ANISOTROPIC DIFFUSION IN TOROIDAL GEOMETRIES∗, ∗∗

AHMED RATNANI1, EMMANUEL FRANCK2, BONIFACE NKONGA3, ALINA EKSAEVA4 AND MARIA KAZAKOVA5

Abstract. In this work, we present a new finite element framework for toroidal geometries based on a tensor product description of the 3D basis functions. In the poloidal plan, different discretizations, including B-splines and cubic Hermite-Bézier surfaces are defined, while for the toroidal direction both Fourier discretization and cubic Hermite-Bézier elements can be used. In this work, we study the MHD equilibrium by solving the Grad-Shafranov equation, which is the basis and the starting point of any MHD simulation. Then we study the anisotropic diffusion problem in both steady and unsteady states.

1. Introduction

The context of this work is the simulation and the modeling of nuclear fusion reaction as power source. The aim of magnetic confinement fusion is to develop a power plant that gains energy from the fusion of deuterium and tritium in a magnetically confined plasma. ITER, a tokamak type fusion experiment currently being built in the South of France, is the next step towards this goal. One of the many challenges for the numerical simulation in a tokamak is the modeling and the simulation of magneto-hydrodynamics (MHD) instabilities as disruptions or ELMs [11] [1]. These instabilities which occur at the boundary of the plasma generate a loss of energy and can damage the wall of the tokamak. For this reason it is necessary to understand the behavior of these instabilities and find a way to control them using experiment and simulation. A physical model well suited to describe those large scale instabilities is the set of magneto-hydrodynamic equations (MHD) with resistivity and bi-fluid effects. In the world some codes have been developed to simulate this problem and solve the MHD equations. One of those codes is the JOREK code which implements and solves both some reduced and full MHD models [10] based on assumptions on the velocity and magnetic fields in a toroidal geometry. The equations in the poloidal plane (circular, d-shape or x-point meshes) [6] [5] [7] are discretized using cubic bezier splines and the toroidal discretization use Fourier expansion. In this work, we are interested in the study a small part of the equations used in JOREK which can generate numerical difficulties: the elliptic and parabolic operators, especially the anisotropic diffusion.

In a tokamak, the heat diffusion is mostly in the direction of the magnetic field described by a large toroidal part and a poloidal perturbation (in the JOREK code the toroidal part of magnetic field is assumed constant in time). At the limit (where the ratio between the diffusion in the magnetic field direction and the isotropic diffusion tend to the infinity) the problem is singular [9]. This singularity at the limit generates ill-conditioning, large error in the perpendicular direction and other numerical problems. At the end, we

∗ This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

∗∗ Corresponding author: ahmed.ratnani@ipp.mpg.de

1 Max-Planck Institute für PlasmaPhysik, Garching, Munich
2 Inria Nancy Grand-Est & IRMA, Strasbourg, France
3 Laboratoire Jean Dieudonné, Université Nice Sophia-Antipolis, Nice, France
4 Moscow Engineering & Physics Institute MEPHI, Russia
5 Lavrentyev Institute of Hydrodynamics of SB RAS, Russia

© EDP Sciences, SMAI 2016
want to study the discretization of elliptic operators especially the anisotropic diffusion operators with splines on toroidal geometry to understand and identify the main numerical problems linked to these operators and in the future propose numerical solutions for the JOREK code.

In the first part of this paper we will detail the construction of the 3D finite elements (obtained by tensor product $2D \times 1D$) using a 2D Hermite-Bézier discretization. The second part is on the discretization of the elliptic Grad-Safranov operator. This operator is important in the JOREK code to obtain the magnetic equilibrium and the toroidal current in the MHD. This operator is also a simplified 2D operator in the cylindrical geometry comparing to the anisotropic diffusion and a first step to study finite element method in the cylindrical coordinates. After the Grad - Shafranov equation we will study the discretization of the anisotropic diffusion equation. These problems are studied on some classical geometries in the tokamak contexts: circular or d-shape poloidal section.

2. Bézier Finite Elements Method

In this section, we present a cubic Hermite-Bézier finite elements method. We consider the isoparametric setting, where both the geometry and the discrete space are described by the same basis functions, which are cubic Hermite-Bézier elements. First, we start by recalling Bézier curves and surfaces. Then we introduce a flexible way to create and manipulate the geometry, thanks to B-splines curves and surfaces. We then explain how to convert a B-spline surface into Bézier surfaces and finally to the cubic Hermite-Bézier description. It is very important to keep in mind that the Bézier surfaces are considered as elements, while the B-spline surface/patch can be decomposed into a structured grid where each element is a Bézier surface. Hence, a B-spline surface can be viewed as a macro-element.

2.1. Bézier curves and surfaces

Bézier curves/surfaces are polynomial parametric curves/surfaces. Since their introduction by Bézier in 1962, they became very popular in the Computer Aided Design (CAD) community.

A Bézier curve (also called arc) of degree $p$ is defined as

$$C(u) = \sum_{i=0}^{p} c_i B_i^p(u),$$

where $(B_i^p)_{0 \leq i \leq p}$ denote Bernstein polynomials: $B_i^p(t) = \binom{p}{i} t^i (1-t)^{p-i}$, $0 \leq t \leq 1$.

The coefficients $(c_i)_{0 \leq i \leq p}$ are called control points. Among the interesting properties of Bézier curves, we recall the affine invariance one: any affine transformation, including translations, rotations and scaling, is applied to the curve by applying it to the control points.

A Bézier surface of degrees $(p, q)$ is defined as

$$S(u, v) = \sum_{i=0}^{p} \sum_{j=0}^{q} s_{ij} B_i^p(u) B_j^q(v),$$

where $(B_i^p)_{0 \leq i \leq p}$ and $(B_j^q)_{0 \leq j \leq q}$ are the Bernstein polynomials of degrees $p$ and $q$, respectively. $(s_{ij})_{0 \leq i \leq p, 0 \leq j \leq q}$ are the control points.

Bézier surfaces have many advantages; they are numerically stable [3] and posses different algorithms for sub-division and elevation degree. Moreover, the control points have a geometric interpretation (see the next sections). However, they can not model a surface with a prescribed regularity at given sites nor satisfy a large number of constraints, as the number of degrees of freedom is exactly $(p+1)(q+1)$. Such surfaces are more useful in CAD, as they allow better control of the shape with local modification impact. For this reason, B-splines surfaces have been introduced, where one can control its regularity. The main difference with the Bézier setting, is the piecewise polynomial property of B-splines. Another limitation occurs when considering a collection of Bézier curves/surfaces, sticked together to form a continuous (or smooth) curve/surface, because of the redundancy of control points on their boundaries (extremities in the case of Bézier curves).
Remark 2.1. Bézier curves/surfaces can be generalized to rational Bézier surfaces when considering rational Bernstein polynomials. Rational curves and surfaces are very useful in CAD, as they can model exactly conics.

For more details about Bézier curves/surfaces, as well as rational Bézier surfaces, we refer to the book [8].

2.2. B-splines curves and surfaces

We start this section by recalling some basic properties about B-splines curves and surfaces. We also recall some fundamental algorithms (knot insertion and degree elevation). For a basic introduction to the subject, we refer to the book [8].

A B-splines family, \((N_i)_{0 \leq i \leq n-1}\) of order \(k\), can be generated using a non-decreasing sequence of knots \(T = \{t_i\}_{0 \leq i \leq n+k-1}\).

Definition 2.2 (B-Splines series). The \(j\)-th B-spline of order \(k\) is defined by the recurrence relation:

\[
N_j^k = w_j^k N_j^{k-1} + (1 - w_{j+1}^k) N_{j+1}^{k-1},
\]

where

\[
w_j^k(x) = \frac{x - t_j}{t_{j+k-1} - t_j}; \quad N_j^1(x) = \chi_{[t_j, t_{j+1}]}(x)
\]

for \(k \geq 1\) and \(0 \leq j \leq n - 1\).

We note some important properties of a B-splines basis:

- B-splines are piecewise polynomial of degree \(p = k - 1\),
- When \(n = k\), B-splines are exactly the Bernstein polynomials,
- Compact support; the support of \(N_j^k\) is contained in \([t_j, t_{j+k}]\),
- If \(x \in [t_j, t_{j+1}]\), then only the B-splines \(\{N_{j-k+1}^k, \ldots, N_j^k\}\) are non vanishing at \(x\),
- Positivity: \(\forall j \in \{0, \ldots, n - 1\} \ N_j(x) > 0, \ \forall x \in [t_j, t_{j+k}]\),
- Partition of unity: \(\sum_{i=0}^{n-1} N_i^k(x) = 1, \forall x \in \mathbb{R}\),
- Local linear independence,
- If a knot \(t_i\) has a multiplicity \(m_i\), then the B-spline is \(\mathcal{C}^{(p - m_i)}\) at \(t_i\).

The vectorial space spanned by these B-splines, which we denote \(S_k(T, I)\), where \(I\) denotes the interval \([t_1, t_{n+1}]\), is called the Schoenberg space.

Definition 2.3 (B-splines curve). The B-spline curve in \(\mathbb{R}^d\) associated to knot vector \(T = (t_i)_{0 \leq i \leq n+k-1}\) and the control polygon \((P_i)_{0 \leq i \leq n-1}\) is defined by :

\[
C(t) = \sum_{i=0}^{n-1} N_i^k(t) P_i.
\]

In (Fig. 1), we give an example of a quadratic B-spline curve, and its corresponding knot vector and control points.

We have the following properties for a B-Spline curve:

- \(C\) is a piecewise polynomial curve,
- If \(n = k\), then \(C\) is just a Bézier-curve,
- When the internal knots have a multiplicity equal to the Spline degree, the corresponding curve is a concatenation of Bézier curves. In each, element, the non-vanishing Splines are exactly the Bernstein polynomials.
- The curve interpolates its extremas if the associated multiplicity of the first and the last knot are maximum \((i.e.\ \text{equal to} \ k)\), i.e. open knot vector,
- Invariance with respect to affine transformations,
- Strong convex-hull property:
  - if \(t_i \leq t \leq t_{i+1}\), then \(C(t)\) is inside the convex-hull associated to the control points \(P_{i-p}, \ldots, P_i\),
- Local modification: moving the \(i^{th}\) control point \(P_i\) affects \(C(t)\), only in the interval \([t_i, t_{i+k}]\),
- The control polygon approaches the behavior of the curve.

Remark 2.4. In order to model a singular curve, we can use multiple knots or control points, i.e. \(P_i = P_{i+1}\).
Figure 1. (left) A quadratic B-spline curve and its control points using the knot vector $T = \{0, 0, 0, \frac{1}{2}, \frac{3}{4}, 1, 1, 1\}$, (right) the corresponding B-splines.

Remark 2.5. Here again, we can introduce the rational version of B-splines, called NURBS for Non Uniform B-splines. Please refer to the book [8] for more details.

2.3. Fundamental geometric operations

By inserting new knots into the knot vector, we add new control points without changing the shape of the B-Spline curve. This can be done using the DeBoor algorithm [2]. We can also elevate the degree of the B-Spline family and keep unchanged the curve [4]. In (Fig. 2), we apply these algorithms on a quadratic B-Spline curve and we show the position of the new control points.

Figure 2. (a) A quadratic B-spline curve and its control points. The knot vector is $T = \{0, 0, 0, \frac{1}{2}, \frac{3}{4}, 1, 1, 1\}$. (b) The curve after a h-refinement by inserting the knot $\{\frac{1}{2}\}$ while the degree is kept equal to 2. (c) The curve after a p-refinement, the degree was raised by 1 (using cubic B-splines).
2.4. Deriving a B-spline curve

The derivative of a B-spline curve is obtained as:

\[ C'(t) = \sum_{i=0}^{n-1} N_i^p (t)P_i = \sum_{i=0}^{n-1} \left( \frac{p}{t_i+p} N_i^p(t)P_i - \frac{p}{t_i+p+1} N_i^p(t)P_i \right) = \sum_{i=0}^{n-2} N_i^{p-1} Q_i, \tag{2.3} \]

where \( Q_i = p_{i+1} P_{i+1} - P_i \), and \( \{N_i^{p-1}, \ 0 \leq i \leq n-2\} \) are generated using the knot vector \( T^* \) which is obtained from \( T \) by reducing by one the multiplicity of the first and the last knot (in the case of open knot vector), i.e. by removing the first and the last knot.

More generally, by introducing the B-splines family \( \{N_i^{p-j}, 0 \leq i \leq n-j-1\} \) generated by the knot vector \( T^j \) obtained from \( T \) by removing the first and the last knot \( j \) times, we have the following result:

**Proposition 2.6.** The \( j^{th} \) derivative of the curve \( C \) is given by

\[ C^{(j)}(t) = \sum_{i=0}^{n-j-1} N_i^{p-j} (t)P_i^{(j)}, \quad \text{where} \quad P_i^{(j)} = \frac{p - j + 1}{t_i+p+1} \left( P_{i+1}^{(j-1)} - P_i^{(j-1)} \right) \quad \text{and} \quad P_i^{(0)} = P_i, \]

By denoting \( C' \) and \( C'' \) the first and second derivative of the B-spline curve \( C \), it is easy to show that:

**Proposition 2.7.** We have,

- \( C'(0) = \frac{p}{t_{p+1}}(P_1 - P_0) \), \( C''(0) = \frac{p(p-1)}{t_{p+1}} \left( \frac{1}{t_{p+1}} P_0 - \left( \frac{1}{t_{p+2}} + \frac{1}{t_{p+2}} \right) P_1 + \frac{1}{t_{p+2}} P_2 \right) \),
- \( C'(1) = \frac{p}{1-t_{n-1}}(P_{n-1} - P_{n-2}) \), \( C''(1) = \frac{p(p-1)}{1-t_{n-1}} \left( \frac{1}{1-t_{n-1}} P_{n-1} - \left( \frac{1}{1-t_{n-2}} + \frac{1}{1-t_{n-2}} \right) P_{n-2} + \frac{1}{1-t_{n-2}} P_{n-3} \right) \).

2.5. Multivariate tensor product splines

Let us consider \( d \) knot vectors \( T = \{T^1, T^2, \cdots, T^d\} \). For simplicity, we consider that these knot vectors are open, which means that \( k \) knots on each side are duplicated so that the spline is interpolating on the boundary, and of bounds 0 and 1. In the sequel we will use the notation \( I = [0,1] \). Each knot vector \( T^i \), will generate a basis for a Schoenberg space, \( S_k(T^i, I) \). The tensor product of all these spaces is also a Schoenberg space, namely \( S_k(T) \), where \( k = \{k_1, \cdots, k_d\} \). The cube \( P = I^d = [0,1]^d \), will be referred to as a patch.

The basis for \( S_k(T) \) is defined by a tensor product:

\[ N_i^k := N_{i_1}^{k_1} \otimes N_{i_2}^{k_2} \otimes \cdots \otimes N_{i_d}^{k_d}, \]

where \( i = \{i_1, \cdots, i_d\} \).

A typical cell from \( P \) is a cube of the form: \( Q_i = [\xi_{i_1}, \xi_{i_1+1}] \otimes \cdots \otimes [\xi_{i_d}, \xi_{i_d+1}] \). In (Fig. 3), we apply knot insertion and elevation degree algorithms on a quadratic B-Spline surface and we show the position of the new control points.

**Remark 2.8.** A B-splines surface can be converted to a list of Bézier surfaces. In order to do so, we only need to increase the multiplicity of internal knots to match the spline degree. Therefor, it is easy to extract Bézier surfaces for all logical elements. Finally, we can elevate their degrees to a desired one.

2.6. Software implementation

**CAID** [13] is an open source multi-platform software that has been designed for IsoGeometric analysis pre and post processing. Its design goal is to provide a fast, light and user-friendly designer and meshing tool. The later is provided by **Igasus** [14] which is an open source Python package for solving (systems of) partial differential equations. Moreover, the user can write his own solvers and integrate them in CAID. One particular and interesting capability of CAID is the macro-recording, which provides the user with a Python script of all her interaction with CAID.
2.7. Hermite-Bézier finite elements method

We consider a sequence of domains $(Ω_h)$ that converges to a 2D domain $Ω$. We recall that in the case of the IsoGeometric Analysis approach, we have $Ω_h = Ω$. Let $Q_h$ be a partition of $Ω_h ⊂ Ω$, where every element $e ∈ Q_h$ denotes a Bézier surface. Therefore, an element $e$ is the image of the unit square $P = [0, 1]^2$ by a local mapping, denoted $F_e$ and every point $x$ in the element $e$ can be written as:

\[
x := x(s, t) = F_e(s, t) = \sum_{i,j=0}^{p,q} x_{ij} B_i^e(s) B_j^e(t), \quad s, t ∈ [0, 1].
\] (2.4)

We can then, create a global map $F$ defined on every element of the partition $Q_h$ as:

\[
F|_e := F^e, \forall e ∈ Q_h.
\]

When using B-splines to create Bézier surfaces, we ensure the global regularity of the mapping $F$ depending on the knot vectors.

In order to have a standard finite element formulation, we shall use a change of basis to associate the degrees of freedom (i.e. the control points) to the vertices (the 4 summits of the Bézier surface) of our elements. The new basis is given in Appendix [A], Eq. [A.34]. As we are using the IsoParametric setting, the unknowns are also expressed in the same basis as the position $x$.

**Remark 2.9.** The assembling procedure is done on the logical domain $[0, 1]^2$. Therefore, all derivatives are computed on the logical domain and must be transported to the physical domain.

3. GRAD-SHAFRANOV SOLVER

In the JOREK code, some elliptic and parabolic operators are discretized (2D Grad-Safranov operator for the equilibrium of the plasma in the tokamak). Solving this 2D equation associated with this operator allows...
to validate the finite element method in cylindric coordinates. The MHD equilibrium is described by the force balance, the Ampere’s law of the Maxwell equations and the magnetic divergence constraint (Eq. 3.5)

\[
\begin{align*}
J \times B &= \nabla P, \\
\nabla \times B &= \mu_0 J, \\
\nabla \cdot B &= 0.
\end{align*}
\] (3.5)

The equilibrium of the plasma in a tokamak (Fig. 4) is assumed axisymmetric and the magnetic field \( B \) can be expressed as \( B = \nabla \varphi \times \nabla \psi + g \nabla \varphi \) where \( \varphi \) the toroidal angle (Fig. 4) and \( \psi \) is the poloidal flux function. The current density \( J \) can also be written, in the same form, as \( J = R J_\varphi \nabla \varphi + \frac{1}{\mu_0} \nabla g \times \nabla \varphi \).

Figure 4. The toroidal plasma configuration

Using \( B \cdot \nabla P = 0 \) it is easy to see that the pressure is a function of the flux \( \psi \). On the other hand, \( J \cdot \nabla P = 0 \) implies that the function \( g \) is also a flux function. Inserting these relations in the balance force equation yields to a second order nonlinear elliptic equation, known as Grad-Shafranov-Schlüter equation:

\[
\Delta^* \psi = R \partial_R \left( \frac{1}{R} \partial_R \psi \right) + \partial_{zz}^2 \psi = -\mu_0 R^2 \frac{d}{d\psi} P - \frac{d}{d\psi} g.
\] (3.6)

The right hand side of the equation (Eq. 3.6) is the input data and is given by \( g \) (toroidal magnetic field) and the pressure \( P = P(\psi) \).

### 3.1. Solver for Grad-Shafranov equation

In the subsection, we are interested in the resolution of the non-linear elliptic partial differential equation:

\[
\frac{1}{R^2} \Delta^* \psi := \frac{1}{R} \partial_R \left( \frac{1}{R} \partial_R \psi \right) + \partial_{zz}^2 \psi = \mathcal{F}(R, Z, \psi), \quad \text{on } \Omega,
\] (3.7)

\[
\psi = 0, \quad \text{on } \partial \Omega.
\] (3.8)

Using a Picard method, we have,

- \( \psi^0 \) is given,
- knowing \( \psi^n \), we solve:

\[
\frac{1}{R^2} \Delta^* \psi^{n+1} = \mathcal{F}(R, Z, \psi^n)
\] (3.9)

The domain \( \Omega \) will be describe later. Let \( \psi \in \mathcal{V}_h \subset \mathcal{V} \) a test function with \( \mathcal{V} \) the functional space of the solution. Multiplying it by the Grad-Shafranov equation 3.8 and integrating by parts, in the cylindric coordinates, we
obtain the weak form of the equation:
\[
\int_{\Omega} \frac{1}{R} \nabla \psi^{n+1} \cdot \nabla \phi \, d\Omega = \int_{\Omega} F(R, Z, \psi^n) \phi \, d\Omega, \quad \forall \phi \text{ in } V_h.
\] (3.10)

We assume that \( \psi^n = \sum_{j=1}^{N} \psi^n_j \phi_j \) with \( N \) the total number of degrees of freedom and \( \phi_j(x) \) the basis function associated to the \( j^{th} \) degree of freedom. By taking \( \phi = \phi_i \) in the weak form we obtain
\[
\sum_{i} \psi^n_i \left( \int_{\Omega} \frac{1}{R} \nabla \phi_j \cdot \nabla \phi_i \, d\Omega \right) = \int_{\Omega} F(R, Z, \psi^n) \phi_i \, d\Omega \quad \forall i \in \{1, \ldots, N\}. \] (3.11)
At the end solving (Eq. 3.9) is equivalent to solve the linear system
\[
M X^{n+1} = f^n, \quad \text{with} \quad M_{ij} = \left( \int_{\Omega} \frac{1}{R} \nabla \phi_j \cdot \nabla \phi_i \, d\Omega \right), \quad f^n_i = \int_{\Omega} F(R, Z, \psi^n) \phi_i \, d\Omega \quad \text{and} \quad X^{n+1} = \{\psi^{n+1}_1, \ldots, \psi^{n+1}_N\}.
\]

3.2. Numerical results

In this section we propose numerical results to validate the finite element method for the Grad-Shafranov equation. A first validation is done for an analytical solution on a square domain. To test in the same time, the mapping we propose to solve the equation on the non-Cartesian Collela mesh obtained with the Collela mapping:
\[
x := \eta = F(\eta) = (\eta_1 + \alpha \sin(k_1 \eta_1) \sin(k_2 \eta_2), \eta_2 + \alpha \sin(k_1 \eta_1) \sin(k_2 \eta_2)).
\] (3.12)

This analytical mapping is approximated by B-Splines surfaces, then we convert them into a collection of cubic Bézier surfaces. These surfaces are therefore converted into the Hermite-Bézier form. If the mapping is the identity the mesh is a classical Cartesian mesh. The right hand side and the nonlinear function \( F \) are computed so that the solution vanishes on the boundary, when \( F \) contains a quadratic dependence on \( \psi \). In figures (Fig 6 and 7) and table (Tab. 1), we show the expected convergence order, while in (Fig. 5), the numerical solution and the numerical error are shown. (Fig. 6) and (Fig. 7) show also the evolution of the \( L^2 \) and \( H^1 \) norms depending on Picard’s iterations. As we may notice, our Picard’s algorithm converges after 6 iterations, but this does not seem to be the rule for other geometries and \( F \) functions.

In the sequel, we present different simulations on a realistic geometry (Fig 8) for poloidal plane in the tokamak.

**Figure 5.** Numerical solution and its error for the equation (Eq 3.9) equation on a square domain, using Collela’s mapping with a grid 64 × 64.
Table 1. Grad-Shafranov convergence order for the identity and Colella’s mappings: \( h \)-dependence of the \( L^2 \) and \( H^1 \) norms between the grids \( n \times n \) and \( 2n \times 2n \), with \( n = \{4, 8, 16, 32\} \). For the \( L^2 \) norm, the values correspond to \( \log_2 \frac{\|u-u_H\|}{\|u-u_h\|} \) with \( H = 2h \). For the \( H^1 \) norm, we use the same formulae with the corresponding semi-norm.

Figure 6. \( L^2 \) and \( H^1 \) errors (left) after convergence (right) and the residual error for each step of the Picard algorithm, for the equation (Eq 3.8) on a square domain, using the identity mapping.

Based on characteristic parameters describing the cross-section for ITER, ASDEX-Upgrade and JET tokamaks. These parameters are: the inverse aspect-ratio \( \epsilon \), the elongation \( \kappa \) and the triangularity \( \delta \). They are given by the following formulae:

\[
\begin{align*}
R_0 &= \frac{R_{\text{min}} + R_{\text{max}}}{2}, \\
\epsilon &= \frac{R_{\text{max}} - R_0}{R_0}, \\
\kappa \epsilon &= \frac{Z_{\text{max}}}{R_0}, \\
1 - \delta \epsilon &= \frac{R(Z = Z_{\text{max}})}{R_0},
\end{align*}
\]

where in the last equation, we consider a given flux surface (in our case, defined by \( \psi(R, Z) = 0 \)) and \( R \) as a function of \( Z \).

In the following test, we consider the right hand side function

\[
F(R, Z, \psi) = R^2,
\]

which gives the analytical solution \([12]\).

\[
\psi(R, Z) = \frac{R^4}{8} + d_1 + d_2 R^2 + d_3 (R^4 - 4(RZ)^2).
\]
Figure 7. $L^2$ and $H^1$ errors (left) after convergence (right) and the residual error for each step of the Picard algorithm, for the equation (Eq. 3.8) on a square domain, using Collela’s mapping.

Figure 8. Geometric definition of the parameters $\epsilon, \kappa$ and $\delta$.

Plugging (Eq. 3.13) in (Eq. 3.15) leads to the following system for $d_1, d_2$ and $d_3$:

$$\begin{pmatrix}
1 & (1+\epsilon)^2 & (1+\epsilon)^4 \\
1 & (1-\epsilon)^2 & (1-\epsilon)^4 \\
1 & (1-\delta\epsilon)^2 & (1-\delta\epsilon)^4 - 4(1-\delta\epsilon)^2\kappa^2\epsilon^2
\end{pmatrix}
\begin{pmatrix}
\frac{d_1}{\kappa^2} \\
\frac{d_2}{\kappa^2} \\
\frac{d_3}{\kappa^2}
\end{pmatrix}
= \frac{1}{8}
\begin{pmatrix}
(1+\epsilon)^4 \\
(1-\epsilon)^4 \\
(1-\delta\epsilon)^4
\end{pmatrix}. \tag{3.16}$$
In figure (Fig. 9), we plot the computational domain and the numerical solution for \textit{ITER}, \textit{ASDEX-Upgrade} and \textit{JET} relevant parameters. In (Fig. 11) we plot the $L^2$ and $H^1$ norms for MESH2 (Fig 10). Meshes generated by \texttt{CAID}). In this case, the theoretical convergence order is not achieved; the reason is the bad approximation of the boundary when $\partial R \psi = 0$ or $\partial Z \psi = 0$.

\textbf{Remark 3.1.} In order to construct a 2D mesh from a given boundary given by an analytical formula, we first need to find a set of points that lie on this boundary. This is done using a Python function. However, we found that the distribution of the given points is not uniform and therefore some parts of the boundary have less points for a B-spline or Hermite reconstruction/interpolation. The loss of the convergence order is related to the bad distribution of these points. This must be fixed in the future in order to allow high order approximation, for arbitrary B-spline approximations.

\textbf{Figure 9.} Computational domain in the poloidal plan, in the $(R, Z)$ coordinates, and the analytical solution for: (left) \textit{ITER} $(R_0, \epsilon, \kappa, \delta) = (1, 0.32, 1.7, 0.33)$, (middle) \textit{ASDEX-Upgrade} $(R_0, \epsilon, \kappa, \delta) = (1.645, 0.311, 1.77, 0.0932)$, (right) \textit{JET} $(R_0, \epsilon, \kappa, \delta) = (2.924, 0.323, 1.87, 0.141)$

\textbf{Figure 10.} \textit{ITER} like relevant parameter domain: (left) MESH1 (right) MESH2.

\section{Anisotropic Diffusion}

In the previous section we have validated the finite elements method in cylindrical coordinates. Now we propose to extend this to a more challenging model: the anisotropic diffusion equation. Firstly, this is a 3D...
problem in a toroidal geometry. Secondly this problem is singular when the anisotropy of the diffusion tends to the infinity. This singularity leads to an ill-posed problem and generates numerical issues like ill-conditioning and lack of accuracy. In this section, we are interested in the resolution of the anisotropic diffusion problem for both steady and unsteady cases. The anisotropic diffusion time evolution problem is

\[
\partial_t u = \nabla \cdot (K \nabla u) + f, \quad x \in \Omega,
\]

where \( \Omega \) is the domain, \( u \) describes the temperature inside a tokamak, the conductivity \( K = \kappa_\parallel K_\parallel + \kappa_I I \) is a 3 by 3 tensor. The later is a sum of two contributions, first parallel component is given by:

\[
\kappa_\parallel = \kappa_\parallel \frac{BB^T}{\|B\|^2}
\]

for prescribed magnetic field \( B = \nabla \varphi \times \nabla \psi + g \nabla \varphi \) (with \( \varphi \) the toroidal angle), \( \kappa_\parallel \) is a parallel diffusion coefficient. The second component \( \kappa_I I \) is a standard isotropic diffusion. We are interested in highly anisotropic configurations with \( \frac{\kappa_\parallel}{\kappa_I} \simeq 10^6 \gg 1 \).

**Circular Plasma** For circular cross section plasma, the Grad-Shafranov equation associated to the nonlinear right hand side \( F(R,Z,\psi) = R^2 \), leads to

\[
\psi(R,Z) = C_\psi \ln \left[ 1 + \frac{r^2}{a^2} \right], \quad \text{with} \quad C_\psi = \frac{a^2}{2R_0 q_0}
\]

**Remark 4.1.** When using an analytical equilibrium for anisotropic diffusion or tearing mode problems, the boundary condition for the flux (needed to construct the equilibrium) does not need to be Homogeneous Dirichlet. In fact, in this case, we don’t solve the Grad-Shafranov equation and the flux formula is directly inserted in the conductivity tensor.

**General case** In the general case, we need to solve the equilibrium (Grad-Shafranov) for the potential \( \psi \) in order to define the magnetic field.

### 4.1. The 3D elliptic anisotropic diffusion equation

We start by solving the following anisotropic diffusion problem (Eq. 4.18)

\[
- \nabla \cdot K \cdot \nabla u = f, \quad x \in \Omega.
\]
We solve this 3D problem using a tensor product between a 2D circular mesh for the poloidal plan and 1D uniform mesh for the toroidal direction. Let $\phi_i$ be a test function. This 3D function is given as the tensor product of 2D and 1D basis functions. Multiplying (Eq. 4.18) by $\phi_i$ and integrating by parts leads to

$$\int_{\Omega} \frac{\kappa_{ii}}{\|B\|^2} (B \cdot \nabla u)(B \cdot \nabla \phi_i) + \kappa_I \nabla u \cdot \nabla \phi_i \, d\Omega = \int_{\Omega} f \phi_i \, d\Omega$$

(Eq. 4.19)

Using the expansion $u = \sum_{j=1}^{N} u_j \phi_j$ with $N$ the total number of degrees of freedom, we get

$$\sum_{j=1}^{N} u_j \left( \int_{\Omega} \frac{\kappa_{ii}}{\|B\|^2} (B \cdot \nabla \phi_j) (B \cdot \nabla \phi_i) + \kappa_I \nabla \phi_j \cdot \nabla \phi_i \right) \, d\Omega = \int_{\Omega} f \phi_i \, d\Omega,$$

(Eq. 4.20)

which leads to the linear system $M U = F$ where

$$M_{ij} = \int_{\Omega} \frac{\kappa_{ii}}{\|B\|^2} (B \cdot \nabla \phi_j) (B \cdot \nabla \phi_i) + \kappa_I \nabla \phi_j \cdot \nabla \phi_i \, d\Omega, \quad \text{and} \quad F_i = \int_{\Omega} f \phi_i \, d\Omega, \quad \forall i, j \in [1, n].$$

(Eq. 4.21)

Remark 4.2. For validation, we take any function $u$ that vanishes on the boundary and then compute $f = -\nabla \cdot K \cdot \nabla u$. In the case of circular cross section plasma, we take $u_{\text{exact}} = 1 - \left( \frac{R - R_0}{a} \right)^2 + \frac{Z^2}{a^2}$. For an annulus cross section plasma, an analytical solution is given by $u_{\text{exact}} = \left( 1 - \left( \frac{R - R_0}{a^2} \right)^2 + \frac{Z^2}{a_{\text{center}}^2} \right) \left( \frac{R - R_0}{a^2} + \frac{Z^2}{a_{\text{center}}^2} - 1 \right)$. It is important to notice that the definition of the conductivity tensor remains the same as well as the relation between the flux $\psi$ and the magnetic field $B$. These two analytical solutions are only constructed so that they vanish on the boundary.

We show in (Fig. 12), the $L^2$ error norm for different meshes and values of $\kappa_{ii}$. GMRES was used with a Jacobi preconditioner for a tolerance $t_p = 10^{-11}$. Block-Jacobi preconditioner was also tested and results were quite equivalent. We may notice that for a diffusion $> 10^4$ the solver does not converge, because of the bad conditioning of the linear system. This can be viewed in (Fig. 13), where we show the evolution of the residual error for the Krylov solver.

![Figure 12. Anisotropic diffusion operator: $L^2$ error norm depending on $\kappa_{ii}$ for different meshes. The 2D mesh is the equivalent of the MESH2 (Fig 10) for a circle. The 1D mesh in toroidal direction is uniform.](image)

4.2. Time evolution problem

For the time depending problem, we consider the following implicit scheme

$$\frac{u^{n+1} - u^n}{\Delta t} - \nabla \cdot (K \nabla u^{n+1}) = f,$$
Figure 13. Anisotropic diffusion operator: evolution of the residual error for the Krylov solver, for a mesh (a) $8 \times 8 \times 8$, (b) $16 \times 16 \times 16$ and (c) $32 \times 32 \times 32$

which leads to

$$u^{n+1} - \Delta t \nabla \cdot (K \nabla u^{n+1}) = u^n + \Delta t f. \tag{4.22}$$

Let $\phi_i$ be a test function. Multiplying (Eq. 4.22) by $\phi_i$, integrating by parts and using the expansion of $u = \sum_{j=1}^n u_j \phi_j$, leads to the following linear system problem

$$M^T U^{n+1} = M^T U^n + F,$$
where for $i, j \in [1, n]$, we have

\[
M_{ij}^T = \int_{\Omega} \phi_i \phi_j + \Delta t \left( \frac{\kappa_{||}}{\|B\|^2} (B \cdot \nabla \phi_j)(B \cdot \nabla \phi_i) + \kappa_j \nabla \phi_j \cdot \nabla \phi_i \right) d\Omega
\]  
(4.23)

\[
M_{ij}^E = \int_{\Omega} \phi_i \phi_j d\Omega
\]  
(4.24)

\[
F_i = \int_{\Omega} \Delta t f \phi_i d\Omega
\]  
(4.25)

4.2.1. Transient case

In the sequel, we are interested in the steady state solution of the anisotropic diffusion problem on a circular cross section domain with a mesh type MESH1 and MESH2 (c.f. Fig. 10 for ITER-like relevant parameter domain). In order to avoid the polar-like singularity at the center of the plasma in MESH1, we consider an annulus domain of minimum radius $r_{\text{min}} = 0.1$, while the maximum radius $r_{\text{max}}$ is kept equal to 1 for both meshes. The source terms are computed as the solution of the elliptic anisotropic diffusion equation, for $\kappa_{||} = 0$, with the following analytical solutions

\[
\begin{cases}
  u_{\text{exact}} = 1 - \frac{(R-R_0)^2 + Z^2}{a^2}, & \text{for MESH1}, \\
  u_{\text{exact}} = 1 - \frac{(R-R_0)^2 + Z^2}{a^2_{\text{center}}} \left( \frac{(R-R_0)^2 + Z^2}{a^2_{\text{center}}} - 1 \right), & \text{for MESH2}.
\end{cases}
\]  
(4.26)

Numerical simulations were done with time step $dt = 5.10^{-3}$, 1 and a final time $T_{\text{final}} = 0.5, 20$ respectively. GMRES and Jacobi preconditioner were used. In (Fig. 14,16), we show the final $L^2$ and $H^1$ norms as functions of $\kappa_{||}$ for different meshes while in (Fig. 15,17), we show the time evolution of the $L^2$ and $H^1$ norms. Finally, in table (2), we show the power approximation, $h$-dependence, of the $L^2$ and $H^1$ norms between the grids $8 \times 4 \times 4$ and $16 \times 8 \times 8$. As we can notice, the convergence order is conserved at the final time.

As seen in the convergence study of the Hermite-Bézier elements (table (2), when increasing the grid points, the value of the numerical convergence order achieved between 2 consecutive grids tends to the theoretical value 4. Therefor finer grids should lead to a better numerical convergence order.

**Figure 14.** Short time behavior of $L^2$ and $H^1$ error norms for different values of $\kappa_{||}$ at time $T_{\text{final}} = 0.5$ and $dt = 5.10^{-3}$ using MESH2.
Figure 15. Short time behavior of $L^2$ and $H^1$ error norms evolution for the transient anisotropic diffusion using MESH1 for $dt = 5 \times 10^{-3}$ and a grid $16 \times 8 \times 8$.

Figure 16. Long time behavior of $L^2$ and $H^1$ error norms for different values of $\kappa_\parallel$ at time $T_{\text{final}} = 20$ and $dt = 1$ using MESH2.

4.2.2. Evolution of a Gaussian Pulse

In this last section, we are interested in the time evolution of a Gaussian pulse, without source term, for the initial condition. This test case allows to study the effects of the meshes on the quality of the solution. More precisely we propose to verify that flux aligned grids (given by $\psi$) give more accurate solution. In the following, we use three type of meshes. One non aligned mesh (NAM) with $32 \times 32$ and $64 \times 64$ cells, one perturbed aligned mesh (PAM) $16 \times 32$ and $32 \times 64$ and one aligned mesh (AM). The studied test case is a Gaussian pulse.
Figure 17. Long time behavior of (first line) $L^2$ and (second line) $H^1$ error norms evolution for the transient anisotropic diffusion using MESH2 for and $dt = 1$ and a grid (left) $16 \times 8 \times 6$ and (right) $32 \times 16 \times 6$.

Table 2. Transient state $h$-dependence of the $L^2$ and $H^1$ norms between the grids $8 \times 4 \times 4$ and $16 \times 8 \times 8$. For the $L^2$ norm, the values correspond to $\log_2 \| u - u_H \| / \| u - u_h \|$ with $H = 2h$, evaluated at $t = 0$ and $t = 0.5$. For the $H^1$ norm, we use the same formulae with the corresponding semi-norm.

with a diffusion in the direction of the magnetic field. The initial data is given by

$$u(t = 0, x) = e^{-\frac{1}{2\sigma^2} ((R-R_1)^2 - R_1^2)^2 - Z^2}$$

where $R_1 = 3.5$, $\sigma = 0.1$, $\kappa_I = 0$ and $\kappa_\parallel = 1$. In this case, energy on any given toroidal surface should be conserved. In order to validate our simulation, we compute and plot the mean value of the numerical solution on a toroidal surface

$$\bar{u}(t, R) = \int_0^{2\pi} \int_0^{2\pi} Ru \, d\varphi \, d\theta$$

The results (Fig. 19–21) show that aligned grids lead to more accurate results. Indeed, if we compare the result on aligned grid and on non aligne grid with $32 \times 32$ cells, we notice that we have perpendicular perturbation in the non aligned case (Fig. 19–20–21). Comparing the results of the non aligned grid (with $64 \times 64$ cells) and the aligned grid (with $32 \times 32$ cells), we remark also small perturbations in the perpendicular direction for the non aligned grid (see Fig. 21). For the perturbed aligned grid, we observe less perturbations.
in the perpendicular direction but then solutions in not good (the diffusion is slower and not at the right place). Consequently, the mesh can affect the tensor of diffusion coefficient.

4.3. Future work

In this work, we focused on the development of a finite element method based on Hermite-Bézier surfaces. The code is written in Fortran, using MPI for parallelization. A new feature of the code is the ability to create any finite element space using a linear expansion over Bernstein polynomials on quadrangles and triangles. The code is still in an early stage and further optimization are being studied and will be implemented. A Computer Aided Design tool has been also developed that allows us to create and manipulate the geometry easily. For instance, one can convert the Hermite-Bézier description into a B-spline one and reciprocally. A new implementation of the resolution of $\psi(R,Z) = 0$ must be done in order to fit accurately the boundary.

In a future work, we will derive the h-estimators for the Hermite-Bézier elements. We also need to derive some mesh-quality estimators based on anisotropic estimations. This will allows us for instance to see the impact of the geometry in anisotropic diffusion problem. This quantitative study has never been done and we need to provide a sufficient mathematical framework for the Hermite-Bézier elements.

Finally, as a continuation of this work, we will study a tearing mode and compare different kind of discretizations in a future work.

5. Conclusions

We have developed a finite element method based on Hermite-Bézier elements for the poloidal plane (quadrangular elements) and a toroidal geometry (hexahedral elements). In order to create the partition of the computation domain $\Omega$ into Bézier surfaces, we use B-Spline surfaces, which allows us to control the regularity between elements and when needed (if possible), a $C^1$ regularity can be imposed. Numerical results are shown first on a nonlinear elliptic problem (Grad-Shafranov equation). The expected convergence order (4) is achieved on a (deformed) square but not on a Soloviev (analytic solution of the Grad-Shafranov equation) and this is due
to the bad approximation of the boundary. This point will be treated in a future work. Secondly, we apply our
finite element method to the resolution of an anisotropic diffusion problem, where we have observed the usual
asymptotic error for very strong anisotropy. Future work will focus on this behavior to improve the accuracy
either by meshes alignment or by asymptotic preserving preconditioning.

Acknowledgement. We warmly thank the referees for their useful comments and remarks, which helped
us to improve the quality of this work.

REFERENCES

August 2008.
Figure 20. $\bar{u}(t, R)$ for the time $t = 60$. In the top the non aligned mesh (NAM) with $32 \times 32$ and $64 \times 64$ cells. In the bottom the perturbed aligned mesh (PAM) with $32 \times 64$ cells and the aligned mesh with $32 \times 32$ cells.


A. APPENDIX A:

Let us consider a cubic Bézier patch

$$\mathbf{x}(s, t) = \sum_{i,j=0}^{3} x_{ij} B_i(s) B_j(t), \quad s, t \in [0, 1]$$

(A.29)

Computing derivatives of this surface on the four (logical) points $\{(s, t) = (0, 0), (0, 1), (1, 0), (1, 1)\}$ leads to

$$\begin{align*}
(x(0, 0) = x_{00}, & \quad x_s(0, 0) = 3(x_{10} - x_{00}), \quad x_t(0, 0) = 3(x_{01} - x_{00}), \quad x_{st}(0, 0) = 9(x_{00} + x_{11} - x_{01} - x_{10}) \\
(x(0, 1) = x_{03}, & \quad x_s(0, 1) = 3(x_{13} - x_{03}), \quad x_t(0, 1) = 3(x_{03} - x_{02}), \quad x_{st}(0, 1) = 9(x_{03} + x_{12} - x_{02} - x_{13}) \\
(x(1, 0) = x_{30}, & \quad x_s(1, 0) = 3(x_{30} - x_{20}), \quad x_t(1, 0) = 3(x_{31} - x_{30}), \quad x_{st}(1, 0) = 9(x_{30} + x_{21} - x_{20} - x_{31}) \\
(x(1, 1) = x_{33}, & \quad x_s(1, 1) = 3(x_{33} - x_{23}), \quad x_t(1, 1) = 3(x_{33} - x_{32}), \quad x_{st}(1, 1) = 9(x_{33} + x_{22} - x_{23} - x_{32})
\end{align*}$$

(A.30)
But, since

Therefore, our Bézier patch can be rewritten as

Now, let us introduce the following quantities

Therefore, our Bézier patch can be rewritten as

But, since

Figure 21. $u(t, R)$ for the time $t = 100$. In the top the non aligned mesh (NAM) with $32 \times 32$ and $64 \times 64$ cells. In the bottom the perturbed aligned mesh (PAM) with $32 \times 64$ cells and the aligned mesh with $32 \times 32$ cells.
Now plugging these relations into (Eq. A.29), we get
\[ \mathbf{x}(s, t) = \mathbf{x}_{00} M_{00}(s, t) + a_{00} \mathbf{u}_{00} N_{01}(s, t) + b_{00} \mathbf{v}_{00} P_{02}(s, t) + a_{00} b_{00} \mathbf{w}_{00} Q_{03}(s, t) \]
\[ + a_{01} M_{10}(s, t) + a_{03} \mathbf{u}_{03} N_{11}(s, t) + b_{03} \mathbf{v}_{03} P_{12}(s, t) + a_{03} b_{03} \mathbf{w}_{03} Q_{13}(s, t) \]
\[ + a_{10} M_{30}(s, t) + a_{30} \mathbf{u}_{30} N_{30}(s, t) + b_{30} \mathbf{v}_{30} P_{30}(s, t) + a_{30} b_{30} \mathbf{w}_{30} Q_{30}(s, t) \]
\[ + a_{33} M_{33}(s, t) + a_{33} \mathbf{u}_{33} N_{33}(s, t) + b_{33} \mathbf{v}_{33} P_{33}(s, t) + a_{33} b_{33} \mathbf{w}_{33} Q_{33}(s, t) \]  
\hspace{1cm} (A.34)

where the new basis is (Eq. A.35)

\[
\begin{align*}
M_{00}(s, t) &= B_0(s)B_0(t) + B_1(s)B_0(t) + B_0(s)B_1(t) + B_1(s)B_1(t), \quad N_{00}(s, t) = B_1(s)B_0(t) + B_1(s)B_1(t) \\
P_{00}(s, t) &= B_0(s)B_1(t) + B_1(s)B_1(t), \quad Q_{00}(s, t) = B_1(s)B_1(t) \\
M_{01}(s, t) &= B_0(s)B_2(t) + B_1(s)B_3(t) + B_0(s)B_2(t) + B_1(s)B_2(t), \quad N_{01}(s, t) = B_1(s)B_3(t) + B_1(s)B_2(t) \\
P_{01}(s, t) &= -B_0(s)B_2(t) - B_1(s)B_2(t), \quad Q_{01}(s, t) = B_1(s)B_2(t) \\
M_{02}(s, t) &= B_3(s)B_0(t) + B_2(s)B_0(t) + B_1(s)B_1(t) + B_2(s)B_1(t), \quad N_{02}(s, t) = -B_2(s)B_0(t) - B_2(s)B_1(t) \\
P_{02}(s, t) &= B_0(s)B_1(t) + B_2(s)B_1(t), \quad Q_{02}(s, t) = B_2(s)B_1(t) \\
M_{03}(s, t) &= B_3(s)B_3(t) + B_2(s)B_3(t) + B_3(s)B_2(t) + B_2(s)B_2(t), \quad N_{03}(s, t) = -B_2(s)B_3(t) - B_2(s)B_2(t) \\
P_{03}(s, t) &= -B_3(s)B_2(t) - B_2(s)B_2(t), \quad Q_{03}(s, t) = B_2(s)B_2(t) \\
M_{33}(s, t) &= B_3(s)B_3(t) + B_2(s)B_3(t) + B_3(s)B_2(t) + B_2(s)B_2(t), \quad N_{33}(s, t) = -B_2(s)B_3(t) - B_2(s)B_2(t) \\
P_{33}(s, t) &= -B_3(s)B_2(t) - B_2(s)B_2(t), \quad Q_{33}(s, t) = B_2(s)B_2(t) \
\end{align*}
\]  
\hspace{1cm} (A.35)