Isogeometric Finite Element

Benoit KRIKKE
Victor PACOTTE
Riad SANCHEZ
Hassane MBYAS SAROUKOU
Xuan ZHANG

University of NICE SOPHIA ANTIPOLIS
Departement of Applied Mathematics and Modeling

An Introduction to Isogeometric Analysis

Supervisors

P.DREYFUSS F.RAPETTI

February 3, 2012

Contents

Introduction

The finite element method (FEM) is a numerical technic for finding approximate solutions of partial differential equations (PDE) as well as integral equations.

The principle of the isogeometric finite element method is to use functions from CAD(Computer-aided design) like B-Splines to determine the field where the PDE takes place and to numerically solve it.

The B-splines allows to make a unstructured grid which represents the field of the solution of the PDE. We can use Bézier curves but the shape of a Bézier curve changes globally when a control point is modified. To overcome this problem we need a curve whose shape only changes locally when a control point is modified. One solution is to connect a number of Bézier curves together and force them to act as a signe one. Hopefully, any change made to a control point would only affect some neighboring curve segments. Since this composite curve is denned on a domain. The domain is also divided into sub-intervals, each of which becomes the domain of Bézier curve segment. The compistie curve is a B-spline curve, and the division points in the domain are it knots. A different subdivion yields a different B-spline curve.

1 B-Spline

1.1 Generality

B-spline is a spline function that has minimal support with respect to a given degree, smoothness, and domain partition. B-splines were investigated as early as the nineteenth century by Nikolai Lobachevsky. A fundamental theorem states that every spline function of a given degree, smoothness, and domain partition, can be uniquely represented as a linear combination of B-splines of that same degree and smoothness, and over that same partition.

1.2 B-spline functions

B-spline functions are piecewise polynomial functions with compact support. They are defined in parametric space using a so-called vector of knots $(\xi_1, ..., \xi_k)$ with $\xi_1 \leq \xi_2 \leq ... \leq \xi_k$. The number of knots verifies $k = n + p + 1$, where n is the number of control points and p is the degree of the spline functions. Each knot represents a coordinate value in the parametric space. If the knot vector is chosen equal to a set of following integers, we refer to natural B-spline. If the knots are distributed uniformly, the vector of knot is said to be uniform. It is said to be open if the first and last knots are repeated $p + 1$ times.

The B-spline functions are defined recursively on the vector of knots using the following procedure:

$$\text{for } \; p = 0 : \; \hat{N}_i^0(\xi) = \begin{cases} 1, & \text{if } \xi_i \leq \xi \leq \xi_i + 1, \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{for } \; p \geq 1 : \; \hat{N}_i^p(\xi) = \frac{\xi - \xi_i}{\xi_i + p - \xi_i} \hat{N}_i^{p-1}(\xi) + \frac{\xi_i + p + 1 - \xi}{\xi_i + p + 1 - \xi_i + p} \hat{N}_{i+1}^{p-1}(\xi)$$

$$i = 1, ..., n + p + 1$$

We can note that each $N_{i,p}(u)$ is computed from two B-spline basis functions of degree $p - 1$, each of which is computed from two B-spline basis functions of degree $p-2$. Hence $N_{i,p}(u)$ is recursively built from basis functions of degree 0.

According to the recursive algorithm, we calculate all of the B-spline functions $\hat{N}_i^p(\xi)$ as follows :

$$\begin{pmatrix} \hat{N}_1^0(\xi) & \hat{N}_1^1(\xi) & \dots & \hat{N}_1^p(\xi) \\ \hat{N}_2^0(\xi) & \hat{N}_2^1(\xi) & \dots & \hat{N}_2^p(\xi) \\ \dots & \dots & \dots & \dots \\ \hat{N}_n^0(\xi) & \hat{N}_n^1(\xi) & \dots & \hat{N}_n^p(\xi) \\ \dots & \dots & \dots & \dots \\ \hat{N}_{n+p-1}^0(\xi) & \hat{N}_{n+p-1}^1(\xi) & ... & 0 \\ \hat{N}_{n+p}^0(\xi) & 0 & \dots & 0 \end{pmatrix}$$

An example of quadradic B-spline function are shown on figure 1 with five distinct knots.

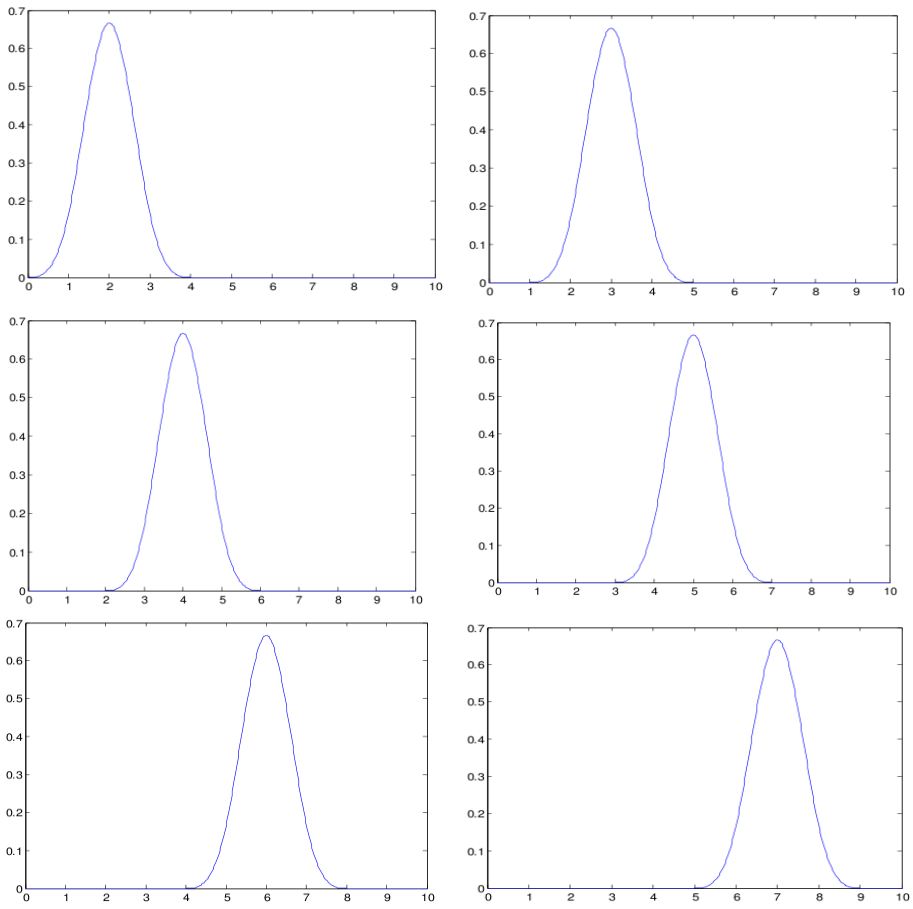

Figure 1: One-dimensional quadratic B-spline functions

1.3 B-spline curve

The B-spline curve of degree p defined by n control points P_1,P_2, ... , P_n is as follows:

$$P(\xi) = (x(\xi), y(\xi), z(\xi)) = \sum_{i=1}^{n} \hat{N}_i^p(\xi) P_i$$

where $P_i = (X_i, Y_i, Z_i)$ are the coordinates of ith control points. P_i can also be interpreted as the weight of the ith B-spline function.

Hence, P(ξ) is the weighted sum of the defining control points. If we change the position of control point P_i, the change made to P_i alters the term $N_{i,p}(\xi)P_i$ only. Since $N_{i,p}(\xi)$ is zero outside of $[\xi_i, \xi_{i+p+1})$, the effect of changing $N_{i,p}(\xi)P_i$ does not propagate outside of $[\xi_i, \xi_{i+p+1})$. Therefore, if P_i is modified, the curve segment on $[\xi_i, \xi_{i+p+1})$ changes and the segment on $[0, \xi_i)$ and $[\xi_{i+p+1}, 1]$ do not. This is exactly an important property of B-spline: the modification made to a control point is localized.

We aim to implement the B-spline in Matlab and we test if the B-spline curve fit the curve provided well.

Programe 1-Bsp : Calculate all the B-spline fonctions

```
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);
                tg=(ksi-ki)/(kip-ki);
                td=(kip1-ksi)/(kip1-ki1);
                sp(i,j)=tg*sp(i,j0)+td*sp(i+1,j0);
            end
        end
    end
end
```

Programe 2-iwBsp : Calculate all the points that go through the B-spline curve

```
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);
x=xi*Nip;
y=yi*Nip;
C(1)=sum(x);
C(2)=sum(y);
end
```

Programe 3-Test B-Spline :

```
function SplineTest()
p=3;
n=9;
points=[-6 -3 -1.5 -1 0 1 2 4 6;0.02 0.1 0.2 0.5 0.999 0.5 0.15 0.07 0];
knotVector=linspace(-4,3,13);
pointsCubic=linspace(-3.2,1.5,900);
m=ip(knotVector,points,p,pointsCubic);
hold on
p=plot(m(:,1),m(:,2));
set(p,'Color','yellow','LineWidth',1.5);
p1=plot(points(1,:),points(2,:),'o');
p2=plot(points(1,:),points(2,:),':');
set(p1,'Color','red','LineWidth',1);
set(p2,'Color','red','LineWidth',2);
ezplot('1/(1+x*x)')
hold off
end
function [P]=ip(ksiVector,points,p,vect)
    n=size(points); %nombre de points de controle
    l=length(vect); %longueur du vecteur vect
    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 Approximation by Splines

In this section we adress the matter of approximating a given function using *splines*, which allow for a piecewise interpolation with a global smoothness.

The problem is to find a B-spline function of degree k of position and velocity at the extremities given by N-1 points Q_i. The problem can be split into two phases : a linear and a nonlinear problem.
We fix a knot vector $\mathbf{t}$ and we seek for a control polygon $\mathbf{P}$ such as the corresponding B-spline curve X_k passes through the Q_i at the nodes. The interpolation results in solving a linear system.
The nonlinear problem consists to optimize the choice of the vector of knot. This question is more difficult than the first point. Besides this part is just an optimization of the choice of vector of knot. So we decided to limite ourselves to the interpolation.

In this report, we aim to approximate the Runge's function using B-spline functions. Runge's function is given by the following equation :

$$f(x) = \frac{1}{1+x^2} \text{ where } x \in \mathbb{R}$$

This function is plotted in figure 2. The Runge's function is a famous example in the field of numerical analysis. Runge's phenomenon is a problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of high degree. It was discovered by Carl Runge. The discovery was important because it shows that going to higher degrees does not always improve accuracy. It's the reason why we have chosen Runge's function to interpolate.

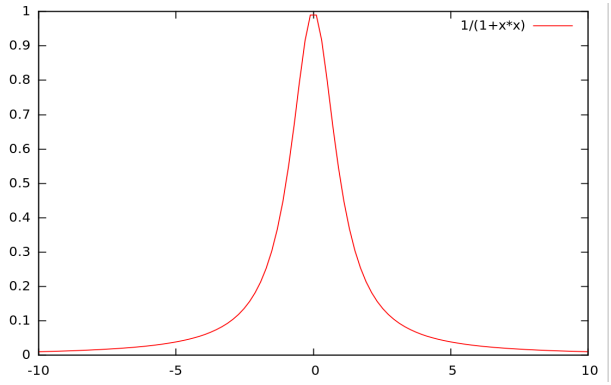

Figure 2: The Runge's function to interpolate

2.1 The linear problem

This interpolation is based on the following theorem for B-splines of degree 3.

Theorem. *Let N+1 nodes Q_j of $\mathbb{R}^n$. Let v_a, v_b two vectors of $\mathbb{R}^n$. Let $\boldsymbol{t}$ a vector of knot clamped at the extremities, such as*
$t_0 = t_1 = t_2 = t_3 = a < t_4 < \ldots < t_{N+2} < b = t_{N+3} = t_{N+4} = t_{N+5} = t_{N+6}$
There exists a unique control polygon $\boldsymbol{P} = (P_0, \ldots, P_{N+2})$ such as the B-spline curve of degree 3 satisfies
$$\forall j = 0, \ldots, N,\ X_3(t_{j+3}) = Q_j,\ X_3'(a) = v_a,\ X_3'(b) = v_b$$

So interpolatory cubic spline are particulary significant since : *i.* they are the splines of minimum degree that yield C^2 approximations; *ii.* they are sufficiently smooth in the presence of small curvatures.

The estimate of the interpolation error is given by the following theorem.

Theorem. *Let $f : [a, b] \to \mathbb{R}$ a C^2 function. Let X_3 the B-spline function of degree 3 witch satisfies the previous theorem. Then*

$$||f - X_3||_\infty \leq \tfrac{h^{3/2}}{2}||f''||_2 \text{ and } ||f' - X_3'||_\infty \leq h^{1/2}||f''||_2$$

where $h = max|t_{i+1} - t_i|$

Now the question is : being given the points Q_i to interpolate, what is the best choice of t_i? To avoid the derivative of the interpolating curve is large, the distant points Q_i and Q_{i+1} must be interpolated with distant values t_i and t_{i+1}. In other words we must correlate spaced t_{i+1} - t_i with the distances $||Q_{i+1} - Q_i||$. The answer is not abvious. There are several ways to choose the t_i. In deed we can choose them uniformly on the interval. Or we can choose a clamped vector that's to say we multiply the points which are at the boundaries.

2.2 Numerical resolution

We have chosen to use a clamped vector to resolve the interpolation problem. Let a knot vector
$t_0 = t_1 = t_2 = t_3 = 0 < t_4 < \ldots < t_{N+2} < N = t_{N+3} = t_{N+4} = t_{N+5} = t_{N+6}$
in the interval $[0, N]$. We seek for the control polygon (with N+3 knots) of the B-spline which passes through the point Q_i at t_{i+3} and which the derivatives are v_0 (resp. v_N) at the extremities.

We have to resolve the following system $\mathrm{A}\mathbf{P} = \mathbf{Q}$ with $\mathbf{Q} = (Q_0, v_0, Q_1, \ldots, Q_{N-1}, v_N, Q_N)$ and $\mathbf{P} \in \mathbb{R}^{N+3}$ and

$$\mathrm{A} = \begin{pmatrix} N_1^p(a) & 0 & \dots & \dots & \dots & 0 \\ N_1^{'p}(a) & N_2^{'p}(a) & 0 & \dots & \dots & 0 \\ 0 & N_2^p(a+h) & N_3^p(a+h) & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & N_N^{'p}(b) \\ 0 & 0 & \dots & \dots & \dots & N_N^p(b) \end{pmatrix}$$

where N_i^p is the spline function of degree p and $N_i^{'p}$ it's derivative. The derivate is given by the following expression

$$(B_i^p)'(x) = p(\frac{B_i^{p-1}(x)}{x_{i+p}-x_i} - \frac{B_{i+1}^{p-1}(x)}{x_{i+p+1}-x_{i+1}})$$

After calculating all the elements, we have the following matrix

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

The matrix A is not symmetric but it's tridiagonal. Even if the resolution of this problem is not difficult we choose to resolve the equivalent system

$${}^T A \mathrm{A} \mathbf{P} = {}^T A \mathbf{Q}$$

The matrix ${}^T AA$ is *coersive*, so the resoltion should be effective. We follow the method given by Pierre Pansu.

2.3 Results

Let's begin by showing the Runge's phenomenon. Let the Runge's function below to interpolate with polynomials

$$f(x) = \frac{1}{1+25x^2},\ -1 \leq x \leq 1$$

Figure 3 plot Runge's function (red) and the interpolated polynomiales with equidistantly spaced data points (green) and with more points at the end (blue). One can see that the polynomial with equidistantly points doesn't approximate the function in the neighborhood of the end points of the interpolation interval. The other interpolated polynomial approximate the function at the end of the interval but elsewere it's a bad approximation.

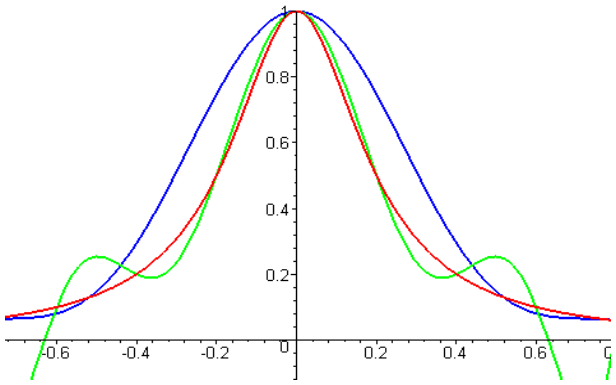

Figure 3: The Runge's phenomenon

Now we plot the interpolation of Runge's function using cubic splines. This function is interpolated between -5 and 5.
In figure 4 we interpolate using 6 points and 15 points in figure 5. We use the method developed in section 1.2.
The Runge's function is drawn in blue and the approximate is drawn in green.

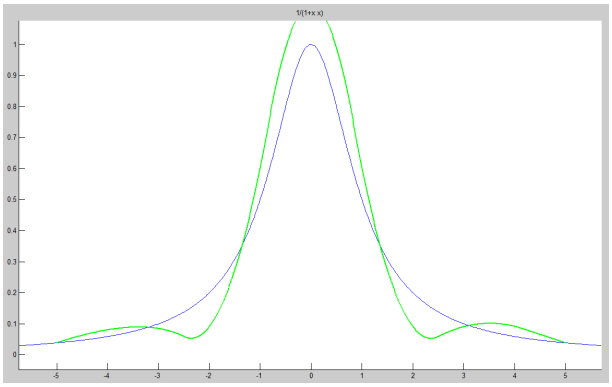

Figure 4: The Runge's function interpolate with 6 points

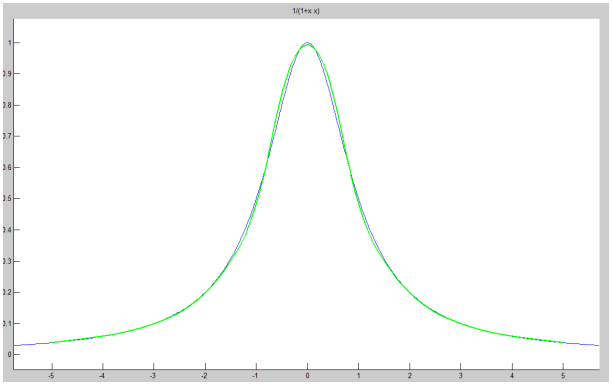

Figure 5: The Runge's function interpolate with 15 points

Once can see in the second case, that's to say interpolation with 15 points, that the interpoltion is good. In fact we have no problem at the end of the intepolation interval as was the case with polynomial interpolation. The Runge's phenomenon does not appear.

3 Grid using B-Spline

In the CAD process, B-spline functions are used to make a representation of the object, this is why we use B-spline functions to describe the computational domain.
First, we need to describe the most difficult side of the boundaries. To do this, we used the one dimension B-spline description.

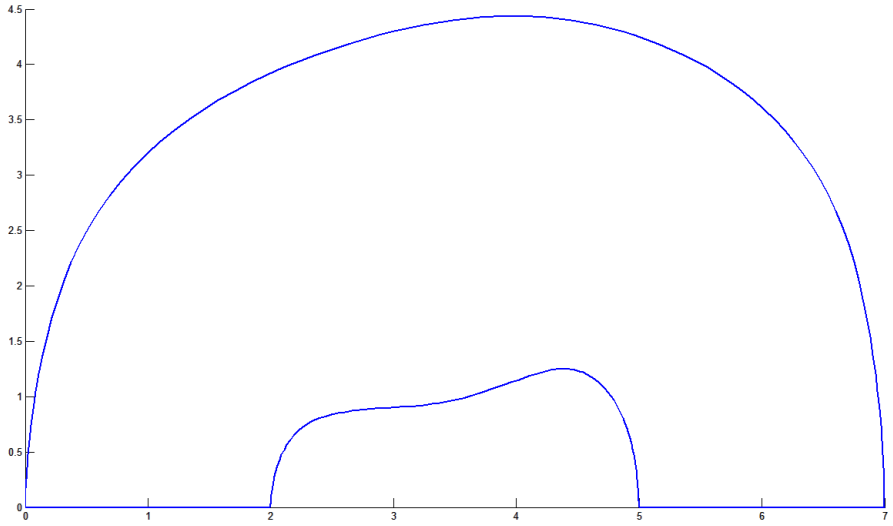

Figure 6: Computational domain around the object

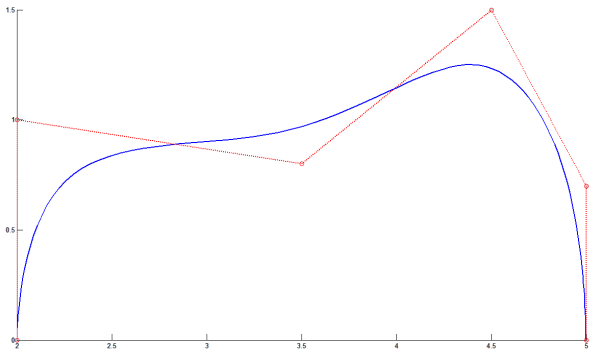

Figure 7: Computational domain around the object

Then we add the other boundaries with the same method.

Once this step is completed, we can insert new control points inside the computational domain without changing the boundaries. For instance, we insert eight points in the domain to obtain an 8x3 matrix which contain the control points (three lines of eight points). The figure 8 represents the three iso-ξ lines.

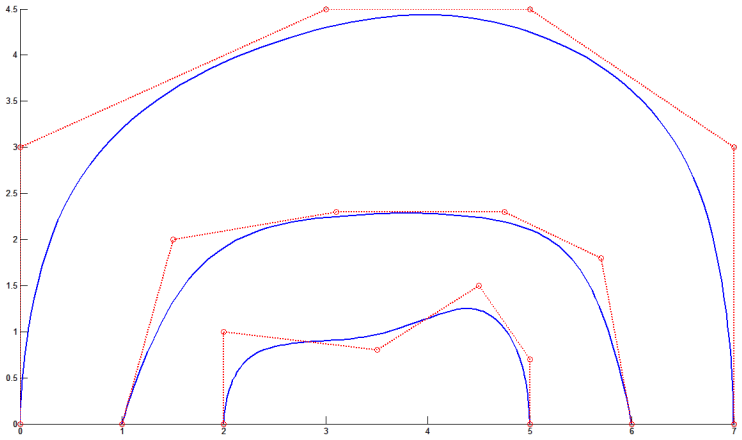

Figure 8: iso-ξ lines

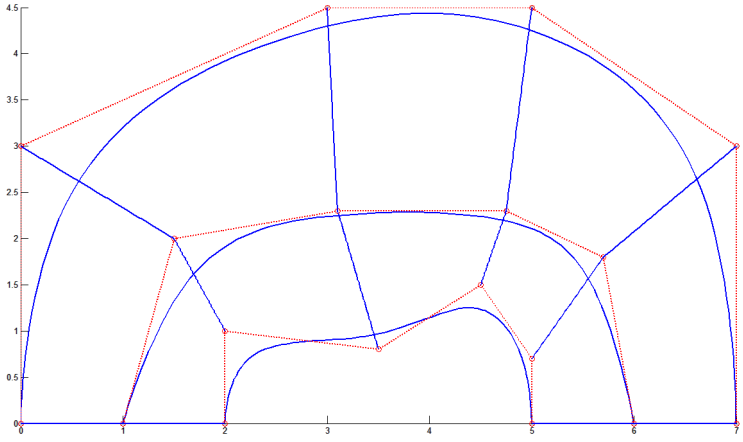

Figure 9: computational domain(iso-ξ and iso-η lines)

The boundaries in y=0 are quite simple therefore we decided to use one degree spline functions for the iso-η. The problem with the figure 9 is the iso-η lines don't start and finish on the computational domain boundaries. So we

decided to use Lagrange piecewise interpolation:

$$L(\eta) = \frac{(\eta-2)(\eta-3)}{(1-2)(3-2)}X_1 + \frac{(\eta-3)(\eta-1)}{(2-3)(2-1)}X_2 + \frac{(\eta-1)(\eta-2)}{(3-1)(3-2)}X_3$$

Where the y_j are the points of the iso-ξ lines where we wanted the iso-η go throught, η is the control points vector of the iso-η line.

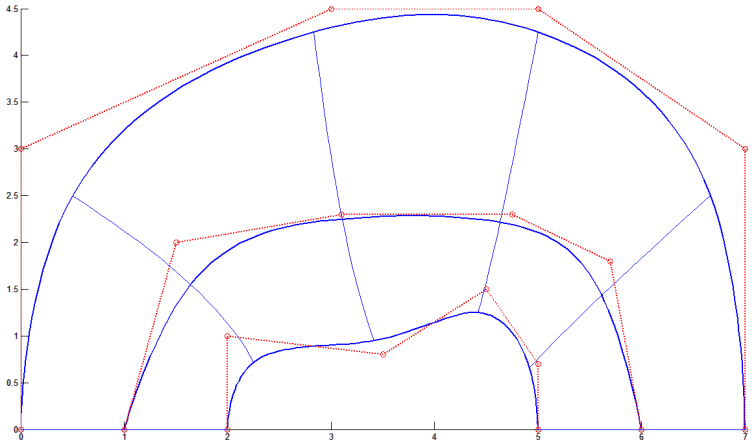

Figure 10: iso-η lines with interpolation)

4 Finite element methode applied to an one-dimensional problem

The finite element method is used to solve differential or partial diferential equations. We adress in this section the resolution of a 2-order differential equation with this method which consists to approach the exact solution of the problem on a mesh of the integration domain with Lagrange finite elements using cubic B-spline functions.

4.1 Problem description

We consider the following differential equation

$$-\alpha \frac{\partial^2 u(x)}{\partial^2 x} + \beta u(x) = f(x),\ x \in \Omega =]0,1[\tag{1}$$

where α, β are given reals, f a real-value continous function. Conditions are imposed at the boundaries, with

$$u(0) = u(1) = 0. \tag{2}$$

4.2 Variational formulation

Let $v \in V = \{\ w \in H^1]0,1[\ |\ v(0) = v(1) = 0\ \}$. We multiply (1) by an arbitrary function v

$$-\alpha \frac{\partial^2 u(x)}{\partial^2 x} v(x) + \beta u(x)v(x) = f(x)v(x)$$

Then we integrate by parts, to obtain

$$[-\alpha u'(x)v(x)]_0^1 \text{ - } \int_0^1 -\alpha u'(x)v'(x)dx + \int_0^1 \beta u(x)v(x)dx = \int_0^1 f(x)v(x)dx$$

which is equivalent to

$$\int_0^1 -\alpha u'(x)v'(x)dx + \int_0^1 \beta u(x)v(x)dx = \int_0^1 f(x)v(x)dx \tag{3}$$

Equation (3) is called variational formulation of the one-dimensional basic problem.

4.3 Uniqueness of the solution and Lax-Milgram's theorem

Let's remind the variational formulation

$$\int_0^1 -\alpha u'(x)v'(x)dx + \int_0^1 \beta u(x)v(x)dx = \int_0^1 f(x)v(x)dx \tag{4}$$

We seek for $u \in V$ such as (3) is satisfied for all $v \in V$. Define

$$a(u,v) = \int_0^1 \alpha u'(x)v'(x)dx + \int_0^1 \beta u(x)v(x)dx \tag{5}$$

and

$$\phi = \int_0^1 f(x)v(x)dx \tag{6}$$

Problem (4) is then remplaced by

$$a(u,v) = \phi(v), \forall v \in V \tag{7}$$

Subject to right conditions, equation (4) has an unique solution.

Theorem. *Let V be a Hilbert space and assume a a continous bilinear form on V. Let ϕ a continous linear form on V. Then there is a unique function $u \in V$ such as* $a(u,v) = \phi(v), \forall v \in V$
Moreover, the linear application is continuous. For example, choose $\alpha = -1$ and $\beta = 0$ guaranteed the hypothesis of Lax-Milgram's theorem and therefore a unique solution of the equation (7).

4.4 Approached resolution and spacial discretization

We are going to replace the space V which has in general a infinite dimension by one of its subspaces and the approached problem are going to be solved : Find $u_h \in V_h$ $a(u_h, v_h) = \phi(vh)$ with $dim(V_h) < \infty$, V_h being a Hilbert space. The space V is built in practice from a mesh of the domain Ω, the index h designating the typical size of the grid cells. In the case of the differential equation (1), we have:

$$-\alpha u''(x) = f(x), u \in]0,1[\text{ where } u(0) = u(1)$$
$$\text{V} = \{\ \omega \in H^1(]0,1[)\ /\ v(0) = v(1) = 0\ \}$$

V_h is a set of continuous functions of ω and polynomial on every mesh . Let $(\phi_1, \ldots, \phi_n)$ be a basis of V_h. The decomposition of the approached solution u_h have the form:

$$u_h = \sum_{i=1}^{N_h} u_i \phi_i$$

The problem amounts to find reals, $u_1, u_2, \ldots, u_{n_h}$ such as

$$\sum_{i=1}^{N_h} u_i a(\phi_i, v_h) = l(v_h)\ \forall v_h \in V$$

By linearity of applications a and ϕ

$$\sum_{i=1}^{N_h} u_i a(\phi_i, \phi_j) = l(\phi_j)\ \forall j \in 1, .., N_h$$

Because any function v_h can be decomposed in the basis of V_h. Finally, the problem is equivalent to solve the linear system:

$$AU = L$$

Where A (stiffness matrix) is a square matrix of size n_h with coefficients $A_{ji} = a(\phi_i, \phi_j)$, U and L (load vector) are column vectors with respective coefficients $u_1, u_2, \ldots, u_{nh}$ and $l(\phi_1), \ldots, l(\phi_{nh})$.

A is a full matrix , a judicious choice of functions ϕ called global basis functions provide a sparse matrix. The functions have compact support very small and the term $a(\phi_i, \phi_j)$ will be often null when the functions ϕ_i and ϕ_j will have disjoint support.

4.5 Choice of global basis functions, mesh and finite-element

B-spline functions of degree p will be used as global basis functions. So: $\forall \phi_i \in [1, N_h]$, $\phi_i = N_i^p$.

There are the functions defined in the first part of this document. The mesh of the domain Ω consists to partition it in small intervals which are chosen with the same length for simplicity.

In fact, the meshes of the domain are not only intervals, but Lagrange finite elements which are chosen affine equivalent to a same finite element called reference element.
We note that all calculations or operations on the global basis functions can be reduced to calculations on local basis functions and then to calculations.

4.6 Resolution of the problem

The finite-element method helps to solve many problems with very complex geometries and constraints. The differential equation (1) to solve is very simple. The mesh and the finite-elements chosen are also simple. We fix $\alpha = 1$ and $\beta = 0$.

The basis functions are B-splines functions of degree p, cubic ($p = 3$) unless orther to compare the exact and approached solutions.

A uniform mesh of nbef finite-elements, so (nbef+1) knots of mesh. The dimension of the subspace V_h is (nbef+1) for a consistency method.

Finally, in this very particular case, we have to determine the matrix A with the coefficients

$$A_{ji} = a(\phi_i, \phi_j) = \int_0^1 N_i^{'p}(x) N_j^{'p}(x) dx$$

and the l vector with

$$l_j = l(\phi_i) = \int_0^1 f(x) N_j^p(x) dx$$

The boundary conditions being imposed null, the matrix A and L must be modified (their first and last lines). The system and all the calculations required will be computed for numeric results.

4.7 Results

In this section we plot the solution of the differential equation (1). We consider foremost the case with $\alpha = -1$, $\beta = 0$, $nbef = 12$, $p = 3$. The exact solution is $u(x) = -\frac{1}{12\alpha}x^4 + \frac{1}{12\alpha}x$. In figure 11 we drawn this function and the approximate solution calculated by the finite element method.

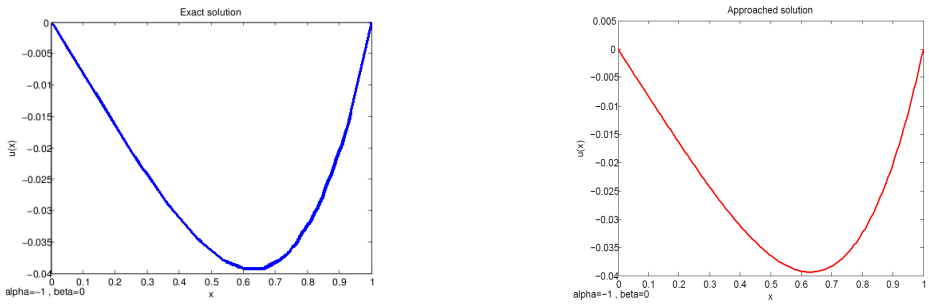

Figure 11: The exact solution (left) and the approximate solution (right)

Once can see also the error between the exact and the approached solution (figure 12).
And we check that the first derivative of the approached is a continuous function.

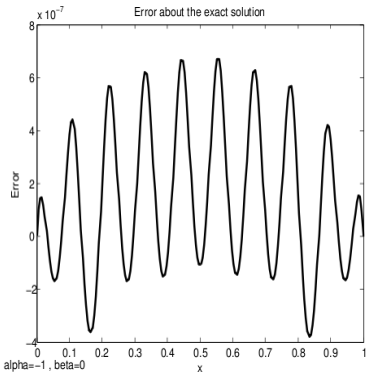

Figure 12: Error

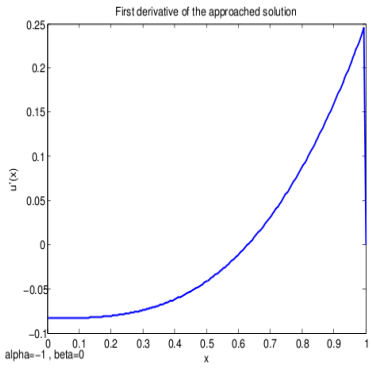

Figure 13: The first derivative of the solution

Let's go on with the second case : $\alpha = -1$, $\beta = 4$, $nbef = 15$, $p = 3$. We follow the same step that the first case.

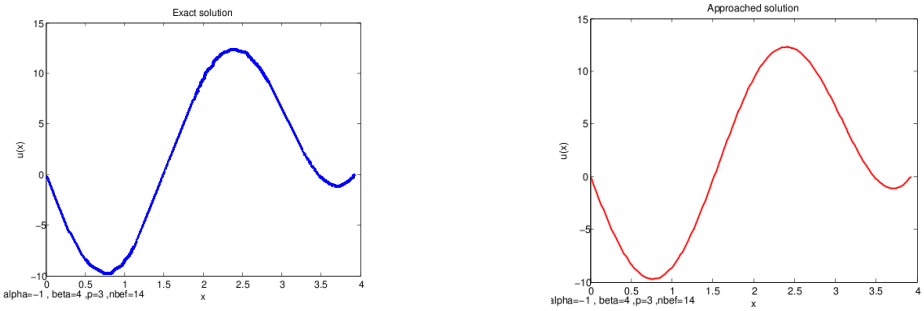

Figure 14: The exact solution (left) and the approximate solution (right)

Now we plot the error (figure 15). We can say that increase the number of finite element reduces the error.

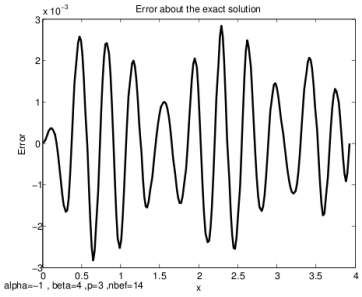

Figure 15: Error between the exact and approached solution

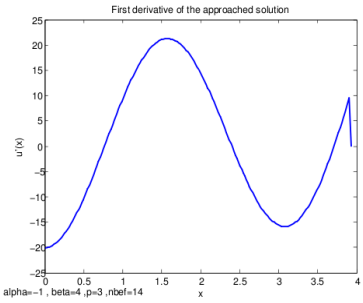

Figure 16: The first derivative of the solution

We finish with the case below $\alpha = -1$, $\beta = 4$, $p = 1$.

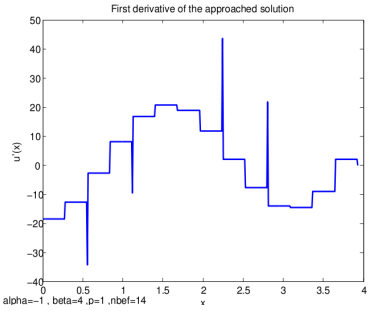

Figure 17: First derivate of the solution

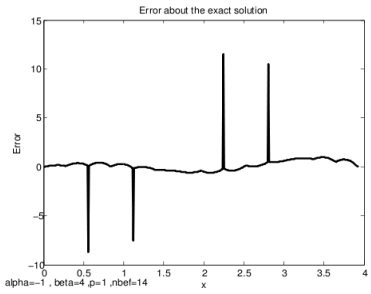

Figure 18: Error between the exact and approached solution

In this case the solution of the problem isn't right and the derivative isn't continuous.

Conclusion

In definitive this project which focused on the B-spline functions as the finite element method and their use in practice like CAD in particular was very interesting.

References

[1] Regis Duvigneau, *An Introduction to Isogeometric Analysis with Application to Thermal Conduction* June 2009.

[2] Alfio Quarteroni, Riccardo Sacco and Fausto Saleri, *Numerical Mathetic* 2007.

[3] Pierre Pansu, *Interpoltion et Approximpation par des B-splines* Fabruary 2004,

$http://www.math.u-psud.fr/\ pansu/web_maitrise/interpolation.pdf$

[4] John Lowther and Ching-Kuang Shene, *Teaching B-splines Is Not Difficult!* Michigan Technological University, 2003 .