i+1} + \frac{1}{6}v_{i+2}\right)v_{i+1} + \left(\frac{7}{12}v_{N-1} + \frac{1}{6}v_{N-2}\right)v_{N-1} \\ &= \frac{1}{3} \sum_{i=2}^{N-2} v_i^2 + \frac{1}{6} \sum_{i=1}^{N-2} (v_i + v_{i+1})^2 + \frac{5}{12}v_1^2 + \frac{5}{12}v_{N-1}^2 > 0 \quad \forall v \in \mathbb{R}^{N-1} \setminus \{0\}. \end{aligned}$$

Cela indique que la matrice B est inversible.

En Matlab, on a les programmes suivants pour calculer le polygone de contrôle et tester l'interpolation de la courbe par calcul automatique des points de contrôle :

Programme 5 - control : Calcule le polygone de contrôle.

% fonction qui renvoie les ordonnées des points de controle, leurs abcisses sont les noeuds

% N+1 : nombre de points d'interpolation

% p : degré de la spline

% a, b : bornes de l' intervalle [a,b]sur laquelle on trace la courbe

% vi, vf : tangentes à la courbe aux bornes

% Qpoints : ordonnées des points d'interpolation, leurs abcisses sont les noeuds

function [Polygon]=control(N,p,vi,vf,a,b,Qpoints)

% vecteur nodal avec des noeuds multiples en debut et fin

ksiVector=zeros(1,N+2*p+1);

ksiVector(1:p)=a;

ksiVector(p+1:N+p+1)=linspace(a,b,N+1);

ksiVector(N+p+2:N+2*p+1)=b;

%matrice A du système linéaire

A=zeros(N+3,N+3);

%premiere ligne

r=Bsp(ksiVector,N+3,p,a);

rr=r(1:N+3,p+1);

A(1,:)=rr';

%deuxieme ligne

s=deriveBsp(ksiVector,N+3,p,a);

ss=s(1:N+3,p+1);

A(2,:)=ss';

%avant-dereniere ligne

```

t=deriveBsp(ksiVector,N+3,p,b);
tt=t(1:N+3,p+1);
A(N+2,:)=tt';

%derniere ligne
u=Bsp(ksiVector,N+3,p,b);
uu=u(1:N+3,p+1);
A(N+3,:)=uu';

%lignes de A
for i=3:N+1
    v=Bsp(ksiVector,N+3,p,ksiVector(i+p-1));
    vv=v(1:N+3,p+1);
    A(i,:)=vv';
end

Q=zeros(N+3,1);
Q(1)=Qpoints(1);
Q(2)=vi;
Q(3:N+1)=Qpoints(2:N);
Q(N+2)=vf;
Q(N+3)=Qpoints(N+1);

Polygon= (A'*A)\(A'*Q);

end

```

Programme 6 - spline1 : Test d'interpolation de la courbe par calcul automatique des points de contrôle.

```

function test1=spline1()
p=3;
a=-5;
b=5;
N=6;
points2trace=linspace(a,b,1000);
ksiVector=zeros(1,N+2*p+1);
ksiVector(1:p)=a;
ksiVector(p+1:N+p+1)=linspace(a,b,N+1);
ksiVector(N+p+2:N+2*p+1)=b;
vi=0;
vf=0;

Qp=zeros(N+1);
for i=1:N+1
    w=ksiVector(i+p);
    Qp(i)=1/(1+w^2);
end

```

```

end

pc=control(N,p,vi,vf,a,b,Qp);
pointsControles=zeros(2,N+3);
pointsControles(1,:)=ksiVector(p:N+p+2);
for i=1:N+3
pointsControles(2,i)=pc(i,1);
end

m2=ip(ksiVector,pointsControles,p,points2trace);

hold on
ezplot('1/(1+x*x)');
q=plot(m2(:,1),m2(:,2),'green');
hold off
end

```

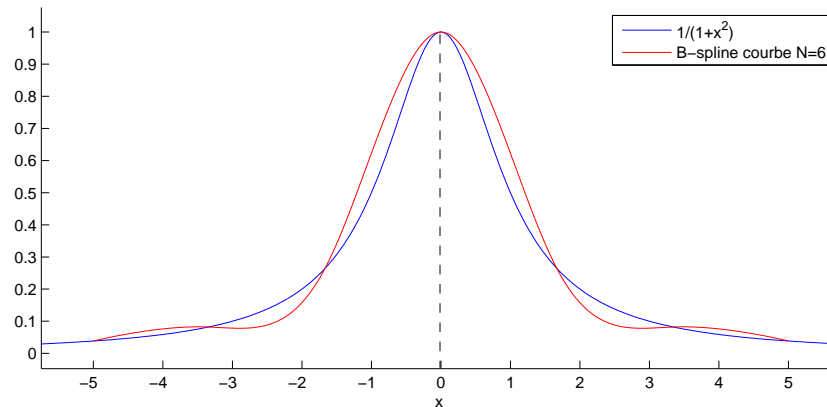


FIGURE 17 – Interpolation sur $[-5, 5]$ avec $N = 6$

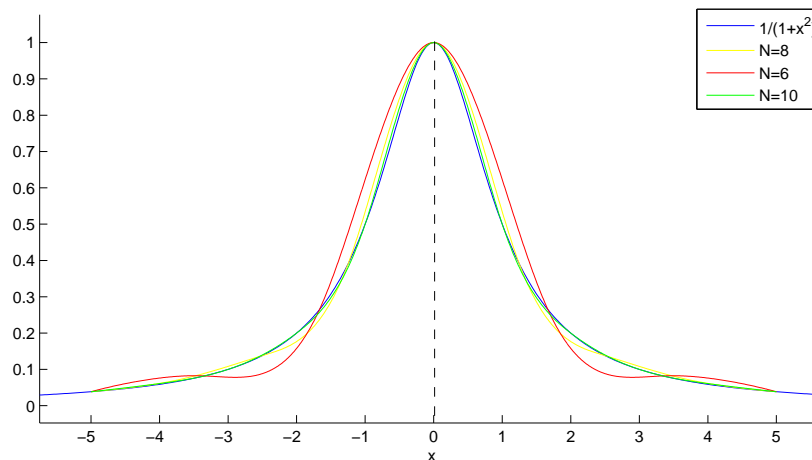


FIGURE 18 – Interpolation sur $[-5, 5]$ avec $N = 6, N = 8, N = 10$

2.6 La méthode d'éléments finis par B-splines

Dans cette méthode, on reconsidère le problème de Dirichlet homogène dans la méthode d'éléments finis classique. Mais, on choisit les fonctions bases de V_h sont les fonctions B-splines de degré p : $w_i := N_{i,p}$, $i = 1, \dots, n$. Alors on peut écrire le problème approché dans $V_{0,h}$ suivant : Trouver la fonction u_h appartenant à $V_{0,h}$ telle que :

$$\begin{aligned} \int_a^b \alpha u_h'(x) N_{i,p}'(x) dx + \alpha u_h'(a) N_{i,p}(a) - \alpha u_h'(b) N_{i,p}(b) + \int_a^b \beta u_h(x) N_{i,p}(x) dx \\ = \int_a^b f(x) N_{i,p}(x), \forall i = 2, \dots, n-1. \end{aligned}$$

D'autre part, si u_h est une solution du problème approché dans $V_{0,h}$, on peut exprimer :

$$u_h(x) = \sum_{j=2}^{n-1} u_j N_{j,p}(x).$$

Ainsi, le problème approché : Trouver u_2, u_3, \dots, u_{n-1} tels que

$$\begin{aligned} \sum_{j=2}^{n-1} \left(\int_a^b \alpha N_{j,p}'(x) N_{i,p}'(x) dx + \alpha N_{j,p}'(a) N_{i,p}(a) - \alpha N_{j,p}'(b) N_{i,p}(b) + \int_a^b \beta N_{j,p}(x) N_{i,p}(x) dx \right) u_j \\ = \int_a^b f(x) N_{i,p}(x), \forall i = 2, \dots, n-1. \end{aligned}$$

Posons

$$F_i := \int_a^b f(x) N_{i,p}(x) dx,$$

$$A_{ij} := \int_a^b \alpha N_{j,p}'(x) N_{i,p}'(x) dx + \alpha N_{j,p}'(a) N_{i,p}(a) - \alpha N_{j,p}'(b) N_{i,p}(b) + \int_a^b \beta N_{j,p}(x) N_{i,p}(x) dx.$$

On remarque que le problème approché prenant la forme d'un système linéaire de $n-2$ équations à $n-2$ inconnues, qui peut s'écrire sous la forme matricielle suivante :

$$AU = F.$$

Calcul matrice membre gauche

Chaque composante A_{ij} de la matrice membre gauche s'écrit

$$A_{ij} = m_{ij} + c_{ij},$$

où

$$m_{ij} = \int_a^b \alpha N_{j,p}'(x) N_{i,p}'(x) dx + \int_a^b \beta N_{j,p}(x) N_{i,p}(x) dx$$

et

$$c_{ij} = \alpha N_{j,p}'(a) N_{i,p}(a) - \alpha N_{j,p}'(b) N_{i,p}(b).$$

Chaque composante m_{ij} est également calculée par assemblage de contributions élémentaires :

$$\begin{aligned} m_{ij} &= \int_a^b (\alpha N'_{j,p}(x)N'_{i,p}(x) + \beta N_{j,p}(x)N_{i,p}(x))dx \\ &= \sum_{s=1}^{n-p} \int_{T_s} (\alpha N'_{j,p}(x)N'_{i,p}(x) + \beta N_{j,p}(x)N_{i,p}(x))dx \end{aligned}$$

On remarque que sur l'élément T_s , il n'a que $p+1$ fonctions B-splines $N_{s,p}, N_{s+1,p}, \dots, N_{s+p,p}$ non nulles et il n'a que $p+1$ dérivés $N'_{s,p}, N'_{s+1,p}, \dots, N'_{s+p,p}$ non nulles. On peut donc utiliser la technique d'assemblage pour calculer les composantes m_{ij} .

Calcul des composantes du second - membre

Chaque composante F_i du vecteur second - membre global est également calculée par assemblage de contributions élémentaires :

$$F_i = \int_a^b f(x)N_{i,p}(x)dx = \sum_{s=1}^{n-p} \int_{T_s} f(x)N_{i,p}(x)dx$$

Sur élément T_s , il n'a que $p+1$ fonctions B-splines $N_{s,p}, N_{s+1,p}, \dots, N_{s+p,p}$ non nulles. On peut donc utiliser la technique d'assemblage pour calculer les composantes F_i .

On a le programme suivant en matlab pour approximation solution u par B-splines :

```
function test=ElementFini1DparBspline ()
ComparaisonparBspline ();
end
function test0=ComparaisonparBspline ()
    nh=15; % nombre de liberte
    p=3;
    alpha=1;
    beta=1;
    a=0;
    b=1;
    nbpt=200;
    shift=(b-a)/1000;
    I=linspace(a,b-shift,nbpt);
    function y=f(x)
        y=x^4;
    end

    [g,G]=solution(p,nh,alpha,beta,@f,a,b,I);
    gp=derSolution(p,nh,G,I,a,b);
    x=I;
    c2=(37-24*exp(1))/(exp(1)-exp(-1));
    c1=-24-c2;
    s_exact=zeros(1,size(x,2));
```



```
for i=1:size(x,2)
    s_exact(i)=c1*exp(x(i)) + c2*exp(-x(i))+ x(i)^4 + 12*x(i)^2 + 24
end
err=g-s_exact;
em=max(abs(err))

hold on
close all;
e=plot(I, err);
xlabel('x'); ylabel('Erreur'); title('Erreur du solution exacte');
set(e, 'Color', 'black', 'LineWidth', 2);
hold off
figure;
p=plot(I, s_exact);
xlabel('x'); ylabel('u(x)'); title('Solution exacte');
set(p, 'Color', 'blue', 'LineWidth', 2);

figure;
q=plot(I, g);
xlabel('x'); ylabel('u(x)'); title('Solution approchée');
set(q, 'Color', 'red', 'LineWidth', 1.5);
hold off
figure;
d=plot(I, gp);
xlabel('x'); ylabel('u''(x)');
title('Première dérivée de la solution approchée');
set(d, 'Color', 'blue', 'LineWidth', 1.5);
% hold off

end

function [sp]=Bsp(ksiVector, n, p, ksi)
sp=zeros(n+p, p+1);
for j=1:p+1
    j0=j-1;
    for i=1:n+p-j0
        ki=ksiVector(i);
        ki1=ksiVector(i+1);
        if(j0==0)
            if ( ksi>=ki && ksi<ki1)
                sp(i, j)=1;
            else
                sp(i, j)=0;
            end
        else
            kip=ksiVector(i+j0);
```

```

        kip1=ksiVector(i+j0+1);

        if (kip==ki)
            tg=0;
        else
            tg=(ksi-ki)/(kip-ki);
        end

        if (kip1==ki1)
            td=0;
        else
            td=(kip1-ksi)/(kip1-ki1);
        end
        sp(i,j)=tg*sp(i,j0)+td*sp(i+1,j0);
    end
end
end

end

end

% calcule les dérivées de fonctions splines au point ksi
function [dsp]=deriveBsp(ksiVector,n,p,ksi)
    U=Bsp(ksiVector,n,p,ksi);
    dsp=zeros(n+p,p+1);
    for j=1:p+1
        j0=j-1;
        for i=1:n+p-j0
            ki=ksiVector(i);
            ki1=ksiVector(i+1);
            if(j0==0)
                dsp(i,j)=0;
            else
                kip=ksiVector(i+j0);
                kip1=ksiVector(i+j0+1);
                if (kip==ki)
                    dtg=0;
                else
                    dtg=j0/(kip-ki);
                end

                if (kip1==ki1)
                    dtd=0;
                else
                    dtd=-j0/(kip1-ki1);
                end
                dsp(i,j)=dtg*U(i,j0)+dtd*U(i+1,j0);
            end
        end
    end
end
end

```

```

    end
end

function nip=splineKsi(ksiVector ,n,p,ksi)
    mat=Bsp(ksiVector ,n,p,ksi);
    nip=mat(n,p+1);
end

function dnip=dsplineKsi(ksiVector ,n,p,ksi)
    mat=deriveBsp(ksiVector ,n,p,ksi);
    dnip=mat(n,p+1);
end

function [ap]=aproximation(g,x0,x1)
    w=[(18-sqrt(30))/36,(18+sqrt(30))/36,(18+sqrt(30))/36,(18-sqrt(30))/36];
    xpq=[-sqrt((3+2*sqrt(6/5))/7),-sqrt((3-2*sqrt(6/5))/7),
        sqrt((3-2*sqrt(6/5))/7),sqrt((3+2*sqrt(6/5))/7)];
    gpq=[g((x0+x1)/2+(x1-x0)/2*xpq(1)),g((x0+x1)/2+(x1-x0)/2*xpq(2)),
        g((x0+x1)/2+(x1-x0)/2*xpq(3)),g((x0+x1)/2+(x1-x0)/2*xpq(4))];
    ap=0;
    for i=1:4
        ap=ap+w(i)*gpq(i);
    end
    ap=ap*(x1-x0)/2;
end

function [ies]=integreElement_s(elFinis ,s,g)
    x0=elFinis(1);
    x1=elFinis(2);
    lk=size(elFinis,2)-1;
    pas=(elFinis(lk+1)-elFinis(1))/lk;
    function y=h(x)
        y=g(x+(s-1)*pas);
    end
    ies=aproximation(@h,x0,x1);
end

function mge=ijfunction(ksiVector ,p,alpha,beta,i,j,x)
    Nip=splineKsi(ksiVector ,i,p,x);
    Njp=splineKsi(ksiVector ,j,p,x);
    dNip=dsplineKsi(ksiVector ,i,p,x);
    dNjp=dsplineKsi(ksiVector ,j,p,x);
    mge=(alpha*dNip*dNjp)+(beta*Nip*Njp);
end

function [ijfs]=intg_ijfunction_Element_s(ksiVector ,p,alpha,beta,i,j,elFinis)
    ijfs=integreElement_s(elFinis ,s,@aij);

```

```

function aij=aij(x)
    aij=zeros(size(x,1),size(x,2));
    for k=1:size(x,2)
        aij(k)=ijfunction(ksiVector,p,alpha,beta,i,j,x(k));
    end
end

function [ms]=matsolution(ksiVector,elFinis,p,nh,alpha,beta)
    ms=zeros(nh,nh);
    c=zeros(nh,nh);
    lh=nh-p;
    r=size(elFinis,2);
    for i=1:nh
        for j=1:nh
            Nipa=splineKsi(ksiVector,j,p,elFinis(1));
            dNjpa=dsplineKsi(ksiVector,i,p,elFinis(1));
            Nipb=splineKsi(ksiVector,j,p,elFinis(r));
            dNjpb=dsplineKsi(ksiVector,i,p,elFinis(r));
            c(i,j)=alpha*(dNjpa*Nipa-dNjpb*Nipb);
        end
    end
    for s=1:lh
        for i=s:s+p
            for j=s:s+p
                ms(i,j)=ms(i,j)+intg_ijfunction_Element_s(ksiVector,p,alpha,beta,i,j,elF
            end
        end
    end
end

function mgem2=jfunctionm2(ksiVector,p,f,j,x)
    Njp=splineKsi(ksiVector,j,p,x);
    mgem2=f(x)*Njp;
end

function [jfs]=intg_jfunction_Element_s(ksiVector,p,alpha,beta,j,elFinis,
jfs=integreElement_s(elFinis,s,@bj);
function bj=bj(x)
    bj=zeros(size(x,1),size(x,2));
    for k=1:size(x,2)
        bj(k)=jfunctionm2(ksiVector,p,f,j,x(k));
    end
end

function [l]=vectMember2(ksiVector,elFinis,p,alpha,beta,nh,f)

```

```
l=zeros(nh,1);
lh=nh-p;
for s=1:lh
    for j=s:s+p
        l(j)=l(j)+intg_jfunction_Element_s(ksiVector,p,alpha,beta,j,c);
    end
end
end
end
```

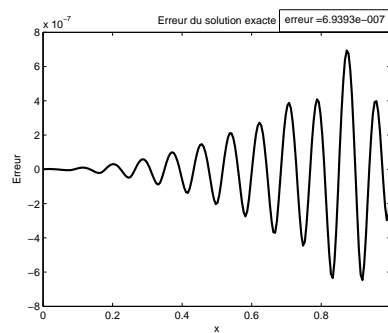
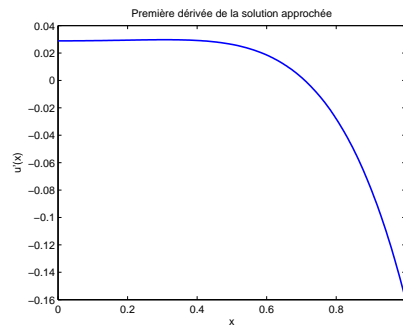
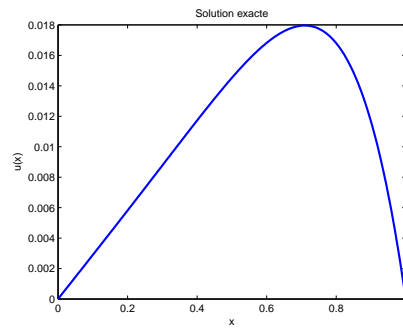
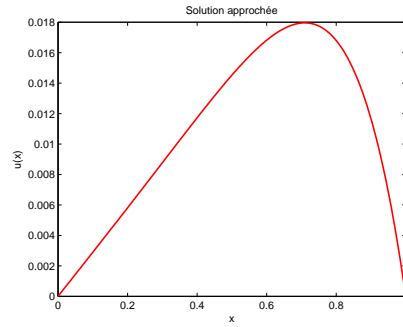
```
function [U]=degres_de_liberte(p,nh,alpha,beta,f,a,b)
ksiVector=zeros(1,nh+p+1);
lh=nh-p;
ksiVector(1,1:p)=a;
ksiVector(1,p+1:lh+p+1)=linspace(a,b,lh+1);
ksiVector(1,lh+p+2:nh+p+1)=b;
elFinis=ksiVector(1,p+1:lh+p+1);

A=matsolution(ksiVector,elFinis,p,nh,alpha,beta);
A(1,:)=0;A(1,1)=1;
A(nh,:)=0;A(nh,nh)=1;
L=vectMember2(ksiVector,elFinis,p,alpha,beta,nh,f);
L(1)=0;
L(nh)=0;
U=A\L;
end
```

```
function [u,U]=solution(p,nh,alpha,beta,f,a,b,I)
ksiVector=zeros(1,nh+p+1);
lh=nh-p;
ksiVector(1,1:p)=a;
ksiVector(1,p+1:lh+p+1)=linspace(a,b,lh+1);
ksiVector(1,lh+p+2:nh+p+1)=b;
u=zeros(1,size(I,2));
U=degres_de_liberte(p,nh,alpha,beta,f,a,b);
for x=1:size(I,2)
    u(x)=0;
    for i=1:nh
        u(x)=u(x)+U(i)*splineKsi(ksiVector,i,p,I(x));
    end
end
end
end
```

```
function u=derSolution(p,nh,U,I,a,b)
ksiVector=zeros(1,nh+p+1);
lh=nh-p;
ksiVector(1,1:p)=a;
ksiVector(1,p+1:lh+p+1)=linspace(a,b,lh+1);
```

```
ksiVector(1, lh+p+2:nh+p+1)=b;  
u=zeros(1, size(I, 2));  
for x=1:size(I, 2)  
    u(x)=0;  
    for i=1:nh  
        u(x)=u(x)+U(i)*dsplineKsi(ksiVector, i, p, I(x));  
    end  
end  
end  
end
```



3 Commentaire

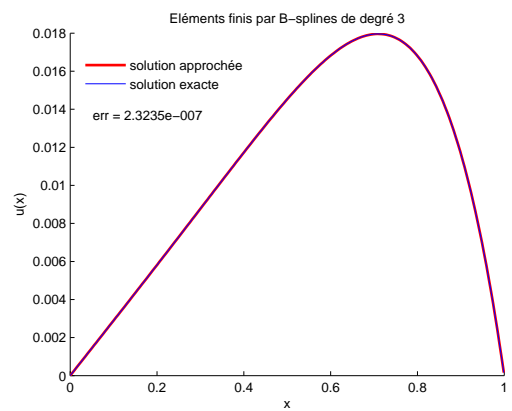
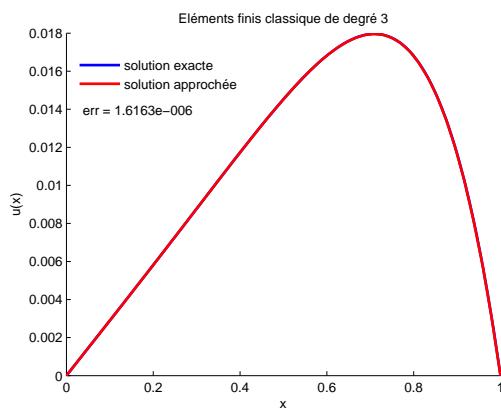
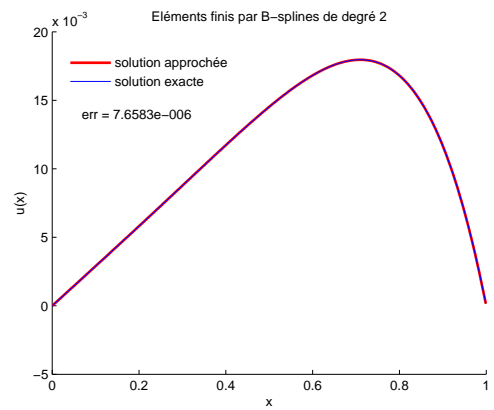
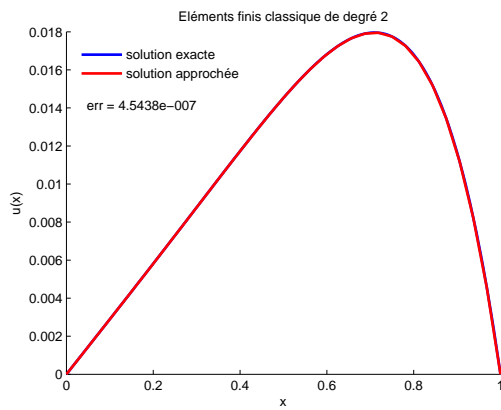
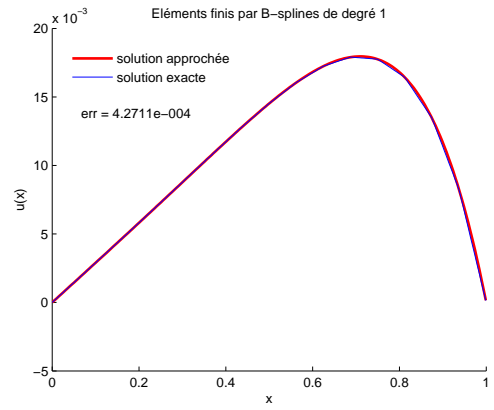
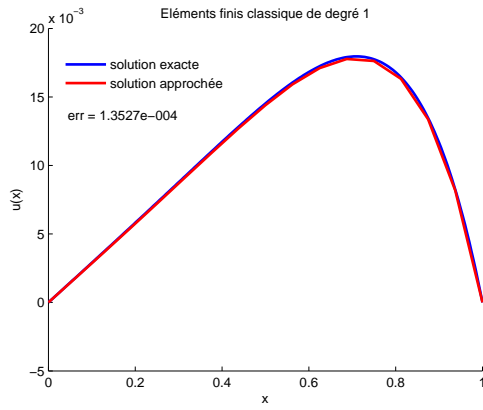
Commentaire sur la méthode d'éléments finis classique et la méthode d'éléments finis par B-splines

1. La similarité

Dans les deux méthodes, on utilise la même formulation variationnelle et la même technique l'assemblage.

2. La différence

- La base : Dans la méthode d'éléments finis classique, on choisit les fonctions d'interpolations de Lagrange et dans la méthode d'éléments finis par B-splines, on choisit les fonctions B-splines. Pour le degré un, les bases de deux méthodes sont coïncides, mais elles sont différentes en degré supérieur à 1.
- La continuité : Les fonctions d'interpolations de Lagrange de degré p sont seulement de classe C^0 mais les fonctions B-splines de degré p sont de classe C^{p-1} .
- Les fonctions d'interpolations de Lagrange peuvent recevoir des valeurs négatives, mais les fonctions B-splines sont toujours positifs, donc tous les composants de la matrice de raideur dans la méthode des éléments finis par B-splines sont toujours positifs.
- Le nombre de fonctions de base : Avec la même N éléments, dans la méthode des éléments finis classique pour degré p , on doit calculer $N \times p + 1$ fonctions de base, mais dans la méthode des éléments finis par B-splines, on doit seulement calculer $N + p$ fonctions de base.
- L'erreur entre deux méthodes :



RÉFÉRENCES

- [1] P.A.RAVIART, J.M.THOMAS, Introduction à l'analyse numérique des équations aux dérivées partielles, MASSON (1983)
- [2] Pierre PANSU, Interpolation et Approximation par des B-splines, February 9, 2004.
- [3] J.Austin Cottrell, Thomas J.R.Hughes, Yuri Bazilevs, Isogeometric Analysis : toward integration of CAD and FEA, WILEY(2009)