Isogeometric analysis in reduced magnetohydrodynamics

To cite this article: A Ratnani and E Sonnendrücker 2012 Comput. Sci. Disc. 5 014007

View the article online for updates and enhancements.

Related content
- Ray tracing of optical systems using NURBS surfaces
  Davide Bianchi, Manuel Sánchez del Río and Claudio Ferrero
- Isogeometric finite element analysis using polynomial splines over hierarchical T-meshes
  Nhon Nguyen-Thanh, Hung Nguyen-Xuan, Stéphane P A Bordas et al.
- Isogeometric analysis based on scaled boundary finite element method
  Y Zhang, G Lin and Z Q Hu

Recent citations
- Application of MFS--MPS to the current-hole simulation in a Tokamak
  D. Nath et al.
Isogeometric analysis in reduced magnetohydrodynamics

A Ratnani\textsuperscript{1} and E Sonnendrücker\textsuperscript{2}
\textsuperscript{1} CEA/DSM/IRFM, Cadarache, France
\textsuperscript{2} INRIA-Nancy-Grand Est, CALVI Project Team and IRMA, Université de Strasbourg, France
E-mail: ahmed.ratnani@cea.fr and sonnen@math.unistra.fr

Received 3 February 2012, in final form 24 May 2012
Published 23 July 2012
Computational Science & Discovery 5 (2012) 014007 (21pp)
doi:10.1088/1749-4699/5/1/014007

Abstract. Isogeometric analysis (IGA) consists of using computer-aided design (CAD) models defining the geometry of the computational domain using both B-splines and non-uniform rational B-splines (NURBS) to represent the unknowns that are the solution of a partial differential equation using a finite element principle. In this paper, we review the main ideas of IGA and apply it to a reduced magnetohydrodynamic (MHD) model that is used in tokamak simulations. This is a first step towards arbitrary high-order and smooth approximations of reduced MHD generalizing the Bézier splines approach of Czarny and Huysmans (2008 J. Comput. Phys. 227 7423–45).
Since their introduction, B-splines have had much success because their implementation is based on fast and stable algorithms. They are used in industry and in academic research for interpolation, computer-aided design (CAD) and data fitting. The recent work of Hughes et al [2, 3] and the introduction of isogeometric analysis (IGA) added yet another dimension to their use, creating an interface between simulation and modeling.

Before the recent work of Hughes et al [2], the use of splines as basis functions in the finite element method was rare and mostly limited to uniform B-splines using periodic conditions, although the web splines developed by Hoellig and others proposed a strategy for dealing with boundary conditions [4, 5]. The idea of IGA, using geometric transformations-based splines and non-uniform rational B-splines (NURBS), is easier for most applications. Compared with the usual finite elements, the main change due to IGA is undoubtedly the emergence of $k$-refinement, a strategy that can increase the regularity of functions with the interface of the mesh in order to reduce the number of degrees of freedom. Moreover, the exact geometry is preserved after $h/p/k$-refinements.

In plasma physics, it is of great importance to develop solvers that are flexible with respect to geometry and that can provide high-order accuracy. Czarny and Huysmans [1] used Bézier elements for magnetohydrodynamic (MHD) simulations. Bézier surfaces are the most widely used tool in CAD. However, they cannot preserve the exact geometry. As far as we know, the use of Bernstein polynomials, which are linked to Bézier splines, for shape preservation in plasma physics was introduced in [6, 7].

IGA relies on NURBS functions, which are a generalization of spline functions and provide an exact modeling of a large class of computational domains, including conics and all spline surfaces. Moreover, it relies on a Cartesian grid of the parameter space and is fairly easy to use even employing spline basis functions of arbitrary degree. Moreover, for domains that can be represented using a periodic angular variable as is the case with the poloidal plane of the tokamak, we were able to develop a fast solver that is comparable in
Figure 1. B-spline functions associated with the knot vector $T = \{000 111\}$, of order $k = 3$. These are Bernstein polynomials.

computation time [8], for any spline degree, with the spectral and second-order finite difference solver used in GYSELA [9].

This work was inspired by [1], where the authors used cubic Bézier elements to study some MHD models. The IGA approach is a natural generalization of Bézier elements. Thanks to IGA, we can considerably reduce the number of degrees of freedom, which is a crucial problem in JOREK code [1]. Moreover, as we have seen for Maxwell’s equations [10], the use of higher regularity elements allows us to have larger Courant–Friedrichs–Lewy (CFL) condition numbers, which are a necessary condition for convergence while solving partial differential equations systems.

This paper is organized as follows. In section 2, we introduce splines and recall their most interesting properties. In section 3, we introduce splines as a CAD tool and show how one can construct curves and surfaces. In section 4, we introduce IGA and define its framework. Finally, in section 5, we show one application of IGA to the two-dimensional (2D) reduced MHD model: the current-hole problem.

2. Basic properties of B-splines

2.1. B-splines

Definition 2.1 (B-spline). Let $X = \{x_0, \ldots, x_p\}$ be a non-decreasing sequence of $p + 1$ points such that $x_0 \neq x_p$. The B-spline is defined by the following recurrence formula:

$$N(x; x_0, \ldots, x_p) = \frac{x - x_0}{x_{p-1} - x_0} N(x; x_0, \ldots, x_{p-1}) + \frac{x_p - x}{x_p - x_1} N(x; x_1, \ldots, x_p),$$

and the initialization is given by: $N(x; x_0, x_1) = 1$ if $x_0 \neq x_1$, and 0 otherwise.

In figures 1 and 2, we give examples of the generated B-splines families depending on the knot vectors (figure 1) and the order (figure 2).

2.2. B-spline series

As noted before, to construct a B-spline of degree $p$, we need $p + 1$ points that are called knots in spline terminology. Then, to create a family of B-splines, we will need to have a non-decreasing sequence of knots that is called a knot vector.
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 $1 \leq j \leq N$.

Let $T = (t_i)_{1 \leq i \leq N + k}$ be a non-decreasing sequence of knots, with $k = p + 1$. Each set of $p + 1$ consecutive knots $T_j = \{t_j, \ldots, t_{j+p}\}$ will generate a B-spline $N_j$. This leads to the following definition:

**Definition 2.2 (B-spline 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 $1 \leq j \leq N$.

We note some important properties of a B-spline basis.

- B-splines are piecewise polynomials of degree $p = k - 1$
- 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: $\forall j \in \{1, \ldots, N\}, N_j(x) > 0$, $\forall x \in [t_j, t_{j+k}]$
- Partition of unity: $\sum_{j=1}^N N_j^k(x) = 1, \forall x \in \mathbb{R}$
- Local linear independence
- If a knot $t_i$ has a multiplicity $m_i$, then the B-spline is $C^{(p-m_i)}$ at $t_i$

### 2.3. Multivariate tensor product splines

Let us consider $d$ knot vectors $T = \{T^1, T^2, \ldots, T^d\}$. For simplicity, we consider that those 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 following, 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 those spaces is also a Schoenberg space, namely $S_k(T)$, where $k = \{k_1, \ldots, k_d\}$. The cube $P = I^d = [0, 1]^d$ will be denoted as a patch.

The basis for $S_k(T)$ is defined by a tensor product,

$$N_i^k := N_{i_1}^{k_1} \otimes N_{i_2}^{k_2} \otimes \ldots \otimes N_{i_d}^{k_d},$$

where $i = \{i_1, \ldots, i_d\}$.

A typical cell from $P$ is a cube of the form $Q_i = [\xi_{i_1}, \xi_{i_1+1}] \otimes \ldots \otimes [\xi_{i_d}, \xi_{i+d-1}]$. With any cell $Q$ we will associate its extension $\hat{Q}$, which is the union of the supports of basis functions, that intersects $Q$. 

![Figure 2. B-spline functions associated with the knot vector $T = [000 1 2 3 4 4 5 5 5]$, of order $k = 1, 2, 3$ (left to right).](image-url)
3. Splines in computer-aided design

3.1. Modeling a curve using the spline form

In order to have control over the regularity of the curve, we need to use a piecewise polynomial form. That is why the use of B-splines has had much success.

Let \((P_i)_{1 \leq i \leq N} \in \mathbb{R}^d\) be a sequence of control points forming a control polygon.

**Definition 3.1 (B-spline curve).** The B-spline curve in \(\mathbb{R}^d\) associated with \(T = (t_i)_{1 \leq i \leq N+k}\) and \((P_i)_{1 \leq i \leq N}\) is defined by

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

We have the following properties for a B-spline curve.

- If \(N = k\), then \(C\) is just a Bézier-curve
- \(C\) is a piecewise polynomial curve
- The curve interpolates its extremas if the associated multiplicity of the first and the last knot is maximum (i.e. equal to \(k\))
- 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 with the control points \(P_{i-p}, \ldots, P_i\)
- Local modification: moving \(P_i\) affects \(C(t)\), only in the interval \([t_i, t_{i+k}]\)
- The control polygon approaches the behavior of the curve

**Remark 3.2.** We can use multiple control points: \(P_i = P_{i+1}\).

**Deriving a B-spline curve:** We have

\[
C'(t) = \sum_{i=1}^{n} \left( N_i'(t)P_i - \frac{p}{t_{i+p} - t_i} N_i^{k-1}(t)P_i - \frac{p}{t_{i+1+p} - t_{i+1}} N_i^{k-1}(t)P_i \right) = \sum_{i=1}^{n-1} N_i^{k-1*}(t)Q_i, \tag{3.2}
\]

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

**Example.** \(T = \{000 \frac{2}{3} \frac{3}{2} 111\}\), \(p = 2, n = 5\).

We have \(C(t) = \sum_{i=1}^{5} N_i^3(t)P_i\), then

\[
C'(t) = \sum_{i=1}^{4} N_i^{2*}(t)Q_i,
\]

where

\[
Q_1 = 5(P_2 - P_1), \quad Q_2 = \frac{10}{3}(P_3 - P_2),
\]

\[
Q_3 = \frac{10}{3}(P_4 - P_3), \quad Q_4 = 5(P_5 - P_4),
\]

and the B-splines \(\{N_i^{2*}, 1 \leq i \leq 4\}\) are associated with the knot vector \(T^* = \{00 \frac{2}{3} \frac{3}{2} 5 11\}\).
In Figure 3, we show an example of a B-spline curve. In this case, the B-splines generated are simply Berstein polynomials, and the obtained curve is a Bézier curve.

**Definition 3.3 (B-spline surface).** The B-spline surface of order $k$ associated with the knot vectors $\{T^{(1)}, T^{(2)}\}$, the control points $(P_{i,j})_{1 \leq i \leq N_1, 1 \leq j \leq N_2}$, is defined by

$$M(t^{(1)}, t^{(2)}) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} N_{i,j}(t^{(1)}, t^{(2)}) P_{i,j},$$

with $N_{i,j}(t^{(1)}, t^{(2)}) = N_i^{(1)}(t^{(1)}) N_j^{(2)}(t^{(2)})$.

3.2. Fundamental geometric operations

After modification, we denote the new parameters by $\tilde{N}$, $\tilde{k}$, $\tilde{T}$. $(Q_i)$ are the new control points.

3.2.1. Knot insertion. One can insert a new knot $t$, where $t_j \leq t < t_{j+1}$. For this purpose we use the DeBoor algorithm [11]:

$$\tilde{N} = N + 1,$$

$$\tilde{k} = k,$$

$$\tilde{T} = \{t_1, \ldots, t_j, t, t_{j+1}, \ldots, t_{N+k}\},$$

$$\alpha_i = \begin{cases} 1, & 1 \leq i \leq j - k + 1, \\ \frac{t - t_{i-1}}{t_{i+k-1} - t_{i-1}}, & j - k + 2 \leq i \leq j, \\ 0, & j + 1 \leq i, \end{cases}$$

$$Q_i = \alpha_i P_i + (1 - \alpha_i) P_{i-1}.$$

Many other algorithms exist, such as blossoming for fast insertion algorithm. For more details of this topic, see [12].
3.2.2. Order elevation. We can elevate the order of the basis without changing the curve. Several algorithms exist for this purpose. We used the one by Huang et al [13].

\[ \tilde{k} = k + m, \]
\[ \tilde{m}_i = m_i + m, \]
\[ \tilde{N} = N + ms. \]

Differential coefficients are denoted as \( \widetilde{P}_l^j \):

\[ \begin{align*}
\widetilde{P}_l^j & = \begin{cases} \\
\frac{1}{l_{i+1} - l_i} (\tilde{P}_i^{l-1} - \tilde{P}_i^{l-1}), & l > 0, t_{i+k-1} > t_{i+l}, \\
0, & l > 0, t_{i+k-1} = t_{i+l}, \\
\end{cases}
\end{align*} \]

\[ \beta_i = \sum_{l=1}^{i} m_l, 1 \leq i \leq s - 1 \text{ and } \alpha_i = \prod_{l=1}^{i} \frac{k-1-l}{k-1+ms-l}, 1 \leq i \leq k - 2. \]

The algorithm described in [13] is as follows:

(i) Compute \( \tilde{P}_0^j, 0 \leq j \leq k - 1 \) and \( \tilde{P}_l^j, 1 \leq l \leq s - 1, k - m_l \leq i \leq k - 1. \)

(ii) Compute \( \widetilde{Q}_0^j = \prod_{l=1}^{j} \frac{k-l}{k+ms-l} \tilde{P}_0^j, \quad 0 \leq j \leq k - 1. \)

(iii) Compute \( \widetilde{Q}_l^j = \prod_{l=1}^{j} \frac{k-l}{k+ms-l} \tilde{P}_l^j, 1 \leq l \leq s - 1, k - m_l \leq i \leq k - 1. \)

(iv) Compute \( \widetilde{Q}_l^{k-1} = \tilde{Q}_l^{k-1}, 1 \leq l \leq s - 1, 1 \leq i \leq m. \)

(v) Compute \( \widetilde{Q}_0^0. \)

Note that there exist other algorithms that expand the curve into a Bézier curve, then elevate the degree using Bernstein polynomials, and finally come back to a description using B-splines. For more details, see [14, 15]. The one given in [13] is more efficient and much more simple to implement. We can also use a more sophisticated version of this algorithm to insert new knots while elevating the degree.

3.3. Non-uniform rational B-splines

Let \( \omega = (\omega_i)_{1 \leq i \leq N} \) be a sequence of non-negative reals. The NURBS functions are defined by a projective transformation:

**Definition 3.4 (NURBS).** The \( i \)th NURBS of order \( k \), associated with the knot vector \( T \) and the weights \( \omega \), is defined by

\[ R_i^k = \frac{\omega_i N_i^k}{\sum_{j=1}^{N} \omega_j N_j^k}. \]

Note that when the weights are equal to 1 the NURBS are B-splines.

**Definition 3.5 (NURBS curve).** The NURBS curve of order \( k \) associated with the knot vector \( T \), the control points \( (P_i)_{1 \leq i \leq N} \) and the weights \( \omega \) is defined by

\[ M(t) = \sum_{i=1}^{N} R_i^k(t)P_i. \]
3.4. The multivariate tensor product non-uniform rational B-splines

As with splines, one can define the multivariate tensor product NURBS. For surfaces, we have the following definition.

**Definition 3.6 (NURBS surface).** The NURBS surface of order \( k \) associated with the knot vectors \( \{T^{(1)}, T^{(2)}\} \), the control points \( (P_{i,j})_{1 \leq i \leq N_1, 1 \leq j \leq N_2} \) and the weights \( \{(\omega^{(1)}, \omega^{(2)})\} \), is defined by

\[
M(t^{(1)}, t^{(2)}) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} R_{i,j}(t^{(1)}, t^{(2)}) P_{i,j},
\]

with \( R_{i,j}(t^{(1)}, t^{(2)}) = R^{(1)}_{i}(t^{(1)})R^{(2)}_{j}(t^{(2)}) \).

**Remark 3.7.** NURBS functions inherit most of the B-spline properties. We remark that in the interior of a knot span, all derivatives exist and are rational functions with a non-vanishing denominator.

We present here the definition of the perspective mapping. We construct the weighted control points \( P_i^{\omega} = (\omega_1x_i, \omega_1y_i, \omega_1z_i, \omega_1) \); then we define the B-spline curve in 4D space as

\[
M^{\omega}(t) = \sum_{i=1}^{N} N^k_i(t) P_i^{\omega}.
\]

For fundamental geometric operations on NURBS curves, we use the latest transformation and algorithms on B-spline curves.

**Remark 3.8.** NURBS functions allow us to model exactly many more domains than B-splines. In fact, all conics can be exactly represented with NURBS. For more details, see [15].

3.5. Modeling conics using non-uniform rational B-splines

In this section, we will show how to construct an arc of a conic using rational B-splines. Let us consider the following knot vector: \( T = \{000111\} \); the generated B-splines are Bernstein polynomials. The general form of a rational Bézier curve of degree 2 is

\[
C(t) = \frac{\omega_1 N^2_1(t) P_1 + \omega_2 N^2_2(t) P_2 + \omega_3 N^2_3(t) P_3}{\omega_1 N^2_1(t) + \omega_2 N^2_2(t) + \omega_3 N^2_3(t)}.
\]

Let us consider the case of \( \omega_1 = \omega_3 = 1 \). Because of the multiplicity of knots 0 and 1, the curve \( C \) links the control point \( P_1 \) to \( P_3 \). Depending on the value of \( \omega_2 \), we obtain different types of curves (table 1):

3.5.1. **Circle.** Using a 1D-patch. One can draw a circle using only nine control points, and the parameters \( N = 9, p = 2, T = \{000, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 111\} \). Control points and weights are given in table 2.

Using a 2D patch. We can also create a circle using a 2D patch with the following parameters: \( N_\xi = N_\eta = 3, p_\xi = p_\eta = 2, T_\xi = T_\eta = \{000, 111\} \). Control points and weights are given in table 3.

4. Isogeometric analysis

The idea behind the IGA method is to use the same functions that define the physical domain, to reach the solution of a partial differential equation. We will only treat the 2D case.

In the following, we consider two knot vectors \( T^{\xi} = \{\xi_1, \ldots, \xi_{N_1+p_1+1}\} \) and \( T^{\eta} = \{\eta_1, \ldots, \eta_{N_2+p_2+1}\} \). Let \( W^{\xi} = \{\omega^{\xi}_1, \ldots, \omega^{\xi}_{N_1}\} \) and \( W^{\eta} = \{\omega^{\eta}_1, \ldots, \omega^{\eta}_{N_2}\} \) be two weight sequences, and \( (P^{ij})_{1 \leq i \leq N_1, 1 \leq j \leq N_2} \) a sequence of control points. This defines a mapping

\[
F(\xi, \eta) = \sum_{1 \leq i \leq N_1, 1 \leq j \leq N_2} R^{\xi}_i(\xi) R^{\eta}_j(\eta) P^{ij}.
\]

\[
(4.5)
\]
that maps the rectangular patch \([\xi_1, \xi_{N_1}] \times [\eta_1, \eta_{N_2}]\) onto the physical domain \(\Omega\), where \(R^\xi\) and \(R^\eta\) are NURBS functions defined by knot vectors \(T^\xi\) and \(T^\eta\) and weights \(W^\xi\) and \(W^\eta\).

As mentioned before, we consider only open knot vectors. Without loss of generality, we shall consider knot vectors of the form

\[
\xi_1 = \ldots = \xi_{p_1+1} = \eta_1 = \ldots = \eta_{p_2+1} = 0
\]

and

\[
\xi_{N_1+1} = \ldots = \xi_{N_1+p_1+1} = \eta_{N_2+1} = \ldots = \eta_{N_2+p_2+1} = 1.
\]
Let $K$ be a cell in the physical domain. $Q$ is the parametric associated cell such that $K = F(Q)$. Let $J_F$ be the Jacobian of the transformation $F$, which maps any parametric domain point $(\xi, \eta)$ into the physical domain point $(x, y)$ (figure 4).

In figure 5, we give an example of the creation of a domain from a square.

With any function $v$ of $(x, y)$ we associate its representation in the parametric domain

$$\tilde{v}(\xi, \eta) := v \circ F(\xi, \eta) = v(x, y).$$

The basis functions, B-splines (NURBS), will not be affected by these changes, as they are invariant by affine transformations. For a point $(x, y)$ in the physical domain, let us denote

$$(x, y) = F(\xi, \eta), \quad x = \alpha(\xi, \eta) \text{ and } y = \beta(\xi, \eta);$$

then

$$\alpha_1 = \frac{\partial \alpha}{\partial \xi}, \quad \alpha_2 = \frac{\partial \alpha}{\partial \eta}, \quad \beta_1 = \frac{\partial \beta}{\partial \xi}, \quad \beta_2 = \frac{\partial \beta}{\partial \eta};$$

we have for the determinant of the Jacobian

$$\Delta = \det(J_F) = \alpha_1 \beta_2 - \alpha_2 \beta_1,$$

(4.6)

$$J_F = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{pmatrix},$$

(4.7)
Figure 6. Grid generation: (left) the coarsest mesh; (right) the domain after $h$-refinement. The minimal degree of basis functions, in this case, is 2.

$$J_F^{-1} = \frac{1}{\Delta} \begin{pmatrix} \beta_2 & -\alpha_2 \\ -\beta_1 & \alpha_1 \end{pmatrix}. \quad (4.8)$$

Let $u$ be a (scalar or vector) function defined on the physical domain. When we use the patch coordinates, we will write it as $\tilde{u}$, idem for the used spaces.

4.1. Refinement strategies

Refining the grid can be done in three different ways. This is the most interesting aspect of B-spline basis.

1. Using the patch parameter $h$, by inserting new knots. This is the $h$-refinement; it is the equivalent of mesh refinement of the classical finite element method.
2. Using the degree $p$, by elevating the B-spline degree. This is the $p$-refinement; it is the equivalent of using higher finite element order in the classical finite-element method (FEM).
3. Using the regularity of B-splines, by increasing/decreasing the multiplicity of inserted knots. This is the $k$-refinement. This new strategy does not have an equivalent in the classical FEM.

Evans et al [16] studied the $k$-refinement using the theory of the Kolmogorov $n$-widths. As we will see later, the use of this strategy can be more efficient than the classical $p$-refinement, as it reduces the dimension of the basis.

An active area of research is the study of local refinement. It is important to note that the use of tensor products leads to the existence of a number of superfluous control points that might exist because of this cartesian grid in the parametric domain. Sederberg et al [17] defined the notion of T-splines that allows us to reduce the number of those control points. In [18], Dörfel et al used T-splines for local $h$-refinement in IGA. Dokken, Kvamsdal and their team from SINTEF are currently developing another approach for local refinement, based on LR-splines [19]. In [20], the authors proposed the use of a hierarchical local refinement method.

4.2. Patch

The latest construction gives a coarse mesh; we can then use $h/p/k$ refinements to create the grid. We can also use multiple patches to describe more complex domains [2, 21]. There are many ways to stick those patches together.

4.3. Grid generation

For this purpose, we use alternately $h$- and $p$-refinement. The minimal degree of basis functions is imposed by the domain design, which will be given by the coarsest mesh. When inserting knots, we can use uniformly spaced knots or non-uniformly spaced ones. In figure 6, we give an example of such a refinement.
5. Application to 2D reduced magnetohydrodynamics

In this section, we treat a special case of the 2D reduced MHD problem, namely the current-hole problem. The general current-hole can problem \([1]\) can be written as

\[
\begin{align*}
\partial_t \psi &= (1 + \epsilon x)[\psi, \phi] + \eta (J - J_c), \\
\partial_t \omega &= 2\epsilon \frac{\partial \phi}{\partial y} + (1 + \epsilon x)[\omega, \phi] + \frac{1}{1 + \epsilon x}[\psi, J] + \nu \nabla^2 \omega,
\end{align*}
\]

(5.9)

where \([a, b] = \frac{\partial a}{\partial x_1} \frac{\partial b}{\partial x_2} - \frac{\partial a}{\partial x_2} \frac{\partial b}{\partial x_1}\), denotes the Poisson bracket of the functions \(a, b\).

In the following, we consider only the planar cylindrical geometry \((\epsilon = 0)\). The problem of the current hole can then be written as

\[
\begin{align*}
\partial_t \psi &= [\psi, \phi] + \eta (J - J_c), \\
\partial_t \omega &= [\omega, \phi] + [\psi, J] + \nu \nabla^2 \omega, \\
J &= \nabla^2 \psi, \\
\nabla^2 \phi &= \omega.
\end{align*}
\]

(5.10)

The current-hole problem is subject to the initial conditions

\[
\begin{align*}
J(0, x) &= J_c(x), \quad x \in \Omega, \\
\psi(0, x) &= \nabla^{-2} J_c, \quad x \in \Omega, \\
\phi(0, x) &= 0, \quad x \in \Omega, \\
\omega(0, x) &= 0, \quad x \in \Omega,
\end{align*}
\]

and the boundary conditions

\[
\begin{align*}
J(t, x) &= 0, \quad x \in \partial \Omega, \quad t \in [0, T], \\
\psi(t, x) &= 0, \quad x \in \partial \Omega, \quad t \in [0, T], \\
\phi(t, x) &= 0, \quad x \in \partial \Omega, \quad t \in [0, T], \\
\omega(t, x) &= 0, \quad x \in \partial \Omega, \quad t \in [0, T].
\end{align*}
\]

Before going to numerical simulations, let us recall some properties of the current-hole problem. See \([22, 23]\) for the proofs.

**Proposition 5.1 (Conservation).** If \(\nu = \eta = 0\), then all regular solutions of equation (5.10), verify

- **energy conservation:**
  \[
  \frac{d}{dt} \int_\Omega |\nabla \psi|^2 + |\nabla \phi|^2 \, d\Omega = 0.
  \]

- **magnetic helicity conservation:**
  \[
  \frac{d}{dt} \int_\Omega \psi \, d\Omega = 0.
  \]

- **cross-helicity conservation:**
  \[
  \frac{d}{dt} \int_\Omega \nabla \psi \cdot \nabla \phi \, d\Omega = -\frac{d}{dt} \int_\Omega \psi \phi \, d\Omega = 0.
  \]

As mentioned in \([22]\), when \(J_c = 0\) and \(\nu > 0\) and \(\eta > 0\), we have

\[
\frac{d}{dt} \int_\Omega |\nabla \psi|^2 + |\nabla \phi|^2 \, d\Omega \leq 0.
\]

In what follows, we present the time scheme and variational formulation that we used.
5.1. Time scheme

We use a semi-implicit time scheme

\[
\frac{\omega^{n+1} - \omega^n}{\Delta t} = [\omega^n, \phi^n] + [\psi^n, J^n] + \nu \nabla^2 \omega^{n+1},
\]

(5.11)

\[
\nabla^2 \phi^{n+1} = \omega^{n+1},
\]

(5.12)

\[
\frac{\psi^{n+1} - \psi^n}{\Delta t} = [\psi^n, \phi^{n+1}] + \eta \nabla^2 \psi^{n+1} - \eta J_c,
\]

(5.13)

\[
J^{n+1} = \nabla^2 \psi^{n+1}.
\]

(5.14)

This time scheme has the advantage of being very simple; however, it is of the order of 1.

5.2. Variational formulation

Now let us introduce the discrete space,

\[
\mathcal{V}_h = \text{span} (\varphi_b, b \in \Lambda^0),
\]

where the functions \(\varphi_b\) can be B-splines or more generally NURBS.

Multiplying equation (5.11) by \(\varphi_b\) and taking the integral over the whole domain, we obtain

\[
\int_{\Omega} \frac{\omega^{n+1} - \omega^n}{\Delta t} \varphi_b = \int_{\Omega} [\omega^n, \phi^n] \varphi_b + \int_{\Omega} [\psi^n, J^n] \varphi_b + \nu \int_{\Omega} \nabla^2 \omega^{n+1} \varphi_b.
\]

Now using Green’s formula and the boundary condition, we obtain

\[
\int_{\Omega} \frac{\omega^{n+1} - \omega^n}{\Delta t} \varphi_b = \int_{\Omega} [\omega^n, \phi^n] \varphi_b + \int_{\Omega} [\psi^n, J^n] \varphi_b - \nu \int_{\Omega} \nabla \omega^{n+1} \cdot \nabla \varphi_b,
\]

and thus

\[
\int_{\Omega} \omega^{n+1} \varphi_b + \nu \int_{\Omega} \nabla \omega^{n+1} \cdot \nabla \varphi_b = \int_{\Omega} \omega^n \varphi_b + \Delta t \int_{\Omega} [\omega^n, \phi^n] \varphi_b + \Delta t \int_{\Omega} [\psi^n, J^n] \varphi_b.
\]

Let us consider \(\omega_h \in \mathcal{V}_h\) an approximation of \(\omega\). We can expand \(\omega_h\) over the basis of \(\mathcal{V}_h\):

\[
\omega_h = \sum_{b' \in \Lambda^0} [\omega]_{b'} \varphi_{b'}.
\]

Figure 7. Current hole: the current density (left) and its profile (right).
Figure 8. Current hole: evolution of the current density as a function of time for $t = 2400, 3310, 3540, 3600, 3640, 3730, 3830, 3920$, using $\nu = 10^{-6}$ and $\eta = 10^{-5}$.

Therefore, the previous equation leads to the linear system,

$$A^0_\nu[\omega^{n+1}] = M^0[\omega^n] + \Delta t C[\psi^n] + \Delta t \gamma^0[J^n, \omega^n],$$

(5.15)

where we have introduced the matrices

$$M^0 = \left( \int_\Omega \varphi_b \varphi_{b'} \right)_{b, b' \in \Lambda^0},$$

(5.16)

$$S^0 = \left( \int_\Omega \nabla \varphi_b \cdot \nabla \varphi_{b'} \right)_{b, b' \in \Lambda^0},$$

(5.17)

$$A^0_\nu = M^0 + \Delta t \nu S^0,$$

(5.18)
Figure 9. Current hole: evolution of the current density profile as a function of time for $t = 2400, 3310, 3540, 3600, 3640, 3730, 3830, 3920$, using $v = 10^{-6}$ and $\eta = 10^{-5}$.

and $C_{\mathcal{V}_h}[u, v]$ is the $L^2$ contribution over $\mathcal{V}_h^0$ of the Poisson’s bracket $[u, v]$, i.e. a column vector where the element of each line $b$ is $\int_\Omega [u, v] \phi_b$.

For equation (5.12), we have for any $\phi_b \in \mathcal{V}_h^0$

$$\int_\Omega \nabla \phi^{n+1} \cdot \nabla \phi_b = - \int_\Omega \omega^{n+1} \phi_b.$$
which leads to the linear system
\[ S^0[\phi^{n+1}] = -M^0[\omega^{n+1}] \tag{5.19} \]

Now, let us go back to equation (5.13); we have for any \( \varphi_b \in \mathcal{V}_h^0 \)
\[ \frac{\int_{\Omega} \psi^{n+1} \varphi_b - \int_{\Omega} \psi^n \varphi_b}{\Delta t} = \int_{\Omega} [\psi^n, \phi^{n+1}] \varphi_b + \eta \int_{\Omega} \nabla^2 \psi^{n+1} \varphi_b - \eta \int_{\Omega} \mathcal{J}_c \varphi_b; \]
using Green’s formula and the boundary conditions, we obtain
\[ \frac{\int_{\Omega} \psi^{n+1} \varphi_b - \int_{\Omega} \psi^n \varphi_b}{\Delta t} = \int_{\Omega} [\psi^n, \phi^{n+1}] \varphi_b - \eta \int_{\Omega} \nabla \psi^{n+1} \cdot \nabla \varphi_b - \eta \int_{\Omega} \mathcal{J}_c \varphi_b. \]

Let us consider \( \psi_h \in \mathcal{V}_h^0 \) an approximation of \( \psi \). We can expand \( \psi_h \) over the basis of \( \mathcal{V}_h^0 \):
\[ \psi_h = \sum_{b' \in \Lambda^0} [\psi]_{b'} \varphi_{b'}. \]
which leads to
\[ A_h^0[\psi^{n+1}] = M^0[\psi^n] + \Delta t \mathcal{C}_h[\psi^n, \phi^{n+1}] - \eta \Delta t M^0[\mathcal{J}_c] \tag{5.20} \]
Finally, the discretization of the incompressible MHD can be written as

\[ A^0_0 = M^0 + \Delta t S^0. \]  
\[ S^0_0[\psi^{n+1}] = -M^0[J_c^{n+1}]. \]  
\[ A^0_0[\psi^{n+1}] = M^0[\psi^{n+1}] + \Delta t C_V^0[\psi^{n+1}, \psi^{n+1}] - \eta \Delta t M^0[J_c]. \]  
\[ M^0[J_c^{n+1}] = -S^0[\psi^{n+1}]. \]  

**Remark 5.2.** In the appendix, we show how one can perform a fast evaluation of the Poisson bracket on the basis of the tensor product property.

### 5.3. Numerical results

Following \[1, 22\], we consider the case of a circular domain of radius 1 centered at 0. For initialization, we take

\[ J_c = j_1(1 - R^4) - j_2(1 - R^2)^2, \]

with \( R^2 = x^2 + y^2 \), \( j_1 = 0.2 \) and \( j_2 = 0.266 \). As can be seen in figure 7, there is a negative current density close to the axis named the current hole. In order to start our simulations, we must take the Grad–Shafranov equilibrium based on this current as the initial condition. The following results were obtained for \( \nu = 10^{-6} \) and \( \eta = 10^{-5} \), using quadratic NURBS and a grid of \( 64 \times 64 \) meshes. The time step was \( \Delta t = 0.1 \). In figures 8 and 9, we plot the evolution of the current density and its profile; they are comparable to those obtained in [1, 22]. The current density, initially axisymmetric, is not affected during the linear stage while nonlinear effects are still negligible. Then, the profile begins to change: the current density is expelled outward from the central axis, generating a current sheet at the resonant surface. Then, the profile close to the center is flattened, leaving only residual fluctuations around \( J = 0 \). In figure 10, we plot the kinetic energy as a function of time.
Figure 13. Current hole: the growth rate $\gamma$ as a function of resistivity $\eta$, using quadratic NURBS, for the grids: the first line, (left) $32 \times 32$ and (right) $64 \times 64$; the second line, $128 \times 128$. The dashed line indicates the $\eta^1$ asymptote.

for $\nu = 10^{-5}$ and $\eta = 10^{-6}$. In figure 11, we show the magnetic, kinetic and total energies as functions of time. As one can see, the impact of the kinetic energy is small. In figure 12, we show the kinetic energy for long times.

In order to validate our simulations, we follow [1] and compute the growth rate $\gamma$ of the $m = 1$ mode during the linear stage. In figure 13, we plot the growth rate $\gamma$ as a function of resistivity $\eta$ for the $32 \times 32$ and $64 \times 64$ grids using quadratic NURBS. It is very important to note that our results are not as accurate as those presented in [1]. There are several reasons:

- **Time scheme**: our time scheme is of the order of 1.
- **Local h-refinement**: in our PyIGA code, we do not handle local refinement.

**Remark 5.3 (Remark on the instability).** The Soloviev equilibrium, which was the starting point of our simulation, is unstable for the current-hole problem. Because of the numerical noise of the simulation, the plasma will leave this equilibrium and start a linear stage and then enter a nonlinear one. Using IGA, we can exactly understand the cause of this numerical noise. As can be seen in figure 14, the singularity of our mapping at the four extremities of the patch will be propagated and will force the plasma to leave the equilibrium.

5.3.1. **Time stability.** In [10], we noted that the use of higher regular elements gives better CFL numbers. In table 4, we give the numerical CFL numbers for our semi-implicit time scheme.

In [22], the authors used a time step of $dt = 1.0$ using a grid with a space step $h = 0.05$. In our case, we get a CFL of 0.8 using $h = 0.03125$, which is of the same order even when we use a semi-implicit time scheme.
Figure 14. Current hole: the creation of instability. Evolution of $\omega$ from left to right and from top to bottom at $t = 0, 2, 4, 6, 8, 10, 22, 36$ and $160$.

Table 4. Numerical CFL numbers using the grid $32 \times 32$.

<table>
<thead>
<tr>
<th>$\eta$</th>
<th>$\nu$</th>
<th>CFL$^{\text{num}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-5}$</td>
<td>$10^{-6}$</td>
<td>0.8</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>$10^{-7}$</td>
<td>0.6</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>$10^{-8}$</td>
<td>0.55</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>$10^{-9}$</td>
<td>0.5</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>$10^{-10}$</td>
<td>0.45</td>
</tr>
</tbody>
</table>

6. Conclusions

In this paper, we have shown one application of the IGA to reduced MHD. In the general case, we must solve the equilibrium problem and, as suggested in [1], construct aligned meshes, before starting our simulations. In [24], we show one way of constructing aligned meshes using splines. A more efficient way to do so is still
under development and gives some interesting results. As we have noted in figure 13, we do not recover the
good slope. Two reasons are under investigation: our time scheme is semi-implicit and only of the order of 1;
the second reason is the impact of Jacobian singularities, which propagate during the simulation and lead to
a numerical diffusion that may be much more important than the resistivity $\eta$. Further investigations must be
done, especially on the use of different types of meshes.

All simulations were done using PyIGA, which is a 2D Fortran code interfaced with Python. The 3D
version will be available very soon and will be parallel and treat multi-patches.

Appendix

Let us show how we can implement a fast evaluation of the Poisson bracket, which must be performed at each
time step. We have

$$[\psi, J] = \partial_x \psi \partial_y J - \partial_y \psi \partial_x J = \frac{1}{\Delta} \left\{ \partial_x \tilde{\psi} \partial_y J - \partial_y \tilde{\psi} \partial_x J \right\} = \frac{1}{\Delta} [\tilde{\psi}, \tilde{J}].$$

Now we use the fact that $\psi = \sum_i [\psi]^i \varphi_i$, $J = \sum_i [J]^i \varphi_i$, with $i = (i_1, i_2)$, and obtain

$$[\psi, J] = \sum_{i,j} [\psi]^i [J]^j [\varphi_i, \varphi_j].$$

From the variational formulation, we need to compute the term

$$\int_{\Omega} [\psi, J] \varphi_k \, d\Omega = \sum_{i,j} [\psi]^i [J]^j \int_{\Omega} [\varphi_i, \varphi_j] \varphi_k \, d\Omega = \sum_{i,j} [\psi]^i [J]^j \int_{\mathcal{P}} [\tilde{\varphi}_i, \tilde{\varphi}_j] \tilde{\varphi}_k \, d\mathcal{P}$$

$$= \sum_{i,j} [\psi]^i [J]^j \int_{\mathcal{P}} (\tilde{\varphi}_{i_1} \tilde{\varphi}_{j_2} - \tilde{\varphi}_{i_2} \tilde{\varphi}_{j_1}) \tilde{\varphi}_k \, d\mathcal{P}.$$

Note that the Jacobian in the bracket transformation cancels the Jacobian in the integral change of variables.
This can be written as

$$\int_{\mathcal{P}} (\tilde{\varphi}_{i_1} \tilde{\varphi}_{j_2} - \tilde{\varphi}_{i_2} \tilde{\varphi}_{j_1}) \tilde{\varphi}_k \, d\mathcal{P} = A_{i_1,j_1,k_1} B_{i_2,j_2,k_2} - B_{j_1,i_1,k_1} A_{j_2,i_2,k_2},$$

where

$$A_{i_1,j_1,k_1} = \int_0^1 \tilde{\varphi}_{i_1} \tilde{\varphi}_{j_1} \tilde{\varphi}_{k_1} \, d\xi, \quad B_{i_2,j_2,k_2} = \int_0^1 \tilde{\varphi}_{i_2} \tilde{\varphi}_{j_2} \tilde{\varphi}_{k_2} \, d\eta.$$

The $A$ and $B$ terms (that are 1D terms) can be computed and stored once for all. This leads to the following
algorithm.

(i) Loop on $k$.

(ii) For each $k$, get the corresponding support and elements.

(iii) Loop on non-vanishing basis functions on these elements:

→ loop on $i$ and $j$.

(iv) Add the contribution of $A_{i_1,j_1,k_1} B_{i_2,j_2,k_2} - B_{j_1,i_1,k_1} A_{j_2,i_2,k_2}$. 

References