Applications of Isogeometric Analysis Coupled with Finite Volume Method

Carlos Pantaleón
Applications of Isogeometric Analysis
Coupled with Finite Volume Method
Applications of Isogeometric Analysis Coupled with Finite Volume Method

By

Carlos Pantaleón

A Thesis Presented in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering

Approved by:

Dr. Amitabha Ghosh
Department of Mechanical Engineering
(Thesis Advisor)

Dr. Alexander Liberson
Department of Mechanical Engineering

Dr. Stephen Boedo
Department of Mechanical Engineering

Dr. Agamemnon Crassidis
Department of Mechanical Engineering
(Department Representative)

Rochester Institute of Technology
Kate Gleason College of Engineering
Department of Mechanical Engineering
Rochester, NY
July 14th, 2014
Abstract

Computer-aided design (CAD) and computer-aided analysis (CAA) play a vital role in the optimization of the engineering design cycle by generating models that can be easily modified, shared and subjected to simulations of several types. However, translating the CAD files to analysis-suitable geometry is a very time-consuming process and precision is often compromised. This is due to the adoption of different geometric approaches: non-uniform rational basis splines (NURBS) are commonly used to generate geometry in CAD, while interpolating polynomials prevail in finite element analysis (FEA) — used extensively in CAA.

Isogeometric analysis (IGA), introduced by T. Hughes in 2005, uses NURBS directly in FEA to accurately model the geometry and the field variable to be computed. This novel approach has gained an outstanding reception from the scientific community and many applications have been tested, including linear elasticity, vibrations and wave propagation, time dependent problems, non-linear problems, nearly incompressible solids, and fluid dynamics.

To overcome some of the challenges found in IGA, variants and combinations have been developed. In this thesis, one of the variants found in the literature is explored: a finite volume method (FVM) working on NURBS geometries, with applications in fluid flow, heat transfer, and shape optimization. An IGA framework supplemented with FVM is created in MATLAB® to solve problems defined over single patch domains with mesh refinement by node insertion. Additionally, a second-order finite difference method is developed using non-orthogonal curvilinear coordinates and a numerical Jacobian of the NURBS parameterization. The benchmarks and test cases include fully developed laminar flows through ducts, transient heat conduction, linear advection-diffusion, potential flow around a tilted ellipse, and a brief example of shape optimization using a particle swarm technique. The numerical solutions are compared among the methods and subsequently verified using available analytical solutions.

In general, IGA is desirable when dealing with NURBS domains since the solution is given as a function in terms of the basis, whereas FVM and FDM yield a discrete field of the solution. Nonetheless, in examples where a quick solution is required, or multiple iterations need to be simulated (such as shape optimization) the selection of the latter two methods is appropriate.
Acknowledgements

I need to offer my deepest gratitude to my advisor, Dr. Amitabha Ghosh, for being an amazing professor and for having enough patience and confidence to take the risk of working with an exciting subject that was new to both of us. Furthermore, as a result of his recommendation, the mechanical engineering department at RIT, through Dr. Edward Hensel, supported me during the thesis stage with a Kelly Fellowship, and I extremely appreciate that.

Although indirectly, every professor who took the time and made the effort of sharing knowledge with me has influenced this work, including Dr. Kathleen Lamkin-Kennard, Dr. Satish Kandlikar, Dr. Marca Lam and Dr. Steven Day. I am tremendously grateful for this contribution, especially to the committee members, Dr. Alexander Liberson and Dr. Stephen Boedo, for evaluating this Master's Thesis and to Dr. Agamemnon Crassidis for his guidance.

I also thank the mechanical engineering department staff, particularly Jill Ehmann, Diane Selleck and Venessa Mitchell, for their enthusiasm to help with every detail.

My graduate studies at RIT would not have been possible without the full financial support from the Ministry of Higher Education, Science and Technology (MESCyT) of the Dominican Republic. It is my intention to amply repay my country for this great opportunity.

This experience was much more pleasant thanks to my friends in Rochester, such as Pedro Domínguez, Siddharth Dasgupta and Zhen Yin; but also to those overseas, in particular to Nelly Almonte for sharing every joyful moment and offering encouragement when most needed.

Lastly, I am infinitely blessed for the love and support of my family, Carmen Matías and her kids, as well as my parents, José Pantaleón and Filonidia Pérez, to whom I owe my entire life.
Table of Contents

Abstract .................................................................................................................................................. I
Acknowledgements ................................................................................................................................. II
Table of Contents ..................................................................................................................................... III

List of Figures ........................................................................................................................................... V
List of Tables ............................................................................................................................................. VII
List of Symbols ......................................................................................................................................... VII

1. Introduction .......................................................................................................................................... 1
   Objectives ............................................................................................................................................. 1
   Motivation ............................................................................................................................................ 1
   Scope .................................................................................................................................................. 2
   Timeline ............................................................................................................................................. 3

2. Literature Review .................................................................................................................................. 4
   Computer-Aided Engineering ................................................................................................................. 4
   Geometry Representation in Finite Element Analysis ............................................................................... 5
   Geometry Representation in Computer-Aided Design ............................................................................. 7
   Isogeometric Analysis ........................................................................................................................... 9
   Instructional applications of IGA ........................................................................................................ 10
   IGA Variants and Combinations .......................................................................................................... 11
   Fluid-Structure Interaction and Shape Optimization in IGA ............................................................... 13

3. Geometry Parameterization ............................................................................................................... 15
   Non-Uniform Rational Basis Spline ...................................................................................................... 15
   Curvilinear coordinates and transformations ...................................................................................... 20

4. Governing Equations and Boundary Conditions .................................................................................. 23
   Elastostatics ......................................................................................................................................... 24
   Poisson's Equation ............................................................................................................................... 25
   Heat Equation ..................................................................................................................................... 26
   Linear Advection Diffusion ................................................................................................................ 26

5. Discretization and Optimization Methods ............................................................................................ 27
   Isogeometric Analysis ......................................................................................................................... 27
   Finite Volume Method on NURBS ...................................................................................................... 33
   Finite Difference Method on NURBS .................................................................................................. 38
   Rothe's Method ................................................................................................................................. 42

IIII
Particle Swarm Optimization ................................................................. 43
6. Applications .......................................................................................... 47
   Plate with a Hole (under in-plane tension) ............................................. 47
   Fully Developed Flows through Ducts .................................................... 48
   Potential flow around a tilted ellipse ...................................................... 50
   Transient Heat Conduction .................................................................... 52
   Linear Advection-Diffusion ..................................................................... 53
   Shape Optimization .................................................................................. 54
7. Results and Discussion .......................................................................... 56
   Fully Developed Flows through Ducts .................................................... 59
   Potential Flow around a tilted ellipse ...................................................... 66
   Transient Heat Conduction .................................................................... 71
   Linear Advection-Diffusion ..................................................................... 73
   Shape optimization .................................................................................. 76
8. Conclusions and Recommendations ..................................................... 79
   Conclusions ........................................................................................... 79
   Limitations of Current Work .................................................................... 80
   Recommended Future Work ..................................................................... 81
References ................................................................................................. 82
Appendices ................................................................................................. 85
   Appendix A: Geometry Data ................................................................. 85
   Appendix B: MATLAB Codes ................................................................. 86
   Appendix C: Code Dependencies ............................................................ 110
   Appendix D: Numerical Jacobian ............................................................. 115
   Appendix E: Laplacian Equation patch test ............................................ 116
List of Figures

Figure 2-1: Diagram of the steps in the Engineering Design Cycle ............................................. 4
Figure 2-2: Example of linear triangular mesh adaptability to complex geometries .................. 6
Figure 2-3: Family of rectangular serendipity elements ................................................................. 6
Figure 2-4: A Bézier curve and its control points .............................................................................. 7
Figure 2-5: Quadratic Bernstein basis functions .............................................................................. 7
Figure 2-6: A B-spline and its control points .................................................................................... 7
Figure 2-7: Quadratic B-splines basis example ................................................................................... 7
Figure 3-1: Example of a bivariate NURBS surface (a) and its control net (b) ..................... 16
Figure 3-2: Examples of consecutive h-refinements (left to right) ............................................ 17
Figure 3-3: Verification of the findspan routine .............................................................................. 18
Figure 3-4: Verification of the basis routine using a bivariate function ....................................... 18
Figure 3-5: Examples of NURBS basis and its parametric derivatives ........................................ 19
Figure 3-6: Jacobian determinant of a circle using NURBS ....................................................... 21
Figure 3-7: Computed outward unit normal vectors for different shapes ................................. 22
Figure 3-8: Computed element centroids for different domains .................................................. 22
Figure 5-1: Flowchart of typical FEA code and required modifications for IGA .................... 32
Figure 5-2: Control volume and its neighbors in the parametric domain .................................. 35
Figure 5-3: Variable step one-sided second order approximation ............................................ 36
Figure 5-4: Example of advected values in Donor Cell scheme ................................................ 37
Figure 5-5: The von Neumann neighborhood as a subset of Moore neighborhood .............. 40
Figure 5-6: Extrapolation method used in degenerated boundaries ........................................ 41
Figure 5-7: PSO benchmark results (ID:1-4) ............................................................................... 45
Figure 5-8: PSO benchmark results (ID:5-8) ............................................................................... 46
Figure 6-1: Elastic plate with a circular hole: geometry and BCs .............................................. 47
Figure 6-2: Potential flow around a cylinder: geometry and BCs .............................................. 50
Figure 6-3: Transient heat conduction: geometry and BCs ........................................................ 52
Figure 6-4: Linear advection-diffusion: geometry and BCs ......................................................... 53
Figure 6-5: Control points and NURBS geometry of the initial iteration ................................ 54
Figure 7-1: Initial control points and mesh for the plate with a hole benchmark .................. 56
Figure 7-2: Refined mesh for the plate with a hole benchmark ............................................... 56
Figure 7-3: Horizontal displacement in the plate with a hole benchmark ............................... 57
Figure 7-4: Vertical displacement in the plate with a hole benchmark ........................................57
Figure 7-5: Verification of the results in the plate with a hole benchmark ................................58
Figure 7-6: Refined Meshes for the flows through ducts ...........................................................59
Figure 7-7: Results of axial velocity in an eccentric annular duct using IGA ...............................59
Figure 7-8: Qualitative view of the numerical results of the flows through ducts .......................60
Figure 7-9: IGA mesh dependency study for the eccentric annulus example, p=1 .......................61
Figure 7-10: Convergence of $L^2$ norm error for eccentric annulus using IGA ..........................62
Figure 7-11: Error comparison in parametric domain for example b ........................................64
Figure 7-12: Comparison of methods in concentric annulus problem .......................................65
Figure 7-13: Comparison of methods in eccentric annulus problem .........................................65
Figure 7-14: Meshes used for potential flow around a cylinder ..................................................66
Figure 7-15: Velocity potential lines for the potential flow around a cylinder ..............................67
Figure 7-16: Horizontal velocity of potential flow around a cylinder .........................................67
Figure 7-17: Vertical velocity of potential flow around a cylinder .............................................68
Figure 7-18: Pressure distribution of potential flow around a cylinder .......................................68
Figure 7-19: Pressure distribution of potential flow around an ellipse .......................................68
Figure 7-20: Mesh used in the tilted ellipse example ...................................................................69
Figure 7-21: Velocity potential lines of the tilted ellipse example ............................................69
Figure 7-22: Velocity components of the tilted ellipse example .................................................70
Figure 7-23: Pressure distribution comparison in an ellipse and a tilted ellipse ............................70
Figure 7-24: Temperature contour plot of the transient heat conduction example .....................71
Figure 7-25: Comparison of temperature profile at mean radius using IGA ...............................72
Figure 7-26: Comparison of temperature profile at mean radius using FVM ...............................72
Figure 7-27: Diffusion-dominated vs Advection-dominated in central difference FVM ............73
Figure 7-28: Diffusion-dominated vs Advection-dominated in donor cell FVM ...........................73
Figure 7-29: Diffusion-dominated vs Advection-dominated in 1st order upwind FDM ................74
Figure 7-30: Advection-dominated example using 1st order upwind FDM .................................74
Figure 7-31: Advection-dominated example using donor cell FVM ..........................................75
Figure 7-32: Advection-dominated example using 2nd order upwind FDM ...............................75
Figure 7-33: Initial iteration for the shape optimization example ...............................................76
Figure 7-34: Particle evaluations inside the design variable space ............................................76
Figure 7-35: Convergence history for PSO shape optimization ....................................................77
Figure 7-36: Control points and geometry for optimal design vs an exact quarter circle ..........78
List of Tables

Table 1-1: Thesis Timeline and Deliverables ................................................................. 3
Table 5-1: PSO Benchmark results .................................................................................. 44
Table 6-1: Parameters used for elastic plate with a circular hole .................................. 48
Table 6-2: Dimensions used for flows through ducts ...................................................... 49
Table 6-3: Parameters used for potential flow example ................................................... 51
Table 6-4: Parameters used for heat equation example ................................................... 52
Table 6-5: Parameters used for the shape optimization example ..................................... 55
Table 7-1: IGA mesh dependency study for the eccentric annulus example, p=2 ............ 61
Table 7-2: IGA mesh dependency summary for flows through ducts. ............................. 62
Table 7-3: FVM mesh dependency summary for flows through ducts ............................ 63
Table 7-4: FDM mesh dependency summary for flows through ducts ............................ 63
Table 7-5: Iteration history for PSO shape optimization ................................................ 77

List of Symbols

RIT – Rochester Institute of Technology
CAD – Computer-Aided Design
CAA – Computer-Aided Analysis
NURBS – Non-Uniform Rational Basis Spline
FEA – Finite Element Analysis
IGA – Isogeometric Analysis
FVM – Finite Volume Method
PDE – Partial Differential Equation
BC – Boundary Condition
$(x, y)$ – Cartesian coordinates
$(\xi, \eta)$ – parametric coordinates
$(i, j)$ – parametric indices
$N(\xi), M(\eta)$ – B-spline basis functions
$N'(\xi), M'(\eta)$ – B-spline basis parametric derivative
$p, q$ – polynomial orders
\( n, m \) – number of basis functions

\( w \) – weight value, or weighting function

\( B \) – control point, or basis derivative matrix

\( C(\xi) \) – NURBS curve

\( S(\xi, \eta) \) – NURBS surface

\( R(\xi, \eta) \) – NURBS basis

\( \Xi, H \) – knot vectors

\( \overline{\xi} \) – inserted knot

\( \overline{B} \) – inserted control point

\( W \) – basis denominator

\( W' \) – basis denominator parametric derivative

\( J \) – Jacobian matrix

\( J \) – Jacobian determinant

\( g \) – Covariant metric tensor

\( g^{-1} \) – Contravariant metric tensor

\( g \) – Covariant metric tensor determinant

\( n \) – outward unit normal vector

\( ds \) – differential curve segment

\( \Omega \) – domain

\( \Gamma \) – boundary

\((u, \nu)\) – displacement field

\( \sigma_x, \sigma_y, \tau_{xy} \) – stress components

\( \epsilon_x, \epsilon_y, \gamma_{xy} \) – strain components

\( E \) – Young’s Modulus

\( \nu \) – Poisson’s ratio

\( u \) – generic field variable, trial solution

\( f(x, y) \) – source term

\( V \) – velocity field

\( \alpha \) – thermal diffusivity, element Péclet number
D – diffusion coefficient
$a(\cdot, \cdot)$ – bilinear operator
$L(\cdot)$ – linear operator
$u^h$ – approximated solution
$v^h$ – homogeneous solution
$g^h$ – lifting solution
d – displacement vector
F – global load vector
K – global stiffness matrix
k – global basis function index
D – elasticity matrix
t – traction vector
$a$ – coefficient of linear system matrix
$b$ – coefficient of linear system vector
M – mass matrix
$\Delta t$ – time step
p – personal best position
g – global best position
$\Phi$ – pseudo objective function
$\phi$ – velocity potential
$Q$ – volumetric flow rate
$p$ – pressure
$\rho$ – density
$(V_r, V_\theta)$ – velocity components in polar coordinates
$V_t$ – tangential velocity
$U_\infty$ – velocity at far field
1. Introduction

Objectives

The primary goal of this thesis is to explore a combination of isogeometric analysis and finite volume method on geometries parameterized by non-uniform rational basis splines, by means of comparing the results from different applications, including fluid flow and heat transfer.

Embedded within this aim, the following specific objectives are suggested:

- Design and develop a MATLAB® implementation of a basic isogeometric analysis framework using the guidelines provided in the literature and the instructional codes.
- Supplement the prior framework with a finite volume method extended to non-uniform rational basis splines geometries based on the theoretical platform already available.
- Set up and execute an extensive selection of benchmarks and test cases to verify the proper performance of the code as well as compare the solutions of this approach to isogeometric analysis and other numerical methods.

Motivation

Isogeometric analysis offers a possible solution to a progressively growing obstacle in the integration of geometric design and computational analysis in the engineering design cycle. A significant measure of its relevance is given by the vast amount of related research that has been generated since its inception.

A specific interest in the combination of isogeometric analysis with finite volume method was catalyzed by the suggested extensions found in the literature review (along with the lack of test cases) that truly presented an opportunity for a side-by-side comparison of the capabilities of the finite volume method with isogeometric analysis for different test cases.
Scope

The considered problems are defined over single patch geometries enriched with refinement by node insertion. Benchmarks and test cases will include a linear elasticity problem, fully developed laminar flows through ducts, transient heat conduction, linear advection-diffusion (with discontinuities), potential flow around a tilted ellipse, and a brief example of shape optimization.

The observed tendency in the literature review of developing instructional software whose primary feature is readability (not speed) will be adopted. This approach supports the formation of a strong interest in this relatively new subject while avoiding the additional work required to find and implement efficient algorithms for certain subroutines or to compile low-level programming into MATLAB®. Thus, a much clearer view of the methodology is obtained.
## Timeline

**Table 1-1: Thesis Timeline and Deliverables.**

<table>
<thead>
<tr>
<th></th>
<th>SUMMER (2124)</th>
<th>FALL SEMESTER (2131)</th>
<th>INTER (2133)</th>
<th>SPRING SEMESTER (2135)</th>
<th>SUMMER (2138)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preliminary study</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced FEA concepts, IGA and basics of computational geometry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples using instructional codes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preliminary documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IGA subroutine development</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traditional FEA: quadrature, local and global assembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NURBS: basis and derivatives, connectivity mapping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main routine and debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mesh refinement by knot insertion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation update I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Benchmarking</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear elasticity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow through ducts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation update II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Test cases development I</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FVM on NURBS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potential flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transient Heat Conduction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Test cases development II</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shape optimization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conference Presentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDM on NURBS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Advection-Diffusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Final documentation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Literature Review

**Computer-Aided Engineering**

Computer-aided engineering (CAE) has greatly improved the productivity, quality and documentation of the engineering design cycle (see Fig. 2-1). Computer models can be easily modified and information can be shared between each of the steps of the product development process. Furthermore, these models can be subjected to several computational analyses of various natures such as structural mechanics, fluid mechanics, heat transfer and electromagnetics.

Although they are strictly linked, individual fields of CAE such as computer-aided design (CAD) and computer-aided analysis (CAA) were first developed in different times and circumstances. For this reason, some differences in their approaches decelerate their integration and thus generate bottlenecks in complex engineering designs. One such problem is the use of different geometric approaches in CAD and CAA technologies.

![Diagram of the Steps in the Engineering Design Cycle](image)

*Figure 2-1: Diagram of the Steps in the Engineering Design Cycle [1].*
These fields have progressively been improved over the years, yet their geometric bases are still linked to their respective roots. For instance, simulation software (ANSYS, COMSOL Multiphysics, etc.) extensively use finite element analysis (FEA) with piecewise polynomial approximations, while most CAD software (SolidWorks, AutoCAD, Rhinoceros, etc.) use non-uniform rational basis splines (NURBS) to represent the geometry. The basic difference is that the first is an interpolating function such as Lagrange and Hermite polynomials that fits the curve to a discrete set of points whereas the latter approach is defined by a set of control points that determine the shape of the curve and a knot vector that defines how the control points affect the curve. Further explanation of these topics and the problems that arise from their difference will be discussed in more detail in later sections.

Early efforts to unify the computational geometry approaches included mechanical-based principles [2] as well as using splines directly in FEA [3]. But a breakthrough concept of isogeometric analysis (IGA) was led by Thomas Hughes in 2004, which extended classical FEA using NURBS to generate basis functions. This method gained an outstanding reception from the scientific community as can be quantified by the number of citations of the original paper [4].

**Geometry Representation in Finite Element Analysis**

The general approach in FEA is to divide the domain in a set of elements with simple shapes and apply a variational formulation to determine a set of element equations which are then assembled to a global system of equations where the field variable is to be solved. Typically, the discretization uses interpolating functions as basis of the solution, such as Lagrange and Hermite polynomials. The discussions that are presented hereafter are mainly limited to the element types and their differences; for a comprehensive coverage of other aspects of FEA, refer to Reddy [5].

One of the earliest type of elements, which is still very popular today, is the linear triangle element (see Fig. 2-2) introduced by R. Courant in 1943. The main advantage of this approach over the linear quadrilateral elements is its adaptability to complex geometries. Also, the automation of the mesh generation process was possible by using Delaunay triangulation [6]. Thus, Courant's contribution added substantial momentum to the development of FEA.
By 1960's, several scholars had developed to a fairly complete level the underlying theory of FEA. Although rectangular elements were already created and usually obtained better accuracy than triangular elements, the rectangular shape was limited for applications with simple geometries. Related to this, the most important contribution to FEA was the concept of isoparametric elements, introduced by B. Irons in 1968, that allowed curved boundaries by representing the geometry of the element with the same interpolating function used for the field variable. This process maps a square (or cube) in local coordinates (parent element) into a quadrilateral (or hexahedral) element in the physical domain that could have curved boundaries, thus allowing more flexibility in the geometry.

More recently, numerous improvements have been introduced and adopted by commercial packages. Some examples are: h-refinement (subdivision of elements), p-refinement (increase the order of interpolating polynomial) and "serendipity" elements (shown in Fig. 2-3) that are created by adding nodes to existing elements to increase accuracy. Adaptive finite element techniques are thoroughly explained in Flaherty [8]. A fairly broad exposure of serendipity elements can be obtained from Arnold and Awanou [9].

Figure 2-3: Family of rectangular serendipity elements.
Geometry Representation in Computer-Aided Design

Geometry representation in CAD has been continuously improved since its origin back in 1960's when two French engineers working in the automotive design area, Pierre Bézier and Paul de Casteljau, simultaneously developed a parametric curve, namely Bézier curve (see Fig. 2-4), based on the Bernstein polynomial (see Fig. 2-5). This invention allowed designers to draw smooth curves on a computer instead of using strips of wood or metal with fixed edges and loaded in control points.

In 1972, a generalization of the Bézier curve, known as B-spline (see Fig. 2-6) was introduced by the doctoral work of R. Reisenfeld, allowing users to design free-form curves and surfaces on a computer. About two years later, non-uniform rational basis splines (NURBS) were introduced by K. Versprille as another significant improvement that can accurately handle both analytic and modeled curves. For details on the evolution of splines, refer to Farin [10] and Bertka [11].
Nowadays, NURBS are used in most computer graphics applications, and specifically in CAE. Renowned industry standards such as Initial Graphics Exchange Specification (IGES), Standard for the Exchange of Product model data (STEP) and ACIS use this technology as well.

As some kind of analogy to the 'parent element' in FEA geometry representation, splines utilize a 'parameter space' which is local to patches, instead of elements. The knots divide the parameter space into elements. The knot vector is a set of non-decreasing parametric coordinates:

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$$  \hspace{1cm} (2-1)

where $p$ is the spline order, and $n$ is the number of basis functions used to construct the curve. Basis functions (see Fig. 2.7) for B-splines can be found by de Boor's algorithm [12]:

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(\xi) = \frac{\xi - \xi_j}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \hspace{1cm} \forall \ p > 0$$  \hspace{1cm} (2-2)

Only for $p < 2$ these shape functions are the same as the piecewise functions used in FEA.

The index $i$ denotes the control point number. In NURBS, each control point $B_i = (x_i, y_i, z_i)$ has a weight value $w_i$ associated with it that determines how close the curve is to the control point. The general form of the NURBS curve with $k$ control points is stated as follows:

$$C(\xi) = \frac{\sum_{i=1}^{k} N_{i,p}(\xi) w_i B_i}{\sum_{i=1}^{k} N_{i,p}(\xi) w_i}$$  \hspace{1cm} (2-3)

Extensions to surfaces and solids can be obtained by the tensor product of NURBS curves and using control nets or lattices instead of a control points. Furthermore, refinement techniques have already been implemented, such as knot insertion (similar to h-refinement), order elevation (p-refinement) and a recently presented generalization of NURBS called T-splines [13] that allows more grid control using T-junctions. For a more complete coverage of NURBS fundamentals, as well as a historical perspective, refer to Rogers [14].
Isogeometric Analysis

The procedure of translating CAD files into analysis-suitable geometry that can be meshed and entered in a FEA program wastes a huge amount of time from the design process. Furthermore, conventional FEA meshing does not reproduce the "exact" geometry from the CAD, but instead it is only an approximation that can create errors when the solution is very sensitive to geometry disturbance, such as shell buckling, boundary layers and sliding contact [15].

This gap between FEA and CAD represents a severe handicap in the engineering design cycle, especially with complex designs where numerous parts, materials and configurations interact. This problem is still present in industry, although mesh generation and refinement has improved year after year and CAD geometry representation now has a very strong mathematical foundation.

An early attempt [16] to bridge this gap was made by K. Höllig in 2003, using a weighted approximation technique by combining the computational advantages of B-splines and standard finite elements without requiring grid generation. Examples of typical boundary value problems in fluid flow, heat conduction, and elasticity were presented. Despite the advantages of a simple data structure, compatibility with CAD software and algorithm parallelization, the integration over irregularly shaped boundary cells presents a serious drawback for its implementation.

By 2005, T. Hughes et al [4] proposed the concept of isogeometric analysis (IGA). Basis functions generated from NURBS are used to construct an exact geometric model and an isoparametric concept is applied to define field variables for FEA. The basis is refined and its order is elevated without geometry alteration. Linear elastic problems showed that rigid body motions as well as constant strain states are represented precisely. Advection–diffusion with steep internal boundary layers converged towards monotone solutions with a newly created k-refinement strategy. Interesting features of NURBS are viewed in comparison to Lagrangian polynomials, especially when treating discontinuities. If the order is increased, the data fit becomes smoother instead of oscillatory. Several challenges were presented, such as local refinement, volumetric mesh generation and trimmed NURBS manipulation. However, the results obtained from IGA promise a future where CAD and FEA are closer together and a huge amount of related publications have been made.
**Instructional applications of IGA**

A noteworthy detail is that IGA had been adopted earlier for merely instructional applications in the form of tutorials, high-level implementations and free research tools. Four examples are:

**ISOGAT**, a 2D MATLAB® tutorial code for elliptic diffusion-type problems on single patch geometries using IGA, developed by Vuong et al, 2010 [17]. The basic steps of IGA are explained along with a couple of linear examples in a user-friendly interface. The interactive code has a very simple structure in order to help understand the IGA algorithm. An analogy, as well as the differences, between this approach and the traditional FEA is shown with emphasis.

**MIGFEM**, a more advanced MATLAB® implementation, presented by Nguyen et al in 2012 [18]. This program is intended to be used by students or researchers interested in IGA, especially in solid mechanics. It has examples in linear elasticity and fracture mechanics as well as Poisson problems. Both 2D and 3D geometries are allowed and FEA is enriched by the partition of unity method to allow discontinuities such as fractures. In the code, IGA pros and cons are shown.

**GeoPDEs**, started by de Falco et al in 2011 [19] is a suite of software tools for applications of IGA, developed in MATLAB®. It is simple enough to allow users to get acquainted with the practical issues related to IGA while its flexibility permits experienced researchers to implement new methods in a fairly direct manner for rapid prototyping and testing. In its documentation, the basic features and capabilities of the software are presented in a Poisson problem, showing how to use the code as a starting point to develop methods with applications such as elasticity, Stokes, and Maxwell problems. Other subjects discussed are: boundary condition imposition, multipatch geometries, refinement (h, p and k), different quadrature strategies, non-isoparametric methods, T-splines implementation, etc.

**PetIGA**, a framework for high performance IGA, was initiated by Collier in 2011 [20]. Based on the Portable, Extensible Toolkit for Scientific Computation (PETSc), an open source project was written in C, planned for researchers in the numerical solution of PDEs who have applications which require extensive computational resources. It is applicable to a wide range of problem sizes, including large-scale simulations with high performance parallel computing.
IGA Variants and Combinations

Another important aspect to highlight about IGA literature is its ability to be modified by introducing certain variations to overcome some of the known drawbacks of this method or to obtain better results for specific applications. Two contributions of this type are presented:

Anisogeometric Analysis, a variant of NURBS enhanced FEA, was proposed by D. Ernens in 2011 [21]. The main objective is to experiment with getting apart from the isoparametric concept adopted in IGA. Obtained results demonstrate that the use of superparametric elements lead to a more inexpensive computation on high degree geometry spaces, while a subparametric approach failed due to a different continuity of the solution and the geometry space. It was also shown that although the solution obtained from IGA has a non-linear transformation error, this does not produce a negative effect since the higher continuity of the basis compensates for this error.

The IGA variation with the most expected potential is linked to a relatively new technology, previously mentioned, that offers several advantages over NURBS geometry when it comes to IGA: T-splines. Due to its highly desirable features in a design context, T-spline technology was acquired by Autodesk in 2011 and several add-ons for modeling software have been created. Bazilevs et al, 2010, [22] introduced IGA with T-splines. It performed well under extensive applications of elementary problems in structural mechanics and fluid dynamics for bivariate (2D) and trivariate (3D) cases. Although there are still some challenges regarding model generation for three dimensional analysis, numerous advantages were observed in comparison to IGA based on NURBS. Besides allowing local refinement and watertight merging of patches (useful for 3D models), T-splines finally offer a solution for the trimmed surface problem. According to Scott et al, 2010, [23] this generalization only requires the modification of the shape function subroutine of the IGA code. Also, the FEA assembly time is reduced considerably, as well as the complexity of parallelizing the IGA code.

In a similar fashion to the aforementioned variations of IGA, combinations of this method with already available techniques have also been documented with successful results.

A combination of features from IGA and weighted FEA was coined as weighted isogeometric finite elements by Höllig et al, 2012 [24]. This arrangement is preferred for special cases such as
free boundary problems. The weighted approach is favored for constructive solid geometry models and IGA is suggested for non-trimmed NURBS geometry models.

In his Master's thesis, R. Hiemstra, 2011, [25] presented an interesting combination of IGA with a mimetic discretization method. The latter is a fusion of concepts of FEA and finite volume method (FVM). The resulting scheme, using B-splines, resembles a FVM on a staggered grid for the representation of the conservation laws and FEA for the representation of the constitutive equations. It allows a decomposition of the field variable within the global topology. This characteristic was exploited to numerically determine the lift produced by an irrotational, incompressible flow over an airfoil. It was observed that accurate results were obtained even with quite coarse meshes and that this approach leads to stable and consistent estimations of the lift, even under mesh distortion on the trailing edge.

With the main objective of tackling geometric models that are based on constructive solid geometry, such as trimmed NURBS surfaces (not fully supported by IGA), Rank et al, 2012, [26] suggested directly coupling IGA with a finite cell method. In this scheme, a high-order fictitious domain is discretized with a simple meshing for complex domains using a structured grid of cells. This approach eliminates the need to generate a body-fitting finite element mesh, while preserving exponential rates of convergence for smooth problems. The accuracy and efficiency of this method is illustrated with a numerical benchmark as well as a modal analysis of a bicycle brake disc. Furthermore, it was found that k-refinement (increasing smoothness) significantly improves the efficiency.

A finite volume method (FVM) on NURBS geometries was proposed by Heinrich et al, 2012, [27] using a computational grid defined by the knot vectors and thus simplifying the mesh generation process. The method is illustrated by a fluid-structure interaction example where the FVM is coupled with a structural solver based on IGA, using a partitioned solver (flow and displacement solved separately). This novel approach preserves the geometry and physical properties and overcomes a typical problem that occurs with classical methods, where gaps and overlaps occur at the fluid-solid interface if different mesh sizes (non-matching grids) are used.
Fluid-Structure Interaction and Shape Optimization in IGA

Since its inception, IGA has been extensively tested in a wide range of applications including linear elasticity, vibrations and wave propagation, time dependent problems, non-linear problems, nearly incompressible solids and fluid dynamics. But one of the most interesting areas been researched is fluid-structure interaction (FSI), which is frequently linked with shape optimization where some of the features of IGA can be conveniently exploited. For instance, in optimization, control points are used as design variables to allow direct feedback to CAD.

Y. Bazilevs, one of the pioneers of IGA that collaborated with T. Hughes, is coincidentally a very prominent author in FSI that has been continuously contributing in this area by developing test cases such as flow over an elastic beam, inflation of a balloon, and blood flow in a patient-specific model of an abdominal aortic aneurysm.

Residual-based turbulent models for incompressible and advection dominated phenomena were presented by the same author using IGA [28] [29]. This work was based on a multiscale paradigm and tests were made using an unsteady parallel isogeometric incompressible flow solver. Furthermore, a patient-specific arterial blood flow FSI problem was solved using a fully coupled formulation after two numerical benchmarks were executed. The results were consistent with theoretical expectations and appear to be on par with state-of-the-art, eddy-viscosity formulations. Accuracy was notably increased by applying h-refinement to the IGA procedure.

Later on, a monolithic isogeometric FSI formulation of an incompressible fluid on a moving domain with a hyperelastic solid was shown [30]. An arbitrary Lagrangian-Eulerian formulation was used for the fluid domain and a Lagrangian formulation for the solid domain. Conservation properties of the formulations were investigated to conclude that mass conservation and geometric conservation laws are satisfied for the fully discrete case, while momentum was not conserved due to truncation errors in the time integration procedure. Three numerical examples were presented, showing that the formulation behaved robustly in all cases.

Another input to shape optimization using IGA was given by Nielsen [31] in his PhD thesis. Incompressible, steady-state, 2D flows at low to moderate Reynolds numbers were considered. Several new cases of discretization of the velocity and pressure spaces were presented and
investigated in terms of stability and convergence. A lid-driven cavity flow was used as a benchmark to show stability and consistency with previous results. Various cases of shape optimization such as minimizing the pressure drop inside a duct, a Taylor-Couette example, and drag minimization in the flow around an airfoil were developed to demonstrate robustness. Additionally, this methodology was extended to a coupled flow-acoustic model of sound propagation through a duct to show the potential of using this method in multi-physics solvers.

On the same line, with the addition of FSI and an isogeometric FVM hybrid, Heinrich et al, 2011, [32] introduced a bent pipe shape optimization problem based on a coupled FSI system using IGA and compared to a classical grid-based approach. Both the solid and fluid subdomains are discretized using the computational mesh defined by the knot vectors to avoid grid generation. The incompressible Navier-Stokes equations are solved using FVM adapted to NURBS geometry and the FSI algorithm is embedded in an optimization loop where the control points are used directly as design variables. The achieved optimization results did not show any important difference between quadratic and linear NURBS with regards to accuracy and convergence. However, IGA expedites the shape optimization and reduces geometry errors.

The mathematical development of a FVM on NURBS geometry with applications on isogeometric FSI was formally introduced a year later by the same author [27]. It is indicated that FVM is still the most popular method in CFD, mainly for its conservative properties (local conservation of fluxes) as well as its ability to handle discontinuous solutions (such as shocks) or high gradient regions (such as boundary layers). These characteristics are not easily obtained by FEA, without proper modifications in the formulation: enrichment by partition of unity, discontinuous Galerkin and other approaches originally introduced to model fractures and material interfaces. Although it is not explicitly stated in the text, the numerical examples shown in this contribution do not show any of the mentioned benefits of using FVM instead of traditional FEA in CFD applications.
3. Geometry Parameterization

**Non-Uniform Rational Basis Spline**

Using a tensor product of two univariate NURBS curves (Eq. 2-3), a bivariate NURBS surface \( S(\xi, \eta) \) (see Fig. 3-1) can be defined in terms of its parametric coordinates \( \xi \) and \( \eta \), as follows:

\[
S(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} R_{i,j}^{p,q}(\xi, \eta) B_{i,j}
\]

where \( B_{i,j} \) is the control net; \( n \) and \( m \) are the number of basis functions; \( p \) and \( q \) are spline orders; and \( R_{i,j}^{p,q} \) is the bivariate NURBS basis function, all linked to coordinates \( \xi \) and \( \eta \).

The bivariate basis function is obtained from the B-spline basis functions, \( N \) and \( M \) (Eq. 2-2) as well as the weights \( w_{i,j} \) associated with each control point using the rational expression [15]:

\[
R_{i,j}^{p,q}(\xi, \eta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta) w_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta) w_{i,j}}
\]

The first derivatives of a bivariate NURBS basis function is obtained using the product rule in terms of the derivatives of the B-spline basis function. Nguyen [18] stated them as:

\[
\begin{align*}
\frac{\partial R_{i,j}^{p,q}(\xi, \eta)}{\partial \xi} &= w_{i,j} \frac{N_{i,p}'(\xi)M_{j,q}(\eta)W - N_{i,p}(\xi)M_{j,q}'(\eta)W'}{W^2} \\
\frac{\partial R_{i,j}^{p,q}(\xi, \eta)}{\partial \eta} &= w_{i,j} \frac{N_{i,p}(\xi)M_{j,q}'(\eta)W - N_{i,p}(\xi)M_{j,q}(\eta)W'}{W^2}
\end{align*}
\]

where, for example, the terms used in the \( \xi \) derivative (including the B-spline derivative) are:

\[
W = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta) w_{i,j} \\
W' = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}'(\xi)M_{j,q}(\eta) w_{i,j}
\]

\[
N_{i,p}'(\xi) = \frac{p}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) - \frac{p}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi)
\]
Hughes [15] shows that NURBS geometries have several important properties associated with their basis. For a given knot vector in the $\xi$ direction, $\Xi = \{\xi_1, \xi_2, ..., \xi_{n+p+1}\}$, and its counterpart in the $\eta$ direction, $H = \{\eta_1, \eta_2, ..., \eta_{m+q+1}\}$ the basis function exhibits:

- Partition of unity: In a given point, the sum of all basis function is one.
- Non-negativity: All basis functions are non-negative.
- Local support: In a given knot span only $(p+1)(q+1)$ basis functions are non-zero.
- Variation diminishing: The curve oscillates less than the control polygon.
- At a knot $\xi_i$ of multiplicity $k$, linked to a basis $N_{i,p}(\xi)$, the function is $C^{p-k}$ continuous.
- A NURBS geometry with constant weights is a B-spline geometry.

Basis functions can be enriched with three different mechanism of refinements while preserving the geometry and parameterization:

- h-refinement: Consists of inserting knots in the already existing knot vector, and recalculating the appropriate new control points in order to generate new basis functions while preserving the continuity of the curve.
- p-refinement: The order of the spline is elevated by raising the multiplicity of each entry in the existing knot vector and elevating the degree of the polynomial for each segment.
- k-refinement: increases both order and continuity making the curve smoother.
The present work will be limited only to h-refinement. The procedure is formally explained by Nguyen [18] using an initial knot vector \( \Xi = \{ \xi_1, \xi_2, \ldots, \xi_{n+p+1} \} \), an extended knot vector \( \Xi^e \) is created by inserting \( k \) knots within the knot spans, while leaving untouched the initial and final knots of \( \Xi \). After this is done, \( n+k \) new control points \( \overline{B_i} \) must be added while satisfying:

\[
\overline{B_i} = \alpha_i B_i + (1-\alpha_i) B_{i-1}, \quad \text{where}
\]

\[
\alpha_i = \begin{cases} 
1 & 1 \leq i \leq k-p \\
\frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} & k-p+1 \leq i \leq k \\
0 & k+1 \leq i \leq n + p + 2 
\end{cases}
\] (3-5)

Figure 3-2: Examples of consecutive h-refinements (left to right).

In practice, instead of using these equations, more efficient algorithms are already available, such as those developed by Piegl [12]. Most of the basic code used in the NURBS geometry parameterization was built based on algorithms provided by this author, including the routines:

- **findspan(xi,p,KV)**: Determines the knot span index of a given parametric value, when the polynomial order and the knot vector are known.
- **basis(xi,p,KV)**: Determines the univariate B-spline basis function and its parametric derivative at the evaluation point, for a known polynomial order and knot vector.
- **surf_p(xi,p,KV_Xi,eta,q,KV_Eta,B)**: Computes the point on a B-spline surface, at a given coordinate, with known polynomial orders, knot vector and control net.
- **hrefine1D(p,KV,B,X)**: Determines the refined knot vector and control points when a set of knots are inserted in a knot vector with known polynomial order and control points.
The results of the implementation of these algorithms (shown in blue) are verified with already available code (shown in red) in Figures 3-3 and 3-4.

![FindSpan comparison](image)

**Figure 3-3: Verification of the findspan routine.**

![Verification of the basis routine using a bivariate function.](image)

**Figure 3-4: Verification of the basis routine using a bivariate function.**

The NURBS basis function and its derivatives with respect to the parametric coordinates were computed using Equations 3-2, 3-3, and 3-4 and were stored in two different routines:

- derNURBS_1D(Xi,p,KV_Xi,W): Returns the univariate NURBS basis and its parametric derivative at a given point, with known polynomial order, knot vector and weights.
- derNURBS_2D(Xi,Eta,p,q,KV_Xi,KV_Eta,W): Returns the bivariate NURBS basis and its derivatives at a given point, with known polynomial orders, knot vectors and weights.

An example of the results is given in Fig. 3-5. The top nine plots are the basis functions, the nine in the middle are the derivatives with respect to $\xi$, and the lower nine with respect to $\eta$. 
Figure 3-5: Examples of NURBS basis and its parametric derivatives.
Additionally, three more functions were based in the guidelines provided in a contribution by Vuong [17] and the associated instructional code, ISOGAT, which is distributed under The European Union Public License, a free software license:

- `plot_grid(p,q,KV_Xi,KV_Eta,B,W)`: Plots parameter space and physical domain.
- `plot_basis(p,q,KV_Xi,KV_Eta,B,W)`: Plot basis function and stores post processing data.
- `hrefine2D(p,q,KV_Xi,KV_Eta,B,W)`: Surface refinement by knot insertion.

### Curvilinear coordinates and transformations

This section encloses a list of curvilinear geometry definitions as well as transformations that are used extensively throughout this thesis. For a more comprehensive analysis, refer to Warsi [33]:

- **Geometry definitions:**
  - Jacobian matrix: $J = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}$ (3-6)
  - Jacobian determinant: $J = |J| = x_\xi y_\eta - x_\eta y_\xi$ (3-7)
  - Inverse Jacobian matrix: $J^{-1} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} = \frac{1}{|J|} \begin{bmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{bmatrix}$ (3-8)
  - Covariant metric tensor: $g = \begin{bmatrix} x_\xi^2 + y_\xi^2 & x_\xi x_\eta + y_\xi y_\eta \\ x_\xi x_\eta + y_\xi y_\eta & x_\eta^2 + y_\eta^2 \end{bmatrix} = J^T J$ (3-9)
  - Covariant metric tensor determinant: $g = |g| = J^2$ (3-10)
  - Contravariant metric tensor: $g^{-1} = \frac{1}{g} \begin{bmatrix} x_\xi^2 + y_\xi^2 & -(x_\xi x_\eta + y_\xi y_\eta) \\ -(x_\xi x_\eta + y_\xi y_\eta) & x_\eta^2 + y_\eta^2 \end{bmatrix} = J^{-1} J^{-T}$ (3-11)

- **Coordinate transformations:**
  - Gradient: $\nabla \phi = \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} = J^T \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix}$ (3-12)
  - Outward unit normal vector: $n = \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} \nabla \xi_x & \nabla \eta_x \\ \nabla \xi_y & \nabla \eta_y \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix} = J^{-1} \begin{bmatrix} n_x \\ n_y \end{bmatrix}$ (3-13)
Differential curve segment: 
\[ ds = \left| \begin{bmatrix} dx \\ dy \end{bmatrix} \right| = \left| \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \right| d\xi d\eta = \left| J \left[ \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \right] \right| \] (3-14)

Surface integral: 
\[ \int f(x, y) \, dxdy = \int_{\Omega} f(x(\xi, \eta), y(\xi, \eta)) |J| \, d\xi d\eta \] (3-15)

Boundary integral: 
\[ \int_{\Gamma} h(x, y) \, ds = \int_{\Gamma} h(x(\xi, \eta), y(\xi, \eta)) \left| J \left[ \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \right] \right| \] (3-16)

Divergence theorem: 
\[ \oint_{\Gamma} (v \cdot n) \, ds = \int_{\Omega} (\nabla \cdot v) \, dxdy \] (3-17)

Green's first identity: 
\[ \int_{\Omega} \left( \psi \nabla^2 \phi + \nabla \phi \cdot \nabla \psi \right) \, dxdy = \oint_{\Gamma} \psi (\nabla \phi \cdot n) \, ds \] (3-18)

Some of the definitions and transformations were tested along with the Gaussian quadrature routines to verify their proper functioning. For instance, the Jacobian determinant for a circular geometry based on NURBS parameterization is shown in Fig. 3-6. Notice how the Jacobian determinant depends not only on the radius, but also the angle. This might seem counter-intuitive at first sight, but when the non-uniformity of the knot spans is taken into account, it is observed that the results shouldn't be a linear mapping between parametric coordinates and polar coordinates.

![Jacobian determinant of a circle using NURBS.](image)
Other examples included the computation of the outward unit normal vector in Gaussian quadrature point along the boundary, as shown in Fig. 3-7.

![Figure 3-7: Computed outward unit normal vectors for different shapes.](image)

Finally, in the example shown in Fig. 3-8, the centroid of each element was found by integrating the physical coordinates over the area in the parametric domain using Gaussian quadrature.

![Figure 3-8: Computed element centroids for different domains.](image)
4. Governing Equations and Boundary Conditions

In this section, a description of the governing partial differential equations (PDE) along with the associated boundary conditions, and initial condition, if applicable. The equations are given in Cartesian coordinates \((x, y)\) using vector notation. In general, the governing PDEs shown here are linear second order partial differential equations of the type:

\[
A \frac{\partial^2 u}{\partial x^2} + 2B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^3 u}{\partial y^2} = D \left( x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right) \tag{4-1}
\]

where \(u\) is a generic field variable, \(\{A B C\}\) are coefficients that may have spatial dependency, and \(D\) is a linear function or zero in case of homogeneity. The equation holds in the domain \(\Omega\).

Depending on the value of the discriminant, \(\Delta = B^2 - AC\), the governing equation is classified as:

- **Elliptic PDE**: \((\Delta < 0)\) The solutions are generally smooth, like in pure diffusion.
- **Parabolic PDE**: \((\Delta = 0)\) Mostly related to diffusion dynamics, like in the heat equation.
- **Hyperbolic PDE**: \((\Delta > 0)\) The solutions propagate, like in the wave equation.

The problems are also subjected to one or more boundary conditions (BC) of the type:

\[
\alpha u + \beta \nabla u \cdot \mathbf{n} = \gamma(x, y) \tag{4-2}
\]

where \(u\) is a generic field variable, \(\{\alpha \beta \gamma\}\) are coefficients that may have spatial dependency, and \(\mathbf{n}\) is the outward normal unit vector to the boundary \(\Gamma\), where the condition holds. It is known as a homogeneous BC if the term \(\gamma(x, y) = 0\) and it is classified as:

- **Dirichlet BC**: \((\beta = 0)\) The field variable is known at the boundary.
- **Neumann BC**: \((\alpha = 0)\) The flux of field variable through the boundary is known.
- **Robin BC**: \(\{\alpha \beta\} \neq 0\) A relation of the field variable and its flux is imposed.


**Elastostatics**

The present study comprises a case of linear elasticity under equilibrium conditions. The body is homogeneous, isotropic and it is subjected to an in-plane stress state, where the out of plane displacement component is negligible. The governing equations couples the stresses with the displacement field. From the equilibrium equation, which relates stresses with external loads:

\[
\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + F_x = 0 \quad \text{and} \quad \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + F_y = 0
\]

(4-3)

where \(\{\sigma_x, \sigma_y\}\) are the normal stresses, \(\{\tau_{yx}, \tau_{xy}\}\) are the shear stresses and \(\{F_x, F_y\}\) are the external loads, each one of them in the \(x\) and \(y\) direction, respectively.

The relation between the stresses \(\sigma\) and the strains \(\varepsilon\) is given by a set of constitutive equations. According to Hooke’s law, for an isotropic, elastic material of Young’s modulus \(E\) and Poisson’s ratio \(\nu\), the stresses are given in term of the strains as follows:

\[
\begin{bmatrix}
\sigma_x \\
\sigma_y \\
\tau_{xy}
\end{bmatrix} = \frac{E}{(1-\nu^2)} \begin{bmatrix}
1 & \nu & 0 \\
\nu & 1 & 0 \\
0 & 0 & (1-\nu)/2
\end{bmatrix} \begin{bmatrix}
\varepsilon_x \\
\varepsilon_y \\
\gamma_{xy}
\end{bmatrix}
\]

(4-4)

where \(\{\varepsilon_x, \varepsilon_y\}\) are the normal strains (dilation) in the \(x\) and \(y\) direction, and \(\gamma_{xy}\) is shear strain which measures the distortion. The strains and the displacement field \((u, v)\) are related as:

\[
\begin{bmatrix}
\varepsilon_x & \varepsilon_y & \gamma_{xy}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}
\end{bmatrix}
\]

(4-5)

With regards to the boundary conditions, three cases are shown:

- Traction: Applied stress magnitude and direction is known at the boundary.
- Free surface: Stresses in the direction normal to the surface are zero.
- Symmetry: No displacement in the normal direction to the boundary.
**Poisson's Equation**

Poisson's equation is an elliptic partial differential equation that describes steady diffusion. It is used extensively in mechanical engineering and electrostatics and it is named after the French mathematician Siméon Denis Poisson. In terms of the Laplacian operator, it is defined as:

\[
\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)
\]

(4-6)

where \( u \) is a generic field variable subjected to diffusion under a source term \( f(x, y) \).

Its homogeneous counterpart \( (f(x, y) = 0) \) is known as Laplace equation, named after the French mathematician Pierre-Simon Laplace. It is the simplest elliptic PDE and its solutions are known as harmonic functions. The equation is written as:

\[
\nabla^2 u = 0
\]

(4-7)

where \( u \) is a generic field variable.

For the examples to be shown later, two additional boundary conditions are imposed:

- **Periodic BC**: When two boundaries have the same solution, mainly because the solution is repeated or that the two boundaries overlap.
- **Flux imposition in one direction**: Since the dot product of the Neumann BC implies that the flux is computed normal to the surface, a modification must be done to account for another directions. For instance, if the flux in one direction is constant, the field variable can be defined as a linear function in that direction, as a Dirichlet BC.

One interesting way to look at the physical sense of the Laplacian operator is to look at it as the divergence of the gradient operator. Also, by writing down a discrete form of the operator using second finite difference, in a uniform grid with no source term, the solution at the current point will be the average of the four neighbors. This explains the smoothing effect mentioned earlier.
**Heat Equation**

The unsteady heat conduction equation is a parabolic partial differential equation that describes the variation of temperature over time inside a body. It is derived from Fourier's law of heat conduction, which states that the heat transfer though a material is proportional to the gradient of the temperature, and to the area through heat flows. Mathematically, it is expressed as:

\[
\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0 \tag{4-8}
\]

where \( u \) is the temperature and \( \alpha \) is the thermal diffusivity (ratio of thermal conductivity to the product of density and specific heat capacity) which is assumed to be a constant in this case.

A more general version of this equation is the diffusion equation, which applies when the diffusion coefficient is not a constant. The boundary conditions and initial conditions that will be used in the heat equation examples included in this thesis are:

- Non-homogeneous Dirichlet BCs: Temperature imposition on the boundary.
- Homogeneous Neumann: No heat flux on the boundary (perfectly insulated wall).
- Initial conditions: The temperature is defined at the starting time over the whole domain.

**Linear Advection Diffusion**

This example involves a generic scalar transport phenomena in which a variable is transferred inside a body due to both diffusion and advection. The process is steady, with no sources, and the velocity field has zero divergence. The governing equation is parabolic in nature, written as:

\[
\mathbf{V} \cdot \nabla u - D \nabla^2 u = 0 \tag{4-9}
\]

where \( u \) is a generic field variable, \( \mathbf{V} \) is the velocity field and \( D \) is the diffusion coefficient which is assumed to be a constant in this case.

Notice that for relatively small diffusion coefficients, where advection dominates, this equation may behave as a hyperbolic equation and thus discontinuities in the boundaries may be preserved. The effect of varying the diffusivity will be shown later, in the examples.
5. Discretization and Optimization Methods

Isogeometric Analysis

Isogeometric analysis borrows the isoparametric concept from finite element analysis (same basis functions for the geometry and the solution) by using NURBS as the basis functions instead of interpolating polynomials.

According to Hughes [15], there are sufficient conditions for the basic convergence of isogeometric analysis in a wide class of problems due to the properties of this basis:

- It is continuously differentiable inside the element.
- It is continuous on the boundary.
- It is complete: capable of representing all linear function on any given element.

The Galerkin Method, a numerical method typically used in finite element analysis is also chosen for the isogeometric analysis examples that are developed here. The first example shown will be using Poisson's equation (Eq. 4-6), limited to Dirichlet BCs and Neumann BCs.

The governing equation is moved from the strong form, previously shown, to a weak form (variational) by means of multiplying by a weighting function and then integrating by parts (or using the divergence theorem and Green's identities in higher dimensions). The weighting functions and the trial solutions must comply with several conditions formally defined in the literature [15] including:

- Both the trial solution and the weighting function need to be part of the Sobolev space:
  - The functions must be square-integrable: \( \int_{\Omega} u^2 \, dx \, dy < +\infty \)
  - The gradients of the functions must be square-integrable: \( \int_{\Omega} (\nabla u \cdot \nabla u) \, dx \, dy < +\infty \)
- The trial solution must satisfy the Dirichlet boundary conditions.
- The weighting function must be zero at the Dirichlet boundary conditions.
If these conditions are met, then the weak form solution will also satisfy the problem in its strong form. In our example, Poisson's equation is multiplied by a weighting function $w$ and the left hand side is expanded using the Green's first theorem:

$$\int_\Omega (w \nabla^2 u) \, dx \, dy = \oint_\Gamma (w \nabla u \cdot n) \, ds - \int_\Omega (\nabla w \cdot \nabla u) \, dx \, dy = \int_\Omega (f(x,y)) \, dx \, dy$$  \hspace{1cm} (5-1)$$

However, since the weighting function is chosen to be zero at the Dirichlet BCs, the boundary integral is non-zero only in the Neumann BC, $\Gamma_N$. Also, at the Neumann BC, the normal derivative of the trial function is given $(\nabla u \cdot n = h)$ and therefore can be substituted as follows:

$$\oint_{\Gamma_N} (wh) \, ds - \int_\Omega (\nabla w \cdot \nabla u) \, dx \, dy = \int_\Omega (f(x,y)) \, dx \, dy$$  \hspace{1cm} (5-2)$$

The results can be rearranged as $a(w,u) = L(w)$ by defining two operators:

- A bilinear operator: $a(w,u) \equiv \int_\Omega (\nabla w \cdot \nabla u) \, dx \, dy$  \hspace{1cm} (5-3)$$
- A linear operator: $L(w) \equiv \oint_{\Gamma_N} (wh) \, ds - \int_\Omega (wf(x,y)) \, dx \, dy$  \hspace{1cm} (5-4)$$

In the Galerkin approach, the trial solution $u^h$ is the addition of a homogenous term $v^h$ that is part of the weighting functions, and a forcing function $g^h$, that is part of the trial solutions and satisfies the Dirichlet boundary conditions, such that:

$$u^h = v^h + g^h$$  \hspace{1cm} (5-5)$$

In this way, the unknown variables can be organized on the left hand side and the known data is on the right hand side, as follows:

$$a(w^h,v^h) = L(w^h) - a(w^h,g^h)$$  \hspace{1cm} (5-6)$$

However, since our application of Poisson's problem using IGA (fully developed flows through ducts) is restricted to homogeneous Dirichlet BCs, the forcing function term is not necessary.
The basis function for all the cases shown here will be bivariate NURBS basis functions, \( R_{i,j}^{p,q}(\xi, \eta) \), with spline orders \( p \) and \( q \), of the basis in the parametric coordinates \( \xi \) and \( \eta \), respectively. For the homogeneous case, the numerical solution is constructed as a linear combination of the basis functions with unknown coefficients \( d_{i,j} \) [17]:

\[
    u^h(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} R_{i,j}^{p,q}(\xi, \eta) d_{i,j}
\]

(5-7)

where \( n \) and \( m \) are the number of basis functions, in the \( \xi \) and \( \eta \) direction, respectively.

For simplicity, the basis functions indices \( (i, j) \) are mapped to a global index \( k = \{1, ..., n \cdot m\} \), such that \( R_k \equiv R_{i,j}^{p,q} \) and \( d_k \equiv d_{i,j} \). Thus, by taking a given basis \( R_k \) as the weighting function \( \left( w^h = R_k \right) \), in the bilinear form equation (Eq. 5-6) and due to linearity, it is shown [15] that:

\[
    \sum_{k=1}^{n \cdot m} a(R_i, R_k) d_k = L(R_i) - a(R_i, g^h)
\]

(5-8)

Since Eq. 5-8 must hold for every basis function, a linear system of \( n \cdot m \) equations is organized:

\[
    Kd = F
\]

(5-9)

where:

- \( d = \{d_i\} \) is a column vector of the unknown coefficients (used to build the solution field).

- \( F \) is a column vector constructed with known data: \( \left\{ F_i = L(R_i) - a(R_i, g^h) \right\} \)

(5-10)

- \( K \) is a square matrix whose entries are: \( K(l,k) = a(R_l, R_k) \)

(5-11)

In practice, for the assembly process of the global matrix \( K \) (also known as the global stiffness matrix) and the global vector \( F \) (also known as global load vector), instead of looping through each one of the basis functions, the assembly is made by looping through the elements (knot spans) to build a local stiffness matrix \( k^e \) and a local force vector \( f^e \) for each element, taking into account only the non-zero basis functions local to that element.
This approach is comparable to the assembly process in traditional finite element analysis. It saves computation time, since $K$ is a sparse matrix, due to the localized support of each NURBS basis function: a total of $(p+1)(q+1)$ non-zero local basis functions exist for each element.

The appropriate distribution of the local stiffness matrix and the local load vector in their global counterparts is aided by the existence of a connectivity matrix that links every local shape function to its global position. Using the NURBS basis functions indices $(i, j)$, the global basis function index $k$ is defined as:

$$k = i + (j-1)n$$  \hspace{1cm} (5-12)

The integration is done in the parametric domain using a Gaussian quadrature of order $p+1$ and $q+1$ for coordinates $\xi$ and $\eta$, respectively. At each quadrature point, the shape function routine must evaluate the NURBS basis functions, their gradients and the Jacobian determinant.

Regarding boundary imposition, two important things are noticed:

- Neumann BCs are imposed inside the linear operator term of Eq. 5-8, and in the homogeneous case, the imposition is natural since no additional computation is needed.
- Dirichlet BCs are easily imposed using a forcing function as long as the boundary has interpolatory control points at its nodes, otherwise a different approach is needed [37].

For a given mesh, the convergence is evaluated usually by computing the discrete $L^2$ norm error as the difference of the analytical and numerical solutions ($u$ and $u^h$) at every element's centroid:

$$\|u-u^h\|_{L^2(\Omega)} = \sqrt{\sum_{i=1}^{N_ele}(u-u^h)^2} / |J| \Delta_\xi \Delta_\eta$$  \hspace{1cm} (5-13)

The order of convergence is determined as the computed slope from the log-log plot of the $L^2$ norm error vs the maximum element size (minimum diameter that encloses an element) as the mesh is refined. According to Hughes [15], the convergence order of isogeometric analysis and traditional finite element analysis should be the same.
For the elastostatics example, since it was only used as a benchmark and the elasticity formulation is not involved in any of the other examples, only the assembly process is discussed here, starting with a global system of equations (such as Eq. 5-9) obtained by Vuong [18]:

\[
K(l, k) = \int_{\Omega} B_l^T \mathbf{D} B_k \, dx \, dy \\
F(l) = \oint_{\Gamma} \mathbf{t} \, ds
\]  \tag{5-14}

where \( \mathbf{t} \) is the traction being applied to the boundary, \( \mathbf{D} \) is the elasticity matrix previously used in the formulation, and \( \mathbf{B}_l \) is a matrix of the associated basis function derivatives as follows:

\[
\mathbf{D} = \frac{E}{(1-v^2)} \begin{bmatrix}
1 & v & 0 \\
v & 1 & 0 \\
0 & 0 & (1-v)/2
\end{bmatrix}
\]  \tag{5-15}

\[
\mathbf{B}_l^T = \begin{bmatrix}
\frac{\partial R_l}{\partial x} & 0 & \frac{\partial R_l}{\partial y} \\
0 & \frac{\partial R_l}{\partial y} & \frac{\partial R_l}{\partial x}
\end{bmatrix}
\]  \tag{5-16}

Notice that the entries defined in Eq. 5-14 are two dimensional and therefore should be distributed in the \( \mathbf{K} \) matrix and the \( \mathbf{F} \) vector accordingly by partitioning the horizontal displacement terms on the top half and the vertical displacement terms in the bottom half. That way, the size of the linear system is the number of degrees of freedom, or \( 2 \cdot n \cdot m \).

In the example developed, the spline orders were \( p=2 \) and \( q=2 \), such that only six basis functions were non-zero over each element. This allows to expedite the assembly procedure by looping through elements instead of basis functions, as mentioned earlier.

The Dirichlet boundary conditions are imposed directly by modifying the force vector entries whose values are fixed and only allowing diagonal terms in the associated rows of the stiffness matrix. Also, a penalty method is applied to enforce the same displacements on overlapping control points.

Overall, the isogeometric analysis code follows the same fashion as the traditional finite element analysis code. Only certain routines need to be replaced or modified, marked as green in Fig. 5-1.
Figure 5-1: Flowchart of typical finite element code and required modifications for IGA. [15]
Finite Volume Method on NURBS

The discretization of Poisson's equation using the finite volume method follows a similar approach as that used by Heinrich [27]. Starting from the PDE in its strong form (Eq. 4-6), the weak form is reached by multiplying by a test function \( v \) and integrating over the control volume:

\[
\int_{\Omega} (v)(\nabla^2 \phi) dxdy = \int_{\Omega} (v)(f(x,y)) dxdy \tag{5-17}
\]

The Laplacian operator is expanded in the left hand side by using Green's first identity, such that:

\[
\int_{\Omega} (v)(\nabla^2 \phi) dxdy = \oint_{\Gamma} (v)(\nabla \phi \cdot n) ds - \int_{\Omega} (\nabla v) \cdot (\nabla \phi) dxdy \tag{5-18}
\]

The test function is assumed to be constant in each control volume, allowing to factor out this term from every integral, as well as dropping the test function gradient term, obtaining:

\[
(v) \int_{\Omega} (\nabla^2 \phi) dxdy = (v) \oint_{\Gamma} (\nabla \phi \cdot n) ds \tag{5-19}
\]

This result is then substituted in Eq. 5-17, factoring out the test function, to obtain the final integral formulation of the numerical solution. The equation now holds for every control volume, including those at the boundaries.

\[
\oint_{\Gamma} (\nabla \phi \cdot n) ds = \int_{\Omega} f(x,y) dxdy \tag{5-20}
\]

This process has been completely established on Cartesian coordinates, and a coordinate transformation must be done to map it to an arbitrary curvilinear coordinate, using the geometry definitions made in section 3. This involves mainly the use of the Jacobian matrix and the metrics (covariant and contravariant metric tensors) as well as the transformation of vectors and differential operators.

The right hand side (source term) in the integral formulation is easily mapped to parametric coordinates by means of the Jacobian determinant:

\[
\int_{\Omega} f(x,y) dxdy = \int_{\Omega_{\xi\eta}} f(x(\xi,\eta), y(\xi,\eta)) |J| d\xi d\eta \tag{5-21}
\]
The normal derivative (dot product of gradient and normal outward unit vector) is computed as:

\[
(\nabla \phi) \cdot (n) = \left[ J^{-T} \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \right] \cdot (n) = \left[ \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \right]^T J^{-1} \begin{bmatrix} J^{-T} n_0 \\ n \end{bmatrix} = \frac{\left[ \phi_x, \phi_y \right] g^{-1} \begin{bmatrix} n_x \\ n_y \end{bmatrix}}{\left\| J^{-T} n_0 \right\|} \tag{5-22}
\]

From Eq. 5-20, the differential curve, ds, is expanded along with the norm term in Eq. 5-22 and it is simplified for the two main cases in the parametric domain, when \( ds_0 = d\xi \) or \( ds_0 = d\eta \):

\[
\left\| J ds_0 \right\| = \left\| \begin{bmatrix} x_\xi x_\eta \\ y_\xi y_\eta \end{bmatrix} d\xi \right\| = \begin{cases} d\eta = 0 & \rightarrow \sqrt{x_\xi^2 + y_\xi^2} d\xi = \sqrt{g_{11}} d\xi \\ d\xi = 0 & \rightarrow \sqrt{x_\eta^2 + y_\eta^2} |d\eta| = \sqrt{g_{22}} |d\eta| \end{cases}\]

\[
\left\| J^{-T} n_0 \right\| = \left\| \frac{1}{|J|} \begin{bmatrix} y_\eta - y_\xi \\ -x_\eta - x_\xi \end{bmatrix} n_\xi n_\eta \right\| = \begin{cases} d\eta = 0 & \rightarrow \frac{\left[ n_\eta \right]}{|J|} \sqrt{x_\xi^2 + y_\xi^2} = \frac{\sqrt{g_{11}}}{|J|} \\ d\xi = 0 & \rightarrow \frac{\left[ n_\xi \right]}{|J|} \sqrt{x_\eta^2 + y_\eta^2} = \frac{\sqrt{g_{22}}}{|J|} \end{cases} \tag{5-23}
\]

It is clearly seen that the ratio of the two terms expanded in Eq. 5-23 is scaled by the Jacobian determinant. This result is then used to substitute Eq. 5-22 in the left hand side of Eq. 5-20:

\[
\hat{f}_T (\nabla \phi \cdot n) ds = \hat{f}_T \left[ \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \right] g^{-1} n_0 \left\| J ds_0 \right\| = \begin{cases} n_x = 0 & \rightarrow \hat{f}_T \left[ \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \right] g^{-1} \begin{bmatrix} 0 \\ n_\eta \end{bmatrix} |J| d\xi \\ n_\eta = 0 & \rightarrow \hat{f}_T \left[ \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} \right] g^{-1} \begin{bmatrix} n_x \\ 0 \end{bmatrix} |J| d\eta \end{cases} \tag{5-24}
\]

By pairing Eq. 5-24 with the discretized source term (Eq. 5-21), and applying midpoint rule at the center P of a control volume (Fig. 5-2) with indices s, w, n, e denoting the faces, the result is:

\[
\sum_{m \in \{s, w, n, e\}} \left( k_m \right)^T \begin{bmatrix} \phi_x \\ \phi_y \end{bmatrix} = \left( f(x(\xi, \eta), y(\xi, \eta))|J|\Delta\xi\Delta\eta \right)_p \tag{5-25}
\]

where \( k_m = (g^{-1} n_0 |J|\Delta s_0)_m \) is a 2×1 coefficient vector that must be computed at each face.
The fluxes in parametric domain are computed using a central difference scheme for every face that is not in a boundary. For the general case (non-orthogonal grids) fluxes in both parametric direction are needed in each face, requiring the use of 8 neighbors around the control volume (Moore neighborhood). For the orthogonal case, only fluxes normal to the faces are needed, resulting in four neighbors around the control volume (von Neumann neighborhood).

The approximation of the fluxes at each face using central difference is summarized below:

- **East face, e:**
  \[
  \phi_z = \frac{\phi_E - \phi_P}{\Delta \xi}; \quad \phi_\eta = \frac{\left(\phi_N + \phi_{NE}\right) - \left(\phi_S + \phi_{SE}\right)}{4\Delta \eta}; \quad (5-26)
  \]

- **West face, w:**
  \[
  \phi_z = \frac{\phi_P - \phi_W}{\Delta \xi}; \quad \phi_\eta = \frac{\left(\phi_N + \phi_{NW}\right) - \left(\phi_S + \phi_{SW}\right)}{4\Delta \eta}; \quad (5-27)
  \]

- **North face, n:**
  \[
  \phi_z = \frac{\left(\phi_E + \phi_{NE}\right) - \left(\phi_W + \phi_{NW}\right)}{4\Delta \xi}; \quad \phi_\eta = \frac{\phi_N - \phi_P}{\Delta \eta}; \quad (5-28)
  \]

- **South face, s:**
  \[
  \phi_z = \frac{\left(\phi_E + \phi_{SE}\right) - \left(\phi_W + \phi_{SW}\right)}{4\Delta \xi}; \quad \phi_\eta = \frac{\phi_P - \phi_S}{\Delta \eta}; \quad (5-29)
  \]
For the orthogonal case \( g_{12}^{-1} = g_{21}^{-1} = 0 \), after applying the flux approximations, the left hand side, LHS, of Eq. 5-25 is reduced to a simplified expression in terms of five unknowns:

\[
LHS = \Delta \zeta \left[ \left( |J| g_{22}^{-1} \frac{\phi_w - \phi_r}{\Delta \eta} \right) - \left( |J| g_{22}^{-1} \frac{\phi_x - \phi_r}{\Delta \xi} \right) \right] + \Delta \eta \left[ \left( |J| g_{11}^{-1} \frac{\phi_z - \phi_r}{\Delta \xi} \right) - \left( |J| g_{11}^{-1} \frac{\phi_x - \phi_r}{\Delta \xi} \right) \right]
\]

Thus, the discretized governing equation for an internal control volume can be organized as a linear system of the von Neumann neighborhood:

\[
a_s \phi_s + a_w \phi_w + a_N \phi_N + a_E \phi_E + a_p \phi_p = b_p
\]

(5-31)

where the coefficients are defined as:

\[
a_s = \frac{\Delta \eta}{\Delta \xi} \left( |J| g_{22}^{-1} \right)_s; \quad a_w = \frac{\Delta \eta}{\Delta \xi} \left( |J| g_{22}^{-1} \right)_w;
\]

\[
a_N = \frac{\Delta \xi}{\Delta \eta} \left( |J| g_{22}^{-1} \right)_N; \quad a_E = \frac{\Delta \eta}{\Delta \xi} \left( |J| g_{11}^{-1} \right)_E;
\]

\[
a_p = -a_E - a_w - a_N - a_s
\]

\[
b_p = \int_{\Omega} f(x(\xi, \eta), y(\xi, \eta))|J|d\xi d\eta
\]

(5-32)

There is a special treatment in the computation of fluxes through faces that are in a boundary:

- For an orthogonal mesh, the Neumann BC value is directly entered as the flux normal to the boundary face in Eq. 5-25. Non-homogeneous terms are moved to the right hand side.
- Dirichlet BC values are used to compute the flux by means of a one-sided approximation. For instance, a linear approximation from the boundary to the control volume centroid or by using a variable step one-sided second order approximation shown in Fig. 5-3.

\[
\Delta \eta = \frac{1}{2} \Delta x \quad \Delta \xi = \frac{1}{2} \Delta x
\]

\[
y = ax^2 + bx + c;
\]

\[
\frac{dy}{dx}_h = 2ax_i + b = \frac{y_1 - 9y_2 + 8y_3}{3\Delta x}
\]

Figure 5-3: Variable step one-sided second order approximation.
In the case a linear advection term \((\mathbf{V} \cdot \nabla \phi)\) is included, it is discretized by integrating over the control volume and applying the Divergence Theorem:

\[
\int_{\Omega} (\mathbf{V} \cdot \nabla \phi) \, dxdy = \oint_{\Gamma} (\phi)(\mathbf{V} \cdot \mathbf{n}) \, ds - \int_{\Omega} (\phi)(\nabla \cdot \mathbf{V}) \, dxdy
\]  \tag{5-33}

For a solenoidal (divergence-free) velocity field, and using the same expansions of Eq. 5-24, this term is reduced for the two main cases in the parametric domain, when \(ds_0 = d\xi\) or \(ds_0 = d\eta\):

\[
\oint_{\Gamma} (\phi)(\mathbf{V} \cdot \mathbf{n}) \, ds = \oint_{\Gamma_0} (\phi) \left( \mathbf{V}^T \mathbf{J}^{-T} \left[ \begin{array}{c} n_x \\ n_x \end{array} \right] \right) |\mathbf{J}| \, ds_0 \Rightarrow \begin{cases} n_x = 0 & \rightarrow \oint_{\Gamma_0} (\phi) \left( \mathbf{V}^T \mathbf{J}^{-T} \left[ \begin{array}{c} 0 \\ n_x \end{array} \right] \right) |\mathbf{J}| \, d\xi \\ n_x = 0 & \rightarrow \oint_{\Gamma_0} (\phi) \left( \mathbf{V}^T \mathbf{J}^{-T} \left[ \begin{array}{c} n_x \\ 0 \end{array} \right] \right) |\mathbf{J}| \, d\eta \end{cases}
\]  \tag{5-34}

The flux coefficient \(\left( \mathbf{V}^T \mathbf{J}^{-T} \left[ n_x \ 0 \right]^T |\mathbf{J}| \right)\) can be computed at each face without any trouble.

However, the new flux quantity is the field variable itself, and the direction of the flow must be taken into account to stabilize the scheme. Several flux limiters methods were reviewed by Dullemond [34] and a first order method known as Donor Cell was chosen for its simplicity.

In the Donor Cell differencing scheme, the advected value of the field variable is taken from the upstream control volume of the current face. For instance, if the horizontal component of the velocity is positive, the flux coefficient of the eastern face multiplies the value of the field variable at \(P\), as seen in Fig. 5-4. After this proper modification of the selection of the flux variable for each face, Eq. 5-24 is used in a discretized form such as the one shown in Eq. 5-25.

![Figure 5-4: Example of advected values in Donor Cell scheme.](image-url)
Finite Difference Method on NURBS

For the finite difference formulation, the governing PDE is directly mapped to the parametric domain in its strong form. For Poisson's equation (Eq. 4-6), the Laplacian operator is expanded with the help of the divergence properties. Using the Jacobian matrix for the transformation:

\[
\nabla^2 \phi = \nabla \cdot \nabla \phi = \left( J^{-T} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} \right)^T \left( J^{-T} \begin{bmatrix} \phi_x \\ \phi_{\eta} \end{bmatrix} \right) \tag{5-35}
\]

This expression is expanded in terms of the elements of the contravariant metric tensor, \( g^{-1} \):

\[
\frac{1}{|J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{11}^{-1} \phi_x \right) + \frac{\partial}{\partial \eta} \left( |J| g_{12}^{-1} \phi_x \right) \right) + \frac{1}{|J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{12}^{-1} \phi_{\eta} \right) + \frac{\partial}{\partial \eta} \left( |J| g_{22}^{-1} \phi_{\eta} \right) \right) \tag{5-36}
\]

Notice that some of the terms include the derivatives of the metric tensor and the Jacobian determinant. Also, it can be rearranged based on orthogonality as:

\[
\frac{1}{|J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{11}^{-1} \phi_x \right) + \frac{\partial}{\partial \eta} \left( |J| g_{12}^{-1} \phi_x \right) \right) + \frac{1}{|J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{12}^{-1} \phi_{\eta} \right) + \frac{\partial}{\partial \eta} \left( |J| g_{22}^{-1} \phi_{\eta} \right) \right) \tag{5-37}
\]

Product rule is used to expand the derivatives of product terms:

\[
\frac{1}{|J|} \left( |J| g_{11}^{-1} \phi_{\xi} + \frac{\partial (|J| g_{11}^{-1})}{\partial \xi} \phi_x + \frac{\partial (|J| g_{11}^{-1})}{\partial \eta} \phi_{\eta} \right) + \frac{1}{|J|} \left( |J| g_{12}^{-1} \phi_{\eta} + \frac{\partial (|J| g_{12}^{-1})}{\partial \xi} \phi_x + \frac{\partial (|J| g_{12}^{-1})}{\partial \eta} \phi_{\eta} \right) \tag{5-38}
\]

Clearly, non-orthogonal terms are associated with mixed derivatives as well as mixed product of derivatives (in different parametric coordinates) of the metric tensor and the field variable.
The discretization for the derivatives of the metrics is made by using a central difference scheme in a smaller subgrid. For instance, for a term $M$ at a position $i$, the derivative with respect to $\xi$ is:

$$\frac{\partial M}{\partial \xi} = \frac{M_{i+1} - M_{i-1}}{2\Delta \xi} \quad (5-39)$$

The derivatives of the field variable are also computed using a second order finite difference in the traditional grid generated in the NURBS mesh:

$$\frac{\partial^2 \phi}{\partial \xi^2} = \frac{\phi_{E} - 2\phi_{i} + \phi_{W}}{(\Delta \xi)^2}; \quad \frac{\partial^2 \phi}{\partial \eta^2} = \frac{\phi_{N} - 2\phi_{i} + \phi_{S}}{(\Delta \eta)^2}$$

$$\frac{\partial^2 \phi}{\partial \xi \partial \eta} = \frac{1}{4\Delta \xi \Delta \eta} \left( \phi_{NE} + \phi_{SW} - \phi_{SE} - \phi_{NW} \right) \quad (5-40)$$

$$\frac{\partial \phi}{\partial \xi} = \frac{\phi_{E} - \phi_{W}}{2\Delta \xi}; \quad \frac{\partial \phi}{\partial \eta} = \frac{\phi_{N} - \phi_{S}}{2\Delta \eta}$$

Substituting these approximations in Eq. 5-38, a linear system for the each node is given in terms of the von Neumann neighborhood for the case of an orthogonal grid; otherwise, it is based on the Moore neighborhood (Fig. 5-5), $MN=\{S, W, N, E, SW, NW, NE, SE, \}$:

$$\sum_{k \in MN} a_{k} \phi_{k} = b_{p}, \quad \text{where:}$$

$$a_{N} = \left( \frac{g_{22}^{-1}}{(\Delta \eta)^2} \right) + \frac{1}{2\Delta \eta |J|} \left( \frac{\partial}{\partial \eta} \left( |J| g_{22}^{-1} \right) + \frac{\partial}{\partial \xi} \left( |J| g_{12}^{-1} \right) \right)$$

$$a_{E} = \left( \frac{g_{11}^{-1}}{(\Delta \xi)^2} \right) + \frac{1}{2\Delta \xi |J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{11}^{-1} \right) + \frac{\partial}{\partial \eta} \left( |J| g_{21}^{-1} \right) \right)$$

$$a_{W} = \left( \frac{g_{11}^{-1}}{(\Delta \xi)^2} \right) - \frac{1}{2\Delta \xi |J|} \left( \frac{\partial}{\partial \xi} \left( |J| g_{11}^{-1} \right) + \frac{\partial}{\partial \eta} \left( |J| g_{21}^{-1} \right) \right)$$

$$a_{S} = \left( \frac{g_{22}^{-1}}{(\Delta \eta)^2} \right) - \frac{1}{2\Delta \eta |J|} \left( \frac{\partial}{\partial \eta} \left( |J| g_{22}^{-1} \right) + \frac{\partial}{\partial \xi} \left( |J| g_{12}^{-1} \right) \right)$$

$$a_{SW} = \frac{g_{12}^{-1} + g_{21}^{-1}}{4\Delta \xi \Delta \eta}; \quad a_{SE} = -\frac{g_{12}^{-1} + g_{21}^{-1}}{4\Delta \xi \Delta \eta}$$

$$a_{NE} = \frac{g_{12}^{-1} + g_{21}^{-1}}{4\Delta \xi \Delta \eta}; \quad a_{NW} = -\frac{g_{12}^{-1} + g_{21}^{-1}}{4\Delta \xi \Delta \eta}$$

$$a_{p} = -2 \left( \frac{g_{11}^{-1}}{(\Delta \xi)^2} + \frac{g_{22}^{-1}}{(\Delta \eta)^2} \right); \quad b_{p} = \left( f \left( x(\xi, \eta), y(\xi, \eta) \right) \right)_{p}$$
Figure 5-5: The von Neumann neighborhood (yellow) as a subset of Moore neighborhood (red).

The previous FDM formulation only applies for internal nodes. BCs are imposed directly at each boundary node, \( n \), by modifying the \( n \)th row in the linear system \( \mathbf{A}\{\mathbf{x}\} = \{\mathbf{b}\} \):

- **Dirichlet BCs**: The right hand side is set to be Dirichlet value \( f_D(n) \) and a unity is imposed in the main diagonal entry. In other words, \( A(n,n) = 1 \) and \( b(n) = f_D(n) \).

- **Neumann BCs**: The right hand side is set to be the Neumann value whereas as a second order one-sided finite difference is used to compute the gradient in the parametric space. Following the same approach used in FVM, the normal derivative can be reduced to

\[
\nabla \phi \cdot \mathbf{n} = \begin{cases} 
\frac{[\phi_\xi \phi_\eta] g_\eta^{-1} \begin{bmatrix} 0 \\ n_\eta \end{bmatrix}}{\sqrt{g_{22}}} = \frac{n_\eta (g_{11}^{-1} \phi_\xi + g_{22}^{-1} \phi_\eta)}{\sqrt{g_{22}}} & \text{for left boundary} \\
\frac{[\phi_\xi \phi_\eta] g_\xi^{-1} \begin{bmatrix} n_\xi \\ 0 \end{bmatrix}}{\sqrt{g_{11}}} = \frac{n_\xi (g_{11}^{-1} \phi_\xi + g_{22}^{-1} \phi_\eta)}{\sqrt{g_{11}}} & \text{for right boundary}
\end{cases}
\]

The fluxes are obtained using a three-point approximation of the derivative. For example, at a point \( j \) in the main grid and using \( \theta = \pm 1 \) for left and right boundaries, respectively:

\[
\left( \frac{\partial \phi}{\partial \xi} \right)_j = \theta \left( \frac{-3 \phi_j + 4 \phi_{j+\theta} - \phi_{j+2\theta}}{2 \Delta \xi} \right)
\]

- **Robin BCs**: Combine the same approach used in Dirichlet and Neumann BCs.
Special treatment is given to the nodes in a degenerate boundary (Jacobian determinant is zero). Every node at the degenerated boundary is mapped to the same corner in the computational domain and the average of two quadratic extrapolations (in the two adjacent boundaries) is used to compute that corner, as shown in Fig. 5-6, where the degenerated boundary is marked as red.

For the examples where an advection term \( \mathbf{V} \cdot \nabla \phi \) is included, an upwind scheme is adopted. This approach uses an adaptive stencil that considers the direction of the flow field [35]. Consider a node located at \( j \), with a horizontal velocity \( u \), and a direction \( \theta = \text{sign}(u) \):

- First order upwind scheme: Uses a first order one-sided difference (two points). This scheme may introduce considerable numerical diffusion in boundary layers.

  \[
  \frac{\partial \phi}{\partial x} \bigg|_j = \theta \frac{\phi_j - \phi_{j-\theta}}{\Delta x} \quad (5-44)
  \]

- Second order upwind scheme: Uses a second order one-sided difference (three points). This scheme is less diffusive than the first order but it may introduce some numerical dispersion near discontinuities in the solution.

  \[
  \frac{\partial \phi}{\partial x} \bigg|_j = \theta \frac{3\phi_j - 4\phi_{j-\theta} + \phi_{j-2\theta}}{2\Delta x} \quad (5-45)
  \]

Using these methods, the instability of the central difference scheme in advective problems is surpassed. In both cases, the numerical error can be decreased by refining the mesh.
**Rothe's Method**

For transient problems of models using the heat equation, Rothe's Method is used as suggested by Elgeti [36]. Using the same numerical infrastructure already programmed for Poisson's equation, the source term is substituted for a time derivative term which is discretized using an implicit Euler scheme. The heat equation (Eq. 4-8) is then (where $n$ denotes the time step):

$$\frac{u^{n+1} - u^n}{\Delta t} - \alpha (\nabla^2 u)^{n+1} = 0$$  \hspace{1cm} (5-46)

For the case of IGA, the Laplacian operator is discretized using the already defined matrix $K$, whereas a newly introduced mass matrix $M$, has entries given by the area integral [36]:

$$M(l,k) = \int_{\Omega} R_l R_k \, dx \, dy$$  \hspace{1cm} (5-47)

Note that every entry of this matrix is positive since NURBS basis functions are non-negative. Also, due to the localized support of the NURBS basis functions, $M$ must be a sparse matrix. In terms of programming, the mass matrix can be computed by creating a modified version of the already presented assembly routine for $K$. Remembering that both matrices are used without imposing BCs, the differential equation can be posed in a matrix form, rearranging as:

$$(M + \Delta t \alpha K) \cdot u^{n+1} = M \cdot u^n$$  \hspace{1cm} (5-48)

To account for non-homogeneous Dirichlet BCs, the problem is homogenized by carrying the Dirichlet values to the right hand side. Thus, splitting the solution as $u^n = u^n_0 + u^n_D$, where $u^n_D$ is a vector that stores the Dirichlet values at the corresponding nodes, serving as a lifting function. The homogeneous part of solution at time step $n+1$, $u^{n+1}_0$, is determined using previous time step solution, $u^n = u^n_0 + u^n_D$, by solving the linear system:

$$(M + \alpha \Delta t K) \cdot u^{n+1}_0 = M \cdot u^n - (M + \alpha \Delta t K) u^n_D$$  \hspace{1cm} (5-49)

In the case of FVM, the non-homogeneous BC terms are moved to the right hand side, and $M$ becomes a diagonal matrix with the areas of the elements. Thus, an equivalent approach is used.
Particle Swarm Optimization

The particle swarm optimization (PSO) is a zeroth order method in which a set of particles are initially positioned with a random position and velocity inside the domain. The particles evaluate a pseudo-objective function and share that information with the other particles of the swarm. A leader particle is determined in each iteration as the one with the position of the best solution.

The velocity of the particle is determined by a combination of three factors:

- A random factor: allows to freely move around space to avoid local minimum.
- A personal thrust factor: leads the particle where the personal best was found.
- A social thrust factor: leads the particle where the leader is.

For the $i^{th}$ particle, the position $x_i$ and velocity $v_i$ are updated in each iteration $k$ as follows [37]:

$$
\begin{align*}
    v_i^{k+1} &= w v_i^k + c_1 r_1 (p_i^k - x_i^k) + c_2 r_2 (g_i^k - x_i^k) \\
    x_i^{k+1} &= x_i^k + v_i^{k+1} \Delta t
\end{align*}
$$

(5-50)

where $w$, $c_1$, $c_2$ and $\Delta t$ are tuning parameters that will be discussed further; $r_1$ and $r_2$ are random numbers between 0 and 1; $p_i^k$ is the best location obtained by the current particle and $g_i^k$ is the global best location obtained by the swarm.

The tuning parameters selection is essential depending on the nature of the problem:

- The inertia parameter, $w$, increases the random velocity component and allows free movement in the design variable space. It can be dynamically implemented and usually it is set to be between 1.4 (for global movement) and 0.8 (for localized movement).
- The personal trust parameter, $c_1$, defines how much confidence the particle has in itself. It is typically chosen to be between 1 and 2.
- The social trust parameter, $c_2$, defines how much confidence the particle has in the swarm leader. It is typically chosen to be between 1 and 2.
- A restriction factor, $\Delta t$, is a swarm iteration time step, usually chosen as 1.
The traditional PSO technique is designed to solve unconstrained problem with only one objective function, \( f(x) \). When the design variables are constrained, a pseudo-objective function, \( \Phi(x, r) \), is introduced to combine the objective function and the \( m \) constraints, \( g_j(x) \leq 0 \), by means of a penalty function in terms of the penalty parameter \( r \) [37]:

\[
\Phi(x, r) = f(x) + r \sum_{j=1}^{m} \max(0, g_j(x))^2
\]  

(5-51)

The penalty parameter is a large positive number, such as \( 10^6 \), and can be updated dynamically. Thus, the pseudo-objective function significantly increases when the constraints are violated.

After a routine for PSO was coded in MATLAB\textsuperscript{®}, several benchmark functions were obtained from an optimization survey [38] in order to evaluate its proper functionality and convergence. Every optimization benchmark is shown in Table 5-1, using 50 particles and \( ns \) iterations.

**Table 5-1: PSO Benchmark results.**

<table>
<thead>
<tr>
<th>ID</th>
<th>Function Name</th>
<th>Exact Solution</th>
<th>Analytical Best Location</th>
<th>Numerical Solution</th>
<th>Numerical Best Location</th>
<th>ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sphere</td>
<td>0</td>
<td>0</td>
<td>2.74E-10</td>
<td>-1.26E-05</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Ackley</td>
<td>0</td>
<td>0</td>
<td>5.68E-05</td>
<td>-1.83E-05</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Beale's</td>
<td>0</td>
<td>3</td>
<td>2.99E-10</td>
<td>3.00E+00</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Rosenbrock</td>
<td>0</td>
<td>1</td>
<td>1.44E-07</td>
<td>1.00E+00</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Rastrigin</td>
<td>0</td>
<td>0</td>
<td>3.43E-07</td>
<td>3.64E-05</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Easom</td>
<td>-1</td>
<td>0.1571</td>
<td>-1.00E+00</td>
<td>1.57E-01</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Styblinski-Tang</td>
<td>-78.332</td>
<td>-2.9035</td>
<td>-7.83E+01</td>
<td>-2.90E+00</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>Schaffer #2</td>
<td>0</td>
<td>0</td>
<td>1.32E-09</td>
<td>-5.21E-05</td>
<td>25</td>
</tr>
</tbody>
</table>

Furthermore, a visualization subroutine was developed to enhance the user interaction with the results, especially since this routine can work for an arbitrary function and may be useful in the near future. A collection of 3D surface plots and a particle evaluation history in the design variable space (on top of a contour plot) is shown in Figs. 5-7 and 5-8. They clearly show that the PSO is a robust algorithm that converges in functions with different behaviors and features.
Figure 5-7: PSO Benchmark results (ID:1-4)
Figure 5-8: PSO Benchmark results (ID:5-8)
6. Applications

In this section, using the governing equations presented in section 4, a set of applications are described using the involved field variables, material properties and secondary variables. The assumptions and simplifications of each model formulation is emphasized and boundary conditions are shown in each domain. Any available exact solutions is also presented.

**Plate with a Hole (under in-plane tension)**

A common benchmark in finite elements is an infinite plate with a circular hole at its center is subjected to uniaxial tension. Due to symmetry along both axes, only a quarter domain is studied. The plate is constituted by a homogeneous, isotropic and perfectly elastic material and it is subjected to an in-plane stress state. The governing equations are posed in Eqs. 4-3 through 4-5.

![Figure 6-1: Elastic plate with a circular hole: geometry and BCs.](image)

The boundary conditions are detailed as shown in Fig. 6-1:

- Traction: The boundaries EA and DE are subjected to an exact traction (Eq. 6-1).
- Free surface: The boundary BC has zero stress in the radial direction.
- Symmetry: The boundaries AB and CD have no displacement in the normal direction.
Table 6-1: Parameters used for elastic plate with a circular hole.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>R</th>
<th>L</th>
<th>E</th>
<th>υ</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Value</td>
<td>1</td>
<td>4</td>
<td>$10^5$</td>
<td>0.3</td>
<td>10</td>
</tr>
</tbody>
</table>

Notice that a set of compatible units needs to be used for each parameter. The problem parameters, detailed in Table 6-1 are given in normalized units, taken from [15].

The exact stresses for an infinite plate with a circular hole under uniaxial tension, $T$, are given by Kirsch equations, obtained by using the Airy Stress function. In order to solve the problem in a finite rectangular domain, the exact traction is evaluated and imposed at the boundaries [18]:

$$
\sigma_{xx}(r, \theta) = T \left( 1 - \left( \frac{R}{r} \right)^2 \left( \frac{3}{2} \cos 2\theta + \cos 4\theta \right) + \frac{3}{2} \left( \frac{R}{r} \right)^4 \cos 4\theta \right)
$$

$$
\sigma_{yy}(r, \theta) = T \left( -\left( \frac{R}{r} \right)^2 \left( \frac{1}{2} \cos 2\theta - \cos 4\theta \right) - \frac{3}{2} \left( \frac{R}{r} \right)^4 \cos 4\theta \right)
$$

$$
\tau_{xy}(r, \theta) = T \left( -\left( \frac{R}{r} \right)^2 \left( \frac{1}{2} \sin 2\theta + \sin 4\theta \right) + \frac{3}{2} \left( \frac{R}{r} \right)^4 \cos 4\theta \right)
$$

(6-1)

The exact displacement field is also available [18] and will be used to compare the numerical results of this benchmark without recurring to any further post-processing:

$$
u(r, \theta) = -\frac{8R(1+\nu)T}{E} \left( \frac{r}{R(1+\nu)} \cos \theta + \frac{R}{r} \left( \frac{4}{(1+\nu)} \cos \theta + \cos 3\theta \right) - \left( \frac{R}{r} \right)^3 \cos 3\theta \right)
$$

$$
u(r, \theta) = -\frac{8R(1+\nu)T}{E} \left( \frac{r}{R(1+\nu)} \sin \theta + \frac{R}{r} \left( \frac{-2\nu}{(1+\nu)} \sin \theta + \sin 3\theta \right) - \left( \frac{R}{r} \right)^3 \sin 3\theta \right)
$$

(6-2)

**Fully Developed Flows through Ducts**

The axial velocity in fully developed laminar flows is governed by Poisson's equation (Eq. 4-6) with a source term which is proportional to a constant pressure gradient. This model also assumes a steady flow with a Newtonian fluid. Thus, the governing equation is given as:

$$
\nabla^2 u = \frac{1}{\mu} \frac{dp}{dz} = \text{const}
$$

(6-3)

where $u$ is the axial velocity, $\mu$ is the viscosity and $dp/dz$ is the pressure gradient along $z$.  


Two different boundary condition impositions are accounted for:

- Homogeneous Dirichlet BC: No slip condition on the stationary walls.
- Homogeneous Neumann BC: Symmetry condition to reduce the computational domain.

Seven different test cases were computed and compared with available analytical solutions [39]. For a circular duct of radius $r_0$, the PDE becomes an ordinary differential equation, with only radial dependency, with a velocity distribution, $u(r)$, and flow rate, $Q$:

$$u(r) = -\frac{1}{4\mu} \frac{dp}{dz} (r_0^2 - r^2)$$

and

$$Q = \int udA = \frac{\pi r_0^4}{8\mu} \left(-\frac{dp}{dz}\right)$$

(6-4)

In other geometries, the solution depends on two coordinates and it is obtained by several methods such as change of variables, separable equations or complex variable transformations. For illustration purposes, the velocity distribution and flow rate through a rectangular duct with horizontal length $2a$, and vertical length $2b$, with origin at the geometric centroid is also shown here:

$$u(x, y) = \frac{16a^2}{\mu \pi^2} \left(-\frac{dp}{dz}\right) \sum_{n=1}^{\infty} \left((-1)^{n-1} \frac{\cosh \left(\frac{(2n-1)\pi x}{2a}\right)}{\cosh \left(\frac{(2n-1)\pi b}{2a}\right)} \frac{\cos \left(\frac{(2n-1)\pi y}{2a}\right)}{(2n-1)^3}\right)$$

(6-5)

$$Q = \frac{4ba^3}{3\mu} \left(-\frac{dp}{dz}\right) \left(1 - \frac{192}{\pi^5 b} \sum_{n=1}^{\infty} \left(\frac{\tanh \left(\frac{(2n-1)\pi b}{2a}\right)}{(2n-1)^5}\right)\right)$$

For all cases a pressure gradient $dp/dz = -1 Pa/m$ was used with a viscosity $\mu = 0.1 Pa \cdot s$.

The dimensions of the geometries (Table 6-2) take the same notation as in [39] and use SI units:

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Circle</th>
<th>Rectangle</th>
<th>Ellipse</th>
<th>Concentric Annulus</th>
<th>Eccentric Annulus</th>
<th>Circular Sector</th>
<th>Equilateral Triangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>-</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>$\pi/6$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1/4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Potential flow around a tilted ellipse

A potential flow model assumes a steady and inviscid flow with an irrotational velocity field $V = (u, v)$, and defines a velocity potential, $\phi$, that directly satisfies continuity with (Eq. 4-7):

$$\nabla \cdot V = \nabla \cdot (\nabla \phi) = \nabla^2 \phi = 0$$

Three types of boundary conditions are imposed, as shown for the case of a circle (Fig. 6-2):

- Homogeneous Neumann BC (green): No penetration ($V \cdot n = 0$) at the body surface.
- Periodic BC (red): At $\theta = 0$, the mesh generated has two joined boundaries.
- Dirichlet with linear distribution: At far-field (blue), $\phi = U_\infty x$ to force that $V = \nabla \phi = (U_\infty, 0)$. This condition is typically imposed at $R_\infty \geq 5a$.

![Figure 6-2: Potential flow around a cylinder: geometry and BCs.](image)

For this example, the solution of the velocity potential is given by [40]:

$$\phi(r, \theta) = U_\infty \left( r + \frac{a^2}{r} \right) \cos \theta$$

(6-7)

The velocity components are obtained as the gradient velocity potential:

$$V_r = U_\infty \left( 1 - \frac{a^2}{r^2} \right) \cos \theta \quad \text{and} \quad V_\theta = -U_\infty \left( 1 + \frac{a^2}{r^2} \right) \sin \theta$$

(6-8)
The pressure can be obtained using Bernoulli’s principle from the far field to any point:

$$\|V\| = \sqrt{V_r^2 + V_\theta^2} \rightarrow p = \frac{1}{2} \rho_\infty \left[ U_\infty^2 - \|V\|^2 \right] + p_\infty \quad (6-9)$$

The velocity and pressure at the body surface are computed by substituting $r = a$, to get a tangential velocity, $V_t$, and a surface pressure, $p_s$, with the following distribution:

$$V_t = 2U_\infty |\sin \theta| \rightarrow p_s = \frac{1}{2} \rho_\infty U_\infty^2 \left( 1 - 4\sin^2 \theta \right) + p_\infty \quad (6-10)$$

According to d’Alembert’s Paradox, the resultant force over the cylinder, $F$, should be zero. It can be verified by integrating the pressure over the cylinder surface, as follows:

$$F = \oint -p_n dA \quad (6-11)$$

In the case of an ellipse (or any symmetric body with respect to the $x$ axis), the results should be identical. The results for a cylinder using normalized data from Table 6-3, are compared with the exact solution in the next section. Numerical results are also obtained for a tilted ellipse at an angle of attack of 45° with parameters shown in the same table, although the exact solution is not given.

| Table 6-3: Parameters used for potential flow example. |
|---------------------------------|-----|-----|
| Parameter | Circle | Ellipse |
| Normalized Value | | |
| a | 1 | 1 |
| b | 1 | 1/2 |
| α | 0 | π/4 |
| $R_\infty$ | 5 | 5 |
| $p_\infty$ | 0 | 0 |
| $\rho_\infty$ | 1 | 1 |
| $U_\infty$ | 1 | 1 |
**Transient Heat Conduction**

The following example was proposed as an assignment in an isogeometric analysis graduate course [36]. It involves a body with constant thermal diffusivity in which heat is being conducted transiently (Eq. 4-8) due to a difference of temperature in the boundaries, as shown in Fig. 6-3:

![Figure 6-3: Transient heat conduction: geometry and BCs.](image)

The domain geometry is a quarter annulus, with inner and outer radii of 1 cm and 2 cm, respectively. Boundaries at the inner and outer radii are subjected to a Neumann boundary condition with zero heat flux, whereas the lowest (blue) and leftmost (red) boundaries are subjected to Dirichlet BCs of \(-10^0\text{C}\) and \(100^0\text{C}\), respectively. The thermal diffusivity is 0.01 cm²/s and a solution is found at a final time 15s with a time step of 1s. An initial condition over the domain was set as a fixed temperature of \(30^0\text{C}\). The parameters are summarized in Table 6-4:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>(r_1)</th>
<th>(r_2)</th>
<th>(T_a)</th>
<th>(T_b)</th>
<th>(T_0)</th>
<th>(\lambda)</th>
<th>(t_f)</th>
<th>(\Delta t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1 cm</td>
<td>2 cm</td>
<td>(-10^0\text{C})</td>
<td>(100^0\text{C})</td>
<td>(30^0\text{C})</td>
<td>0.01 cm²/s</td>
<td>15s</td>
<td>1s</td>
</tr>
</tbody>
</table>

Since the exact solution for this example is not given, the results will be compared with a simulation using traditional finite element analysis in COMSOL Multiphysics.
**Linear Advection-Diffusion**

This linear advection-diffusion example, as proposed by Hughes [4][15], "... is frequently the starting point for research in fluids as many of the difficulties encountered in more complicated nonlinear systems, such as the Navier-Stokes equation, also appear in this simple linear setting. This is a model problem, but a rich one". In a simple unit square domain (shown in Fig. 6-4), a concentration is forced in parts of the boundary. As shown earlier in Eq. 4-9, the concentration is propagated by means of both diffusion and advection. The ratio of advection to diffusion determines which nature dominates, in FEA it is related to the element Péclet number:

$$\alpha = \frac{V h}{2D}$$  \hspace{1cm} (6-12)

where \( \alpha \) is the element Péclet number, \( \|V\| \) is the velocity magnitude, \( h \) is the edge-length of the element and \( D \) is the diffusion coefficient. When this dimensionless number is greater than one, advection dominates and diffusion only matters in boundary layers.

![Figure 6-4: Linear Advection-Diffusion: geometry and BCs.](image)

When the process is dominated by diffusion, the concentration is distributed smoothly near the discontinuity in the lower boundary FA and AB. However, when advection dominates, the concentration is abruptly distributed over the domain creating an internal layer at the same angle as the velocity field, as well as boundary layers in BC and CD. The problem will be simulated using normalized parameters with different mesh sizes and diffusion coefficients to analyze the behavior of the numerical schemes.
The results from this example are already available using IGA [15]. According to Hughes [4], the Galerkin method (in both FEA and IGA) "...becomes unstable and exhibits spurious oscillation (with completely inaccurate concentration values) due to the failure to account for the effect of the scales that are too small to be represented explicitly with the basis employed." In traditional FEA, this effect has been studied extensively, as reviewed by Oñate [41] and some of the most popular approaches (applicable to both traditional FEA and IGA) are Variational Multiscale (VMS) and streamline upwind/Petrov-Galerkin (SUPG).

One advantage of IGA over traditional FEA is that, due to the non-interpolatory nature of NURBS, increasing the spline order does not generate oscillations (Runge's phenomenon) in problems with discontinuities. [4]

For the case of finite volume and finite differences, a series of methods are reviewed in [34] and the best results are obtained by non-linear methods to compute the fluxes, since the linear methods either generate diffusive or dispersive errors. For illustrative purposes, this problem will be evaluated using a donor cell scheme in FVM, whereas in FDM a first order upwind scheme and a second order upwind scheme are used.

**Shape Optimization**

In this simple optimization example, the shape of a symmetrical duct with respect to the horizontal axis $x$ and the vertical axis $y$, is designed to maximize the flow rate, $Q$, while maintaining a fixed total area. A quarter of the shape is parameterized using NURBS with an initial iteration that forms a square duct, as seen in Fig. 6-5.

![Figure 6-5: Control points and NURBS geometry of the initial iteration.](image-url)
The design variables to be chosen are based on three control points' locations (B, C and D) as well as the weight of the control point that determines the shape Dirichlet boundary (C). Several constraints are imposed: the control point A is fixed at the original, control points D and B are only allowed to move along the boundary the same distance, and finally the control point C is allowed to move along a line at an angle of 45°. The design then involves three design variables ($x_B$, $r_C$ and $x_C$) with the normalized positions and parameters shown in Table 6-5:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$(x_A,y_A,w_A)$</th>
<th>$(x_B,y_B,w_B)$</th>
<th>$(x_C,y_C,w_C)$</th>
<th>$(x_D,y_D,w_D)$</th>
<th>$dp/dz$</th>
<th>$\mu$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalized Values</td>
<td>(0,0,1)</td>
<td>(x_B,0,1)</td>
<td>(x_C,x_C,w_C)</td>
<td>(0,x_B,1)</td>
<td>-1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The original iteration uses the following design variables: $x_B = 1$, $x_C = \frac{1}{2}$ and $w_C = 0$. The fixed area must be equal to $\frac{1}{2}$. In order to obtain the flow rate in, three refinements are used after each iteration is determined and a finite difference formulation for non-orthogonal mesh is utilized.

In mathematical terms, the optimization problem is stated as:

$$
\begin{align*}
\text{minimize} & \quad -Q(x_B, x_C, w_C) \\
\text{subject to:} & \quad \begin{cases}
\frac{1}{2} \leq x_B \leq 2 \\
\frac{1}{2} \leq x_C \leq 2 \\
0 \leq w_C \leq 1 \\
\int_{\Omega} dxdy = \frac{1}{2}
\end{cases}
\end{align*}
$$

The solution, to be obtained using particle swarm optimization, is expected to be comparable to a quarter circle, with radius $R = \sqrt{2/\pi}$, yielding the optimal NURBS design variables:

$$
\begin{align*}
x_B &= \sqrt{2/\pi} = 0.7979 \\
x_C &= \sqrt{2/\pi} = 0.7979 \\
w_C &= \cos(\pi/4) = \sqrt{2}/2 = 0.7071
\end{align*}
$$
7. Results and Discussion

**Plate with a Hole under (in-plane tension)**

The mesh used in this benchmark was originally developed by Hughes [4], and its data was obtained from Nguyen [18]. The original mesh and its control points are shown in Fig. 7-1:

![Initial control points and mesh for the plate with a hole benchmark.](image1)

After three knot insertion refinements, the parametric lines can be distinguished better. Both directions need to be constructed with at least second order splines in order to account for the curvature. The total number of nodes is 561, distributed over 32x16=512 elements.

![Refined mesh for the plate with a hole benchmark.](image2)
Figure 7-3: Horizontal displacement in the plate with a hole benchmark.

Figure 7-4: Vertical displacement in the plate with a hole benchmark.
There are two hindrances to notice about this mesh (Fig. 7-2). The first and most important is that for this particular problem no mesh refinement should be needed near the top left corner, but due to the non-uniform nature of the NURBS parameterization and the way the mesh is constructed, the elements generated during the knot insertion refinement become relatively large near the lower left and the top right corners, as well as overly refined and skewed on the top left corner. Hughes [15] mentioned that although local refinement is not possible at the moment, this particular drawback could be solved by modifying the h-refinement routine. Another issue, is that the Jacobian determinant becomes zero on the top left corner which generates a problem at the post-processing part. Nguyen [18] temporally overcomes this issue by adding a small position shift in the computation of the Jacobian in that corner.

The numerical results for the displacements in Figs. 7-3 and 7-4 show that the normal displacements to the symmetrical boundaries are zero, as they should be, and overall a smooth contour plot is generated by the NURBS, despite the mesh problems mentioned earlier. Lastly, in Fig. 7-5, the top view of a 3D surface plot of the numerical resultant displacement along with a 3D scatter plot (black circles) shows that the numerical results completely agree with the exact solution on every node since the scatter plot intersects the surface plot in the three-dimensional space.

Figure 7-5: Verification of the results in the plate with a hole benchmark.
**Fully Developed Flows through Ducts**

The meshes used in these examples were created using the general guidelines given by Piegl [12]. The following geometries (Fig. 7-6) were used: a) circle, b) rectangle, c) ellipse, d) concentric annulus, e) eccentric annulus, f) angular sector, and g) equilateral triangle. For the simplest NURBS data used to construct these shapes, refer to Appendix A.

![Figure 7-6: Refined Meshes for the flows through ducts examples.](image)

Notice that, due to symmetry, the domains were divided in equal subdomains with four parts (circle, rectangle, ellipse, concentric annulus), two parts (concentric annulus and circular sector) and six parts (equilateral triangle). Also, some of the meshes are non-orthogonal (ellipse, eccentric annulus and equilateral triangle) and will not be usable with the 5 cells FVM approach.

![Figure 7-7: Results for the axial velocity in an eccentric annular duct using IGA.](image)
For illustration purposes, the results obtained for one of the geometries (eccentric annulus) will be shown in extension. The velocity profile contour plot and a mesh plot of the numerical results compared with the analytical results along the boundaries (black dots) are shown in Fig. 7-7.

![Figure 7-8: Qualitative view of the numerical results for the flows through ducts](image)

The solution over the entire domain is reconstructed from the numerical results from the studied subdomain, as shown in Fig. 7-8. The symmetry lines are marked as black, whereas the normalized velocity value is denoted by the color inside the contour plot, from blue (lowest velocity) to red (highest velocity).
In all geometries, mesh dependency was assessed by means of the error convergence rate, obtained as the slope of the log-log plot of the $L^2$ norm error vs the maximum element size (minimum diameter that encloses an element) as the mesh is refined. Table 7-1 shows an example where only with four quadratic IGA elements the results can be obtain with a flow rate accuracy greater that 99%, whereas four linear IGA elements only yield a 75% accuracy, as seen in Fig. 7-9. This agrees with the comparison of the $L^2$ norm error using linear and quadratic IGA elements, shown in Fig. 7-10. The advantage of quadratic elements over linear elements is clearly demonstrated, especially in domains with curved boundaries.
In most of the cases, the IGA numerical results (summarized in Table 7-2) reasonably reach the exact solution after one mesh refinement. A quadratic convergence rate was attained for the linear spline meshes, whereas quadratic meshes showed at least a cubic convergence rate.

Table 7-2: IGA mesh dependency summary for flows through ducts.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Spline order</th>
<th>Convergence Rate, $R^2 &gt; 0.99$</th>
<th>$U_{\text{max}}$, m/s</th>
<th>$Q$, m³/s</th>
<th>Sample: Mesh refinement #3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$H_{\text{max}}$, m</td>
</tr>
<tr>
<td>Circular</td>
<td>1</td>
<td>2.21</td>
<td>2.500</td>
<td>3.9270</td>
<td>0.230</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td></td>
<td></td>
<td>8.22E-09</td>
</tr>
<tr>
<td>Rectangular</td>
<td>1</td>
<td>1.96</td>
<td>1.139</td>
<td>1.1434</td>
<td>0.140</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.08</td>
<td></td>
<td></td>
<td>1.47E-03</td>
</tr>
<tr>
<td>Elliptical</td>
<td>1</td>
<td>2.23</td>
<td>1.000</td>
<td>0.7854</td>
<td>0.229</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.76</td>
<td></td>
<td></td>
<td>8.22E-09</td>
</tr>
<tr>
<td>Concentric Annulus</td>
<td>1</td>
<td>2.24</td>
<td>0.317</td>
<td>0.4947</td>
<td>0.209</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.96</td>
<td></td>
<td></td>
<td>2.97E-04</td>
</tr>
<tr>
<td>Eccentric Annulus</td>
<td>1</td>
<td>2.27</td>
<td>0.671</td>
<td>0.6667</td>
<td>0.219</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.53</td>
<td></td>
<td></td>
<td>2.70E-05</td>
</tr>
<tr>
<td>Circular Sector</td>
<td>1</td>
<td>2.04</td>
<td>0.147</td>
<td>0.0169</td>
<td>0.064</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.03</td>
<td></td>
<td></td>
<td>2.26E-03</td>
</tr>
<tr>
<td>Equilateral Triangle</td>
<td>1</td>
<td>2.22</td>
<td>0.278</td>
<td>0.0541</td>
<td>0.123</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.17</td>
<td></td>
<td></td>
<td>6.40E-04</td>
</tr>
</tbody>
</table>
The numerical solutions obtained using finite volume method (Table 7-3) and finite difference method (Table 7-4) were assessed in the same manner. Except that quadratic and linear meshes behaved the same way (in both FVM and FDM) and that the 5 cells finite volume method developed here only works in orthogonal meshes (4 out of 7 cases). Although the interior fluxes are computed using central difference, two different integration schemes are compared in FVM:

1. Midpoint rule integration on faces with linear approximation of the fluxes in boundaries.
2. Second order Gaussian quadrature with second order approximation in the boundaries.

### Table 7-3: FVM mesh dependency summary for flows through ducts.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Integ. order</th>
<th>Convergence Rate, R² &gt; 0.99</th>
<th>U max, m/s</th>
<th>Q, m³/s</th>
<th>Sample: Mesh refinement #3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H max, m</td>
</tr>
<tr>
<td>Circular</td>
<td>1</td>
<td>2.06</td>
<td>2.500</td>
<td>3.928</td>
<td>0.230</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td></td>
<td></td>
<td>0.20</td>
</tr>
<tr>
<td>Rectangular</td>
<td>1</td>
<td>1.91</td>
<td>1.139</td>
<td>1.143</td>
<td>0.140</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.76</td>
<td></td>
<td></td>
<td>0.33</td>
</tr>
<tr>
<td>Concentric Annulus</td>
<td>1</td>
<td>2.07</td>
<td>0.317</td>
<td>0.495</td>
<td>0.209</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3.11</td>
<td></td>
<td></td>
<td>0.79</td>
</tr>
<tr>
<td>Circular Sector</td>
<td>1</td>
<td>1.81</td>
<td>0.147</td>
<td>0.129</td>
<td>0.0645</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.86</td>
<td></td>
<td></td>
<td>0.34</td>
</tr>
</tbody>
</table>

### Table 7-4: FDM mesh dependency summary for flows through ducts.

<table>
<thead>
<tr>
<th>Geometry</th>
<th>Conv. Rate, R² &gt; 0.99</th>
<th>U max, m/s</th>
<th>Q, m³/s</th>
<th>Sample: Mesh refinement #3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H max, m</td>
</tr>
<tr>
<td>Circular</td>
<td>2.10</td>
<td>2.500</td>
<td>3.927</td>
<td>0.23</td>
</tr>
<tr>
<td>Rectangular</td>
<td>1.85</td>
<td>1.139</td>
<td>1.143</td>
<td>0.14</td>
</tr>
<tr>
<td>Elliptical</td>
<td>2.10</td>
<td>1.000</td>
<td>0.785</td>
<td>0.229</td>
</tr>
<tr>
<td>Concentric Annulus</td>
<td>2.13</td>
<td>0.317</td>
<td>0.494</td>
<td>0.209</td>
</tr>
<tr>
<td>Eccentric Annulus</td>
<td>2.37</td>
<td>0.672</td>
<td>0.667</td>
<td>0.219</td>
</tr>
<tr>
<td>Circular Sector</td>
<td>1.62</td>
<td>0.147</td>
<td>0.017</td>
<td>0.064</td>
</tr>
<tr>
<td>Equilateral Triangle</td>
<td>1.83</td>
<td>0.278</td>
<td>0.054</td>
<td>0.123</td>
</tr>
</tbody>
</table>
One really important thing to notice when assessing the errors obtained either by FVM or FDM is that the discrete error should not be compared by percentage (relative to exact solution) but instead compared to the truncation error of the discretization scheme. For instance, the results for the flow in a rectangular duct using the simplest FVM, with a mesh refined three consecutive times, produce a flow rate error of 1.65% (Table 7-3), but the discrete percentage error reaches nearly 20% in the Dirichlet corner, as seen in the blue plot of Fig. 7-11. At first sight, this appears to be caused by a combination of three factors:

1. A high velocity gradient near the Dirichlet corner.
2. Flux approximations near Dirichlet boundary (in case of FVM).
3. A relatively small velocity magnitude near Dirichlet boundary.

However, the second argument can be discarded because the effect is still observed using FVM with a second order approximation of the fluxes at the Dirichlet boundaries. Also, a comparable behavior (near the boundary) is seen with FDM. Both methods use a central difference scheme (2nd order approximation) and therefore the discrete absolute error should always be bounded by the element side length squared, as seen in the red plot of Fig. 7-11. In this mesh, the nearest point near the Dirichlet corner has a velocity magnitude of 0.0368 m/s. For an error near 5% (0.0018 m/s), the mesh size (h=0.140m) should be smaller than $\sqrt{0.0018} = 0.0429$m. However, further refinement generates points even nearer to the corner, with smaller magnitudes, thus the effect will still be present. The bottom line is to compare absolute error, not relative error.
In order to compare the results obtained by these methods, two geometries are taken as examples, one has an orthogonal grid (Fig. 7-12) whereas the other has a non-orthogonal grid (Fig. 7-13). As expected, the quadratic IGA yields the most accurate results for every case. For meshes with more than two refinements, the linear IGA and the FDM yield the same results; while for coarser meshes, the FD has a bigger error. This is since in IGA, one of the parametric coordinates (angular) is required to be quadratic in order to preserve the geometry while the other parametric coordinate (radial) can be chosen as linear or quadratic. In coarser meshes, the second order integration is preferable for FVM.

![Concentric Annulus: Comparison of methods](image)

**Figure 7-12: Comparison of methods in concentric annulus problem.**

![Eccentric Annulus: Comparison of methods](image)

**Figure 7-13: Comparison of methods in eccentric annulus problem.**
Potential Flow around a tilted ellipse

For this example, two cases were analyzed, first a cylinder (as a benchmark) and then a tilted ellipse. This was made only using finite difference for two important reasons:

- A non-orthogonal mesh is created for the ellipse (5 cells FVM formulation is not valid).
- In IGA, the application of Dirichlet BCs on physical nodes that are non-interpolatory requires special treatment either as a weak imposition (with parameters to be selected) [42] or by a spline interpolation method [15] using the control points.

The results were evaluated with three different meshes shown in Fig. 7-14; the finest is chosen. For the coarsest grid, the cylinder surface included 9 points, whereas for the finest grid 33 points are evaluated. The solutions for all three meshes are in agreement with the exact solution, in all points except at the highest velocity positions \((\theta = \pm 90^\circ)\) where an overshoot (see Fig. 7-16) was created in the horizontal velocity computation. This effect was diminished for finer grids.

![Figure 7-14: Meshes used for potential flow around a cylinder.](image)

After obtaining the velocity potential (Fig. 7-15), its gradient was computed to obtain the velocities. As expected, the velocity profiles (Figs. 7-16 and 7-17), along the cylinder are symmetrical, yielding a symmetric pressure distribution (Fig. 7-18) and resultant zero net force. The same effect is observed in an ellipse with zero angle of attack (Fig. 7-19) or any other geometry that is symmetrical with respect to the \(x\) axis, according to d'Alembert's paradox.
Figure 7-15: Velocity potential lines for the potential flow around a cylinder.

Figure 7-16: Horizontal velocity of potential flow around a cylinder.
**Figure 7-17:** Vertical velocity of potential flow around a cylinder.

**Figure 7-18:** Pressure distribution of potential flow around a cylinder.

**Figure 7-19:** Pressure distribution of potential flow around an ellipse.
The analytical results for the potential flow around a tilted ellipse were not available; however, the numerical results can be assessed by verifying (Fig. 7-21) the orthogonality of the potential lines with respect to the surface of the ellipse as well as checking that the potential lines are completely vertical in the far field. This simulation used only the mesh shown in Fig. 7-20.

Figure 7-20: Mesh used in the tilted ellipse example.

Figure 7-21: Velocity potential lines of the tilted ellipse example.
The velocity components obtained in the potential flow around a tilted ellipse (Fig. 7-22) clearly show how the stagnation points changed compared to the ellipse with zero angle of attack. The pressure distribution is still periodic (Fig. 7-23) but now it is not symmetrical as the previous case. However, this should only be taken as a naïve example, since potential flow does not occur in nature, it is only a mathematical abstraction. According to current models, the lift generation process requires the existence of viscosity.
Transient Heat Conduction

This example uses the same mesh from Fig. 7-6 d). A transient heat conduction example including non-homogeneous boundary conditions was simulated using Rothe's method and discretizing the domain with 64 quadratic isogeometric elements.

As seen in the contour plot of the solution at the final time (Fig. 7-24), the isothermal lines are dependent of the angle as well, except at (θ=0, θ=π/2 and θ=π/4), indicating that the process has not reached steady state yet.

![Temperature contour plot of the transient heat conduction example.](image)

Notice that the non-homogeneous Dirichlet conditions could be easily imposed using IGA in this particular case because in those boundaries (θ=0 and θ=π/4) the control points are completely interpolatory. If the Dirichlet Boundaries were in the radial direction instead, the imposition would be much harder, as mentioned in the potential flow example.
The results successfully agree (Fig. 25) with a model created in COMSOL Multiphysics with comparable characteristics and using 380 quadratic triangular elements of traditional FEA.

Figure 7-25: Comparison of temperature profile at mean radius using IGA.

Figure 7-26: Comparison of temperature profile at mean radius using FVM.

The results using FVM (Fig. 7-26) required a finer mesh (at least 32 points) to fully capture the temperature distribution due to the averaging process (the mean radius is between two elements) and because the IGA solution is quadratic whereas a linear interpolation is made in the FVM.
Linear Advection-Diffusion

As mentioned in the Applications section, the numerical methods used this type of problem need to be modified in order to capture the phenomena at the appropriate length scale. This holds for FEA, IGA, FVM, FDM and even meshless methods [41]. In the following models, two cases are chosen, $D=10^{-1}$ ($\alpha=1/3$) for the diffusion-dominated case and $D=10^{-3}$ ($\alpha=100/3$) for the advection-dominated case, and 4 mesh refinements as default. For illustration purposes, in Fig. 7-27, a central difference FVM on NURBS is used in to observe the instability effects:

![Figure 7-27: Diffusion-dominated vs Advection-dominated in central difference FVM.](image)

Notice how the oscillations in the latter case reach concentration values between 4 and -1, which makes no physical sense. This error has exactly the same behavior as the examples using IGA [15]. After modifying the FVM using a donor cell algorithm for the advective flux computation, the results stabilized but added substantial numerical diffusion, as seen in Fig. 7-28:

![Figure 7-28: Diffusion-dominated vs Advection-dominated in donor cell FVM.](image)
Exactly the same effect of numerical diffusion is observed in Fig. 7-29, were the same problem and mesh are used in a finite difference with a first order upwind scheme:

![Figure 7-29: Diffusion-dominated vs Advection-dominated in first order upwind FDM.](image)

The only difference (observed in the advection-dominated plot) is that the plotting tool is only using centroid values to create the contour plot; therefore, the sharp boundary layer at the east and north boundaries is not shown, but it is still present due to the homogeneous Dirichlet BC.

One way to avoid the introduced numerical diffusion of these first order schemes is to refine the mesh, as shown qualitatively in Fig. 7-30 for finite differences upwind scheme, and quantitatively in Fig. 7-31 for the FVM using donor cell technique.

![Figure 7-30: Advection-dominated example using 1st order upwind FDM with 4, 5 and 6 refinements.](image)
However, mesh refinement implies additional computation and a better approach might be choosing a second order scheme, as shown Fig. 7-32, which cuts back numerical diffusion.

The drawback that comes with the adoption of a second order scheme is a dispersive error that can be quantified by means of an overshoot of the solution near the discontinuity. For instance, the examples shown in Fig. 7-32 have a maximum percentage overshoot of 2.27%, 1.59% and 0.646%, respectively. As it can be seen, this effect is reduced when the mesh is refined. If a solution is required with less numerical diffusion than a first order scheme but less numerical dispersion than a second order scheme is wanted, a more complex method needs to be chosen.
Shape optimization

Using the original control points shown in Fig. 7-33, the physical domain has the shape of a square, then the PSO algorithm moves along the design variable (DV) space (Fig. 7-34), to determine the optimal location. In each particle position, the iterated mesh is refined and evaluated to obtain the pseudo objective function (POF) value based on the flow rate and the constraints, using a finite difference discretization of Poisson's equation.

Figure 7-33: Initial iteration for the shape optimization example.

Figure 7-34: Particle evaluations inside the design variable space.
The swarm had a total of 15 particles and 10 iterations (time steps) were evaluated. Notice how particles are attracted towards the optimal location inside the design variable space, while still moving around randomly to avoid local minima. The history of best positions found and their respective fitness values are shown in Table 7-5 along with the convergence plot in Fig. 7-35.

**Table 7-5: Iteration history for PSO shape optimization.**

<table>
<thead>
<tr>
<th>Swarm Iteration</th>
<th>Best DV location</th>
<th>POF value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>XB</td>
<td>XC</td>
</tr>
<tr>
<td>1</td>
<td>0.6739</td>
<td>0.9513</td>
</tr>
<tr>
<td>2</td>
<td>0.6739</td>
<td>0.9513</td>
</tr>
<tr>
<td>3</td>
<td>0.8397</td>
<td>0.799</td>
</tr>
<tr>
<td>4</td>
<td>0.807</td>
<td>0.7492</td>
</tr>
<tr>
<td>5</td>
<td>0.807</td>
<td>0.7492</td>
</tr>
<tr>
<td>6</td>
<td>0.807</td>
<td>0.7492</td>
</tr>
<tr>
<td>7</td>
<td>0.7699</td>
<td>0.7912</td>
</tr>
<tr>
<td>8</td>
<td>0.7699</td>
<td>0.7912</td>
</tr>
<tr>
<td>9</td>
<td>0.7852</td>
<td>0.7671</td>
</tr>
<tr>
<td>10</td>
<td>0.7852</td>
<td>0.7671</td>
</tr>
</tbody>
</table>

**Figure 7-35: Convergence history for PSO shape optimization.**
The final shape obtained is marked as black in Fig. 7-36, compared to an exact quarter circle. The total flow rate obtained was 0.3753, and the area (computed by a second order Gaussian quadrature) is 0.4974, for a difference of about 0.5%.

![Figure 7-36: Control points and geometry for optimal design (black) vs an exact quarter circle (red).](image)

Although the obtained radius (0.7852) is within 2% of the expected value (0.7979), it is noticeable that the location of control point C (top right) is nearer the origin compared to the expected position. However, this effect is counteracted by a bigger weight value of that control point (0.9166) which finally allows the generation of a curve that is fairly similar to a circle.

The NURBS parameterization process clearly allows the manipulation of geometry by means of using the control points and their weights as design variables of the shape optimization routine. This also streamlines information exchange from the geometry design to the analysis procedure.

However, there are still some challenges, such as the parameterization of domain interiors, since every time the parameterization of the boundary is changed, the interior nodes (in this case obtained from the refinement process) must be recomputed.
8. Conclusions and Recommendations

Conclusions

The aim of this thesis was to explore and compare the results from IGA and FVM on geometries parameterized by NURBS, in different applications including fluid flow and heat transfer. In order to do that, a basic IGA framework was developed in MATLAB® and supplemented with a FVM extension for NURBS geometries. Several examples were set up as benchmarks and test cases to verify the proper performance and compare the results among the numerical methods.

In most cases, the IGA numerical results reasonably reached the exact solution after one mesh refinement, especially when quadratic elements were used. The IGA results show that a quadratic convergence order was attained for the linear spline meshes, whereas quadratic meshes showed at least a cubic convergence order. FVM and FDM exhibited a quadratic convergence.

For the Poisson solver, the linear IGA and the FDM yield the same results for meshes with more than two refinements; however, for coarser meshes, the FD had a significantly bigger error. In orthogonal meshes, the FVM with an improved integration scheme displayed better results than the FDM and a linear IGA, in terms of average flow rate error.

In the transient heat conduction case, the FVM (due to the averaging process) required a finer mesh to fully capture the inner distribution of the temperature compared to IGA, which uses a quadratic NURBS to describe the solution.

In advection-dominated problems, the same instability behavior was observed for every method. However, the implementation of a numerical scheme to deal with this issue seems to be simpler in FDM and FVM compared to IGA, where a variational multiscale method is usually adopted.

In general, the IGA approach is desirable when dealing with NURBS geometries since it takes advantage of the basis properties and the solution is given in terms of a continuous function whereas the FVM and FDM yield a discrete field of the solution. Nonetheless, in examples where a quick solution is required, or multiple iterations need to be simulated (such as shape optimization) the selection of these two last methods is appropriate.
**Limitations of Current Work**

Most of the CAD generated files used in engineering applications have multiple patches that are fitted together to be geometrically continuous. However, the work presented in this document was only based on geometry representation using single patch NURBS parameterization, since only simple geometries were needed for the examples shown.

In the finite volume formulation, shown in section 5, only orthogonal meshes are considered. Although many geometries can be represented using orthogonal meshes, which are frequently preferred in analysis, the addition of non-orthogonal cases would generalize the formulation. Nevertheless, this general approach requires both the normal and tangential fluxes at each face. For internal control volumes, this is easily done by considering the central difference scheme shown here to obtain the derivatives in both directions using a Moore neighborhood instead of a von Neumann neighborhood. The hindrance comes with the computations at the boundary, where the treatment gets convoluted, especially in the decomposition of Neumann BCs, and mixed boundaries in corners. This involves a tremendous number of conditional equations that would increase the length of the code and decrease its readability; therefore, it was avoided.

In isogeometric analysis, a special treatment is required for the imposition of Dirichlet boundary conditions in boundaries where the curve does not pass exactly through the control points. The examples shown here only cover the interpolatory cases, although a general case is desirable.

All the applications covered are based on linear equations with simplified coefficients and various assumptions. A more realistic formulation would consider coefficients that may be spatially dependent or based on constitutive models. It would also consider multiple effects being coupled, such as fluid-structure interaction or thermal dependency in fluid problems.

Additionally, a comprehensive comparison between the numerical methods would require testing the performance based on other variables such as computational cost, simulation time and the amount of useful data generated. This was not done because the primary feature intended in the code is readability, not performance, in order to promote interest in this relatively new subject while avoiding the additional work required to find and implement efficient algorithms.
**Recommended Future Work**

In addition to overcoming the limitations previously shown, there are multiple interesting points of future research that might bring successful outcomes, including:

- In the finite volume method on NURBS, the extensive amount of available schemes (such as advanced flux limiters) that are especially designed for particular problems could be extended to curvilinear coordinates and applied in geometries parameterized by NURBS.

- In the finite difference method on NURBS, since its implementation is straightforward, more advanced examples could be easily developed and a framework could be created to account for rapid prototyping of numerical schemes for use in academia and the industry. A numerical Hessian matrix computation of the NURBS geometry could also be useful.

- In isogeometric analysis, it would be interesting to verify the effect of degenerated boundaries in different problems and check if the formulation still holds at the degenerated point. For instance, when computing the Jacobian determinant in the linear elasticity example, a slight but arbitrary location shift was needed in order to avoid the determinant going to zero. This effect was not observed in the Poisson examples that used a geometry with a degenerated boundary; nevertheless, it is suitable to know if the degenerated elements still comply with the formal conditions of finite element analysis.

- In shape optimization, it would be beneficial to know how the optimal geometry results are affected when geometry enrichment using p-refinement and k-refinement is performed. Furthermore, the generation of orthogonal meshes could be developed by using an objective function that relates to non-orthogonality, such as the area integral of the absolute value of the non-orthogonal entry of the covariant metric tensor. The design variables would be the control points and the respective weights in the domain interior.
References


## Appendices

### Appendix A: Geometry Data

#### Quarter circle geometry data

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>Ξ</th>
<th>H</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>0.70711</th>
<th>1</th>
<th>0.70711</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Rectangle geometry data

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>Ξ</th>
<th>H</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Ellipse geometry data

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>Ξ</th>
<th>H</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>0.70711</th>
<th>1</th>
<th>0.70711</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Triangle geometry data

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>Ξ</th>
<th>H</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.28868</td>
<td>0.28868</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Concentric annulus geometry data

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>Ξ</th>
<th>H</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>0.70711</th>
<th>1</th>
<th>0.70711</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
</table>
Appendix B: MATLAB Codes

To avoid an overextended appendix, only a selection of the main routines is exposed here. The dependencies for each example are shown in Appendix C.

% IGA DUCT FLOW
clc; disp('IGA DUCT FLOW');

%%% INPUT DATA

% Physical properties:
global dpdx miu
dpdx=input('Pressure gradient, dp/dx= ');
miu=input('Dynamic viscosity, miu= ');

% Order and refinement:
order=input('Lowest element order (1-2), q= ');
ref=input('Refinement level (0-5): ');

% Select geometry
fprintf('
Geometries:
Circle (1)
Rectangle (2)
Ellipse (3)
Concentric annulus (4)
Eccentric annulus (5)
Circular sector (6)
Equilateral triangle (7)
');
geo=input('Geometry ID: ');

Circular sector geometry data
\[
\begin{array}{cccccccc}
p & 2 \\
q & 1 \\
\Xi & 0 & 0 & 0.5 & 0.5 & 1 & 1 & 1 \\
H & 0 & 0.5 & 1 & 1 \\
\end{array}
\]
\[
\begin{array}{cccccccccccc}
W & 1 & 1 & 1 & 1 & 1 & 1 & 0.9979 & 1 & 0.9979 & 1 & 0.9979 & 1 \\
B & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0.4957 & 0.4914 & 0.4830 & 1 & 1 & 0.9914 & 0.9829 & 0.9659 \\
\end{array}
\]

Eccentric annulus geometry data
\[
\begin{array}{cccccccccccc}
p & 2 \\
q & 1 \\
\Xi & 0 & 0 & 0.5 & 0.5 & 1 & 1 & 1 \\
H & 0 & 0.5 & 1 & 1 \\
\end{array}
\]
\[
\begin{array}{cccccccccccccccc}
W & 1 & 0.7071 & 1 & 0.7071 & 1 & 1 & 0.7071 & 1 & 0.7071 & 1 & 0.7071 & 1 \\
B & -1 & -1 & 0 & 1 & 1 & -0.875 & -0.875 & -0.125 & 0.625 & 0.625 & -0.75 & -0.75 & -0.25 & 0.25 & 0.25 \\
\end{array}
\]

0 1 1 1 0 0 0.75 0.75 0.75 0 0 0.5 0.5 0.5 0
a=input('a = '); if all(geo==[1 7])
    b=input('b = '); if geo==5
        c=input('c = '); end
end

%% GENERATE NURBS GEOMETRY
tic; cd NGG; fprintf('
Generating geometry...
'); switch geo
    case 1 % Circle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circ(a,order); k=4;
    case 2 % Rectangle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_rect(a,b,order); k=4;
    case 3 % Ellipse
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_ellip(a,b,order); k=4;
    case 4 % Concentric annulus
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_con(a,b,order); k=4;
    case 5 % Eccentric annulus
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_ecc(a,b,c,order); k=2;
    case 6 % Circular sector
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circs(a,b,order); k=2;
    case 7 % Equilateral triangle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_trian(a,order); k=6;
end

cd ../

%% REFINEMENT
disp('Refining mesh...'); for i=1:ref
    [KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end

%% ASSEMBLY
disp('Assembling global matrix...'); [AA,bb]=assem(KV_Xi,KV_Eta,p,q,B,W,eye(2)); bb0=bb*mu/-dpdx;

%% BCs
disp('Applying boundary conditions...'); [AA,bb]=applyBC(AA,bb,BC,KV_Xi,KV_Eta,p,q);

%% SOLVE
disp('Solving linear system...');
x=AA\bb;

% POST-PROCESSING

% Plot
fprintf('Plotting solution...

[SU,SV,SOL]=plot_x(x,KV_Xi,KV_Eta,p,q,B,W);
title('IGA solution'); xlabel('y'); ylabel('z'); zlabel('U');
axis equal; colorbar; view([0 90]); shading interp;

% Other simulation data
Umax=max(SOL(:)); Uave=(bb0'*x)/sum(bb0); Q=Uave*k*sum(bb0); DT=toc;
disp('Number of elements:');
disp((length(unique(KV_Xi))-1)*(length(unique(KV_Eta))-1));
disp('Max velocity:'); disp(Umax);
disp('Average velocity:'); disp(Uave);
disp('Volumetric flow rate:'); disp(Q);
disp('Elapsed time:'); disp(DT);

% Compare to analytic solution
figure; mesh(SU,SV,SOL); hold all; cd AS

switch geo
  case 1 % Circle
    r=(0:a/50:a);
    U=U_circle(r,a,dpdx,miu);
    plot3(r,0*r,U,'--ok','MarkerFaceColor','k');
    plot3(0*r,r,U,'--ok','MarkerFaceColor','k');
    Qt=Q_circle(a,dpdx,miu);
  case 2 % Rectangle
    y=0:a/50:a; z=0:b/50:b; U1=y*0; U2=z*0;
    for i=1:50
      U1(i)=U_rectangle(y(i),0,a,b,dpdx,miu);
      U2(i)=U_rectangle(0,z(i),a,b,dpdx,miu);
    end
    plot3(y,0*z,U1,'--ok','MarkerFaceColor','k');
    plot3(0+y,z,U2,'--ok','MarkerFaceColor','k');
    Qt=Q_rectangle(a,b,dpdx,miu);
  case 3 % Ellipse
    y=0:a/50:a; z=0:b/50:b;
    U1=U_ellipse(y,0,a,b,dpdx,miu);
    U2=U_ellipse(0,z,a,b,dpdx,miu);
    plot3(y,0*z,U1,'--ok','MarkerFaceColor','k');
    plot3(0+y,z,U2,'--ok','MarkerFaceColor','k');
    Qt=Q_ellipse(a,b,dpdx,miu);
  case 4 % Concentric annulus
    r=(b:(a-b)/50:a);
    U=U_concentric(r,a,b,dpdx,miu);
```matlab
plot3(r,0*r,U,'--ok','MarkerFaceColor','k');
plot3(0*r,r,U,'--ok','MarkerFaceColor','k');
Qt=Q_concentric(a,b,dpdx,miu);

case 5 % Eccentric annulus
y=(-a:a/50:a); U=0*y;
for i=1:length(y)
    U(i)=U_eccentric(0,y(i),a,b,c,dpdx,miu);
    if and(y(i)>-c-b,y(i)<-c+b)
        U(i)=NaN;
    end
end
plot3(y,0*y,U,'--ok','MarkerFaceColor','k');
Qt=Q_eccentric(a,b,c,dpdx,miu);

case 6 % Circular sector
r=(0:a/50:a); U=0*r;
for i=1:50
    U(i)=U_csector(r(i),0,a,b,dpdx,miu);
end
plot3(r,0*r,U,'--ok','MarkerFaceColor','k');
Qt=Q_csector(a,b,dpdx,miu);

case 7 % Equilateral triangle
z=(0:(a/sqrt(3))/50:a/sqrt(3));
U=U_triangle(0,z,a,dpdx,miu);zp=fliplr(z);
plot3(zp/2,zp*sqrt(3)/2,U,'--ok','MarkerFaceColor','k');
Qt=Q_triangle(a,dpdx,miu);
end

hold off; cd ../
title('IGA and analytic solutions');
xlabel('y'); ylabel('z'); zlabel('U');
disp('Flow rate error %:'); disp(abs(100*(Qt-Q)/Qt));
```

% Rothe's Method using IGA
p=2;
q=2;
KV_Xi = [0 0 0 1 1 1];
KV_Eta = [0 0 0 1 1 1];
W= [ 1 1/sqrt(2) 1 ... 
    1 1/sqrt(2) 1 ... 
    1 1/sqrt(2) 1 ];
B= [ 0 1; 1 1; 1 0; 
    0 3/2; 3/2 3/2; 3/2 0; 
    0 2; 2 2; 2 0 ];
% Refinement
for i=1:3
[KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end

%% ASSEMBLY
[AA,bb]=assem(KV_Xi,KV_Eta,p,q,B,W,[1 0; 0 1]); %stiffness matrix
MM=assem2(KV_Xi,KV_Eta,p,q,B,W,[1 0; 0 1]); %mass matrix

%% SETUP IC
xx=30*ones(length(AA),1);
[~,xx]=applyBC(AA,xx,[0 1 0 1],KV_Xi,KV_Eta,p,q);

%% SETUP DIRICHLET VALUES
xxD=applyBC2(KV_Xi,KV_Eta,p,q); xx=xx+xxD;

%% SOLVE
dt=1; tf=15; lambdab=0.01; C=MM+dt*lambda*AA; Cp=C;
for t=0:dt:tf
    fprintf('%2f s

',t)
    b=MM*xx-C*xxD;
    [Cp,b]=applyBC(Cp,b,[0 1 0 1],KV_Xi,KV_Eta,p,q);
    xx=Cp\b+xxD;
end

%% POST-PROCESSING

% Plot
[SU,SV,SOL]=plot_x(xx,KV_Xi,KV_Eta,p,q,B,W);
title('IGA solution'); xlabel('y'); ylabel('z'); zlabel('U'); axis equal; view([0 90]); shading interp; colorbar

% 5 CVs + simplest flux approximation ; mesh study
clc; disp('DUCT FLOW - FVM 5 CVs - MESH STUDY');

%% INPUT DATA

% Physical properties:
global dpdx miu
dpx=-1;
miu=.1;
% Order and refinement:
order=1;
DATA=zeros(9,7);

% Select geometry
geo=2;

a=1;
b=1/2+(geo==6)*(-1/2+pi/6);
c=1/4;

for ref=0:4
    %% GENERATE NURBS GEOMETRY
    tic; cd NGG; fprintf('
Generating geometry...
');
    switch geo
        case 1 % Circle
            [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circ(a,order); k=4;

        case 2 % Rectangle
            [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_rect(a,b,order); k=4;

        case 3 % Ellipse
            error('NOT AVAILABLE');

        case 4 % Concentric annulus
            [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_con(a,b,order); k=4;

        case 5 % Eccentric annulus
            error('NOT AVAILABLE');

        case 6 % Circular sector
            [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circs(a,b,order); k=2;

        case 7 % Equilateral triangle
            error('NOT AVAILABLE');
    end

cd ../

%% DEFINE SOURCE TERM
S=@(x,y)dpdx/miu;

%% REFINEMENT
disp('Refining mesh...');
for i=1:ref
    [KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end
%% Connectivity

% Unique knot values
uXi=unique(KV_Xi); uEta=unique(KV_Eta);

% Number of elements per direction
numel_Xi=length(uXi)-1; numel_Eta=length(uEta)-1;
numel_tot=numel_Xi*numel_Eta;

% Number of univariate basis functions
n=length(KV_Xi)-p-1; m=length(KV_Eta)-q-1;

% connectivity arrays
[eR_Xi,eR_Eta,~,~,eNO,iNO] = conn2(KV_Xi,KV_Eta,p,q,n,m,numel_Xi,numel_Eta);

% NURBS to projected coordinates
projcoord=B;
for i=1:(length(KV_Xi)-p-1)*(length(KV_Eta)-q-1)
    projcoord(i,:)=projcoord(i,:)*W(i);
end
projcoord2=[projcoord,W];

%% ASSEMBLY

disp('Assembling global matrix and applying BCs...');

% Compute grid quantities that are reused
dXi=diff(uXi(1:2)); dEta=diff(uEta(1:2)); % control volume sizes
en=@(i,j)i+(j-1)*numel_Xi; A=zeros(numel_tot); bb=A(:,1); bb0=bb;

% Internal CVs
for j=1:numel_Eta
    for i=1:numel_Xi
        e=i+(j-1)*numel_Xi; % element number
        xIE=eR_Xi(iNO(e,1),:); etaE=eR_Eta(iNO(e,2),:); % int. limits
        xIC=mean(xIE); etaC=mean(etaE); % centroids
        % Coefficient of fluxes
        % south face
        Eta=etaE(1); [~,~,detJ,Gu,~]=nJGu(xiC,Eta,p,q,KV_Xi,KV_Eta,B,W);
        Ks=-dXi*abs(detJ)*Gu(2,2);
        % west face
        Xi=xiE(1); [~,~,detJ,Gu,~]=nJGu(Xi,etaC,p,q,KV_Xi,KV_Eta,B,W);
        Kw=-dEta*abs(detJ)*Gu(1,1);
% north face
Eta=etaE(2); [~,~,detJ,Gu,~]=nJGu(xiC,Eta,p,q,KV_Xi,KV_Eta,B,W);
Kn=dXi*abs(detJ)*Gu(2,2);

% east face
Xi=xiE(2); [~,~,detJ,Gu,~]=nJGu(Xi,etaC,p,q,KV_Xi,KV_Eta,B,W);
Ke=dEta*abs(detJ)*Gu(1,1);

% RHS (source term)
bb(e)=Gquad2D(@(xi,eta)fun3(xi,eta,p,q,KV_Xi,
... ,xiE(1),xiE(2),etaE(1),etaE(2),2,2);
bb0(e)=abs(bb(e)*miu/dpdx);

eBC=[j==1 i==1 j==numel_Eta i==numel_Xi];
if all(~eBC) % Internal CVs coefficients
  A(e,en(i,j+1))=Kn/dEta; %n
  A(e,en(i,j-1))=-Ks/dEta; %s
  A(e,en(i+1,j))=Ke/dXi; %e
  A(e,en(i-1,j))=-Kw/dXi; %w
  A(e,e)=-(Kn-Ks)/dEta-(Ke-Kw)/dXi; %p
else % BOUNDARY
  % Identify BC index
  iBC=[eBC, and(eBC(1),eBC(2)), and(eBC(3),eBC(2)),
  ... ,and(eBC(1),eBC(4)), and(eBC(3),eBC(4))];
  if sum(eBC)==2
    iBC(1:4)=0;
  end
  iBC=find(iBC); BCpos={'S' 'W' 'N' 'E' 'SW' 'NW' 'SE' 'NE'};
  BCpos=BCpos{iBC};

  % SOUTH
  switch BCpos
    case {'S' 'SW' 'SE'}
      if BC(1) %Dirichlet
        A(e,e)=A(e,e)+2*Ks/dEta; %p
      end
      otherwise
        A(e,e)=A(e,e)+Ks/dEta; %p
        A(e,en(i,j-1))=-Ks/dEta; %s
      end

    % WEST
    switch BCpos
      case {'W' 'SW' 'NW'}
        if BC(2) %Dirichlet
          A(e,e)=A(e,e)+2*Kw/dXi; %p
        end
        otherwise
          A(e,e)=A(e,e)+Kw/dXi; %p
          A(e,en(i-1,j))=-Kw/dXi; %w
% NORTH
switch BCpos
    case {'N' 'NW' 'NE'}
        if BC(3) %Dirichlet
            A(e,e)=A(e,e)-2*Kn/dEta; %p
        end
    otherwise
        A(e,en(i,j+1))=Kn/dEta; %n
        A(e,e)=A(e,e)-Kn/dEta; %p
    end
end

% EAST
switch BCpos
    case {'E' 'SE' 'NE'}
        if BC(4) %Dirichlet
            A(e,e)=A(e,e)-2*Ke/dXi; %p
        end
    otherwise
        A(e,en(i+1,j))=Ke/dXi; %e
        A(e,e)=A(e,e)-Ke/dXi; %p
    end
end
end
end

%% SOLVE
disp('Solving linear system...');
X=A\bb;

%% POST-PROCESSING

% Plot
% Other simulation data
Umax=max(X); Uave=(bb0'*X)/sum(bb0); Q=k*X'*bb0; DT=toc;

cd AS
switch geo
    case 1 % Circle
        Qt=Q_circle(a,dpdx,miu);
    case 2 % Rectangle
        Qt=Q_rectangle(a,b,dpdx,miu);
    case 3 % Ellipse
        Qt=Q_ellipse(a,b,dpdx,miu);
case 4 % Concentric annulus
    Qt=Q_concentric(a,b,dpdx,miu);

case 5 % Eccentric annulus
    Qt=Q_eccentric(a,b,c,dpdx,miu);

case 6 % Circular sector
    Qt=Q_csector(a,b,dpdx,miu);

case 7 % Equilateral triangle
    Qt=Q_triangle(a,dpdx,miu);

end

cd ../

Qerr=abs(100*(Qt-Q)/Qt);

ERROR=zeros(1,numel_tot); Xiplot=ERROR; Etaplot=ERROR;
Xplot=ERROR; Yplot=ERROR;
Uexact=ERROR; Unum=ERROR;

% NURBS to projected coordinates
numel_Xi=(length(unique(KV_Xi))-1);
numel_Eta=(length(unique(KV_Eta))-1);
umeltot=numel_Xi*numel_Eta;

HMAX=hmax(p,q,KV_Xi,KV_Eta,B,W);

L2err=0;
for e=1:numeltot
    % ELEMENT CENTROID
    xiE=eR_Xi(iNO(e,1),:); etaE=eR_Eta(iNO(e,2),:); % integration limits
    xic=mean(xiE); etac=mean(etaE); % centroids (parametric)
    Xplot(e)=xic; Etaplot(e)=etac;

    % compute physical coordinates
    temp=surf_p(xic,p,KV_Xi,etaC,q,KV_Eta,projcoord2);
xp=temp(1)/temp(3); yp=temp(2)/temp(3);
    Xplot(e)=xp; Yplot(e)=yp;

    % ELEMENT AREA (parametric domain)
    Aelp=abs(diff(xiE)*diff(etaE));

    % JACOBIAN DETERMINANT
    [~,~,detJ,~,~]=nJGu(xiC,etaC,p,q,KV_Xi,KV_Eta,B,W);

    % NUMERICAL SOLUTION
    Unum(e)=X(e);

    % ANALYTICAL SOLUTION
    cd AS
switch geo

    case 1 % Circle
        r=norm([xp yp]);
        Gan=U_circle(r,a,dpdx,miu);

    case 2 % Rectangle
        Gan=U_rectangle(xp,yp,a,b,dpdx,miu);

    case 3 % Ellipse
        Gan=U_ellipse(xp,yp,a,b,dpdx,miu);

    case 4 % Concentric annulus
        r=norm([xp yp]);
        Gan=U_concentric(r,a,b,dpdx,miu);

    case 5 % Eccentric annulus
        Gan=U_eccentric(yp,xp,a,b,c,dpdx,miu);

    case 6 % Circular sector
        r=norm([xp yp]); t=atan(yp/xp);
        Gan=U_csector(r,t,a,b,dpdx,miu);

    case 7 % Equilateral triangle
        t=pi/6; L=a/(2*sin(pi/3));
        XY=( [cos(t) -sin(t); sin(t) cos(t)]*[xp; yp]-[0; L]).*[1; -1];
        Gan=U_triangle(XY(1),XY(2),a,dpdx,miu);

end
cd ../

Uexact(e)=Gan;

% L2 error
L2err=L2err+Aelp*(Uexact(e)-Unum(e))^2*abs(detJ);

end
L2err=sqrt(L2err);
DATA(:,ref+1)=[ref Umax Uave Q Qerr DT numeltot HMAX L2err]';

fprintf('
\n\n\n\n\nNUMERICAL RESULTS:\n\n');
fprintf('Refinement: '); disp(ref);
disp('Volumetric Flow rate percentage error: '); disp(Qerr);
disp('Maximum velocity: '); disp(Umax);
disp('Average velocity: '); disp(Uave);
end
% FLOW OVER AN ELLIPSE
clc; disp('Ideal flow over an ellipse:');

%% INPUT DATA

% Flow conditions
Uinf=input('Far-field speed, Uinf: ');
Pinf=input('Far-field pressure, Pinf: ');
rho=input('Fluid density, rho: ');
ang=input('Angle of attack (deg): ') * pi/180;

% Dimensions
a=input('Semi-major axis, a: ');
b=input('Semi-minor axis, b: ');
R=input('Far-field radius, R: ');
ref=input('Refinement level (3-5): ');

%% Geometry Definition

disp('Generating geometry...');
p=2; q=1;
KV_Xi=[0 0 0 1 1 2 2 3 3 4 4 4]/4;
KV_Eta=[0 0 1/2 1 1];

% weights
inW=[1 1/sqrt(2) 1 1/sqrt(2) 1 1/sqrt(2) 1 1/sqrt(2) 1 1/sqrt(2) 1];
outW=[1 1/sqrt(2) 1 1/sqrt(2) 1 1/sqrt(2) 1 1/sqrt(2) 1]
midW=(inW+outW)/2; W=[inW, midW, outW];

% inner control points
inB=[1 0; 1 1; 0 1; -1 1; -1 0; -1 -1; 0 -1; 1 -1; 1 0];
inB=[a*inB(:,1) b*inB(:,2)]; % apply eccentricity
outB=[R 0; R R; 0 R; -R R; -R -R; 0 -R; R -R; R 0];
midB=(inB+outB)/2; % middle line
B=[inB; midB; outB];
ROT=[cos(ang) -sin(ang); sin(ang) cos(ang)]; % rotation matrix
B=B*ROT;

% Refine mesh
for i=1:ref
    [KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end

%% Initialization and connectivity

% Unique knot values
uXi=unique(KV_Xi); uEta=unique(KV_Eta);

% number of nodes
nXi=length(uXi); nEta=length(uEta); ntot=nXi*nEta;
% step size
dXi=diff(uXi(1:2)); dEta=diff(uEta(1:2));

% step sizes of subgrid
sdXi=(KV_Xi(end)-KV_Xi(1))/1024;
sdEta=(KV_Eta(end)-KV_Eta(1))/1024;

% matrix system allocation
A=zeros(ntot); bb=zeros(ntot,1); nn=@(i,j) i+(j-1)*nXi;

% Plotting positions
Xplot=bb; Yplot=bb; Xiplot=bb; Etaplot=bb; invJ=zeros(2,2,ntot);

% Projected coordinates
projcoord=B;
for i=1:(length(KV_Xi)-p-1)*(length(KV_Eta)-q-1)
    projcoord(i,:)=projcoord(i,:)*W(i);
end
projcoord2=[projcoord,W];

%% ASSEMBLY
disp('Assembling global matrix and applying BCs...');
for j=1:nEta
    for i=1:nXi
        n=i+(j-1)*nXi; % current node number
        xi=uXi(i); eta=uEta(j); % parametric position

        % Physical coordinates
        temp=surf_p(xi,p,KV_Xi,eta,q,KV_Eta,projcoord2);
        Xplot(n)=temp(1)/temp(3); Yplot(n)=temp(2)/temp(3);

        % Compute metric
        [~,invJ(:,n),detJ,Gu,~]=nJGu(xi,eta,p,q,KV_Xi,KV_Eta,B,W);

        % periodic conditions
        ileft=(i~=1)*(i-1)+(i==1)*(nXi-1);
        iright=(i~=nXi)*(i+1)+(i==nXi)*2;
        xileft=(i~=1)*(xi-sdXi)+(i==1)*(uXi(end)-sdXi);
        xiright=(i==nXi)*(xi+sdXi)+(i==nXi)*(uXi(1)+sdXi);

        if and(j~=1,j~=nEta)
            % Contravariant metric tensor at five points subgrid
            [~,~,dJs,GuS,~]=nJGu(xi,eta-sdEta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJw,GuW,~]=nJGu(xileft,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJn,GuN,~]=nJGu(xi,eta+sdEta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJe,GuE,~]=nJGu(xiright,eta,p,q,KV_Xi,KV_Eta,B,W);
            % Derivatives of metrics using CDS on subgrid
            dM_dxi=(abs(dJe)*GuE-abs(dJw)*GuW)/(2*sdXi);
            dM_deta=(abs(dJn)*GuN-abs(dJs)*GuS)/(2*sdEta);
        end
    end
end
\% Internal nodes
A(n,nn(i,j+1))=Gu(2,2)/dEta^2+
(dM_deta(2,2)+dM_dxi(1,2))/(2*dEta*abs(detJ)); \% N
A(n,nn(iright,j))=Gu(1,1)/dXi^2+
(dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); \% E
A(n,nn(ileft,j))=Gu(1,1)/dXi^2-
(dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); \% W
A(n,nn(i,j-1))=Gu(2,2)/dEta^2-
(dM_deta(2,2)+dM_dxi(1,2))/(2*dEta*abs(detJ)); \% S
A(n,n)=-2*(Gu(1,1)/dXi^2+Gu(2,2)/dEta^2); \% P
A(n,nn(ileft,j-1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \% SW
A(n,nn(iright,j-1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \% SE
A(n,nn(ileft,j+1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \% NE
A(n,nn(iright,j+1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \% NW

bb(n)=0;

else \% BCS (other than periodic)

if j==1 \% Homogeneous Neumann in airfoil surface (j=1)

\% Compute fluxes coefficients
n0=[0; -1]; [-,~,~,Gu,~]=nJGu(xi,eta,p,q,KV_Xi,KV_Eta,B,W);
K=Gu*n0/sqrt(Gu(2,2));

\% CDS for d/dXi
A(n,nn(iright,j))=K(1)/(2*dXi);
A(n,nn(ileft,j))=-K(1)/(2*dXi);

\% 2nd order one-sided for d/dEta
A(n,n)=A(n,n)-3*K(2)/(2*dEta);
A(n,nn(i,j+1))=4*K(2)/(2*dEta);
A(n,nn(i,j+2))=-K(2)/(2*dEta);

b(n)=0;

elseif j==nEta
\% Far-field linear Dirichlet imposition (j==nEta)
A(n,n)=1; \%bb(n)=Uinf*Xplot(n);
end

end
end

\%
SOLVE
disp('Solving linear system...');
x=A\bb;

\%
POST-PROCESSING
disp('Post-processing...'); figure(1);
contour(reshape(Xplot,nXi,nEta),reshape(Yplot,nXi,nEta),reshape(x,nXi,nEta));
colorbar; view([0 90]); axis equal; xlabel('x'); ylabel('y');
title('Potential lines')

% Approximate fluxes at the boundary (2nd order)
dPHI_dXi=zeros(1,nXi); dPHI_dEta=zeros(1,nXi); j=1;
U=zeros(nXi,1); V=U; Vt=U;
for i=1:nXi
  % PARAMETRIC
  dPHI_dEta(i)=(-3*x(nn(i,j))+4*x(nn(i,j+1))-1*x(nn(i,j+2)))/(2*dEta);
  ileft=(i~=1)*(i-1)+(i==1)*(nXi-1); iright=(i~=nXi)*(i+1)+(i==nXi)*2;
  dPHI_dXi(i)=(x(nn(iright,j))-x(nn(ileft,j)))/(2*dXi);
end

% PHYSICAL DOMAIN
  grad=invJ(:,:,nn(i,j))'*[dPHI_dXi(i); dPHI_dEta(i)];
  U(i)=grad(1); V(i)=grad(2); Vt(i)=norm([U(i) V(i)]);
end

% Compute additional variables
P=Pinf+rho*(Uinf^2-Vt.^2)/2;
CP=(P-Pinf)/(1/2*rho*Uinf^2);

% Plot velocities as a function of the angle
Xp=Xplot(1:nXi); Yp=Yplot(1:nXi);
th=180/pi*(atan2(Yp,Xp)+2*pi*(atan2(Yp,Xp)<0));
figure(2); plot(th,U,'ob'); grid on; xlim([0 360]);
xlabel('theta (deg)'); ylabel('Horizontal Velocity');
figure(3); plot(th,V,'or'); grid on; xlim([0 360]);
xlabel('theta (deg)'); ylabel('Vertical Velocity');
figure(4); plot(th,Vt,'og'); grid on; xlim([0 360]);
xlabel('theta (deg)'); ylabel('Tangential Velocity');
figure(5); plot(th,P,'ok'); grid on; xlim([0 360]);
xlabel('theta (deg)'); ylabel('Pressure');
figure(6); plot(th,CP,'ok'); grid on; xlim([0 360]);
xlabel('theta (deg)'); ylabel('Pressure Coefficient');

% DUCT FLOW using FD
clc; disp('DUCT FLOW using FD');

% INPUT DATA
% Physical properties:
  global dpdx miu
dpx=global dpdx miu
  dpdx=global dpdx miu
  dpdx=input('Pressure gradient, dp/dx = ');
miu=input('Dynamic viscosity, miu= ');

% Order and refinement:
order=input('Lowest spline order (1-2), q= ');
ref=input('Refinement level (2-5): ');

% Select geometry
fprintf('nGeometries:
Circle (1)
Rectangle (2)
Ellipse (3)
Concentric annulus (4)
Eccentric annulus (5)
Circular sector (6)
Equilateral triangle (7)\n\n');
geo=input('Geometry ID: ');

a=input('a= ');
if all(geo==[1 7])
    b=input('b= ');
    if geo==5
        c=input('c= ');
    end
end

% GENERATE NURBS GEOMETRY
tic; cd NGG; fprintf('nGenerating geometry...\n');
switch geo
    case 1 % Circle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circ(a,order); k=4;
    case 2 % Rectangle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_rect(a,b,order); k=4;
    case 3 % Ellipse
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_ellip(a,b,order); k=4;
    case 4 % Concentric annulus
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_con(a,b,order); k=4;
    case 5 % Eccentric annulus
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_ecc(a,b,c,order); k=2;
    case 6 % Circular sector
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_circs(a,b,order); k=2;
    case 7 % Equilateral triangle
        [p,q,KV_Xi,KV_Eta,B,W,BC]=NGG_trian(a,order); k=6;
end

cd ../

% DEFINE SOURCE TERM
S=@(x,y)dpdx/miu;
%% REFINEMENT

disp('Refining mesh...');
for i=1:ref
    [KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end

% Unique knot values
uXi=unique(KV_Xi); uEta=unique(KV_Eta);

% number of nodes
nXi=length(uXi); nEta=length(uEta); ntot=nXi*nEta;

% step size
dXi=diff(uXi(1:2)); dEta=diff(uEta(1:2));

% matrix system allocation
A=zeros(ntot); bb=zeros(ntot,1); nn=@(i,j) i+(j-1)*nXi; degen=[1 1 1 1];

%% LOOP OVER ALL NODES

disp('Assembling FD matrix system and BCs...');
for j=1:nEta
    for i=1:nXi

        n=i+(j-1)*nXi; % current node number
        xi=uXi(i); eta=uEta(j); % current position

        nBC=[j==1 i==1 j==nEta i==nXi];

        if all(~nBC) % Internal node

            % step sizes of subgrid
            sdXi=(KV_Xi(end)-KV_Xi(1))/1024;
            sdEta=(KV_Eta(end)-KV_Eta(1))/1024;

            % Contravariant metric tensor at five points subgrid
            [~,~,detJ,Gu,~]=nJGu(xi,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,Gs,GuS,~]=nJGu(xi-sdEta,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,Gw,GuW,~]=nJGu(xi-sdXi,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,Gn,GuN,~]=nJGu(xi,eta+sdEta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,Ge,GuE,~]=nJGu(xi+sdXi,eta,p,q,KV_Xi,KV_Eta,B,W);

            % Derivatives of metrics using CDS on subgrid
            dM_dxi=(abs(dJe)*GuE-abs(dJw)*GuW)/(2*sdXi);
            dM_deta=(abs(dJn)*GuN-abs(dJs)*GuS)/(2*sdEta);

            % Internal nodes
            A(n,nn(i,j+1))=Gu(2,2)/dEta^2+...;
            (dM_deta(2,2)+dM_dxi(1,2))/(2*dEta*abs(detJ)); \% N
            A(n,nn(i+1,j))=Gu(1,1)/dXi^2+...;
            (dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); \% E
            A(n,nn(i-1,j))=Gu(1,1)/dXi^2+...;
            (dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); \% W
            A(n,nn(i,j-1))=Gu(2,2)/dEta^2+...;

        end
    end
end
\[
\frac{dM_\text{deta}(2,2)+dM_\text{dxi}(1,2)}{2*dEta*abs(detJ)}; \quad \text{S}
\]
\[
A(n,n)=-2\cdot\frac{Gu(1,1)}{dXi^2}+Gu(2,2)/dEta^2; \quad \text{P}
\]
\[
A(n,nn(i-1,j-1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \quad \text{SW}
\]
\[
A(n,nn(i+1,j-1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \quad \text{SE}
\]
\[
A(n,nn(i-1,j+1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \quad \text{NE}
\]
\[
A(n,nn(i+1,j+1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); \quad \text{NW}
\]

bb(n)=dpdx/miu;

else %BOUNDARY

% Dirichlet condition
dirBC=and(nBC,BC);
neuBC=and(nBC,~BC); % Neumann condition

if any(dirBC) %homogeneous Dirichlet
    A(n,n)=1; bb(n)=0;
else

    % contravariant metric tensor
    [~,~,~,Gu,~]=nJGu(xi,eta,p,q,KV_Xi,KV_Eta,B,W);

    % choose first Neumann BC
    neupos=find(neuBC,1,'first'); neuBC=((1:4)==neupos);

% outward normal vector (parametric domain)
    n0=[neuBC(4)-neuBC(2) neuBC(3)-neuBC(1)];

% Compute fluxes coefficients
    Kxi=n0*[sqrt(Gu(1,1)); Gu(1,2)/sqrt(Gu(2,2))];
    Keta=n0*[Gu(2,1)/sqrt(Gu(1,1)); sqrt(Gu(2,2))];

% Compute fluxes approximation
    dh=[dXi dEta]; s=[nBC(2)-nBC(4) nBC(1)-nBC(3)]; K=[Kxi Keta];

    for dir=1:2
        if s(dir)==0 % CDS
            di=(dir==1)*sign(2*s(1)+1);
            dj=(dir==2)*sign(2*s(2)+1);
            A(n,nn(i+di,j+dj))=K(dir)/(2*dh(dir));
            A(n,nn(i-di,j-dj))=-K(dir)/(2*dh(dir));
        else % 2nd order one-sided
            di=(dir==1)*s(1); dj=(dir==2)*s(2);
            A(n,n)=A(n,n)-3*K(dir)*s(dir)/(2*dh(dir));
            A(n,nn(i+di,j+dj))=4*K(dir)*s(dir)/(2*dh(dir));
            A(n,nn(i+2*di,j+2*dj))=-K(dir)*s(dir)/(2*dh(dir));
        end
    end
end
% RHS (homogeneous Neumann)
bb(n)=0;

% Check for degenerate side
if any(isnan(A(n,:)))
    degBC=[j==1,i==1,j==nEta,i==nXi];
    degen=and(degen,degBC);
    A(n,:)=0; bb(n)=0;
end
end

% Fix degenerate side
if sum(degen)==1
    switch find(degen)
        case 1 % south
            id1=nn(1:nXi,1); id2=sub2ind(size(A),id1,id1);
            id3=sub2ind(size(A),id1,nn(1*ones(1,nXi),1));
            A(id2)=1; A(id3)=-1; % set all to corner (1,1)

            % interpolate corner (1,1)
            n=nn(1,1); A(n,n)=-1;
            A(n,nn(1,2))=3/2; A(n,nn(nXi,2))=3/2;
            A(n,nn(1,3))=-3/2; A(n,nn(nXi,3))=-3/2;
            A(n,nn(1,4))=1/2; A(n,nn(nXi,4))=1/2;

        case 2 % west
            id1=nn(1,1:nEta); id2=sub2ind(size(A),id1,id1);
            id3=sub2ind(size(A),id1,nn(1*ones(1,nXi),1));
            A(id2)=1; A(id3)=-1; % set all to corner (1,1)

            % interpolate corner (1,1)
            n=nn(1,1); A(n,n)=-1;
            A(n,nn(2,1))=3/2; A(n,nn(nXi,2))=3/2;
            A(n,nn(2,3))=-3/2; A(n,nn(nXi,3))=-3/2;
            A(n,nn(2,4))=1/2; A(n,nn(nXi,4))=1/2;

        case 3 % north
            id1=nn(1:nXi,nEta); id2=sub2ind(size(A),id1,id1);
            id3=sub2ind(size(A),id1,nn(nXi*ones(1,nXi),nEta));
            A(id2)=1; A(id3)=-1; % set all to corner (nXi,nEta)

            % interpolate corner (nXi,nEta)
            n=nn(nXi,nEta); A(n,n)=-1;
            A(n,nn(1,nEta-1))=3/2; A(n,nn(nXi,nEta-1))=3/2;
            A(n,nn(1,nEta-2))=-3/2; A(n,nn(nXi,nEta-2))=-3/2;
            A(n,nn(1,nEta-3))=1/2; A(n,nn(nXi,nEta-3))=1/2;

        case 4 % east
            id1=nn(1:nXi,nEta); id2=sub2ind(size(A),id1,id1);
            id3=sub2ind(size(A),id1,nn(nXi*ones(1,nXi),nEta));
            A(id2)=1; A(id3)=-1; % set all to corner (nXi,nEta)

            % interpolate corner (nXi,nEta)
            n=nn(nXi,nEta); A(n,n)=-1;
            A(n,nn(nXi,nEta-1))=3/2; A(n,nn(nXi,nEta-1))=3/2;
            A(n,nn(nXi,nEta-2))=-3/2; A(n,nn(nXi,nEta-2))=-3/2;
            A(n,nn(nXi,nEta-3))=1/2; A(n,nn(nXi,nEta-3))=1/2;
    end
end
id1=nn(nXi,1:nEta); id2=sub2ind(size(A),id1,id1);
id3=sub2ind(size(A),id1,nn(nXi*ones(1,nXi),nEta));
A(id2)=1; A(id3)=-1; % set all to corner (nXi,nEta)

% interpolate corner (nXi,nEta)
n=nn(nXi,nEta); A(n,n)=-1;
A(n,nn(nXi-1,1))=3/2; A(n,nn(nXi-1,nEta))=3/2;
A(n,nn(nXi-2,1))=-3/2; A(n,nn(nXi-2,nEta))=-3/2;
A(n,nn(nXi-3,1))=1/2; A(n,nn(nXi-3,nEta))=1/2;

%% SOLVE LINEAR SYSTEM
disp('Solving linear system...');
x=A\bb;

%% POST-PROCESSING
% Get areas and solution at centroid

% Plot
fprintf('Plotting solution...
');
plot_solFD; DT=toc;

% Other simulation data
fprintf('Post-processing...
');
areaFD; % Area and centroid computation

% solution interpolation at centroid
Umax=max(x); Uave=(bb0*Unum')/sum(bb0); Q=k*Uave*sum(bb0);

disp('Number of nodes:');
disp(length(uXi)*length(uEta));
disp('Max velocity:'); disp(Umax);
disp('Average velocity:'); disp(Uave);
disp('Volumetric flow rate:'); disp(Q);
disp('Elapsed time:'); disp(DT);
disp('Flow rate error %:'); disp(abs(100*(Qt-Q)/Qt));
disp('L2 norm error:'); disp(L2err);

% Linear advection-diffusion
% 1st order upwind
fprintf('Linear advection-diffusion
');
fprintf('1st order upwind
');

% define velocity field and diffusivity
theta=pi/4;
V=[cos(theta); sin(theta)];
kappa=10^-3;
ref=6;

% UNIT RECTANGLE
p=1;
q=1;
KV_Xi=[0 0 1 1];
KV_Eta=[0 0 1 1];
W=[1 1 1 1]';
B=[0 0; 1 0; ...
   0 1; 1 1];

% REFINEMENT
for i=1:ref
    [KV_Xi,KV_Eta,B,W]=hrefine2D(p,q,KV_Xi,KV_Eta,B,W);
end

% Unique knot values
uXi=unique(KV_Xi); uEta=unique(KV_Eta);

% number of nodes
nXi=length(uXi); nEta=length(uEta); ntot=nXi*nEta;

% step size
dXi=diff(uXi(1:2)); dEta=diff(uEta(1:2));

% matrix system allocation
A=zeros(ntot); bb=zeros(ntot,1); nn=@(i,j) i+(j-1)*nXi;

for j=1:nEta
    for i=1:nXi
        n=i+(j-1)*nXi; % current node number
        xi=uXi(i); eta=uEta(j); % current position
        nBC=[j==1 i==1 j==nEta i==nXi];
        if all(~nBC) % Internal node
            % step sizes of subgrid
            sdXi=(KV_Xi(end)-KV_Xi(1))/1024;
            sdEta=(KV_Eta(end)-KV_Eta(1))/1024;
            % Contravariant metric tensor at five points subgrid
            [~,invJ,detJ,Gu,~]=nJGu(xi,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJs,GuS,~]=nJGu(xi,eta-sdEta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJw,GuW,~]=nJGu(xi-sdXi,eta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJn,GuN,~]=nJGu(xi,eta+sdEta,p,q,KV_Xi,KV_Eta,B,W);
            [~,~,dJe,GuE,~]=nJGu(xi+sdXi,eta,p,q,KV_Xi,KV_Eta,B,W);
            % Derivatives of metrics using CDS on subgrid
            dM_dx=abs(dJe)*GuE-abs(dJw)*GuW/(2*sdXi);
            dM_deta=abs(dJn)*GuN-abs(dJs)*GuS/(2*sdEta);
% Internal nodes

% DIFFUSION TERM
A(n,nn(i,j+1))=Gu(2,2)/dEta^2+...
   (dM_deta(2,2)+dM_dxi(1,2))/(2*dEta*abs(detJ)); % N
A(n,nn(i+1,j))=Gu(1,1)/dXi^2+...
   (dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); % E
A(n,nn(i-1,j))=Gu(1,1)/dXi^2+...
   (dM_dxi(1,1)+dM_deta(2,1))/(2*dXi*abs(detJ)); % W
A(n,nn(i,j-1))=Gu(2,2)/dEta^2+...
   (dM_deta(2,2)+dM_dxi(1,2))/(2*dEta*abs(detJ)); % S
A(n,n)=-2*(Gu(1,1)/dXi^2+Gu(2,2)/dEta^2); % P
A(n,nn(i-1,j-1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); % SW
A(n,nn(i+1,j-1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); % SE
A(n,nn(i+1,j+1))=(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); % NE
A(n,nn(i-1,j+1))=-(Gu(1,2)+Gu(2,1))/(4*dXi*dEta); % NW

% ADVECTION TERM
coef=-V'*invJ'/kappa;
A(n,n)=A(n,n)+coef(1)/dXi;
A(n,nn(i-1,j))=A(n,nn(i-1,j))-coef(1)/dXi;
A(n,n)=A(n,n)+coef(2)/dEta;
A(n,nn(i,j-1))=A(n,nn(i,j-1))-coef(2)/dEta;

% SOURCE TERM
bb(n)=0;
else % DIRICHLET BC
A(n,n)=1;
   if or(j==1,and(i==1,j<=nEta/5));
      bb(n)=1;
   end
end

%% SOLVE LINEAR SYSTEM
x=A\bb;

%% POST-PROCESSING
projcoord=B;
for i=1:(length(KV_Xi)-p-1)*(length(KV_Eta)-q-1)
    projcoord(i,:)=projcoord(i,:)*W(i);
end
projcoord2=[projcoord,W];

Xplot=zeros(1,ntot); Yplot=Xplot;
for i=1:nXi
    for j=1:nEta
% position parametric
xi=uXi(i); eta=uEta(j);
n=i+(j-1)*nXi;

% position physical
temp=surf_p(xi,p,KV_Xi,eta,q,KV_Eta,projcoord2);
xp=temp(1)/temp(3); yp=temp(2)/temp(3);
Xplot(n)=xp; Yplot(n)=yp;

end
end
dim=sqrt(length(x));
fig=surf(reshape(Xplot,dim,dim)',reshape(Yplot,dim,dim)',reshape(x,dim,dim)',
...    'FaceColor','interp','EdgeColor','none');
colorbar; axis equal; view(0,90)
disp('Overshoot:'); disp(100*(max(x)-1)/1)

% PSO SHAPE OPTIMIZATION
% POF: pseudo-objective function
% ns: number of swarm iterations
% np: number of particles
% DVlim: DV limits size(DVlim)=[nDV,2]

%% INPUT DATA
global DVlim dpdx miu
DVlim=[1/2 1; 1/2 1; 1/2 1];
dpdx=-1; miu=0.1;
ns=10;
np=15;

POF=fcnchk(@(DV)evalgeo_opt(DV), 'vectorized'); % inline function

%% LIMITS
nDV=size(DVlim,1);
Xl=DVlim(:,1)'; Xu=DVlim(:,2)';

%% INITIAL SWARM
X=zeros(np,nDV); V=X; % initialize matrices
F=zeros(np,1); % initialize function vector
for i=1:np
    X(i,:)=Xl+rand(1,nDV).*((Xu-Xl)); % position
    V(i,:)=(-abs(Xu-Xl)+2*rand(1,nDV).*abs(Xu-Xl)); % velocity
% function evaluation
end

FP=F; % initialize personal best solution
P=X; % initialize personal best position
FG=min(FP); % swarm best solution
G=P(F==FG,:); % swarm best position

%% SWARM LOOP
w=linspace(1,0.5,ns); % dynamic inertia parameter
C1=1.5; C2=2; % trust parameters
for j=1:ns
    for i=1:np
        % velocity update
        V(i,:)=w(j)*V(i,:)+C1*rand*(P(i,:)-X(i,:))+C2*rand*(G-X(i,:));
        X(i,:)=X(i,:)+V(i,:); % position update
        F(i)=POF(X(i,:)); % function evaluation

        if F(i)<FP(i) % update personal best
            FP(i)=F(i); % function value
            P(i,:)=X(i,:); % position
        end
    end
end

scatter3(X(:,1),X(:,2),X(:,3)); hold on
axis([DVlim(1,:) DVlim(2,:) DVlim(3,:)]))

% update global best
if min(FP)<FG
    FG=min(FP); % swarm best solution
    G=X(FP==FG,:); % swarm best position
end

disp('SWARM #:'); disp(j);
disp('Current solution is:'); disp(FG);
disp('Current location is:'); disp(G);
end

disp('Best solution is:'); disp(FG);
disp('Best location is:'); disp(G);
hold off;
Appendix C: Code Dependencies

Flows through ducts using IGA.

Transient heat conduction using IGA.
Plate with a hole example using IGA.

Flows through ducts using FVM on NURBS.
Transient heat conduction using FVM on NURBS.

Linear advection-diffusion using FVM on NURBS.
Flows through ducts using FDM on NURBS.

Linear advection-diffusion using FDM on NURBS.
Potential flow around a tilted ellipse using FDM on NURBS.

Shape optimization using PSO and FDM on NURBS.
Appendix D: Numerical Jacobian

The numerical Jacobian computation is done using second order finite difference schemes. Given parametric coordinates, \((\xi, \eta)\) and the NURBS geometry data, the following steps are done:

1. Determine step size in a subgrid for each coordinate.
2. For each direction, independently:
   i. If the point is at a boundary, use a central difference scheme.
   ii. Else, use a 2\textsuperscript{nd} order one-sided finite difference moving away from the boundary.
3. Store the solution in the appropriate entry of the Jacobian matrix.

This process can be simplified by defining two variables inside each direction loop:

- **Step size vector, \(\Delta h\):** It is a \(2 \times 1\) vector originally containing the chosen steps sizes for each direction, but sets to zero the entry that is not linked to the current direction.
- **Sign vector, \(\theta\):** It is only computed if the evaluation point is at a boundary in the current direction. The value is set to +1 for lower boundaries and −1 for upper boundaries.

Thus, the gradient with respect to the current coordinate reduce to:

- **Central difference scheme:**
  \[
  \frac{(x, y)_{(\xi, \eta) + \Delta h} - (x, y)_{(\xi, \eta) - \Delta h}}{2(\Delta h_1 + \Delta h_2)}
  \]

- **One-sided 2\textsuperscript{nd} order scheme:**
  \[
  \theta \frac{-3(x, y)_{(\xi, \eta)} + 4(x, y)_{(\xi, \eta)+\Delta h} - (x, y)_{(\xi, \eta)+2\Delta h}}{2(\Delta h_1 + \Delta h_2)}
  \]

Since the parametric coordinates are looped independently, this approach could be generalized to compute the gradient in each coordinate simultaneously or in several points in a parallel computation. Furthermore, it can be easily extended to three-dimensional geometries.
Appendix E: Laplacian Equation patch test

A common test for numerical methods is to solve the Laplace equation with a constant Dirichlet value on all the boundaries. For any arbitrary geometry, the numerical solution should be the same specified constant all over the domain. For this test, two geometries were used:

- A pipe, given by Vuong et al [17], described using the following parameters:

<table>
<thead>
<tr>
<th>p</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ξ</th>
<th>0.25</th>
<th>0.25</th>
<th>0.5</th>
<th>0.5</th>
<th>0.75</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
<th>1</th>
<th>1</th>
<th>0.7071</th>
<th>1</th>
<th>0.7</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0.7071</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- The eccentric annulus domain, as a non-orthogonal mesh example.

The isogeometric analysis program successfully passed the patch test, as shown below:

Patch test on pipe geometry.

Patch test on eccentric annulus geometry.