

An Introduction to Isogeometric Analysis with Application to Thermal Conduction

Régis Duvigneau

▶ **To cite this version:**

Régis Duvigneau. An Introduction to Isogeometric Analysis with Application to Thermal Conduction. [Research Report] RR-6957, INRIA. 2009, pp.28. inria-00394151

HAL Id: inria-00394151
https://inria.hal.science/inria-00394151

Submitted on 10 Jun 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Introduction to Isogeometric Analysis with Application to Thermal Conduction

R. Duvigneau

N° 6957

Juin 2009

Thème NUM



ISSN 0249-6399 ISRN INRIA/RR--6957--FR+ENG

INRIA
SOPHIA ANTIPOLIS

An Introduction to Isogeometric Analysis with Application to Thermal Conduction

R. Duvigneau*

Thème NUM — Systèmes numériques
Projet Opale

Rapport de recherche n° 6957 — Juin 2009 — 28 pages

Abstract: In this report, we present a new method, namely isogeometric analysis, proposed by T. Hughes and his collaborators, whose objective is to combine Computer Aided Design (CAD) tools and Finite-Element (FE) solvers into a single software entity. This consists in replacing grids by parametric surfaces and volumes to define the integration elements used in a variational formulation. This approach allows to define a computational domain that matches exactly the geometry of the problem, whatever the number of degrees of freedom. High-order, h- and p-adaptive schemes, as well as hierarchical solving strategies can be constructed. After presenting the method and comparing with classical finite-element approach, some numerical experiments are carried out using a thermal conduction test-case.

Key-words: Isogeometric analysis, finite-elements, thermal conduction

* Opale Project-Team

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Une introduction à l'analyse isogéométrique avec application à la conduction thermique

Résumé : On présente dans ce rapport une nouvelle méthode, appelée analyse isogéométrique, proposée par T. Hughes et ses collaborateurs pour fusionner les outils de Conception Assistée par Ordinateur (CAO) et les solveurs éléments-finis (EF) en une seule entité logicielle. Cela consiste à remplacer les maillages par des surfaces et volumes paramétrés pour définir les éléments d'intégration utilisés dans une formulation variationnelle. Cette approche permet de définir un domaine de calcul qui correspond exactement à la géométrie du problème, quel que soit le nombre de degrés de liberté. On peut construire des schémas d'ordre élevé h- et p-adaptatifs, ainsi que des stratégies hiérarchiques de résolution. Après une présentation de la méthode et une comparaison avec une méthode éléments-finis classique, quelques expériences numériques sont menées, dans le cadre d'un problème de conduction thermique.

Mots-clés : Analyse isogéométrique, élément-finis, conduction thermique

Contents

1 A thermal conduction problem **6**

2 Variational formulation **6**

3 Classical finite-element method **6**
3.1 The finite-element method . . . 6
3.2 Spatial discretization . . . 7
3.3 Ritz functions . . . 8
3.4 Integration . . . 8
3.5 Boundary conditions . . . 8

4 Isogeometric analysis **12**
4.1 B-spline functions . . . 12
4.2 B-Spline curves . . . 12
4.3 B-Spline surfaces and volumes . . . 14
4.4 Hierarchical representation . . . 14
4.5 Representation of the geometry in isogeometric analysis . . . 14
4.6 Representation of the solution field . . . 15
4.7 Procedure for isogeometric computation . . . 16
4.8 Boundary conditions . . . 17
4.9 Overview of the algorithm . . . 17

5 Application to thermal conduction **18**
5.1 Problem with a simple geometry . . . 18
5.1.1 Problem description . . . 18
5.1.2 Numerical results . . . 18
5.1.3 Accuracy study . . . 18
5.1.4 Computational time study . . . 19
5.2 Problem with a more complex geometry . . . 22
5.2.1 Problem description . . . 22
5.2.2 Numerical results . . . 22
5.2.3 Accuracy study . . . 23

Introduction

Automated design loop including state-of-the-art simulation codes and optimization methods are now used for industrial applications, for instance in automotive or aeronautic industries. These loops are faced with several requirements and constraints, that are scientific and technological, but also related to the human organization of the companies. A major difficulty comes from the fact that these design loops include various tools, such as Computer Aided Design (CAD) tools, simulation tools (e.g. Finite-Element (FE) solvers), optimization tools, each of them being complex and belonging to a specific field. A typical design loop is depicted in Fig. 1.

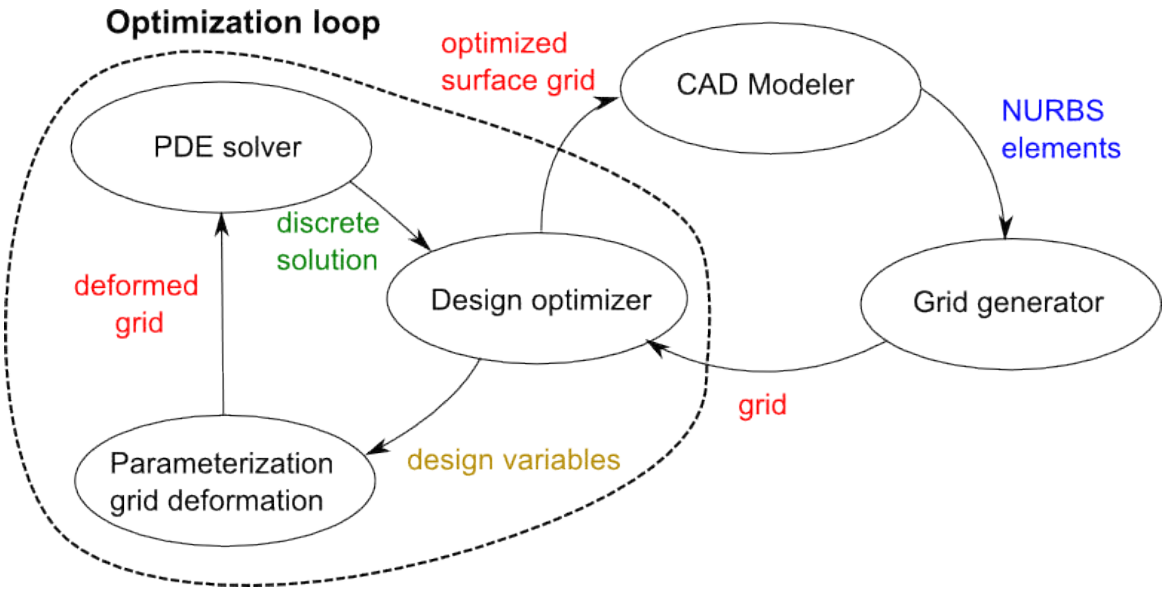


Figure 1: Typical design cycle.

A CAD modeler is first employed to define a first draft of the system to be produced. A grid generation tool is then used to build a computational grid on the basis of the CAD output. Here occurs a first and critical modification of the geometry, since the smooth description from the CAD (for instance NURBS surfaces) is replaced by a piecewise-linear description (for instance surface triangulation). A design optimization procedure is then carried out on the basis of the grid, that is deformed to maximize an objective function, estimated thanks to a simulation tool. At this step, the CAD representation is usually ignored and replaced by some ad-hoc parameterization technique, because the CAD tool is too complex to embed in the design loop. This is the most computationally expensive task, since several calls to the simulation tool are mandatory. Finally, the optimized design, i.e. the optimized grid, is transfered back into the CAD tool, because it is necessary for the manufacturing process to describe the system in an adequate CAD format. However, this projection can yield significant changes and performance degradation. To summarize, this classical design loop has several drawbacks:

- It includes several scientific fields, whose specialists do not comunicate to each other easily ;
- It embeds several sophisticated softwares, which have different operating conditions and input/output formats ;
- It is based on several geometrical and physical representations, yielding several projections or approximations.

The last item is the most relevant from a scientific point of view, but the two first ones should not be neglected in practice. To overcome these difficulties, T. Hughes has proposed recently a new paradigm[4], namely *isogeometric analysis*, for which the design loop entirely relies on a CAD basis. Fig. 2 illustrates such an isogeometric design loop.

The main principle is to *carry out computations on parametric surfaces and volumes instead of grids*. Thus, the grid generation step is replaced by the construction of a CAD domain. Moreover, the simulation code

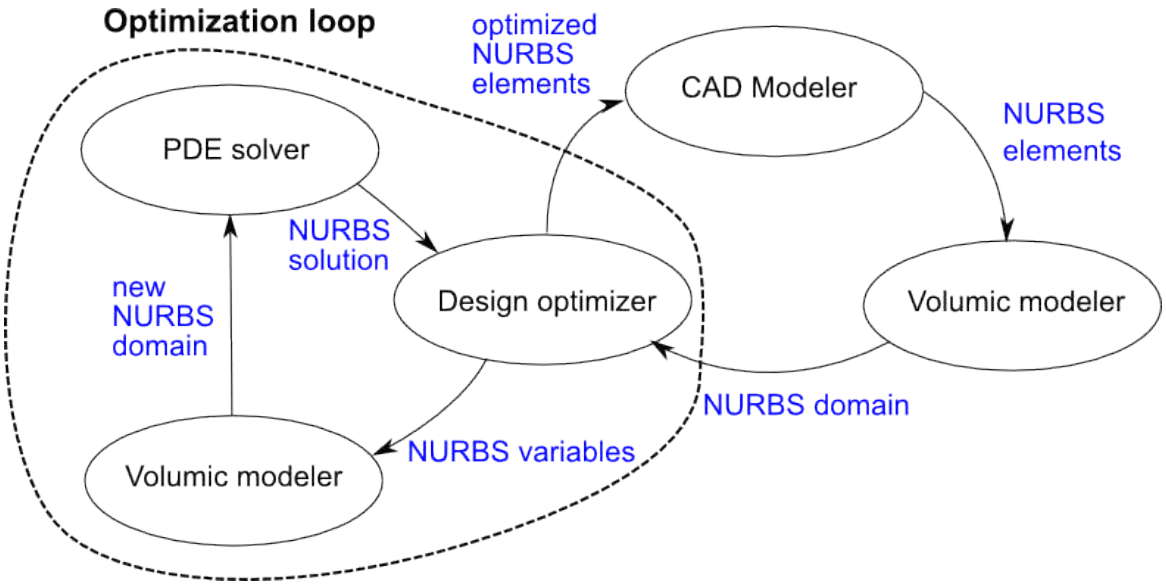


Figure 2: Typical design cycle based on isogeometric concepts.

(presently a FE solver) is modified to perform PDEs integration on NURBS surfaces and volumes, and define solution fields as NURBS objects.

As consequence, the whole design loop is re-organized:

- Only one geometrical and physical representation basis is employed ;
- All numerical ingredients can be fused together into one single software.

The objective of this report is to present this new approach, study its advantages and drawbacks, and illustrate the method for a simple test-case dealing with thermal conduction. In the first section, we present a typical thermal conduction problem and the corresponding partial differential equations that govern the system. Then, a variational formulation is derived and the basics of a classical FE method based on Lagrange functions is reminded, for comparison purpose. In the fourth section, the isogeometric analysis method is presented. Finally, some numerical experiments are carried out.

1 A thermal conduction problem

Given a domain Ω closed by the boundary $\Gamma = \Gamma_D \cup \Gamma_N$, we consider the following thermal conduction problem:

$$\begin{aligned} &\boldsymbol{\nabla}(\kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x})) = 0 \quad \text{in } \Omega \\ &T(\mathbf{x}) = T_0(\mathbf{x}) \text{ on } \partial\Omega_D \\ &\kappa(\mathbf{x})\frac{\partial T}{\partial \mathbf{n}}(\mathbf{x}) = \Phi_0(\mathbf{x}) \text{ on } \partial\Omega_N, \end{aligned} \tag{1}$$

where $\mathbf{x}$ are the Cartesian coordinates, T represents the temperature field and κ the thermal conductivity. Dirichlet and Neumann boundary conditions are applied on $\partial\Omega_D$ and $\partial\Omega_N$ respectively, T_0 and Φ_0 being the imposed temperature and thermal flux ($\mathbf{n}$ unit vector normal to the boundary).

2 Variational formulation

According to a classical variational approach, we seek for a solution $T \in H^1(\Omega)$, such as $T(\mathbf{x}) = T_0(\mathbf{x})$ on $\partial\Omega_D$ and:

$$\int_\Omega \boldsymbol{\nabla}(\kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x}))\ \psi(\mathbf{x})\ d\Omega = 0 \quad \forall \psi \in H^1_{\partial\Omega_D}(\Omega), \tag{2}$$

where $\psi(\mathbf{x})$ are test functions. By integrating by parts, we obtain:

$$-\int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x})\ \boldsymbol{\nabla}\psi(\mathbf{x})\ d\Omega + \int_{\partial\Omega} \kappa(\mathbf{x})\frac{\partial T}{\partial \mathbf{n}}(\mathbf{x})\ \psi(\mathbf{x})\ d\Gamma = 0 \quad \forall \psi \in H^1_{\partial\Omega_D}(\Omega). \tag{3}$$

Using Neumann boundary conditions, we obtain finally the weak formulation:

$$-\int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x})\ \boldsymbol{\nabla}\psi(\mathbf{x})\ d\Omega + \int_{\partial\Omega_N} \Phi_0(\mathbf{x})\ \psi(\mathbf{x})\ d\Gamma = 0 \quad \forall \psi \in H^1_{\partial\Omega_D}(\Omega). \tag{4}$$

3 Classical finite-element method

3.1 The finite-element method

According to the Ritz-Galerkin method, we seek for a solution field T in the form:

$$T(\mathbf{x}) = \sum_{j=1}^{n} T_j \psi_j(\mathbf{x}), \tag{5}$$

where $\psi_j(\mathbf{x}), j = 1, \ldots, N$ are the Ritz functions. T_j are the *degrees of freedom*. By choosing successively the tests functions equal to the Ritz functions, we obtain:

$$\sum_{j=1}^{n} T_j \int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}\psi_j(\mathbf{x})\ \boldsymbol{\nabla}\psi_i(\mathbf{x})\ d\Omega = \int_{\partial\Omega_N} \Phi_0(\mathbf{x})\ \psi_i(\mathbf{x})\ d\Gamma \quad i = 1, \ldots, n. \tag{6}$$

Therefore, the solution can be found by solving the following linear system:

$$M\,T = S, \tag{7}$$

with:

$$\begin{aligned} M_{i,j} &= \int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}\psi_j(\mathbf{x})\ \boldsymbol{\nabla}\psi_i(\mathbf{x})\ d\Omega \\ S_i &= \int_{\partial\Omega_N} \Phi_0(\mathbf{x})\ \psi_i(\mathbf{x})\ d\Gamma. \end{aligned} \tag{8}$$

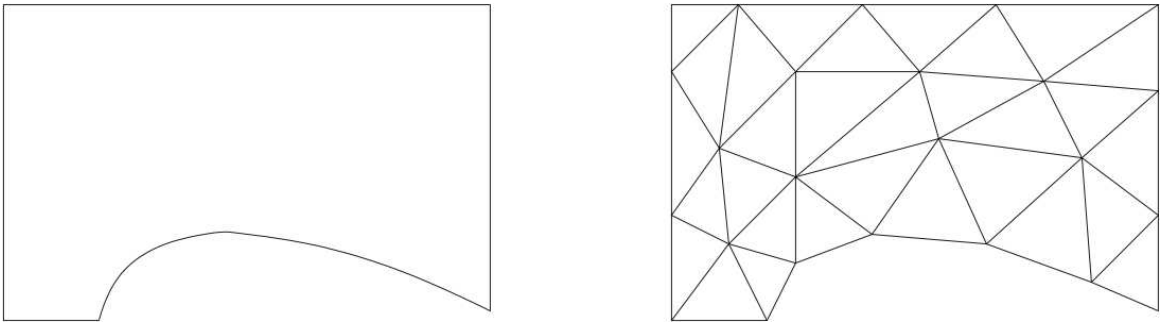


Figure 3: Exemple of physical domain Ω (left) and discretized domain Ω_h (right).

3.2 Spatial discretization

In the classical approach the physical domain Ω, that can be defined for instance by NURBS using a CAD software, is discretized using triangles or quadrangles in 2D, named *elements*, yielding the discretized domain Ω_h. It should be noted that the geometry of the domain is changed during the discretization process since *the curved boundaries are approximated by piecewise linear boundaries* (see Fig. 3). The main consequence is that a refined mesh is required in the vicinity of curved boundaries in order to have a satisfactory description of the geometry, whatever the behavior of the solution field.

The coefficients of the linear system 8 are computed element by element, yielding for the element K:

$$\begin{aligned} M_{i,j}^K &= \int_K \kappa(\mathbf{x}) \boldsymbol{\nabla} \psi_j^K(\mathbf{x}) \; \boldsymbol{\nabla} \psi_i^K(\mathbf{x}) \; d\Omega \\ S_i^K &= \int_{\partial K_N} \Phi_0(\mathbf{x}) \; \psi_i^K(\mathbf{x}) \; d\Gamma. \end{aligned} \tag{9}$$

However, to reduce the computational cost, integrations are not performed directly on each element, but on one reference element, by introducing a spatial transformation. Each node ${\mathbf{x_i}}^K = (x_i^K, y_i^K)$ (in 2D) of an element K corresponds to a node $\boldsymbol{\xi}_i = (\xi_i, \eta_i)$ on the reference element $\hat{K}$. Thus, we define the following transformation for each element K:

$$\begin{array}{rccc} \mathcal{T}^K : & \hat{K} & \rightarrow & K \\ & \boldsymbol{\xi} = (\xi, \eta) & \rightarrow & \mathbf{x} = (x, y) \end{array} \tag{10}$$

Fig. 4 illustrates this (linear) transformation for triangle and quadrangle elements. The Jacobian matrix of the transformation is:

$$D\mathcal{T}^K = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{pmatrix}. \tag{11}$$

Ritz functions $\hat{\psi}$ can be defined on the reference element using:

$$\psi^K(\mathbf{x}) = \psi^K(\mathcal{T}^K(\boldsymbol{\xi})) = \hat{\psi}(\boldsymbol{\xi}). \tag{12}$$

The gradient of the functions can be evaluated thanks to the Jacobian matrix:

$$\boldsymbol{\nabla}_{\boldsymbol{\xi}} \hat{\psi}(\boldsymbol{\xi}) = (D\mathcal{T}^K)^\top \boldsymbol{\nabla} \psi^K(\mathbf{x}). \tag{13}$$

If we introduce this transformation into integrals, we obtain finally for each element K:

$$\begin{aligned} M_{i,j}^K &= \int_{\hat{K}} \kappa(\mathcal{T}^K(\xi)) \boldsymbol{\nabla}_{\boldsymbol{\xi}} \hat{\psi}_j(\boldsymbol{\xi}) \; B^K(\boldsymbol{\xi})^\top \; B^K(\boldsymbol{\xi}) \; \boldsymbol{\nabla}_{\boldsymbol{\xi}} \hat{\psi}_i(\boldsymbol{\xi}) \; J^K(\boldsymbol{\xi}) \; d\hat{\Omega} \\ S_i^K &= \int_{\partial \hat{K}_N} \Phi_0(\mathcal{T}^K(\boldsymbol{\xi})) \; \hat{\psi}_i(\boldsymbol{\xi}) \; J^K(\boldsymbol{\xi}) \; d\hat{\Gamma}, \end{aligned} \tag{14}$$

where J^K is the Jacobian of the transformation and B^K is the transposed of the inverse of the Jacobian matrix. Using such manipulations, the Ritz functions can be evaluated once, on the reference element only.

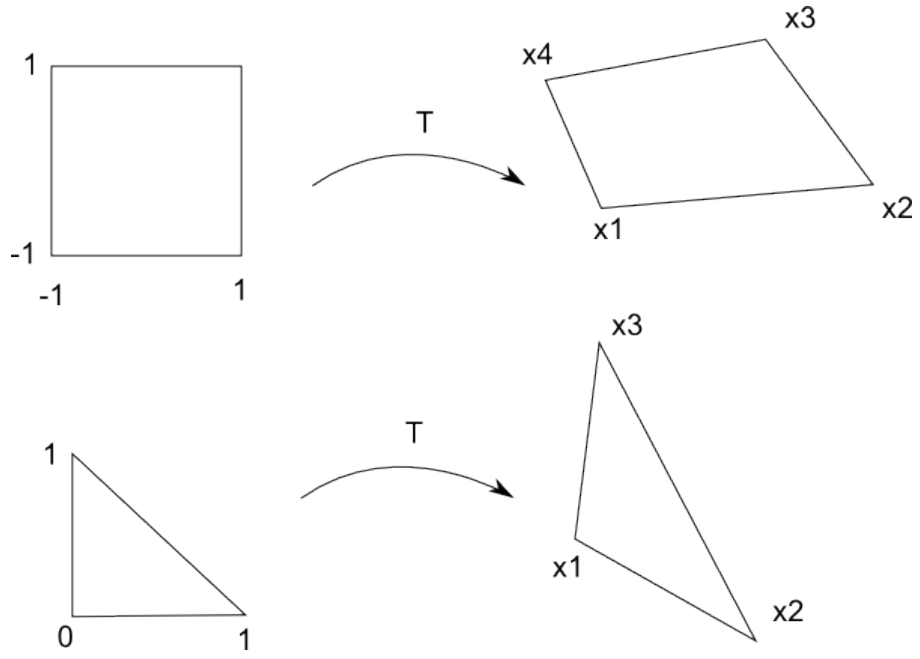


Figure 4: Spatial transformations from reference element $\hat{K}$ to element K.

3.3 Ritz functions

The usual approach to define the Ritz functions consist in imposing that they interpolate the degrees of freedom at each node $\mathbf{x_i}$:

$$T(\mathbf{x_i}) = \sum_{j=1}^{N} T_j \psi_j(\mathbf{x_i}) = T_i. \tag{15}$$

Therefore, the degrees of freedom $T_i, i = 1, \ldots, N$ can be interpreted as the value of the solution field at the nodes $\mathbf{x_i}, i = 1, \ldots, N$. These conditions can be fulfiled by choosing Lagrange interpolating polynomials as Ritz functions. Indeed, Lagrange polynomials are constructed in such a way that:

$$\psi_j(\mathbf{x_i}) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{16}$$

However, the use of low-order polynomials that interpolate the solution in each element only is preferred in practice, in order to handle functions with compact supports yielding sparse matrices. Therefore, most codes use linear or quadratic functions in each element. We illustrate the linear and quadratic Lagrange functions for the one-dimensional case. When linear functions are used, an element is composed of two nodes and two functions are defined in each element, whereas for quadratic functions an element is composed of three nodes and three functions are defined in each element. Fig. 5 represents the functions for one element. If we consider the Ritz functions defined on the whole computational domain, we obtain as many functions with compact support as the number of nodes (see Fig. 6 and 7). These examples can be extended in 2D with triangles and quadrangles without difficulty.

3.4 Integration

Integrals in Eq. 14 can be evaluated analytically for simple functions $\kappa(\mathbf{x})$ and $\Phi_0(\mathbf{x})$. In practice, numerical integration is usually performed using Gauss quadrature methods, that allow to compute the exact integral values for polynomials of a given degree, on the basis of a few number of evaluations.

3.5 Boundary conditions

In the case of Dirichlet boundary conditions, degrees of freedom located at the boundary are directly imposed. Therefore, the corresponding variables T_D can be removed from the vector of unknowns when the linear system

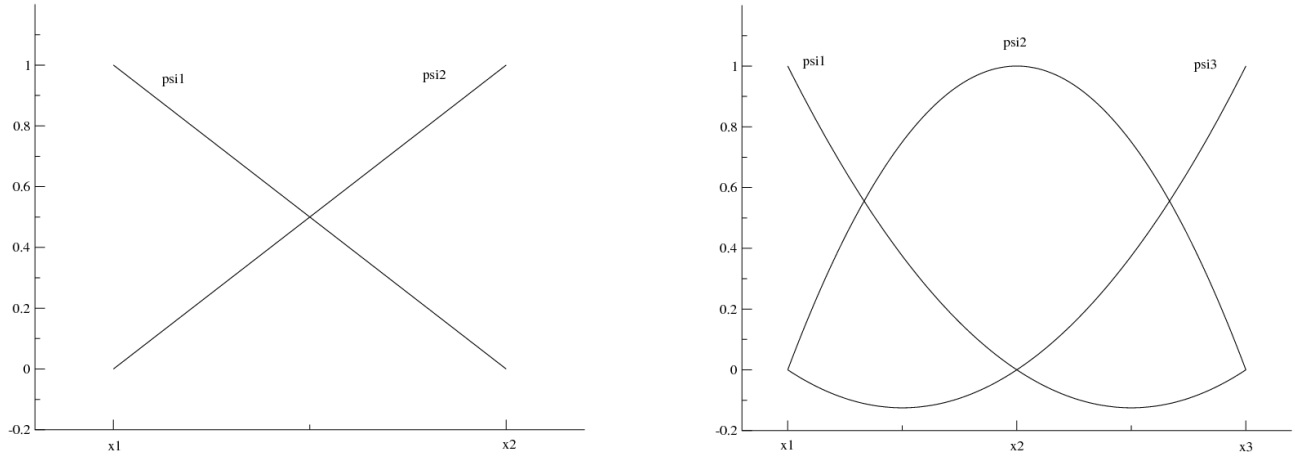


Figure 5: Example of one-dimensional linear (left) and quadratic (right) Ritz functions for one element.

is solved. If one modifies the variable numbering to locate at the end of the vector of unknowns the variables corresponding to the Dirichlet boundary condition, the system can be written as:

$$\begin{pmatrix} M^{11} & M^{12} \\ M^{21} & M^{21} \end{pmatrix} \cdot \begin{pmatrix} T^1 \\ T_D \end{pmatrix} = \begin{pmatrix} S^1 \\ S_D \end{pmatrix}. \tag{17}$$

The system to solve is simply:

$$M^{11} \cdot T^1 = S^1 - M^{12} \cdot T_D. \tag{18}$$

In the case of Neumann boundary conditions, the degrees of freedom located on the boundary are unknown. The thermal flux at the boundary is imposed in the source term in a weak sense.

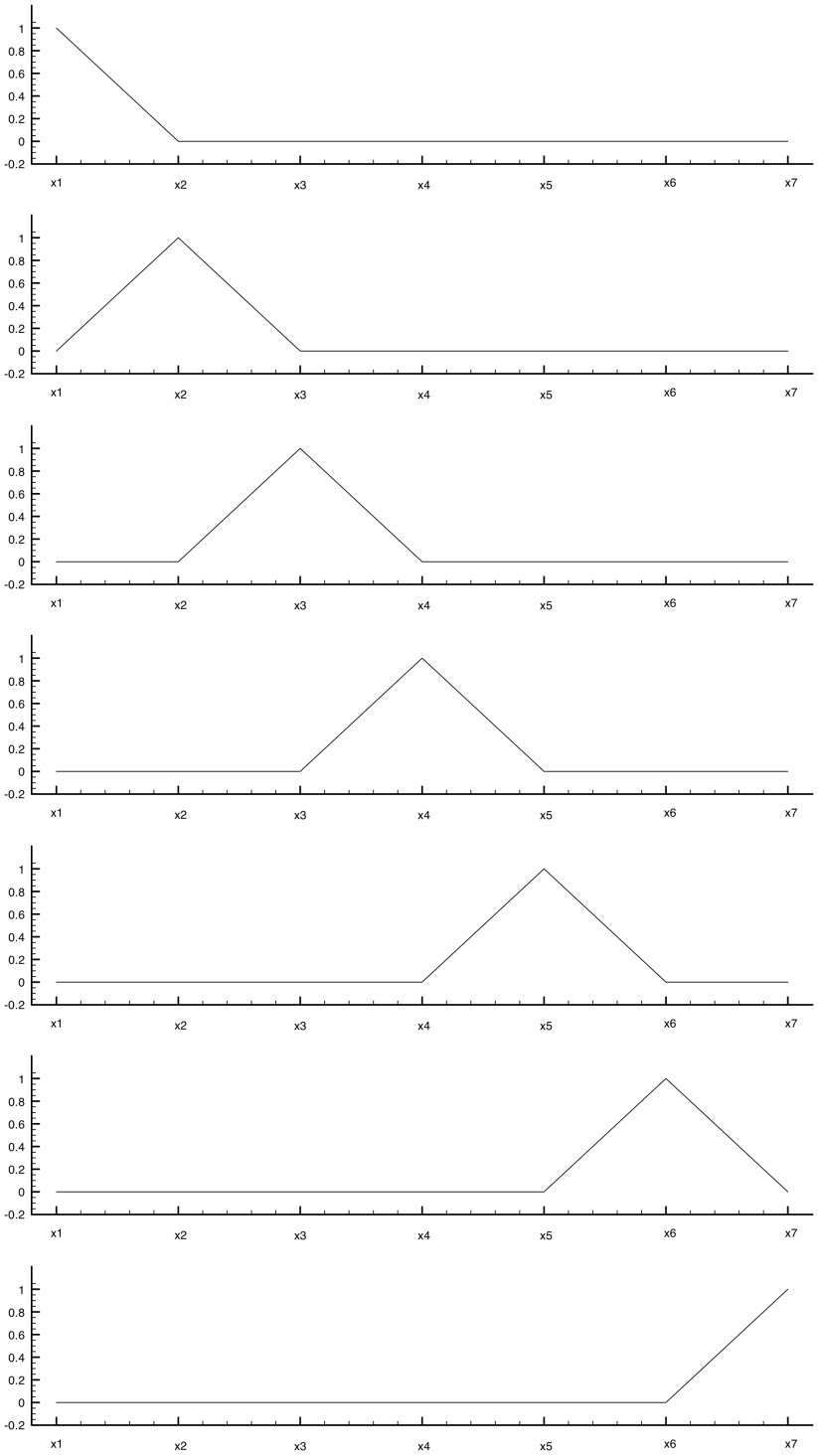


Figure 6: Example of one-dimensional linear Ritz functions.

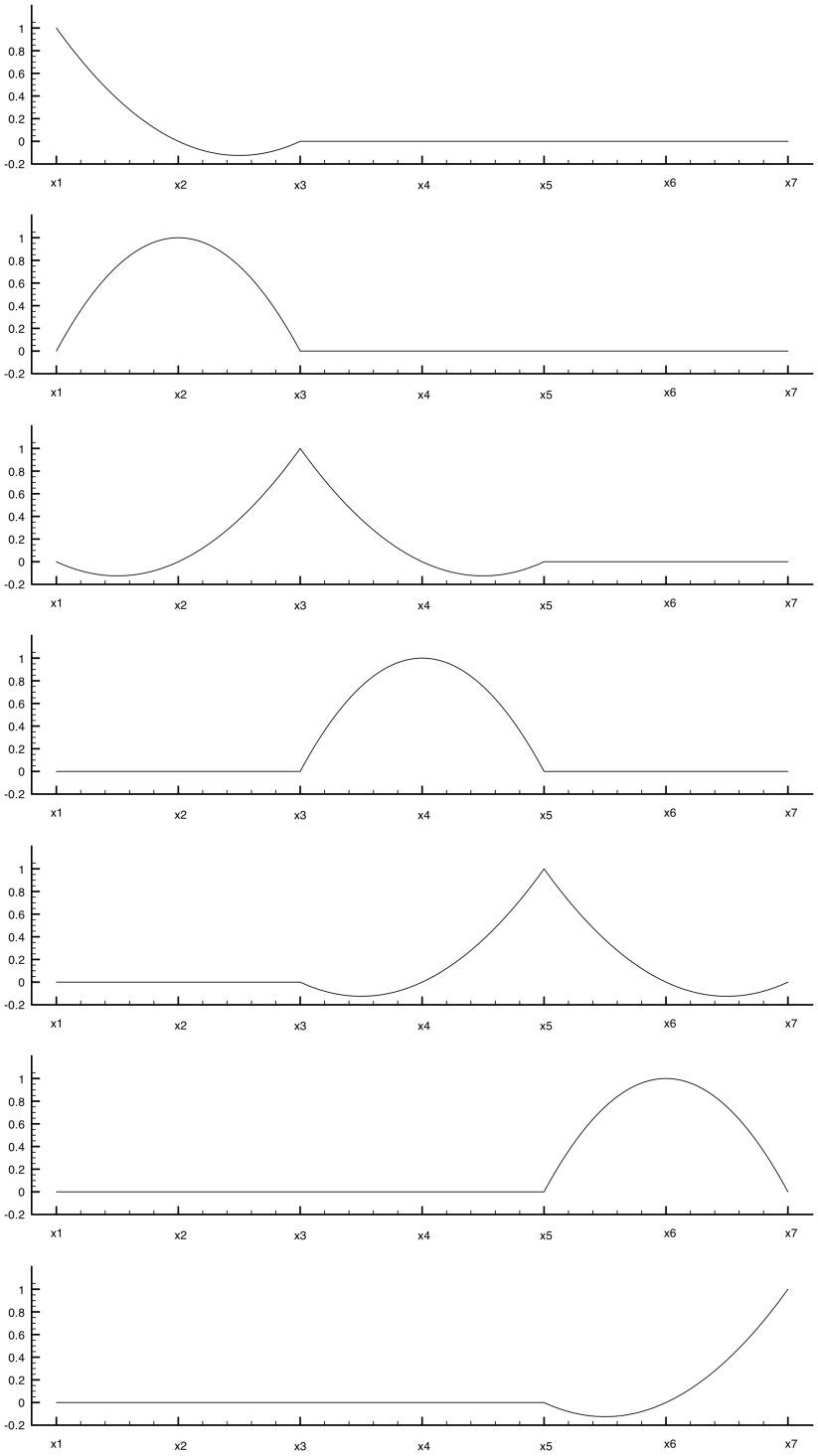


Figure 7: Example of one-dimensional quadratic Ritz functions.

4 Isogeometric analysis

Isogeometric analysis relies on the same basis as classical finite-element methods, except that Lagrange interpolating polynomials used as Ritz functions are replaced by CAD basis functions, such as B-splines or NURBS. For the sake of simplicity, we will describe the method in the next sections using B-spline functions.

4.1 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* $(\xi_1, \ldots, \xi_k)$ with $\xi_1 \leqslant \xi_2 \leqslant \ldots \leqslant \xi_k$. The number of knots verifies $k = n + p + 1$, where n is the number of control points and p 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-splines*. 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 interval $[\xi_1, \ldots, \xi_k]$ is called *a patch*, whereas the interval $[\xi_i, \xi_{i+1})$ is called *a knot span*.

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

$$\begin{aligned} &\text{for } p = 0 : \hat{N}_i^0(\xi) = \begin{cases} 1 & \text{if } \xi_i \leqslant \xi \leqslant \xi_{i+1} \\ 0 & \text{otherwise.} \end{cases} \\ &\text{for } p \geqslant 1 : \hat{N}_i^p(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} \hat{N}_i^{p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_i + p} \hat{N}_{i+1}^{p-1}(\xi) \quad i = 1, \ldots, n + p + 1. \end{aligned} \tag{19}$$

The B-spline functions defined using the procedure described above have the following properties:

- The functions $\hat{N}_i^p(\xi)$ are polynomials of degree p ;
- They have compact supports $[\xi_i, \xi_{i+p+1})$;
- The functions $\hat{N}_i^p(\xi)$ are of regularity C^{p-m} at each knot of multiplicity m. Then, for knots of multiplicity one, the functions are C^{p-1} ;
- On the knot span $[\xi_i, \xi_{i+1})$, there are $p + 1$ non zero functions.

An example of quadratic B-spline functions are shown on Fig. 8, for an open uniform knot vector with seven distinct knots.

4.2 B-Spline curves

The B-spline curve is then defined using:

$$\boldsymbol{P}(\xi) = (x(\xi), y(\xi), z(\xi)) = \sum_{i=1}^{n} \hat{N}_i^p(\xi) \boldsymbol{P}_i, \tag{20}$$

where $\boldsymbol{P}_i = (X_i, Y_i, Z_i)$ are the coordinates of the control points. $\boldsymbol{P}_i$ can also be interpreted as the weight of the ith B-spline function.

The B-spline curves have the following properties:

- If the knot vector is open, the curve starts at the point $\boldsymbol{P}_1$ and ends at $\boldsymbol{P}_n$. Moreover, it is tangent to $(\boldsymbol{P}_0\, \boldsymbol{P}_1)$ and $(\boldsymbol{P}_{n-1}\, \boldsymbol{P}_n)$ at the extremities ;
- If $\xi \in [\xi_i, \xi_{i+1})$, the point $\boldsymbol{P}(\xi)$ belongs to the convex hull defined by the control points $\boldsymbol{P}_{i-p}, \cdots, \boldsymbol{P}_i$;
- Globally, the B-spline curve belongs to the convex hull defined by the set of control points.
- Affine transformations of a B-spline curve are identical to applying the transformations directly to the control points set.

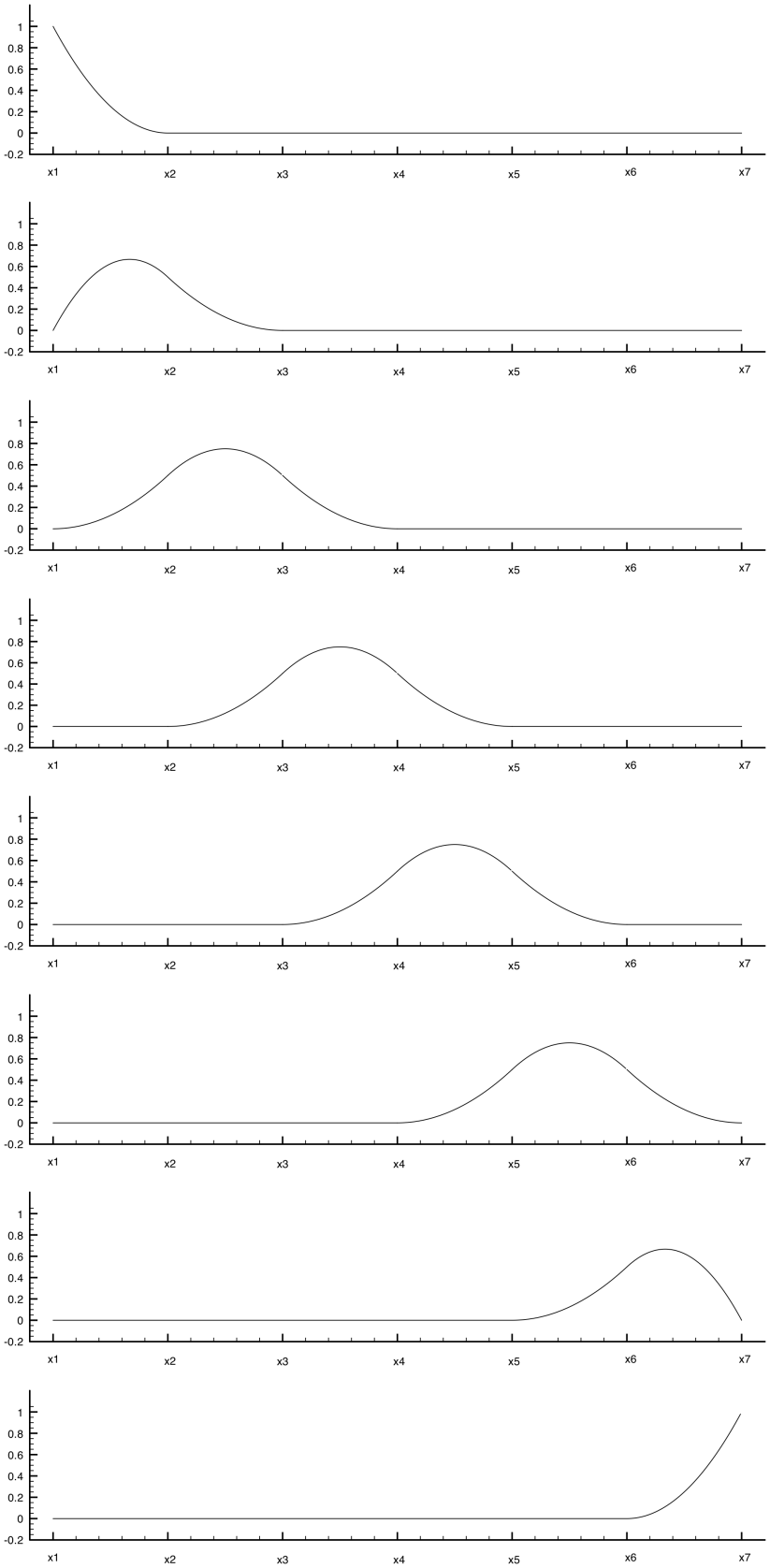


Figure 8: Example of one-dimensional quadratic B-spline functions.

4.3 B-Spline surfaces and volumes

One can easily extend the previous concepts to define B-spline surfaces and volumes, by using tensor products of B-spline functions. More precisely, B-spline surfaces are defined using a second-order tensor product of B-spline functions:

$$\boldsymbol{P}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta), z(\xi,\eta)) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j} \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta)\,\boldsymbol{P}_{ij}. \tag{21}$$

B-spline volumes are defined using a third-order tensor product of B-spline functions:

$$\boldsymbol{P}(\xi,\eta,\zeta) = (x(\xi,\eta,\zeta), y(\xi,\eta,\zeta), z(\xi,\eta,\zeta)) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j}\sum_{k=1}^{n_k} \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta)\,\hat{N}_k^{p_k}(\zeta)\,\boldsymbol{P}_{ijk}. \tag{22}$$

4.4 Hierarchical representation

The B-spline functions allow to have a hierarchical representation of the curves, surfaces or volumes. Basically, it means that a B-Spline object defined using n control points can be *exactly* represented using an other B-Spline object defined using a larger number $n' > n$ of control points. This property allows to define *refinement* procedures to increase the number of control points, if required.

This can be done using two different approaches, that are similar to h- and p-refinement process as in classical finite-element methods. Basically, h-refinement consists in adding new grid nodes, whereas p-refinement consists in using Ritz functions of higher degree. Both strategies result in increasing the number of degrees of freedom. Using B-Splines, these processes correspond to:

- knot insertion : given a B-spline curve of degree p defined for a knot vector $(\xi_1,\ldots,\xi_k)$, it is possible to insert a knot, yielding a new knot vector $(\xi_1,\ldots,\xi_{k+1})$, without modifying the curve. The $n+1$ control points $\tilde{\boldsymbol{P}}_i$ $i = 1,\cdots,n+1$ that define the new curve are obtained from the control points of the original curve $\boldsymbol{P}_i$ $i = 1,\cdots,n$:

$$\begin{aligned} \tilde{\boldsymbol{P}}_i &= \alpha_i \boldsymbol{P}_i + (1-\alpha_i)\boldsymbol{P}_{i-1} \\ \alpha_i &= \begin{cases} 1 & \text{if } i \leqslant k-p \\ \frac{\xi-\xi_i}{\xi_{i+p}-\xi_i} & \text{if } k-p+1 \leqslant i \leqslant k \\ 0 & \text{if } i \geqslant k+1 \end{cases} \end{aligned} \tag{23}$$

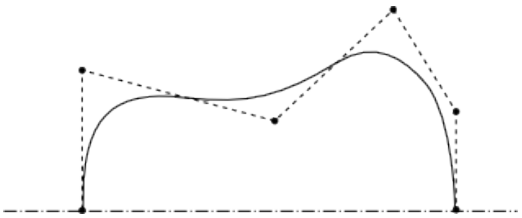
- degree-elevation : given a B-spline curve of degree p defined for a knot vector $(\xi_1,\ldots,\xi_k)$, it is possible to define an other B-Spline curve of degree $p+1$ that is identical to the original one. It is defined using $n+1$ control points for the original knot vector. The process to obtain the new control points coodinates is more complex than for knot insertion. It is composed of three steps:
  - Subdivision of the curve into several Bezier curves of degree p ;
  - Degree-elevation for each Bezier curve ;
  - Recombination into a B-Spline curve of degree $p+1$.

4.5 Representation of the geometry in isogeometric analysis

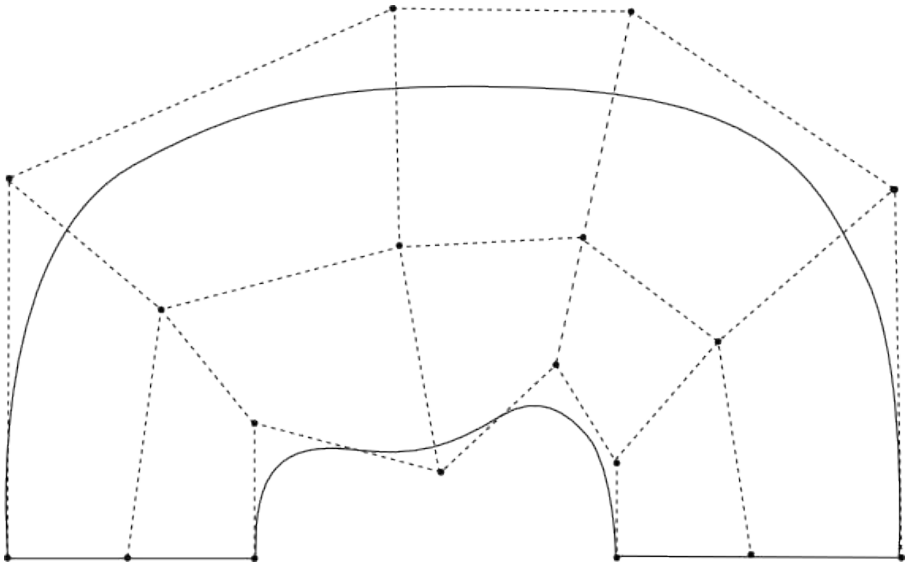
In the isogeometric analysis framework, the computational domain is not approximated, but "exactly" described using the same representation as that employed in the CAD process. For instance, if one uses cubic B-spline functions to define the geometrical domain of interest, isogeometric analysis will use a cubic B-spline representation for the computational domain. Consequently, the geometry of the computational domain is "exact", whatever the representation of the solution field. In particular, the geometry is not approximated by a piecewise linear representation, as in classical finite-element methods.

However, there is a major difference between the CAD description of the geometry and the representation used in isogeometric analysis: usually, in the CAD process, one describes only the boundary of the domain, that defines the skin of the object of interest. For isogeometric analysis, we need to compute solution fields. Therefore, one should also build a CAD representation of the whole computational domain that surrounds the

object. For a 2D problem, it means that we represent the computational domain as a flat B-spline surface. Fig. 9 illustrates these difference. The choice of the control points located at the interior of the computational domain is similar in some sense to the construction of a grid in the classical finite-element approach.



(a) CAD representation of the object alone.



(b) CAD representation of the computational domain around the object.

Figure 9: Example of CAD representations for a 2D object with a symmetry.

4.6 Representation of the solution field

According to the isogeometric paradigm, the solution fields, such as the temperature field for heat conduction problems, is also represented using B-spline basis functions. For a 2D problem, we have:

$$T(\xi,\eta) = \sum_{i=1}^{n_i} \sum_{j=1}^{n_j} \hat{N}_i^{p_i}(\xi)\, \hat{N}_j^{p_j}(\eta)\, T_{ij}. \tag{24}$$

This representation is similar to that used in classical finite-element methods, according to Eq. 5. For linear representations, B-spline functions and functions based on Lagrange interpolating polynomials are identical. However, this is not the case for representations of higher degree. If one compares quadratic B-spline basis functions on Fig. 8 with basis functions based on Lagrange interpolating polynomials shown in Fig. 7, one can notice some critical differences:

- For seven distinct knots, height B-spline functions are defined, whereas seven basis functions based on Lagrange interpolating polynomials can be constructed with seven nodes ;
- Quadratic B-spline functions are C^1, whereas basis functions based on Lagrange interpolating polynomials are only C^0 ;
- B-spline functions remain always positive, whereas Lagrange interpolating polynomials can have negative values ;

- Only first and last B-spline functions at the first and last knot have a unity value, whereas one basis function based on Lagrange interpolating polynomials has a unity value at each node. Consequently, the B-spline curve is not interpolating the control points.

The last property has a critical consequence: the estimation of the solution value at a given point requires the computation of Eq. 24. In particular, it is not possible to know the solution value at a control point $\boldsymbol{P}_{ij}$ or at a given knot (ξ_i, η_j) by considering only the degree of freedom T_{ij}. *The non-interpolatory nature of the basis functions does not lead to the usual interpretation of the degrees of freedom.* Thus an effort, in terms of software development, has to be done to visualize exactly the solution field.

4.7 Procedure for isogeometric computation

Once the CAD representation of the computational domain has been built, one should construct the linear system resulting from the application of isogeometric analysis to the problem studied.

We use the weak formulation of the problem Eq. 4, with the isogeometric representation of the solution field Eq. 24. We would like to use the tensor products of the B-spline basis functions in the parametric space $\hat{N}_{ij}(\xi,\eta) = \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta)$ as tests functions. To define these functions in the physical domain, we introduce a transformation from the parametric space into the physical space:

$$\begin{array}{rccc} \mathcal{T}: & \hat{\mathcal{P}} & \rightarrow & \mathcal{P} \\ & \boldsymbol{\xi} = (\xi,\eta) & \rightarrow & \mathbf{x} = (x,y) \end{array} \tag{25}$$

Note that this transformation is not linear and is simply defined by the parametric representation of the computational domain, according to Eq. 21. Therefore, the Jacobian matrix of this transformation, that will be used further for integration, requires to differentiate this representation. Then, we define the test functions in the physical domain such as:

$$N_{ij}(\mathbf{x}) = N_{ij}(x,y) = N_{ij}(\mathcal{T}(\xi,\eta)) = \hat{N}_{ij}(\xi,\eta) = \hat{N}_i^{p_i}(\xi)\,\hat{N}_j^{p_j}(\eta). \tag{26}$$

Then, the weak formulation Eq. 4 reads:

$$\sum_{k=1}^{n_k}\sum_{l=1}^{n_l} T_{kl}\int_{\mathcal{P}} \kappa(\mathbf{x})\boldsymbol{\nabla}N_{kl}(\mathbf{x})\;\boldsymbol{\nabla}N_{ij}(\mathbf{x})\;d\Omega = \int_{\partial\mathcal{P}_N}\Phi_0(\mathbf{x})\;N_{ij}(\mathbf{x})\;d\Gamma \quad i=1,\ldots,n_i \quad j=1,\ldots,n_j. \tag{27}$$

Finally, we obtain a linear system similar to that resulting from the classical finite-element methods, with a matrix and a right-hand side defined as:

$$\begin{aligned} M_{ij,kl} &= \int_{\mathcal{P}} \kappa(\mathbf{x})\boldsymbol{\nabla}N_{kl}(\mathbf{x})\;\boldsymbol{\nabla}N_{ij}(\mathbf{x})\;d\Omega \\ S_{ij} &= \int_{\partial\mathcal{P}_N}\Phi_0(\mathbf{x})\;N_{ij}(\mathbf{x})\;d\Gamma. \end{aligned} \tag{28}$$

By performing the integration in the parametric space, we obtain:

$$\begin{aligned} M_{ij,kl} &= \int_{\hat{\mathcal{P}}} \kappa(\mathcal{T}(\boldsymbol{\xi}))\boldsymbol{\nabla}_{\boldsymbol{\xi}}\hat{N}_{kl}(\boldsymbol{\xi})\;B(\boldsymbol{\xi})^{\top}\;B(\boldsymbol{\xi})\;\boldsymbol{\nabla}_{\boldsymbol{\xi}}\hat{N}_{ij}(\boldsymbol{\xi})\;J(\boldsymbol{\xi})\;d\hat{\Omega} \\ S_{ij} &= \int_{\partial\hat{\mathcal{P}}_N}\Phi_0(\mathcal{T}(\boldsymbol{\xi}))\;\hat{N}_{ij}(\boldsymbol{\xi})\;J(\boldsymbol{\xi})\;d\hat{\Gamma}, \end{aligned} \tag{29}$$

where J is the Jacobian of the transformation, B^K is the transposed of the inverse of the Jacobian matrix. Tthe gradient of the tests functions is:

$$\boldsymbol{\nabla}_{\boldsymbol{\xi}}\hat{N}_{ij}(\boldsymbol{\xi}) = \begin{pmatrix} \frac{\partial\hat{N}_i^{p_i}}{\partial\xi}(\xi)\;\hat{N}_j^{p_j}(\eta) \\ \hat{N}_i^{p_i}(\xi)\;\frac{\partial\hat{N}_j^{p_j}}{\partial\eta}(\eta) \end{pmatrix} \tag{30}$$

Since the Ritz functions have a compact support and are defined on each knot span independently, according to Eq. 19, the integrations to compute the system coefficients are performed for each knot span successively. Therefore, it is convenient to *consider the knot spans as elements*, to adopt the same terminology as the classical method. These integration are carried out numerically, using Gauss quadrature rules.

4.8 Boundary conditions

Neumann boundary conditions are imposed in the right-hand side term, in a weak sense. For Dirichlet boundary conditions, degrees of freedom are directly imposed at the boundary and can be removed from the system. However, this is possible only using open knot vectors, for which the solution field interpolates the degrees of freedom at the extremity of the patch. Moreover, in case of non-constant values (e.g. the solution field is given by a function of the coordinates on the boundary), the degrees of freedom should be computed to match the desired function along the boundary.

4.9 Overview of the algorithm

The whole algorithm for isogeometric analysis can be described as:

1. Construction of the CAD representation of the computational domain (pre-process);
2. Computation of the matrix coefficients
   For each knot span:
   (a) Computation of the B-spline function and gradient values at the Gauss points according to Eqs. 19 and 30;
   (b) Computation of the transformation matrix and Jacobian at the Gauss points according to Eqs. 11 and 21 ;
   (c) Calculation of the matrix coefficients according to Eq. 29 ;
   (d) Assembly of the matrix ;
3. Computation of the right-hand side terms
   For each knot span on boundaries with Neumann condition:
   (a) Computation of the B-spline function and gradient values at the Gauss points according to Eq. 19 ;
   (b) Computation of the transformation matrix and Jacobian at the Gauss points according to Eqs. 11 and 20;
   (c) Computation of the imposed flux value at the Gauss points ;
   (d) Calculation of the right-hand side term according to Eq. 29;
   (e) Assembly of the vector ;
4. Imposition of values for boundaries with Dirichlet conditions ;
5. Solving of the linear system ;
6. Possible refinement process according to Eq. 23 ;
   (a) Refinement for the geometry ;
   (b) Refinement for the solution field ;

   Goto step (2) until a stopping criterion is satisfied ;
7. Reconstruction of the solution field (post-process) according to Eq. 24.

5 Application to thermal conduction

5.1 Problem with a simple geometry

5.1.1 Problem description

We first consider a problem with a very simple geometry, for which even classical methods have an exact description of the geometry. Therefore, the following tests only aim at validating the method and testing the numerical schemes.

The problem considered is depicted in figure Fig. 10. The geometry is composed of a single square. Two uniform Dirichlet conditions are imposed at left and right faces, with $T_1 = 1$ and $T_2 = 2$. A zero Neumann condition is imposed at the top (adiabaticity condition), whereas a sine thermal flux is imposed at the bottom boundary. In the whole computational domain the thermal conductivity is assumed to be constant and equal to unity.

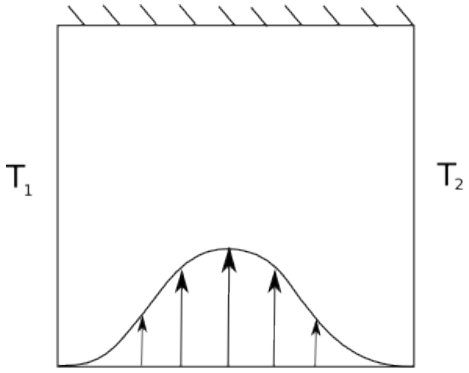


Figure 10: Test-case with simple geometry.

5.1.2 Numerical results

We first parameterize the computational domain using a natural B-Spline surface. As exemple, we consider a cubic B-Spline surface, with 5×5 control points. As result, the number of knots is $k = 9$ for each direction (e.g. $(0, 0, 0, 0, 1, 2, 2, 2, 2)$). Fig. 11 represents the iso-ξ and iso-η lines of the computational domain, for each knot value. Since four knots are equal at each extremity of the patch (cubic B-Spline), there is only one knot in the interior area. As consequence, the computational domain is composed of only four elements, used for the integration.

After computing the matrix and right-hand side coefficients as described above, the linear system is inverted using the Gauss-Seidel algorithm, until the residual norm is reduced by a factor six. For this example, the size of the linear system is only 5×5.

Once the vector of unknowns is computed, the physical solution is reconstructed according to Eq. 24. Results in terms of temperature field and iso-T lines are depicted in Figs. 12 and 13. As can be seen, a very smooth solution is obtained, using only 5×5 degrees of freedom. It is convenient to represent the solution as a B-Spline surface, by plotting the temperature as vertical axis (see Fig. 14). This figure allows to clearly understand how the solution field is defined and the role of the degrees of freedom, as B-Spline control points. The non-interpolatory nature of the degrees of freedom is underlined.

5.1.3 Accuracy study

An accuracy study is performed using this test-case. As the geometry is exactly described whatever the number of degrees of freedom, only the accuracy of the approximation scheme is tested. We employ successively linear, quadratic and cubic B-Spline functions and carry out a systematic refinement procedure by inserting knots uniformly. The L2 norm of the error is computed to measure the accuracy of the numerical scheme. We consider the solution obtained using a very large number of degrees of freedom as the exact solution in order to

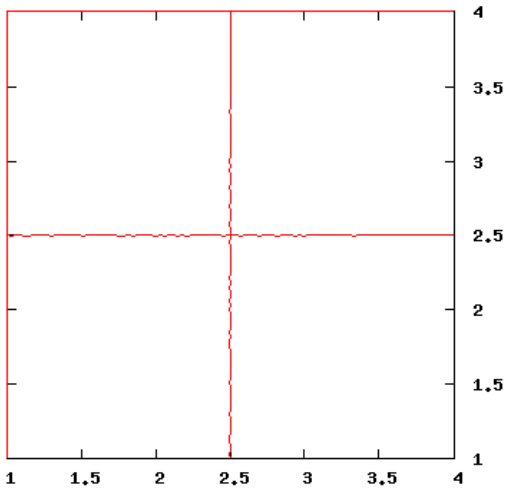


Figure 11: Computational domain (iso-ξ and iso-η lines).

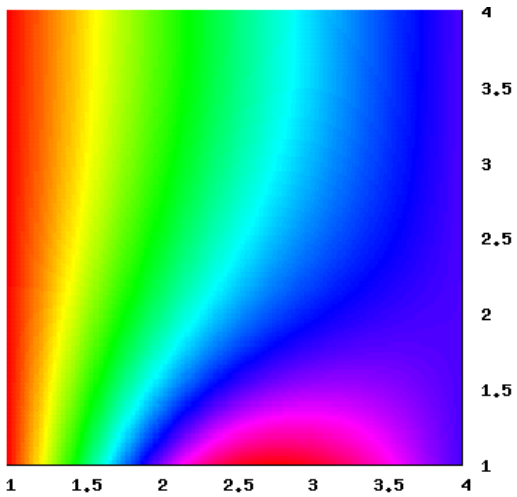


Figure 12: Temperature field (5×5 degrees of freedom).

compute the error. The evolution of the error with respect to the number of degrees of freedom is depicted in Fig. 15.

As expected, we obtain a second-order accuracy when using linear functions, a third-order accuracy for quadratic functions and a fourth-order accuracy for cubic functions.

5.1.4 Computational time study

The use of high-order discretization schemes yields a significant error reduction for a given number of degrees of freedom. However, this is not free and the computational cost increases during the evaluation of the system coefficients and the linear system inversion. Therefore, it is interesting to measure the computational time also.

The table 1 presents the evolution of the CPU time, for different function degrees, as well as the evolution of the error. These tests are performed using an Intel Core2 Duo 2.8 GHz. As can be seen, for an error level of 10^{-3}, the CPU time required using linear functions is about 12s whereas it takes 0.25s using quadratic functions and 0.08s using cubic functions. This example shows the critical impact on the CPU time of using high-order schemes.

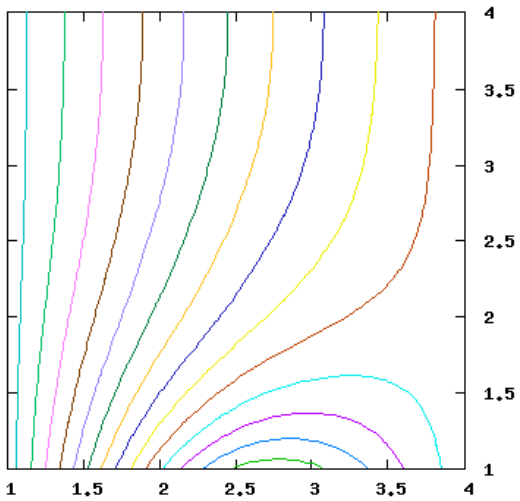


Figure 13: Iso-T lines (5×5 degrees of freedom).

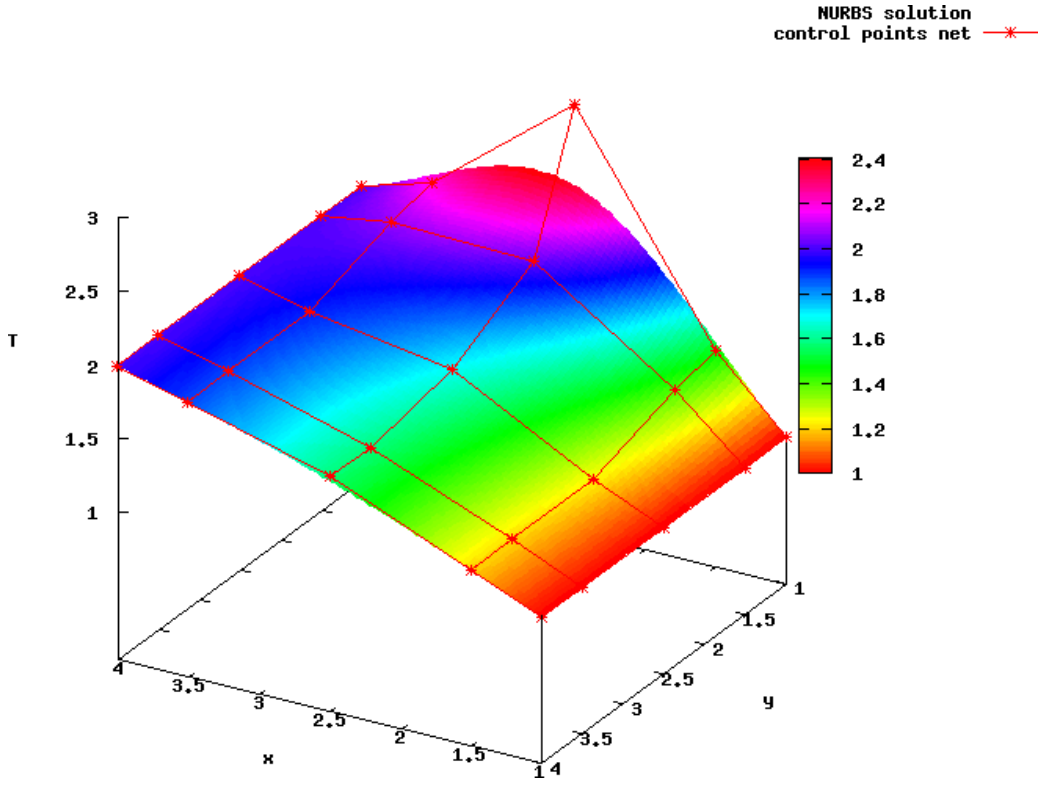


Figure 14: Temperature as B-Spline surface with control points (5×5 degrees of freedom).

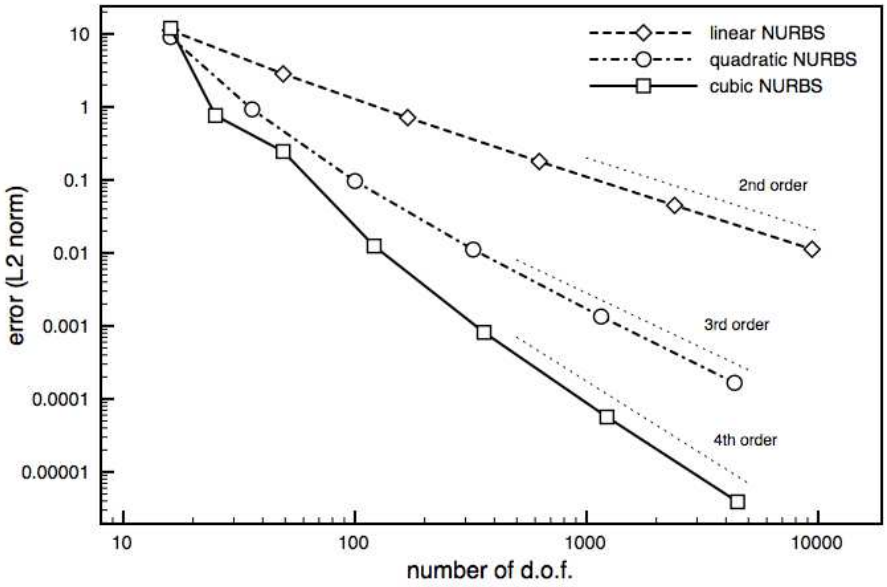


Figure 15: Accuracy study.

Degree	d.o.f.	Error	CPU (s)
linear	49	$0.28\ 10^{1}$	0.004
linear	625	$0.17\ 10^{0}$	0.072
linear	2401	$0.45\ 10^{-2}$	0.911
linear	9409	$0.11\ 10^{-2}$	12.150
quadratic	36	$0.91\ 10^{0}$	0.004
quadratic	324	$0.11\ 10^{-1}$	0.030
quadratic	1156	$0.13\ 10^{-2}$	0.252
quadratic	4356	$0.16\ 10^{-3}$	2.832
cubic	49	$0.24\ 10^{0}$	0.007
cubic	361	$0.81\ 10^{-3}$	0.078
cubic	1225	$0.56\ 10^{-4}$	0.461
cubic	4489	$0.38\ 10^{-5}$	4.509

Table 1: Computational time study.

5.2 Problem with a more complex geometry

5.2.1 Problem description

We now consider a problem with a more complex geometry, for which the boundaries are represented by cubic B-Spline curves. The boundary conditions are similar to those of the previous exercise. Fig. 16 shows the geometry of the test-case, whereas Fig. 17 presents the definition of the boundaries as B-Spline curves (with six control points).

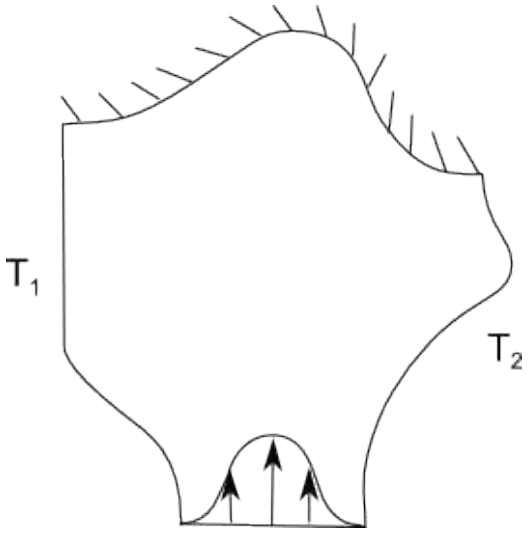


Figure 16: Test-case with a more complex geometry.

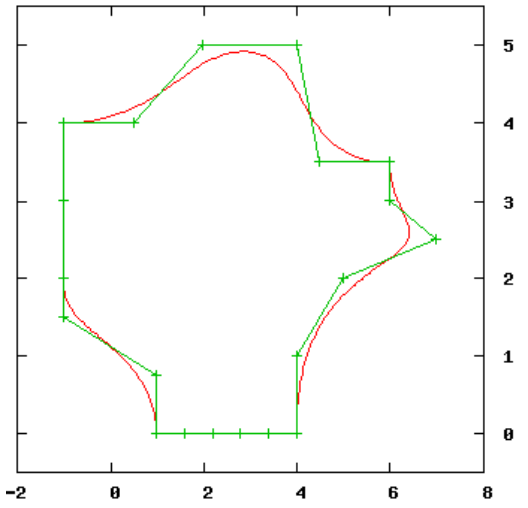


Figure 17: Boundaries as cubic B-Spline curves.

5.2.2 Numerical results

Since the geometry of the problem is defined using cubic B-Spline curves with six control points, the use of a cubic B-Spline surface including 6×6 control points at least is mandatory to exactly represent the geometry. As example, we consider a computational domain defined by a cubic B-Spline surface including 9×9 control points. The location of these control points is determined in order to match exactly the boundaries of the domain. Fig. 18 and 19 show respectively the iso-ξ and iso-η lines of the computational domain for each knot value, and the control points net.

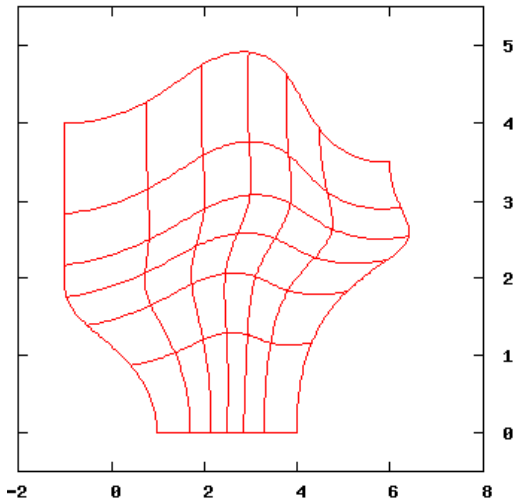


Figure 18: Computational domain (iso-ξ and iso-η lines).

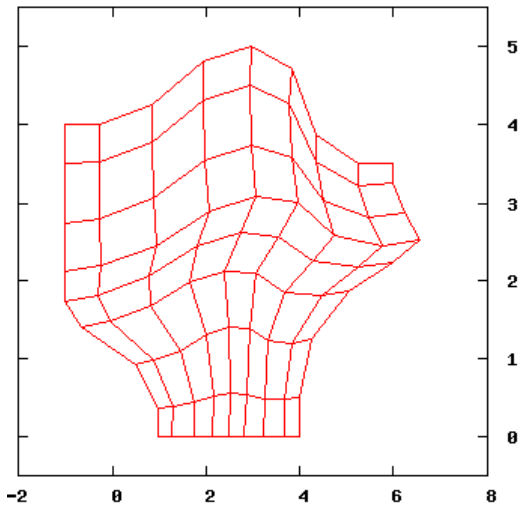


Figure 19: Control points net.

For each direction, there are $k = 13$ knots, since we use cubic functions and 9 control points. As four knots are identical at each extremity, there are five interior knots for each direction, yielding 6×6 elements for the integration.

It should be emphasized that a set of computational domains can be constructed by knot insertion from the patch including 6×6 control points. This can be done without altering the boundary geometry, that always matches the one defined in the original problem. Fig 20 presents the computational domains obtained by applying some successive knot insertion procedures to the patch including 6×6 control points.

As example, we consider back the computational domain defined by 9×9 control points. The solution is computed by inverting the 9×9 linear system. Then, the solution field is plotted in Fig. 21 and iso-T lines in Fig. 22 . We also represent the solution field as B-Spline surface with the control points in Fig. 23.

5.2.3 Accuracy study

As in the first exercise, we carry out an accuracy study, by computing the evolution of the L2 norm of the error with respect to the number of degrees of freedom. A solution computed with a very large number of degrees of

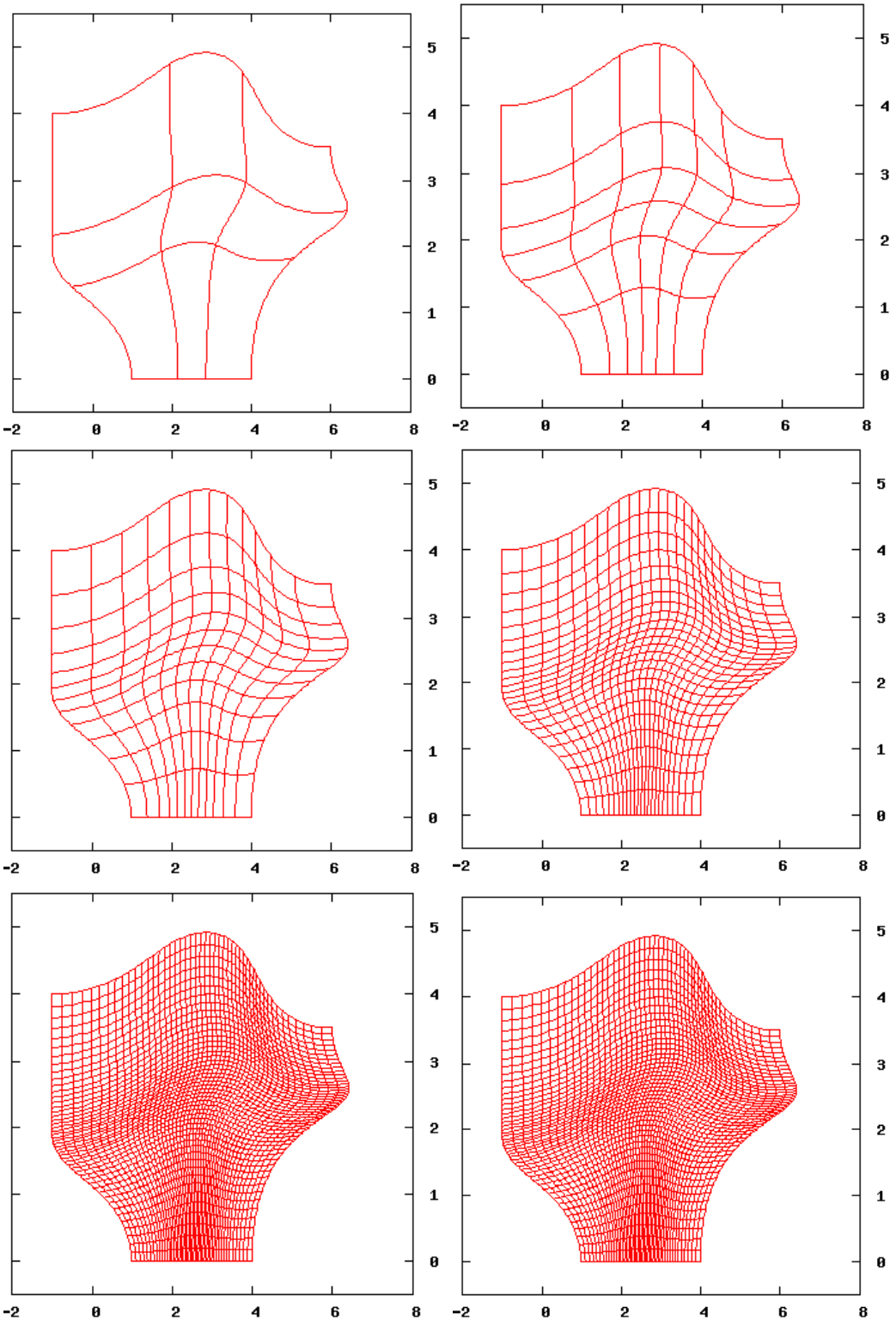


Figure 20: Illustration of the knot insertion procedure.

freedom is considered as the exact solution for the error estimation. Results can be seen in Fig. 24. As can be observed, a fourth-order accuracy is obtained, even though the geometry is more complex.

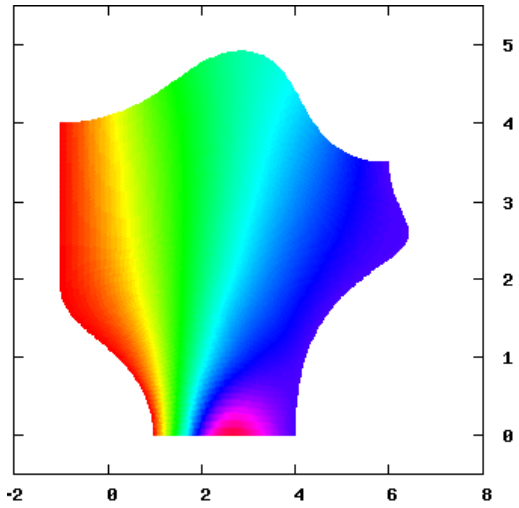


Figure 21: Temperature field (9×9 degrees of freedom).

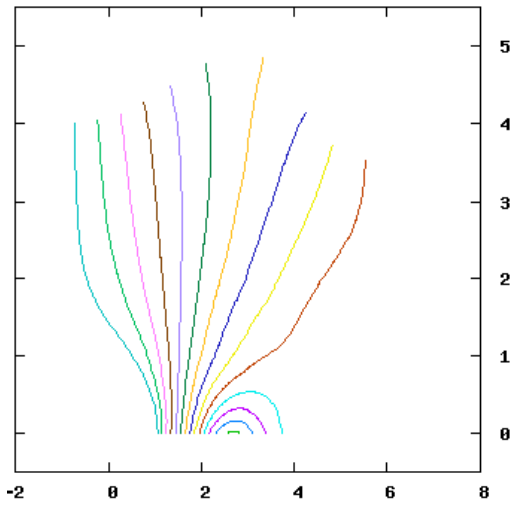


Figure 22: Iso-T lines (9×9 degrees of freedom).

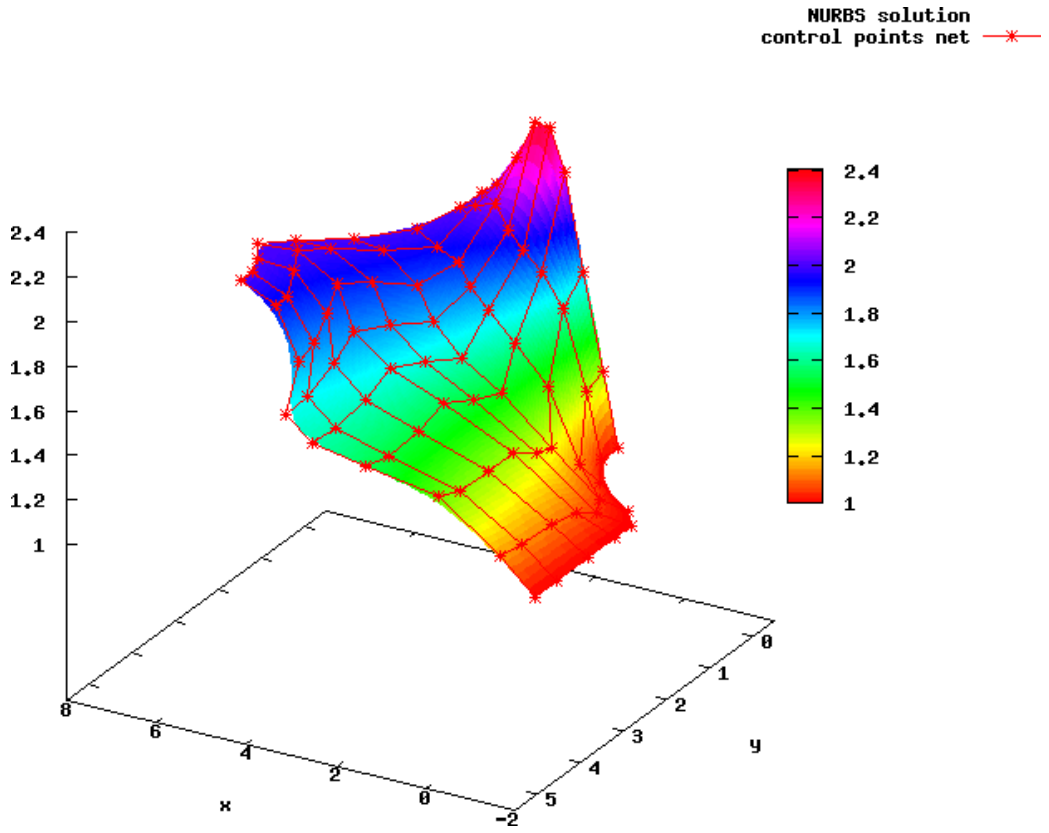


Figure 23: Temperature as B-Spline surface with control points (9×9 degrees of freedom).

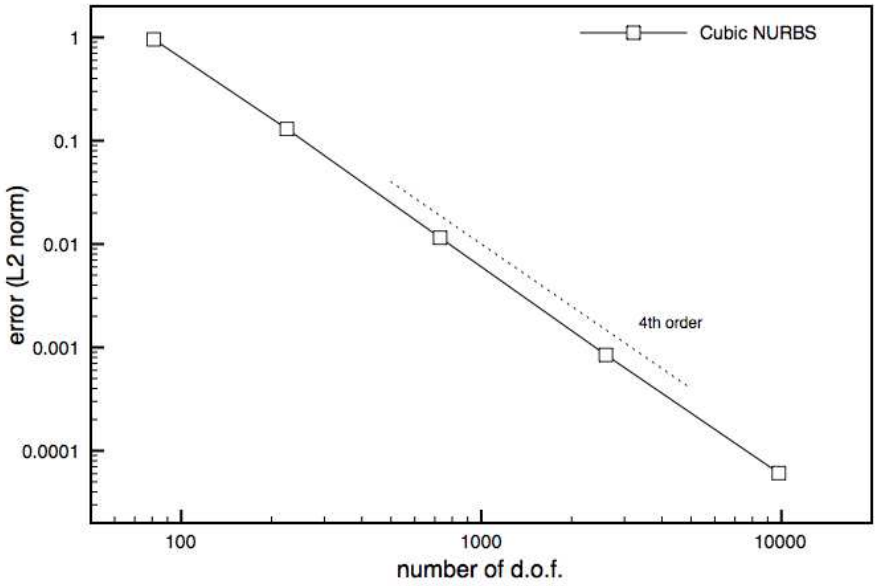


Figure 24: Accuracy study.

Conclusion

In this report we have presented the isogeometric analysis method, proposed by T. Hughes and his collaborators to overcome some difficulties that occur in classical design loop. The method has been described and some numerical experiments have been carried out, yielding the following comments:

- Isogeometric analysis consist in a classical Finite-Element (FE) framework, with NURBS functions as Ritz functions and control points as degrees of freedom ;
- The computational domain matches the boundaries geometry whatever the number of degrees of freedom. Then, the number of degrees of freedom is chosen according to the solution field and not to represent the geometry ;
- Classical FE solvers have to be modified significantly to adopt the isogeometric paradigm. Especially, data structures have to be modified to take into account parametric domains and integration procedures should be carried out on the knot spans of parametric domains instead of grid elements ;
- High-order schemes can be constructed without additional coding efforts thanks to the recursive definition of NURBS functions ;
- H- and p-adaptive schemes can be constructed ;
- Degrees of freedom cannot be interpreted as the solution values at a given location (non-interpolatory approach), contrary to classical FE methods ;
- Geometry and solution values at a given location are not known or accessed without computations to reconstruct the surface or volume NURBS field ;
- Some pre- and post-treatment tools are required, for instance to construct the parametric computational domain or visualize the solution field ;
- A nice coding framework can be carried out thanks to the use of an identical data structure for geometry and physical fields.

Several points have not been discussed nor presented in this report, such as theoretical results[1], the combination of degree-elevation and knot insertion procedures[4], the use of T-Splines for local refinement[3], the use of several connected patches[2], the construction of hierarchical methods for solving PDEs or the application of optimization methods in the isogeometric analysis context[5]. Some of these aspects are currently under study.

References

[1] Bazilevs, Y., de Veiga, L. B., Cottrell, J., Hughes, T., and Sangalli., G. Isogeometric analysis: approximation, stability and error estimates for refined meshes. *Mathematical Models and Methods in Applied Sciences*, 6 (2006), 1031–1090.

[2] Cottrell, J., Hughes, T., and Reali, A. Studies of refinement and continuity in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 196 (2007), 4160–4183.

[3] Dorfel, M., Juttler, B., and Simeon, B. Adaptive isogeometric analysis by local h-refinement with t-splines. *Computer Methods in Applied Mechanics and Engineering* (2009). to appear.

[4] Hughes, T., Cottrell, J., and Bazilevs, Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194 (2005), 4135–4195.

[5] Wall, W. A., Frenzel, M. A., and Cyron, C. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering 197*, 33-40 (2008), 2976–2988.



Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399