

Mimetic Spectral Element Method for 2D Potential Flow

Implementation, Test and Comparison

Shuming Liu

Master Thesis

Mimetic Spectral Element Method for 2D Potential Flow Implementation, Test and Comparison

by Shuming Liu

to obtain the degree of Master of Science,
in Aerospace Engineering and Wind Energy,
at the Delft University of Technology,
to be defended on Thursday, April 5th, 2018 at 10:00 AM.

Student number: 4490444
Project duration: June, 2017 – February, 2018
Thesis committee: Dr.ir. M.I. Gerritsma, TU Delft, supervisor
Prof.dr. S.J. Watson, TU Delft
Dr.ir. A.H. van Zuijlen, TU Delft

Faculty of Aerospace Engineering

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

List of Figures

2.1	Tangent vector v in tangent space $T_p\mathcal{M}$, taken from [1]	4
2.2	Tangent bundle over a sphere, taken from [2]	5
2.3	Operation of differential forms	7
2.4	Geometrical understanding of 2-form	8
2.5	Outer and inner-orientation of k-form in $\mathbb{R}^3$, taken from [3]	13
2.6	Inner-(bottom) and outer-orientation(top) of geometries in 2D	17
2.7	Cell complex sample	17
2.8	Mapping between reference domain and physical domain	19
2.9	Element example for transfinite mesh	20
2.10	Transfinite interpolation components	21
2.11	Generated transfinite mesh	21
2.12	Transfinite mesh for one of the test cases	21
3.1	Sample of basis functions to be used in this study	24
3.2	Nodal reconstruction of $u(x)$ using Lagrange polynomials	25
3.3	Reconstruction using edge function	27
3.4	Comparison of different nodes	28
4.1	Reference domain and differential forms defined	34
4.2	Local labeling and global labeling	38
4.3	Physical domain and boundaries for the test	40
4.4	Numerical solution for the manufactured case	41
4.5	L2-norm error for the manufactured case (polynomials order vs. error)	41
4.6	L2-norm error for the manufactured case (element size vs. error)	41
4.7	Convergence plot for simple manufactured solution	42
4.8	Potential contour of $\phi=x$	42
4.9	Potential contour of $\phi=(3-x)y(2-y)$	42
4.10	Boundary conditions for the case 1	43
4.11	Results for channel flow solution case	43
4.12	Boundary conditions for the case 2	44
4.13	Results for analytical solution case	45
4.14	Comparison between numerical results and exact solution	45
4.15	L2-norm error for the analytical case (polynomials order vs. error)	46
4.16	L2-norm error for the analytical case (element size vs. error)	46
4.17	Mimetic spectral element method for airfoil	46

Contents

1	Introduction	1
1.1	Thesis Outline	2
2	Background Theories	3
2.1	Differential Geometry	3
2.1.1	Manifold	3
2.1.2	Differential Form	6
2.1.3	Exterior Derivative	9
2.1.4	Generalized Stokes Equation	11
2.1.5	Hodge Operator	11
2.1.6	Double de Rham complex	13
2.1.7	Pullback Operator	14
2.1.8	Inner Product and Codifferential	15
2.2	Algebraic Topology	16
2.2.1	Cell, Chain, and Cochain	16
2.2.2	Coboundary Operator	18
2.3	Transfinite Mesh	18
3	Mimetic Spectral Implementation	23
3.1	Reduction Operator	23
3.2	Reconstruction Operator	24
3.3	Basis Function	24
3.3.1	Nodal Function	24
3.3.2	Edge Function	26
3.4	Quadrature	27
3.5	Multi-dimensional Case	28
3.6	Pullback Operator	30
3.7	Weak Form and Test Function	30
3.8	About Python	32
4	Poisson and Laplace Equation	33
4.1	Reference domain	33
4.2	Equation in Matrix Form	33
4.2.1	Derivation of $u^{(1)} = -d\star\phi^{(2)}$	35
4.2.2	Derivation of $du^{(1)} = f^{(2)}$	36
4.3	Multiple Elements	38
4.4	Manufactured solution	39
4.4.1	Results	40
4.5	Laplace Equation	42
4.5.1	Results	43
5	Conclusion and Recommendations	47

Abstract

The thesis aims to solve partial differential equations numerically using mimetic spectral element method. The method rewrites the PDEs with differential forms so the resulting equations are able to preserve geometrical characteristics in physics. The equation we focus is the Poisson equation, which can be utilized for potential flow problems.

The physical domain to be tested is a typical flow around cylinder domain. Due to the curved surface, a curvilinear mesh is generated by transfinite interpolation so that the mesh can be perfectly fitted to the body.

The implementation of the mimetic spectral element method involves a Python package called **MimeticFEM** developed by Group of Aerodynamics, Faculty of Aerospace Engineering. The package provides a series of handy functions and methods to build system equations efficiently. A manufactured solution is made for error analysis of the numerical results. In addition, both h-refinement and p-refinement are made to test the convergence rate.

1 | Introduction

In the field of wind energy, people are pursuing more efficient and more durable wind turbine designs. That requires the blade design to be optimized. In order to achieve such blade or airfoil design, reliable and accurate method of calculating the air flow is necessary. That is why the analyzing tools had been transformed from the blade element method to panel method and eventually we witnessed the increase of using CFD methods. At the same time, since the blade has a curved surface, the discretization on geometries also becomes more refined. In this thesis, we want to investigate a CFD method that is highly accurate and compatible with curved surface.

In the fluid dynamics, phenomenon are described in forms of partial differential equations and it is well known that most of the PDEs do not have exact solution thus numerical solutions are required. That is what CFD focuses on. Currently, there are several ways to solve them and they can be categorized either as Finite Volume Method (FVM), Finite Difference Method (FDM), or Finite Element Method (FEM). In this study, we use Mimetic Spectral Element Method (MSEM), which can be regarded as a variant of Spectral Element Method (SEM) and SEM itself can be deemed as a formulation of the finite element method that uses piecewise high order orthogonal polynomials as basis function other than regular piecewise linear functions as basis.

The mimetic spectral element method is known for its exponential convergence rate. In this thesis we aim to solve the Laplace equation with mimetic spectral element method on a curvilinear mesh. The general idea is that we rewrite the PDEs in interest using differential forms, the one that is able to preserve the geometrical information of the equation. Then, the discretization is implemented with help of algebraic topology to rewrite the system in terms of algebraic equations. Finally a set of discrete equations is established in a matrix form and is solved. In this study, we implement the whole process using a Python library called *MimeticFEM*, which is developed by the Aerodynamics Group of Faculty of Aerospace Engineering.

Differential geometry is used to reinterpret the vector calculus (vectors, differential operators, etc.) by associating physical variables with different geometrical objects, such as points, lines or surfaces. In differential geometry we no longer work with scalars and vectors, instead, they are replaced by the differential k -forms. The reason of not using simply scalars or vectors is that the geometric content is lost while generalizing the physical phenomenon. Geometric content is considered as an very important characteristic of physical variables. Though without the geometric content equations still hold, nevertheless, it would cause inconsistency in physics. Beside replacing scalar and vector, operators from vector calculus such as gradient, curl and divergence are also replaced. They become a single operator, d , the exterior derivative.

The algebraic topology is the discrete counterpart of the differential geometry, where the differential forms and exterior derivatives can all find its corresponding discrete representations. The first one who introduced the differential form to rewrite PDEs and use algebraic topology for discretization is Enzo Tonti [4]. In his introductory paper [5], he claimed that in computational mathematics, each physical variable is represented by one or more numbers but “they also have a physical meaning that mathematics deliberately ignores” and stressed that physical variables are not just numbers but has greater information on the physical phenomenon that should be taken into account while doing computational physics, thus concluded that computational physics is not completely equivalent to computational mathematics.

The curvilinear mesh is used in the study to preserve the physical domain intact. The test case is flow around cylinder, therefore the cylinder surface should be discretized in a way that the mesh can preserve the curved surface as much as possible. One way to do that is to refine the mesh with more smaller elements. But that would often require intensive computation resources and more importantly, no matter how the refinement is done to the surface, the generated mesh is still an approximation to the curved surface. Therefore we will use a curvilinear mesh to discretize the surface, which can effectively preserve the curved domain. The method to create the curvilinear mesh is called transfinite interpolation, or TFI. The merit is that it is an interpolation recipe that can generate grids conforming to specified boundaries, so one can create a mesh that exactly fits to the body.

1.1 Thesis Outline

In the thesis, our investigation includes main questions as following:

- Can the high order method be applied with the curvilinear boundaries?
- If so, how is the accuracy of the method?
- Can the code obtain an exponential convergence rate on the deformed physical domain?
- How is the computation efficiency of the `MimeticFEM` package?

At chapter 2, there is a basic introduction to differential geometry and algebraic topology concepts that will be used to rewrite and discretize the equations. In addition, we will discuss about how to generate the mesh with transfinite interpolation for deformed physical domain.

The chapter 3 is about the implementation of the mimetic spectral element method. First, we will talk about the transformation between differential geometry and algebraic topology. That is the mimetic projection operator that converts the differential forms in infinite dimensional space to the finite dimensional space. To implement such method, some important concepts of spectral element method will also be discussed, such as basis function and weak form. After that, it becomes possible to formulate the system matrix to be solved in the study.

In chapter 4, we implement the method to solve the Laplace equation, which is used in the potential flow. Since there exists an exact solution for potential flow around cylinder, a comparison between the numerical solution and the exact solution is possible to evaluate the overall performance of the method and the code.

The chapter 5 is conclusion and recommendation of this thesis where we conclude our work and point out the possible road map for future research.

2 | Background Theories

In this chapter, basic ingredients of the mimetic spectral element method are introduced. The method is based on differential geometry and its discretized counterpart – the algebraic topology. The general idea is to describe PDEs using differential form and then discretize with help of algebraic topology. A full and thorough discussion of them is impossible and beyond the scope of the thesis. For more in-depth discussion, it is recommended to refer some textbooks and papers as [6, 7] and for Poisson problem [3].

2.1 Differential Geometry

In mimetic spectral element method, governing equations, usually PDEs, will be rewritten in differential form. The reason to do so comes from the idea of assigning physical variables with certain geometries, such as points, lines and surfaces. It is believed that preserving geometric information in the PDEs could better represent the physical variables. The idea of combining variables with geometry is first proposed by Enzo Tonti in his 1972 paper [4].

For people that are not familiar with differential geometry, some concepts might seem abstract, but it should be highlighted that differential geometry is the foundation of the scheme and knowing them paves the way for the further understanding of other concepts.

2.1.1 Manifold

Manifold is the foundation of the differential geometry, where the differential forms are defined. One commonly mentioned definition is that the manifold is a topological space that locally resembles Euclidean space near each point [8]. Though in many cases the manifold is impossible to be visualized, one can regard it as a shape of any dimension – a string (1-dimensional manifold), a surface (2-dimensional manifold), or a volume (3-dimensional manifold) where locally, the space is a Euclidean space.

Tangent Space and Bundles

Over the manifold the tangent space is introduced. In short, tangent space is a vector space consisting of all tangent vectors in the manifold at a certain point. Consider a curve $\gamma(s)$ in a k -dimensional manifold. The curve can be parameterized as

$$\gamma(s) = (x^1(s), x^2(s), \dots, x^k(s)) , \quad (2.1)$$

where $x^1, \dots, x^k$ are the coordinate on the manifold and $s \in (-\epsilon, \epsilon) \subset \mathbb{R}$.

The tangent vector at point p along the curve will be derived as the derivative of the curve at point p : $s=p=0$, see Figure 2.1. Note that the vector does not belong to the manifold $\mathcal{M}^k$.

The tangent space, denoted as $T_p\mathcal{M}^k$ is introduced as a collection of all vectors that are tangent to the manifold at point p , where $\mathcal{M}^k$ indicates that the point lies in the k -dimensional manifold. In addition, the (coordinate) basis vector of the space is written as

$$\frac{\partial}{\partial x^i} := \underbrace{(0, 0, 0, \dots, 1, \dots, 0, 0, 0)}_k . \quad (2.2)$$

Thus the tangent vector can be expressed as

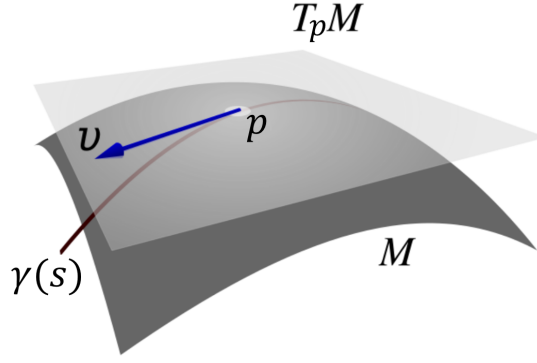


Figure 2.1: Tangent vector v in tangent space $T_p \mathcal{M}$, taken from [1]

$$v = d\gamma = \frac{\partial x^1}{\partial s} \frac{\partial}{\partial x^1} + \frac{\partial x^2}{\partial s} \frac{\partial}{\partial x^2} + \dots + \frac{\partial x^k}{\partial s} \frac{\partial}{\partial x^k} = \frac{\partial x^i}{\partial s} \frac{\partial}{\partial x^i} = v^i \frac{\partial}{\partial x^i} . \quad (2.3)$$

Here, the expression $\frac{\partial x^i}{\partial s} = v^i$ is called expansion coefficient.

One of the reasons that we use the partial derivative to express a basis is due to the fact that the components of vectors are not invariant under coordinate transformations on the manifold, where there is no such issue by using the derivative expression, because every transformation of basis would be simply applying chain rule. A coordinate free approach is helpful, as it guarantees that the law of physics will not change due to different coordinate frame. For example, suppose a new coordinate $(y^1, \dots, y^k)$ where the relationship with the original coordinate system is $y^i = y^i(x^1, \dots, x^k)$, then the new basis vector can be easily derived as

$$\frac{\partial}{\partial y_j} = \frac{\partial x^i}{\partial y_j} \frac{\partial}{\partial x^i} . \quad (2.4)$$

The purpose of introducing a tangent space and tangent vector is that it gives a way of expressing directional derivatives. The tangent vector v acts on a function f that lies on the manifold and gives the derivative of the function at a point (the rate of change of f in the direction of v).

For instance, given a curve $\gamma(s) = (x^1(s), x^2(s), \dots, x^k(s))$ on the manifold, then the tangent vector v at p along this curve is exactly the Equation 2.3.

Now a function f is defined on the manifold as $f(x^k(s))$ that maps point $(x^k(s))$ a real value. Obviously f can be written as a complex function as $f(\gamma(s))$. Take the derivative of f at point p ($s=0$), it gives

$$\frac{df(\gamma(s))}{ds} = \frac{\partial f}{\partial x^1} \frac{dx^1}{ds} + \frac{\partial f}{\partial x^2} \frac{dx^2}{ds} + \dots + \frac{\partial f}{\partial x^k} \frac{dx^k}{ds} \quad (2.5)$$

$$= \frac{dx^i}{ds} \frac{\partial f}{\partial x^i} = v^i \frac{\partial f}{\partial x^i} \quad (2.6)$$

$$:= v(f) . \quad (2.7)$$

Therefore the connection of derivative and vector is established.

After tangent space, the tangent bundle is then defined as a collection or union of tangent space and is denoted as $T\mathcal{M}$.

$$T\mathcal{M} := \cup_{p \in \mathcal{M}} T_p \mathcal{M} . \quad (2.8)$$

An example tangent bundle is illustrated as Figure 2.2. The tangent bundle is the space in which vector fields live. There is another concept, the section of the vector bundle. Essentially, a section assigns every point of the manifold a vector from the bundle. Therefore the section of the tangent bundle is a vector field. That corresponds with the regular definition of vector field, which is an assignment of a vector to every point in

space. In a differential manifold, the vector field assigns a tangent vector with each point on the manifold. One can image a low dimensional case, when the vector bundle is cut, then the “cross section” of the tangent vector bundle is a vector field. Or more loosely, take a tangent vector at every point and the vector field is formed.

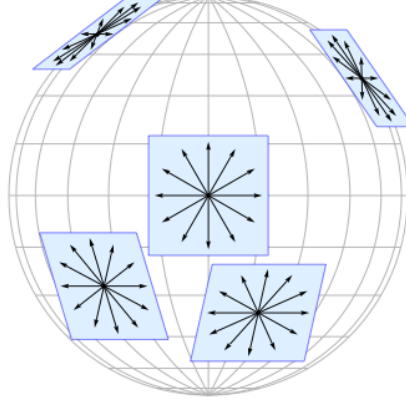


Figure 2.2: *Tangent bundle over a sphere, taken from [2]*

Dual Space

The dual space has different definitions. They can be found in many textbooks about differential geometry or linear algebra. Typically, dual space V^* is defined as a vector space consisting of all linear functionals of the vector space V . In other words, the dual space is a space of functionals that maps from V to $\mathbb{R}$.

Every linear vector space has its dual space and the element in the vector space v is called vector so the element in the dual space $\alpha \in V^*$ is called **covector**. A covector $\alpha \in V^*$, as a functional acts on a vector $v \in V$ and give a scalar product, $\alpha(v) \in \mathbb{R}$.

An important property is, if the basis of the vector space is $(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n)$, then one natural basis of this dual space is defined with upper script as $(\alpha^1, \alpha^2, \dots, \alpha^n)$, which satisfies

$$\alpha^i(v_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i=j, \\ 0 & \text{otherwise} \end{cases} \quad (2.9)$$

Consider the covector α expanded by a basis

$$\alpha = \lambda_1 \alpha^1 + \lambda_2 \alpha^2 + \dots + \lambda_n \alpha^n = \lambda_i \alpha^i . \quad (2.10)$$

The last expression is Einstein summation convention, and the vector $\mathbf{v}$ is also expanded by the basis

$$\mathbf{v} = x^1 \mathbf{v}_1 + x^2 \mathbf{v}_2 + \dots + x^n \mathbf{v}_n = x^j \mathbf{v}_j . \quad (2.11)$$

Then the operation of applying a covector to a vector is written as

$$\alpha(\mathbf{v}) = \lambda_i \alpha^i(x^j \mathbf{v}_j) = \lambda_i x^j \alpha^i(\mathbf{v}_j) = \lambda_i x^i , \quad (2.12)$$

and such operation is called duality pairing. Note that, though this operation is very similar to the inner product, it is not. The inner product is an operation in one vector space while the duality pairing is done between two vector spaces. The above expression can also be written as $\langle \alpha, \mathbf{v} \rangle$

$$\langle \alpha, \mathbf{v} \rangle := \lambda_i x^i = \alpha(\mathbf{v}) . \quad (2.13)$$

Cotangent Space and Bundles

As the name tells, the cotangent space is the dual of the tangent space and the same goes for cotangent bundle, which is the dual of the tangent bundle.

As a dual space, the cotangent space has the same dimension as the tangent space. An asterisk is marked to denote the cotangent space, i.e. $T_p^*\mathcal{M}$. If the basis of the tangent space is $(e_1|_p, e_2|_p, \dots, e_k|_p)$, then the basis of this dual space is defined with superscript as $(e^1|_p, e^2|_p, \dots, e^k|_p)$, where they are formed similarly with the Equation 2.9.

$$e^i(e_j) = \delta_{ij} . \quad (2.14)$$

When the basis of tangent space is a coordinate basis, which is $(\frac{\partial}{\partial x^1}, \frac{\partial}{\partial x^2}, \dots, \frac{\partial}{\partial x^k})$, the basis of cotangent space becomes $(dx^1, dx^2, \dots, dx^k)$. Therefore, any covector (the dual vector in the cotangent vector space) will be expressed by a combination of these basis functionals, such as covector $\alpha(p)$ at point $p \in \mathcal{M}$. For instance,

$$\alpha(p) = \alpha_1 dx^1 + \alpha_2 dx^2 + \dots + \alpha_k dx^k . \quad (2.15)$$

acts as a linear functional that assigns a vector to a real value. In this case, the vector in the tangent space is $v = v^i \frac{\partial}{\partial x^i}$ (Equation 2.3), it is applied by a covector so,

$$\alpha(v) = \alpha_j dx^j (v^i \frac{\partial}{\partial x^i}) = \alpha_j v^i dx^j (\frac{\partial}{\partial x^i}) = \alpha_i v^i , \quad (2.16)$$

where the last step applies the Equation 2.14. From the above we see that the covector is only meaningful when it is applied to a vector.

The cotangent bundle is a collection of cotangent space, and is denoted as $T^*\mathcal{M}$

$$T^*\mathcal{M} := \cup_{p \in \mathcal{M}} T_p^*\mathcal{M} , \quad (2.17)$$

and the differential 1-form is defined as a section of the cotangent bundle. So the differential forms could be interpreted as covectors. The differential 1-form is written as

$$\alpha^{(1)} = \alpha_i(x^1, x^2, \dots, x^n) dx^i . \quad (2.18)$$

Note the upper script (1) indicates it is a 1-form.

2.1.2 Differential Form

The differential form is defined as a way of measuring k -dimensional objects in the manifold, as the covector “eats” a vector and gives out a real number.

Although this definition of differential forms seems to be conceptual, there is some literature to help understand what the differential form is and its significance by visualization. In the literature, the definition becomes more informal but could give people a more intuitive view. Note there are different ways of visualizing the differential form. Burke [9] discussed the visualization in the field of electromagnetism. The book Gravitation [10] also gives a way to visualize the differential forms. Other illustrations can also be found in [11, 12].

One of the ways to understand the differential form is through the projection.

Consider a vector v , and a covector α . As mentioned, the covector gets a vector and gives out a real number. So it can be expressed in the Figure 2.3, where $\alpha(v)$ can be deemed as the projection of the vector v on the covector (differential 1-form) α . The operation is similar to vector multiplication, where the result is also a real number.

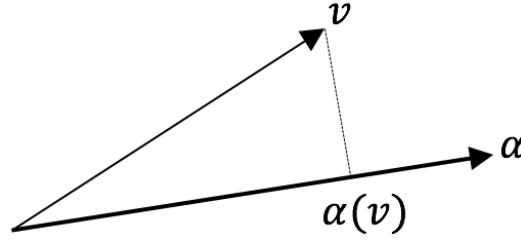


Figure 2.3: Operation of differential forms

It is interesting to point out that practically, covector and vector looks exactly the same. Thus to distinguish between vector and 1-form, we make one a row vector and another a column vector. That also satisfies the fact that only covector is permitted to apply to vector.

$$\alpha(v) = [\alpha_1, \alpha_2, \dots, \alpha_k] \begin{bmatrix} v^1 \\ v^2 \\ \vdots \\ v^k \end{bmatrix} = \alpha_i v^i . \quad (2.19)$$

Note the equation above corresponds with the Equation 2.16.

Wedge Product

Similarly, the 2-form “eats” two vectors and gives out a real number but that is a bit difficult to figure out. Because there is a **wedge product** operator involved. The wedge product (also known as the exterior product) of two 1-forms results in one 2-form and similarly a 3-form is the result of wedge product of three 1-forms or wedge product of one 1-form and one 2-form. Therefore, wedge product is actually a mapping to construct higher-dimensional forms from lower-dimensional forms.

$$\wedge : \Lambda^k(\mathcal{M}) \times \Lambda^l(\mathcal{M}) \rightarrow \Lambda^{(k+l)}(\mathcal{M}) , \quad (2.20)$$

where the space of k -form is denoted as $\Lambda^k(\mathcal{M})$.

The wedge product has the following properties,

- Linearity: $(\alpha^{(k)} + \beta^{(l)}) \wedge \gamma^{(m)} = \alpha^{(k)} \wedge \gamma^{(m)} + \beta^{(l)} \wedge \gamma^{(m)} ;$
- Associativity: $(\alpha^{(k)} \wedge \beta^{(l)}) \wedge \gamma^{(m)} = \alpha^{(k)} \wedge (\beta^{(l)} \wedge \gamma^{(m)}) ;$
- Antisymmetry: $\alpha^{(k)} \wedge \beta^{(l)} = (-1)^{kl} \beta^{(l)} \wedge \alpha^{(k)} .$

Note the third property leads to an important lemma. That is, when k is odd or $k > \frac{n}{2}$, then $a^{(k)} \wedge a^{(k)} = 0$ ■

Just like the 1-form can be expressed by the projection of a vector, the 2-form performs the same projection. For instance, consider two arbitrary vectors u and v , and two 1-forms α and β . The wedge product is defined as

$$(\alpha \wedge \beta)(u, v) := \alpha(u)\beta(v) - \alpha(v)\beta(u) , \quad (2.21)$$

and this $\alpha \wedge \beta$ is a 2-form.

This projection can be illustrated as the area formed by two vectors being projected to the plane formed by two covectors (one 2-form). Using the same example, each projection of vectors u and v , over the $\alpha - \beta$ plane, has two components – one for α and one for β . Therefore, two new vectors lying on the $\alpha - \beta$ plane are derived,

$$u' = (\alpha(u), \beta(u)) , \quad (2.22)$$

$$v' = (\alpha(v), \beta(v)) , \quad (2.23)$$

which form a new area that is exactly the projection operation, as shown Figure 2.4 where the shaded area is the cross product of u' and v' .

$$|u' \times v'| = \alpha(u)\beta(v) - \alpha(v)\beta(u) , \quad (2.24)$$

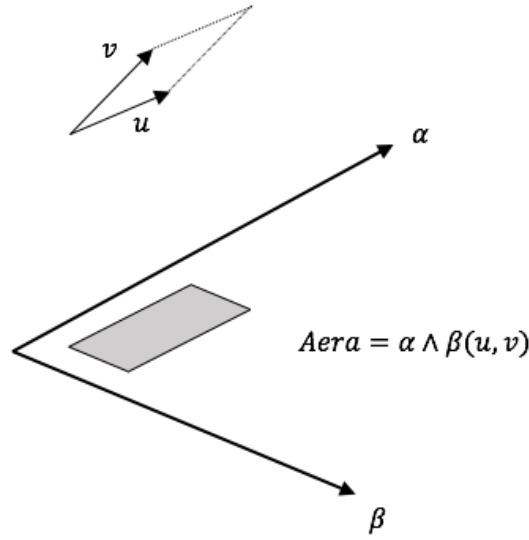


Figure 2.4: Geometrical understanding of 2-form

So there are k -forms corresponding to k -dimensional objects. Here we express the differential form with coordinate basis. For this study, the test case is a 2D problem, thus only 0, 1, 2-forms are needed to express points, edges and surfaces. However, here we assume $\mathbb{R}^3$ and there are three basis dx^1, dx^2, dx^3 for better illustration.

First, 0-form is a special case. Mathematically speaking, 0-form is merely a function, e.g. $\alpha^{(0)} = f(x^1, x^2, x^3)$. All operations applied to the function can be applied on the 0-form. The differential forms can be integrated over the manifold. 0-form in the differential geometry represents points in the domain so the integral of the 0-form is the collection of points in the domain, thus

$$\int_{\mathcal{P}_1, \dots, \mathcal{P}_k} \alpha^{(0)} = \sum_{i=1}^k f(\mathcal{P}_i) , \quad (2.25)$$

where there are k number of points $\mathcal{P}$ and the integration can be considered as a duality pairing shown as

$$\langle \alpha^{(0)}, \mathcal{P} \rangle := f(\mathcal{P}) . \quad (2.26)$$

It should be highlighted that this is not the duality pairing that was discussed before. This duality pairing is the about differential form and manifold while the initially discussed one is about vector and covector.

1-form is given by

$$\lambda^{(1)} = A(x^1, x^2, x^3)dx^1 + B(x^1, x^2, x^3)dx^2 + C(x^1, x^2, x^3)dx^3 . \quad (2.27)$$

In the above equation, $A(x^1, x^2, x^3)$, $B(x^1, x^2, x^3)$, $C(x^1, x^2, x^3)$ are functions called vector proxies. It should be noted that regular vector calculus only applies to these functions and ignore its basis vector dx^1, dx^2, dx^3

[1]. The integration of 1-form could be thought as to measure the length over a curve. Its integration is shown in the Equation 2.28.

$$\int_{\mathcal{C}} \lambda^{(1)} = \int_{\mathcal{C}} A(x^1, x^2, x^3) dx^1 + B(x^1, x^2, x^3) dx^2 + C(x^1, x^2, x^3) dx^3, \quad (2.28)$$

where $\mathcal{C}$ is a smooth curve.

In the formation of duality pairing, the expression can be rewritten as

$$\langle \lambda^{(1)}, \mathcal{C} \rangle := \int_{\mathcal{C}} \lambda^{(1)}. \quad (2.29)$$

2-form is given by wedge product as shown in the equation below

$$\eta^{(2)} = P(x^1, x^2, x^3) dx^2 \wedge dx^3 + Q(x^1, x^2, x^3) dx^3 \wedge dx^1 + R(x^1, x^2, x^3) dx^1 \wedge dx^2, \quad (2.30)$$

where P, Q, R are vector proxies and the basis are $dx^2 \wedge dx^3, dx^3 \wedge dx^1, dx^1 \wedge dx^2$. 2-forms can be integrated over the 2-dimensional manifold $\mathcal{S}$ as

$$\langle \eta^{(2)}, \mathcal{S} \rangle = \int_{\mathcal{S}} \eta^{(2)} = \int_{\mathcal{S}} P(x^1, x^2, x^3) dx^2 \wedge dx^3 + Q(x^1, x^2, x^3) dx^3 \wedge dx^1 + R(x^1, x^2, x^3) dx^1 \wedge dx^2. \quad (2.31)$$

As mentioned before, the 2-form is the wedge product of two 1-forms. A short example of formula expansion and simplification is, say

$$(3dx + dy) \wedge (xdx + 2dy) = 3xdx \wedge dx + 6dx \wedge dy + xdy \wedge dx + 2dy \wedge dy = (6 - x)dx \wedge dy. \quad (2.32)$$

If we continue to the 3-form, it is given by the expression of

$$\omega^{(3)} = G(x^1, x^2, x^3) dx^1 \wedge dx^2 \wedge dx^3. \quad (2.33)$$

The integration over a 3-dimensional manifold $\mathcal{V}$ is

$$\langle \omega^{(3)}, \mathcal{V} \rangle = \int_{\mathcal{V}} \omega^{(3)} = \int_{\mathcal{V}} G(x^1, x^2, x^3) dx^1 \wedge dx^2 \wedge dx^3. \quad (2.34)$$

Up to now, the concepts of differential forms have been introduced. Though some concepts seem to be unfamiliar, we are using them every day. For example the 2-form $dx^1 \wedge dx^2$, when we integrate the area S over a surface Ω in $\mathbb{R}^2$, the integration would be $S_{\Omega} = \int_{\Omega} dS$. The dS is the infinitesimal part of the S . In differential geometry, the dS is actually a 2-form – $dx^1 \wedge dx^2$. Moreover, by adding a scalar weighted function, say $A(x^1, x^2)$, the equation will be exactly the same as the regular integration equation. As a result, a number of references [13, 14] generalize the differential form as “something we can integrate”.

2.1.3 Exterior Derivative

We use differential forms to describe physical variables. Now comes the exterior derivative that describes the derivative of physical variables. The exterior derivative is defined to transform k -form into $(k+1)$ -form on the manifold.

$$d: \Lambda^k(\mathcal{M}) \rightarrow \Lambda^{(k+1)}(\mathcal{M}). \quad (2.35)$$

According to literature, the exterior derivative has several properties as

- $d(\alpha + \beta) = d\alpha + d\beta$;

- $d(\alpha^{(k)} \wedge \beta^{(m)}) = d\alpha \wedge \beta + (-1)^k \alpha \wedge d\beta$;
- $dd\alpha = 0$.

Here, we introduce the exterior derivatives of 0, 1, 2, 3-forms. Since 0-form is a scalar function, the exterior derivative of 0-form $\alpha^{(0)} = f(x^1, x^2, x^3)$ is given by

$$d\alpha^{(0)} = \frac{\partial f}{\partial x^1} dx^1 + \frac{\partial f}{\partial x^2} dx^2 + \frac{\partial f}{\partial x^3} dx^3 . \quad (2.36)$$

The expression of exterior derivative of 0-form, $d\alpha^{(0)}$ is similar to gradient operator. If the scalar function is $f(x^1, x^2, x^3)$, then the gradient is given by $\text{grad } f = \nabla f = \frac{\partial f}{\partial x^1} \mathbf{i} + \frac{\partial f}{\partial x^2} \mathbf{j} + \frac{\partial f}{\partial x^3} \mathbf{k}$.

For 1-form, the exterior derivative is derived as

$$\begin{aligned} d\gamma^{(1)} &= d(Adx^1 + Bdx^2 + Cdx^3) \\ &= d(Adx^1) + d(Bdx^2) + d(Cdx^3) \end{aligned} \quad (2.37)$$

$$= d(A \wedge dx^1) + d(B \wedge dx^2) + d(C \wedge dx^3) \quad (2.38)$$

$$= (dA) \wedge dx^1 + A \wedge \underbrace{(ddx^1)}_0 + (dB) \wedge dx^2 + B \wedge \underbrace{(ddx^2)}_0 + (dC) \wedge dx^3 + C \wedge \underbrace{(ddx^3)}_0 \quad (2.39)$$

$$\begin{aligned} &= \left(\frac{\partial A}{\partial x^1} dx^1 + \frac{\partial A}{\partial x^2} dx^2 + \frac{\partial A}{\partial x^3} dx^3 \right) \wedge dx^1 \\ &+ \left(\frac{\partial B}{\partial x^1} dx^1 + \frac{\partial B}{\partial x^2} dx^2 + \frac{\partial B}{\partial x^3} dx^3 \right) \wedge dx^2 \\ &+ \left(\frac{\partial C}{\partial x^1} dx^1 + \frac{\partial C}{\partial x^2} dx^2 + \frac{\partial C}{\partial x^3} dx^3 \right) \wedge dx^3 \end{aligned} \quad (2.40)$$

$$= \left(\frac{\partial C}{\partial x^2} - \frac{\partial B}{\partial x^3} \right) dx^2 \wedge dx^3 + \left(\frac{\partial A}{\partial x^3} - \frac{\partial C}{\partial x^1} \right) dx^3 \wedge dx^1 + \left(\frac{\partial B}{\partial x^1} - \frac{\partial A}{\partial x^2} \right) dx^1 \wedge dx^2 . \quad (2.41)$$

Equation 2.37 is derived due to linearity, and Equation 2.38 is derived by the fact that the function A can be regarded as a 0-form thus Adx^1 is actually the wedge product of the 0-form, A , and the 1-form, dx^1 . Later the Equation 2.39 refers to the third property of exterior derivative. The last step refers to the fact of wedge product $\alpha \wedge \beta = -\beta \wedge \alpha$ and $\alpha \wedge \alpha = 0$.

The final form of $d\gamma^{(1)}$ is similar to the *curl* operator. If the vector field is $\mathbf{u}(x^1, x^2, x^3) = A(x^1, x^2, x^3)\mathbf{i} + B(x^1, x^2, x^3)\mathbf{j} + C(x^1, x^2, x^3)\mathbf{k}$, then the vector proxies of $d\gamma^{(1)}$ are exactly the components of the *curl* $\mathbf{u} = \nabla \times \mathbf{u} = \left(\frac{\partial C}{\partial x^2} - \frac{\partial B}{\partial x^3} \right) \mathbf{i} + \left(\frac{\partial A}{\partial x^3} - \frac{\partial C}{\partial x^1} \right) \mathbf{j} + \left(\frac{\partial B}{\partial x^1} - \frac{\partial A}{\partial x^2} \right) \mathbf{k}$.

For 2-form $\eta^{(2)} = P(x^1, x^2, x^3)dx^2 \wedge dx^3 + Q(x^1, x^2, x^3)dx^3 \wedge dx^1 + R(x^1, x^2, x^3)dx^1 \wedge dx^2$, the exterior derivative of it is given by the expression of

$$d\eta^{(2)} = \left(\frac{\partial P}{\partial x^1} + \frac{\partial Q}{\partial x^2} + \frac{\partial R}{\partial x^3} \right) dx^1 \wedge dx^2 \wedge dx^3 . \quad (2.42)$$

The similarity with vector calculus also exists for the 2-form, which is about the divergence operator. It is noted that the expansion coefficient is the same as the *div* $\mathbf{V}$ ($\mathbf{V} = P\mathbf{i} + Q\mathbf{j} + R\mathbf{k}$).

Except the resemblance between ∇ operators and exterior derivatives mentioned above, such consistency has also been found with the $dd\alpha = 0$ feature, which is similar to the expression $\nabla \times (\nabla f) = 0$ in vector calculus.

The advantage of exterior derivative is clear – the gradient, curl, and divergence operator are replaced by a single d . The relationship can be depicted as below

$$\begin{array}{ccccccc} \{0\text{-form}\} & \xrightarrow{d} & \{1\text{-form}\} & \xrightarrow{d} & \{2\text{-form}\} & \xrightarrow{d} & \{3\text{-form}\} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \{\text{function}\} & \xrightarrow{\text{grad}} & \{\text{vector field}\} & \xrightarrow{\text{curl}} & \{\text{vector field}\} & \xrightarrow{\text{div}} & \{\text{function}\} \end{array}$$

2.1.4 Generalized Stokes Equation

Integrating the exterior derivative leads to the generalized Stokes theorem. The generalized Stokes equation is defined as

$$\int_{\Omega_{k+1}} d\alpha^{(k)} = \int_{\partial\Omega_{k+1}} \alpha^{(k)} , \quad (2.43)$$

where $\alpha^{(k)}$ is an arbitrary k -form in domain Ω_{k+1} . ∂ is a boundary operator thus $\partial\Omega_{k+1}$ is a k -dimensional domain.

The above equation implies that the integration of exterior derivative of k -form on Ω_{k+1} manifold is the same as the integral of k -form on the manifold boundaries. If $k=0$, the equation becomes gradient integral theorem. If $k=1$, the equation is regular Stokes theorem, and if $k=2$, the equation is divergence theorem.

For example, in case of $k=1$, the above equation becomes:

$$\int_S d\alpha^{(1)} = \int_{\partial S} \alpha^{(1)} , \quad (2.44)$$

which is expanded as:

$$\int_S \left(\frac{\partial C}{\partial x^2} - \frac{\partial B}{\partial x^3} \right) dx^2 \wedge dx^3 + \left(\frac{\partial A}{\partial x^3} - \frac{\partial C}{\partial x^1} \right) dx^3 \wedge dx^1 + \left(\frac{\partial B}{\partial x^1} - \frac{\partial A}{\partial x^2} \right) dx^1 \wedge dx^2 = \int_{\partial S} A dx^1 + B dx^2 + C dx^3 . \quad (2.45)$$

The equation tells integrating the exterior derivative of 1-form over a surface is equal to integrating the 1-form along the boundaries of the surface, which is exactly the classical Stokes theorem:

$$\int_S \nabla \times \mathbf{F} d\mathbf{S} = \int_{\partial S} \mathbf{F} d\mathbf{r} , \quad (2.46)$$

where the term of the curl in equation above is the term of the exterior derivative of 1-form, and is equal to integrating the vector along the lines $\mathbf{r}$ (at boundaries of the surface). Therefore, it is natural to relate 1-form to lines. In case of $k=0$ and $k=2$, we associate 0-form with points and associate 2-form with surfaces. Such association is important as we need to place the discrete unknowns to the mesh according to its geometric type.

We can write the generalized Stokes theorem as duality pairing as

$$\langle d\alpha^{(k)}, \Omega_{k+1} \rangle = \langle \alpha^{(k)}, \partial\Omega_{k+1} \rangle . \quad (2.47)$$

The duality indicates that the exterior derivative is the formal adjoint of the boundary operator. This feature is useful for the algebraic topology section, where the discrete exterior derivative is defined.

2.1.5 Hodge Operator

The Hodge operator is an operator that maps from k -form to $(n-k)$ -form. n is the dimension of ambient space $\mathbb{R}^n$. It is denoted as $\star$ and defined as

$$\star: \Lambda^k(\Omega) \rightarrow \Lambda^{(n-k)}(\Omega) . \quad (2.48)$$

With exterior derivative introduced, it is possible to express the PDEs in terms of differential forms. However, there are some inconsistencies that need to be addressed. That is what the Hodge operator is used for. The problem can be revealed by the example of Poisson equation:

$$\nabla \cdot \mathbf{q} = f, \quad (2.49)$$

$$\nabla \phi = \mathbf{u}, \quad (2.50)$$

$$\mathbf{u} = \mathbf{q}. \quad (2.51)$$

The first equation is applying the divergence operator to a vector, thus it can be regarded as an exterior derivative applied to a 2-form ($\mathbf{q}$ being $q^{(2)}$). As a result, the right hand side f should be a 3-form, $f^{(3)}$. The scalar ϕ is a 0-form $\phi^{(0)}$, so the equation is an exterior derivative applied to the 0-form. Therefore the right hand side is a 1-form $u^{(1)}$. The resulted equations become

$$dq^{(2)} = f^{(3)}, \quad (2.52)$$

$$d\phi^{(0)} = u^{(1)}, \quad (2.53)$$

$$u^{(1)} = q^{(2)}. \quad (2.54)$$

where the last equation shows the conflict, as they are different types of differential forms. If we expand the equations, one can clearly see the difference between the left hand side term $u^{(1)} = u_1 dx^1 + u_2 dx^2 + u_3 dx^3$ and the right hand side term $q^{(2)} = q_1 dx^2 \wedge dx^3 + q_2 dx^3 \wedge dx^1 + q_3 dx^1 \wedge dx^2$.

They can not be equated directly, thus we introduce the Hodge operator. Apply a Hodge operator to $u^{(1)}$, the last equation is

$$\begin{aligned} \star u^{(1)} &= \star(u_1 dx^1 + u_2 dx^2 + u_3 dx^3) \\ &= u_1 \star dx^1 + u_2 \star dx^2 + u_3 \star dx^3 \\ &= q^{(2)}, \end{aligned} \quad (2.55)$$

where for each basis

$$\star dx^1 = dx^2 \wedge dx^3, \quad (2.56)$$

$$\star dx^2 = dx^3 \wedge dx^1, \quad (2.57)$$

$$\star dx^3 = dx^1 \wedge dx^2. \quad (2.58)$$

Now their basis are consistent. Geometrically speaking, one can understand such operator as changing the geometry associated to the differential forms. The projection of 2-form can be thought of the area that the two vectors projected to the covector plane. Instead using 2-form, we can also use a normal vector to express the same area, which is written as a 1-form $\gamma^{(1)}(u' \times v')$.

The theory behind the Hodge operator is the two types of differential forms. Described in [1], it is said that there are inner-oriented forms and outer-oriented forms. Both of the forms are able to represent physical quantities, but they can never be equated together due to different orientation settings. To equate them, the form should be transformed to be consistent with the orientation, and that transformation/switch is exactly what the Hodge operator does.

Figure 2.5 depicts the different orientations of geometry objects in $\mathbb{R}^3$, where the Hodge operator switches the orientation between the two rows.

The inner-oriented is defined as an orientation of a geometry object that is independent of the dimension of the ambient space, and the outer-orientation is the one that depends on the dimension of the ambient space. Take the velocity for example, the velocity vector can be represented as circulation (which is the velocity integrated along the line). Alternatively, it can also be represented as a flux, i.e. integrated as amount of fluid passing through a surface ($\mathbb{R}^3$)/line($\mathbb{R}^2$). Therefore, we have two representations for velocity, one is 1-form and the other is (3-1)-form/(2-1)-form. The first representation is independent of the ambient space

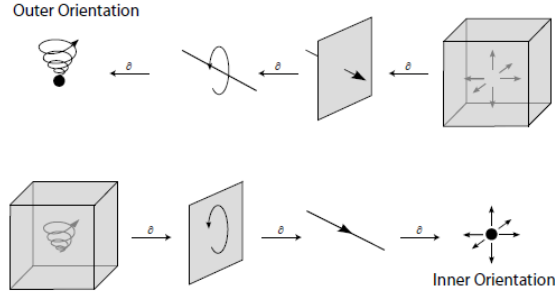


Figure 2.5: Outer and inner-orientation of k -form in $\mathbb{R}^3$, taken from [3]

dimension, thus the 1-form is inner-oriented. The other one, on the contrary, is determined to be 2-form or 1-form by the embedding space and as a result, it is an outer-oriented form.

For the same rule, the Hodge operation on other forms are

$$\star dx^2 \wedge dx^3 = dx^1, \quad (2.59)$$

$$\star dx^3 \wedge dx^1 = dx^2, \quad (2.60)$$

$$\star dx^1 \wedge dx^2 = dx^3, \quad (2.61)$$

$$\star dx^1 \wedge dx^2 \wedge dx^3 = 1, \quad (2.62)$$

$$\star 1 = dx^1 \wedge dx^2 \wedge dx^3. \quad (2.63)$$

In $\mathbb{R}^2$ setting, the Hodge operations are

$$\star 1 = dx^1 \wedge dx^2, \quad (2.64)$$

$$\star dx^1 = dx^2, \quad (2.65)$$

$$\star dx^2 = -dx^1, \quad (2.66)$$

$$\star(dx^1 \wedge dx^2) = 1. \quad (2.67)$$

Applying Hodge operator twice leads to identity with a minus or plus sign.

$$\star\star\alpha^{(k)} = (-1)^{(n-k)k}\alpha^{(k)}. \quad (2.68)$$

2.1.6 Double de Rham complex

A helpful diagram to establish relationship between two orientations of differential forms is the double de Rham complex:

$$\begin{array}{ccccccccc} \mathbb{R} & \longrightarrow & \Lambda^0(\Omega) & \xrightarrow{\frac{d}{\nabla}} & \Lambda^1(\Omega) & \xrightarrow{\frac{d}{\nabla \times}} & \Lambda^2(\Omega) & \xrightarrow{\frac{d}{\nabla \cdot}} & \Lambda^3(\Omega) & \xrightarrow{d} & 0 \\ & & \updownarrow \star & & \updownarrow \star & & \updownarrow \star & & \updownarrow \star & & \\ 0 & \xleftarrow{d} & \Lambda^3(\Omega) & \xleftarrow{\frac{\nabla \cdot}{d}} & \Lambda^2(\Omega) & \xleftarrow{\frac{\nabla \times}{d}} & \Lambda^1(\Omega) & \xleftarrow{\frac{\nabla}{d}} & \Lambda^0(\Omega) & \xleftarrow{} & \mathbb{R} \end{array}$$

The diagram is an analogy of the Figure 2.5 where the top sequence is inner-oriented differential form and the bottom side it is the outer-oriented one. As expected, the Hodge operator is placed between the two orientations and the exterior derivative d is shown in the horizontal line.

With this diagram, double derivative can be constructed, for example the Laplacian operator $\Delta := \nabla \cdot \nabla$. The operator can be constructed as this: first define an inner-oriented 0-form, that is the top left in the

diagram, then apply the exterior derivative which represents gradient operator to get the 1-form. Next, apply the Hodge to convert the 1-form to the outer-orient counterpart 2-form, where the second exterior derivative is applied which corresponds with the divergence operator. Finally, we apply the Hodge operator again to get the 3-form back to the original 0-form. In the equation, the above operations are expressed as

$$\Delta\alpha^{(0)} = \star d\star d\alpha^{(0)} . \quad (2.69)$$

The Laplacian operator for 1-form can be derived using the same rule. This time we start with 1-form on the top row, either we can first apply d then $\star$, d and finally $\star$ or apply the $\star$ first, then d , $\star$ and d to construct the Laplace operator. The Laplacian for a 1-form is the sum of these two operations,

$$\Delta\alpha^{(1)} = d\star d\star\alpha^{(1)} - \star d\star d\alpha^{(1)} , \quad (2.70)$$

which coincides with the vector Laplacian operator, $\Delta\vec{v} = \nabla\nabla\cdot\vec{v} - \nabla\times\nabla\times\vec{v}$. Equation 2.70 is also the general form of Laplace operator in differential k-form. The reason that there is only one term for 0-form is because $d\star\alpha^{(0)} = 0$. For the same reason, the Laplacian of 3-form is $\Delta\alpha^{(3)} = d\star d\star\alpha^{(3)}$.

2.1.7 Pullback Operator

The pullback operator is introduced due to the fact the physical domain is not equivalent to the reference/-computational domain. It is very common in numerical computation, where the calculation and analysis are done in the reference domain and then use mapping to switch to the more complex real physical domain. Particularly, in this work, Lagrange polynomials and edge functions (introduced later) are used as interpolation functions for differential forms and they are all defined on the simple orthogonal elements with a fixed interval, therefore, using reference domain to do computations such as integration is easier to implement.

As the result, the workflow of the program is that a differential form is pulled back from the physical domain to the reference domain, where discretization is performed. At last, an inverse of pullback is applied to the differential form, and brings them from the reference domain to the physical one.

The pullback operator is a mapping that maps the physical domain to the reference domain. Suppose the mapping $\Phi: \hat{\Omega}(\xi, \eta) \rightarrow \Omega(x, y)$, where the **reference domain** is denoted as $\hat{\Omega}$ and the **physical domain** is denoted as Ω . Then pullback operator, denoted as Φ^* , is defined as

$$\Phi^*: \Lambda^k(\Omega(x, y)) \rightarrow \Lambda^k(\hat{\Omega}(\xi, \eta)) . \quad (2.71)$$

Note in the reference domain the coordinates are denoted as ξ and η while in the physical domain, the corresponding coordinates are denoted as x and y .

For example, in case of 1-form, the pullback operator maps $\alpha^{(1)} = A dx + B dy$ defined on physical Ω to reference $\hat{\Omega}$, and get a 1-form $\beta^{(1)}$ with a different basis. In detail, apply the pullback to 1-form is

$$\Phi^*(dx) = \frac{dx}{d\xi} d\xi + \frac{dx}{d\eta} d\eta , \quad (2.72)$$

$$\Phi^*(dy) = \frac{dy}{d\xi} d\xi + \frac{dy}{d\eta} d\eta , \quad (2.73)$$

which is equivalent in matrix form, also known as Jacobian matrix,

$$\Phi^* = \begin{bmatrix} \frac{dx}{d\xi} & \frac{dx}{d\eta} \\ \frac{dy}{d\xi} & \frac{dy}{d\eta} \end{bmatrix} . \quad (2.74)$$

Therefore,

$$\begin{aligned}
 \beta^{(1)} &= \Phi^*(\alpha^{(1)}) = \Phi^*(Adx + Bdy) \\
 &= \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} \frac{dx}{d\xi} & \frac{dx}{d\eta} \\ \frac{dy}{d\xi} & \frac{dy}{d\eta} \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \\
 &= (A \frac{dx}{d\xi} + B \frac{dy}{d\xi}) d\xi + (A \frac{dx}{d\eta} + B \frac{dy}{d\eta}) d\eta ,
 \end{aligned} \tag{2.75}$$

where the term $(A \frac{dx}{d\xi} + B \frac{dy}{d\xi})$ and $(A \frac{dx}{d\eta} + B \frac{dy}{d\eta})$ will be referred as new vector proxies in the corresponding domain. The pullback operator has two properties [1], relating to wedge product and exterior derivative. They are

- $\Phi^*(\alpha^{(k)} \wedge \beta^{(l)}) = (\Phi^*(\alpha^{(k)})) \wedge (\Phi^*(\beta^{(l)}))$
- $\Phi^*(d\alpha^{(k)}) = d\Phi^*(\alpha^{(k)})$

The first property makes it easy to apply the pullback operator to higher k -forms with wedge product, e.g. $\Phi^*(dx \wedge dy) = \Phi^*(dx) \wedge \Phi^*(dy)$. The second one enables free commuting operation, i.e. there is no difference with sequence of operator applied. It also proved that the exterior derivative is coordinate free.

The pullback operator is compatible with the integration of differential form, as the Equation 2.76 shows. It means that one can simply do the integration in a simple computational domain and apply a pullback operator, which yields the same result as direct integration in the physical domain,

$$\int_{\Phi(\hat{\Omega})} a^{(k)} = \int_{\hat{\Omega}} \Phi^* a^{(k)} . \tag{2.76}$$

The inverse of the pullback operator is denoted as Φ^{-*} , given by

$$\Phi^{-*} = \frac{1}{\det(\Phi^*)} \begin{bmatrix} \frac{dy}{d\eta} & -\frac{dx}{d\eta} \\ -\frac{dy}{d\xi} & \frac{dx}{d\xi} \end{bmatrix} . \tag{2.77}$$

This operator is used to put the discretized differential forms from the reference domain into the physical domain. It will be illustrated in the next chapter.

2.1.8 Inner Product and Codifferential

In the previous sections, we saw that we can write the second derivative with the help of the two types of oriented differential forms. It implies that we can explicitly describe the two different oriented differential forms with two different oriented grids.

However, after exterior derivative and Hodge operations, we actually ended at the same oriented grid as where we start (such as $\Delta\alpha^{(0)} = \star d\star\alpha^{(0)}$). Actually we can stay in one type of oriented differential forms to perform the second derivative.

This is done by codifferential, denoted as d^* and is defined as

$$d^* := (-1)^{n(k+1)+1} \star d \star . \tag{2.78}$$

It is a series of Hodge and d operations, so if we use the d^* in the equation, there is only one type of orientation as we use the other oriented differential forms implicitly.

The inner product is defined as

$$(a^{(k)}, b^{(k)})_{\Omega} := \int_{\Omega} a^{(k)} \wedge \star b^{(k)} . \tag{2.79}$$

We use the inner product to derive the weak form of the PDE. Combine the above equation with the generalized Stokes theorem, we can derive an important equation by the integration by parts. Suppose there are two differential forms $a^{(k)}$ and $b^{(k+1)}$, then

$$\int_{\partial\Omega} \text{tr} a^{(k)} \wedge \text{tr} \star b^{(k+1)} = \int_{\Omega} d(a^{(k)} \wedge \star b^{(k+1)}) \quad (2.80)$$

$$= \int_{\Omega} da^{(k)} \wedge \star b^{(k+1)} + (-1)^k \int_{\Omega} a^{(k)} \wedge d\star b^{(k+1)} \quad (2.81)$$

$$= \int_{\Omega} da^{(k)} \wedge \star b^{(k+1)} + (-1)^{k+(k+1)(n-k-1)} \int_{\Omega} a^{(k)} \wedge \star \star d\star b^{(k+1)} \quad (2.82)$$

$$= (da^{(k)}, b^{(k+1)}) + (a^{(k)}, (-1)^{k+(k+1)(n-k-1)} \star d\star b^{(k+1)}) \quad (2.83)$$

$$= (da^{(k)}, b^{(k+1)}) - (a^{(k)}, d\star b^{(k+1)}) . \quad (2.84)$$

In the first line we use Stokes theorem. The second equation uses the property of $d(\alpha^{(k)} \wedge \beta^{(m)}) = d\alpha^{(k)} \wedge \beta^{(m)} + (-1)^k \alpha^{(k)} \wedge d\beta^{(m)}$ mentioned before. In the third line we use the second property of Hodge operator $\star\star = (-1)^{k(n-k)}$ mentioned before. The rest is to rewrite in inner product form and the final expression is

$$(a^{(k)}, d\star b^{(k+1)})_{\Omega} = (da^{(k)}, b^{(k+1)})_{\Omega} - \int_{\partial\Omega} \text{tr} a^{(k)} \wedge \text{tr} \star b^{(k+1)} . \quad (2.85)$$

As the Equation 2.85 shows, the codifferential is canceled by integration by part to be an inner product plus a boundary term. Notation tr is the trace of the differential form, which means differential forms at the boundaries on the manifold, $\text{tr}: \Lambda^k(\Omega) \rightarrow \Lambda^k(\partial\Omega)$.

2.2 Algebraic Topology

The differential geometry provides a tool to describe the physical variable in a more natural way, where these physical quantities are associated with points, surfaces or volumes. In this section, algebraic topology is introduced. This is the discrete version of differential geometry, thus can be put into computer for numerical solving. Here we present a brief introduction to the algebraic topology. For more detailed references we refer to [4] or [15].

2.2.1 Cell, Chain, and Cochain

The basic element of algebraic topology is the p -cell, so there are 0-cells, 1-cells, 2-cells in $\mathbb{R}^2$, which is similar to the definition of differential form. A 0-cell is a point in topological space, denoted by $c_{(0)}$. For higher dimensional cells, 1-cell is the line segment bounded by two 0-cells in the topological space. Similarly, the 2-cell represents the surface bounded by surrounding 1-cells. They are denoted in the subscript, i.e. $c_{(1)}$ and $c_{(2)}$. The collection of p -cells are defined as p -chain, denoted as C_p

$$C_p = \sum_{i=1}^{n_p} m_{(p),i} c_{(p),i} , \quad (2.86)$$

where n_p is the number of p -cells and $m_{(p)}$ is the weight for each p -cell. Usually the weight value is set to be either 1, -1, or 0, which means that the cell has a positive/negative orientation or not being part of the chain. It is clear to see that chain is actually the linear combination of cells.

An important thing to note is that there are also two types of orientations for cells – the outer-oriented or inner-oriented – corresponding to the orientations of differential forms. Note from here we start to talk about $\mathbb{R}^2$ case, since this thesis focuses on solving PDEs in a 2D domain. For $\mathbb{R}^3$, reader can refer to [16].

For 0-cells, they are defined as sink or source (inner-oriented, it has the same literal meaning as a sink or source in fluid dynamics). It also can be defined as clockwise or anti-clockwise (outer-oriented). For 1-cell, the inner orientation is defined as along the line. Nevertheless the outer-oriented 1-cell is defined as the oriented

line orthogonal to the cell. As for 2-cell, the orientation is defined as clockwise/anti-clockwise rotation as inner-oriented or as a sink/source on the surface. In the Figure 2.6, cells on the top are the outer-oriented 0,1,2-cell and on the bottom and the inner-oriented 2,1,0-cell.

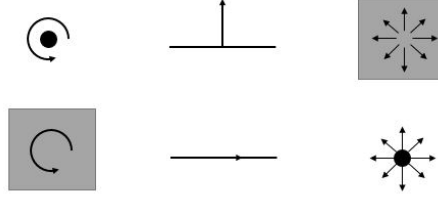


Figure 2.6: Inner-(bottom) and outer-orientation(top) of geometries in 2D

The collection of all the cells is called the cell complex. An example in [1] is shown in the Figure 2.7. This is a simple two-dimensional cell complex. Each 1-cell is given by two 0-cells. For example, the boundary of the 1-cell, L_1 , is given by the 0-cells P_1 and P_2 . When the default orientation of point is defined as sink, the boundary is simply $-P_1 + P_2$. So a minus one corresponds to an orientation in the opposite direction, and a plus one means orientation in the same direction as the line orientation. Similarly, the boundary of surface S_1 is given by surrounding lines, namely $L_1 + L_2 - L_3 - L_4$, where their $+/-$ sign corresponds with their orientation. It should be noted that the 0-cell does not have boundaries.

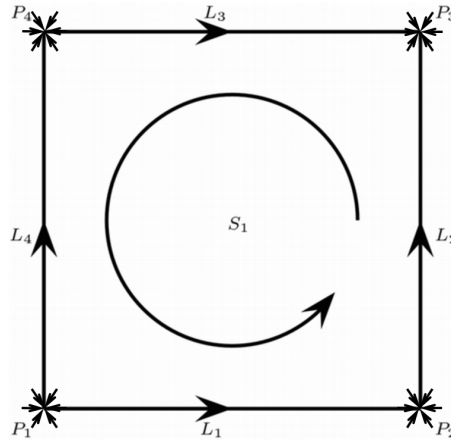


Figure 2.7: Cell complex sample

We can write the boundary of all 1-cells in form of matrix, for example,

$$\partial \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \begin{bmatrix} \partial L_1 \\ \partial L_2 \\ \partial L_3 \\ \partial L_4 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \mathbb{E}^{(1,0)} P_i . \quad (2.87)$$

In the equation, ∂ is the boundary operator. The matrix that contains 0, $+/-1$ is called the incidence matrix and is denoted as $\mathbb{E}^{(1,0)}$ where the upper script indicates that it is the relation between 1-cells and 0-cells. It is the matrix form of the topological boundary operator. In a similar way, the boundary of the surface S_1 can be written as

$$\partial S_1 = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \mathbb{E}^{(2,1)} L_i , \quad (2.88)$$

where $\mathbb{E}^{(2,1)}$ is the incidence matrix. In addition, if we multiple the two incidence matrices $\mathbb{E}^{(2,1)}$ and $\mathbb{E}^{(1,0)}$, the result is 0, which implies that the boundary of the boundary of cells is zero, $\partial \circ \partial C_{(p)} = 0$

It is easy to prove that the set of all p -chains form a linear vector space. Therefore it is reasonable to think about the dual of such vector space. That is where p -cochain comes from, which is defined as the dual of p -chain. It is the discrete version of the differential form [1]. The cochain is defined as

$$C^p = \sum_{i=1}^{n_p} a^{(p),i} c_{(p),i}^{(p)}, \quad (2.89)$$

where $c_{(p),j}^{(p),i} = \delta_i^j$ is a Kronecker delta and $a^{(p),i}$ is a list of weights. Applying the cochain to chain by duality pairing, we get

$$\begin{aligned} \langle C^p, C_p \rangle &:= \sum_{i=1}^{n_p} \sum_{j=1}^{n_p} a^{(p),i} m_{(p),j} \langle c_i^{(p)}, c_{(p),j} \rangle \\ &= \sum_{i=1}^{n_p} a^{(p),i} m_{(p),i}. \end{aligned} \quad (2.90)$$

The duality expression 2.90 resembles the integration of differential k-form over a k-dimensional manifold, where the chain plays the role of manifold and the cochain acts like a differential form. For a duality between p -chain and p -cochain, the comparison can be found in the following table.

$\langle C^0, C_0 \rangle := \sum_{i=1}^{n_p} a^{(0),i} m_{(0),i}$	$\langle \alpha^{(0)}, \mathcal{P} \rangle := f(\mathcal{P})$
$\langle C^1, C_1 \rangle := \sum_{i=1}^{n_p} a^{(1),i} m_{(1),i}$	$\langle \lambda^{(1)}, \mathcal{C} \rangle := \int_{\mathcal{C}} \lambda^{(1)}$
$\langle C^2, C_2 \rangle := \sum_{i=1}^{n_p} a^{(2),i} m_{(2),i}$	$\langle \eta^{(2)}, \mathcal{S} \rangle := \int_{\mathcal{S}} \eta^{(2)}$

2.2.2 Coboundary Operator

In the last section, we talked a bit about the boundary operator ∂ which is in fact a mapping of $C_p \rightarrow C_{p-1}$. Here we introduce the coboundary operator, δ as

$$\langle \delta C^p, C_p \rangle := \langle C^p, \partial C_{(p+1)} \rangle. \quad (2.91)$$

The coboundary operator maps p -cochains into $(p+1)$ -cochains. Such operation resembles the generalized Stokes theorem (Equation 2.47), where the exterior derivatives maps k -form to $(k+1)$ -form. In fact the coboundary operator is the discrete analogy of the exterior derivative [16]. That also means that the vector calculus operators (*grad*, *curl*, *div*) or the exterior derivative d can be represented by the coboundary operator δ . Applying the coboundary operator to a 0-cochain represents gradient operation. Similarly, 1-cochain for curl, and 2-cochain for divergence. The matrix representation of coboundary operator is in fact the transpose of incidence matrix of the boundary operator.

2.3 Transfinite Mesh

In this study, a transfinite mesh is used to depict the physical domain, which in our case is the flow around a cylinder. The transfinite method is actually a tool originally developed for computer aided design but it has been rapidly adapted to the problem of mesh generation for finite element analysis [17]. The transfinite mesh method for finite element method was first introduced by Gordon and Hall [18]. The term “transfinite”, as quoted, is to describe the fact that unlike classical interpolation method that match primitive function at a finite number of points, the transfinite method matches the function with non-denumerable number of points.

Currently, there is no good mesh generator for spectral element method yet and most of the generators are coarse on approximation on geometry. For a case like a cylinder, traditional mesh generator would simply

approximate the geometry to the polygons, a piecewise linear mesh. Refining the mesh will just end at polygons with more sides. The transfinite method is a very accurate method on approximating the geometry that retains the original boundaries, and the more refinement we have, the accuracy increases. Such feature is good for the spectral element method as the case in the study consists of circular boundary. Therefore, in this thesis, all the physical domains that are being used will be meshed by this method.

It should be noted that in spectral element method, increasing the order of polynomials is often used for refinement. That is also why the element in spectral element method can be larger than the one used in the finite element method. In the thesis, for our flow around cylinder case, the cylinder can be bounded by only eight elements and give reasonably accurate result.

The geometry modeling and interpolation work will be explained here. To analyze the flow field with complex geometry, the reference domain $\hat{\Omega}$ is established, as well as a projector Φ that projects from reference domain $\hat{\Omega}$ to the physical one: $\Phi: \Lambda(\hat{\Omega}) \rightarrow \Lambda(\Omega)$.

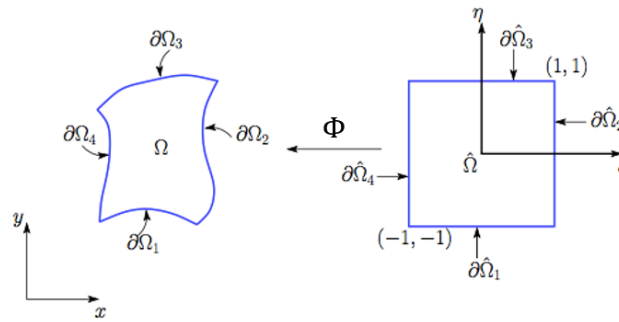


Figure 2.8: Mapping between reference domain and physical domain

In transfinite mesh, the prerequisite is the four boundaries of the physical domain, namely $\partial\Omega_{1,2,3,4} = \Gamma_{1,2,3,4}$ in the Figure 2.8 and are expressed as vector valued functions by:

$$\partial\Omega_1 = \Gamma(\xi, -1) , \quad (2.92)$$

$$\partial\Omega_2 = \Gamma(1, \eta) , \quad (2.93)$$

$$\partial\Omega_3 = \Gamma(\xi, 1) , \quad (2.94)$$

$$\partial\Omega_4 = \Gamma(-1, \eta) . \quad (2.95)$$

The curve $\partial\hat{\Omega}_1$ corresponds to the edge at bottom in the reference domain. For a given value of $\xi \in [-1, 1]$ with η being the constant -1 , the function Γ defines the correspondent point coordinate in physical domain edge $\partial\Omega$.

We take an element with curved boundaries to demonstrate how the mesh is constructed, the shape of the domain is shown in the Figure 2.9. The four boundaries are indicated as Γ_1 , Γ_2 , Γ_3 , and Γ_4 , starting from the bottom edge, anti-clockwise.

These boundaries are functions of reference coordinates, i.e. $\Gamma(\xi, \eta)$. The shape of reference domain is a square with ξ and η varying from -1 to 1 , i.e the $\hat{\Omega}$ in the Figure 2.8. (The reference domain interval minus one to one is not essential and can be altered such as zero to one.) Therefore for this element, the boundaries are defined as:

$$\Gamma(\xi, -1) = \begin{bmatrix} y(\xi, \eta) \\ x(\xi, \eta) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2}(1+\xi) \end{bmatrix} , \quad (2.96)$$

$$\Gamma(1, \eta) = \begin{bmatrix} y(\xi, \eta) \\ x(\xi, \eta) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(1+\eta) \\ 0.25\eta^2 + 0.75 \end{bmatrix} , \quad (2.97)$$

$$\Gamma(\xi, 1) = \begin{bmatrix} y(\xi, \eta) \\ x(\xi, \eta) \end{bmatrix} = \begin{bmatrix} -0.25\xi^2 + 0.75 \\ \frac{1}{2}(1+\xi) \end{bmatrix} , \quad (2.98)$$

$$\Gamma(-1, \eta) = \begin{bmatrix} y(\xi, \eta) \\ x(\xi, \eta) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(1+\eta) \\ 0 \end{bmatrix} . \quad (2.99)$$

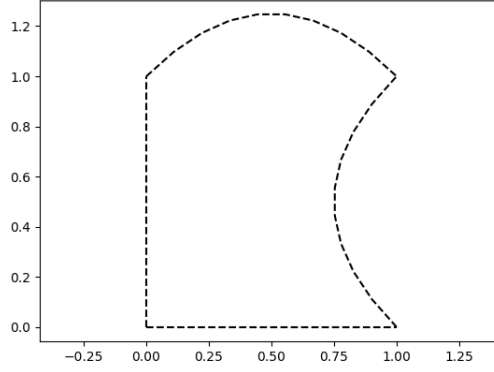


Figure 2.9: *Element example for transfinite mesh*

It is easy to map the boundaries precisely into the physical domain or vice versa, but the key is to ensure the interior points of the domain being properly mapped.

Here we define univariate functions $\phi_i, \psi_i, i=1,2$ that behave like linear basis functions:

$$\phi_0 = \frac{1-\xi}{2}, \quad \psi_0 = \frac{1-\eta}{2}, \quad (2.100)$$

$$\phi_1 = \frac{1+\xi}{2}, \quad \psi_1 = \frac{1+\eta}{2}. \quad (2.101)$$

Then the transfinite mapping consists of three mappings, they are

$$\mathcal{F}_\xi(\xi, \eta) = \phi_0 \Gamma(-1, \eta) + \phi_1 \Gamma(1, \eta), \quad (2.102)$$

$$\mathcal{F}_\eta(\xi, \eta) = \psi_0 \Gamma(\xi, -1) + \psi_1 \Gamma(\xi, 1), \quad (2.103)$$

$$\begin{aligned} \mathcal{F}_{\xi\eta}(\xi, \eta) &= \phi_0 \psi_0 \Gamma(-1, -1) + \phi_0 \psi_1 \Gamma(-1, 1) + \phi_1 \psi_0 \Gamma(1, -1) + \phi_1 \psi_1 \Gamma(1, 1) \\ &= \sum_{i=0}^1 \sum_{j=0}^1 \phi_i \psi_j \Gamma(-(-1)^i, -(-1)^j). \end{aligned} \quad (2.104)$$

Then the transfinite mapping between the reference domain and the physical domain is calculated by the Equation 2.105:

$$\mathcal{F}(\xi, \eta) = \mathcal{F}_\xi + \mathcal{F}_\eta - \mathcal{F}_{\xi\eta}. \quad (2.105)$$

The first component of the mapping preserves the left and right side boundaries, Γ_2 and Γ_4 , as shown in the Figure 2.10a, and the second mapping $\mathcal{F}_\eta$ will preserve the remaining two boundaries, Γ_1 and Γ_3 , (Figure 2.10b), while the third mapping $\mathcal{F}_{\xi\eta}$ only preserves the four points at corners shown in the Figure 2.10c. The transfinite mapping $\mathcal{F}$ is actually a Boolean sum of interpolations in each of the computational coordinates which will preserve all the four boundaries and the interior points in the domain. So the final interpolation is shown in the Figure 2.11

The derivation of the equation is described in [18], detailed references include [17, 19, 20]. The function ϕ and ψ are called blending functions in the paper and they are not necessary being linear as being defined before, being non-linear functions are able to twist the mesh. Nevertheless the linear one is the most straightforward case, also in our case there is no twist inside the element.

One should be cautious that when using the method is that the mapping has to be in the consistent direction. It is obvious that if we substitute the ξ with $-\xi$, the shape being mapped may seem correct, but in fact this mapping just twists the image by 180 degrees to the other side.

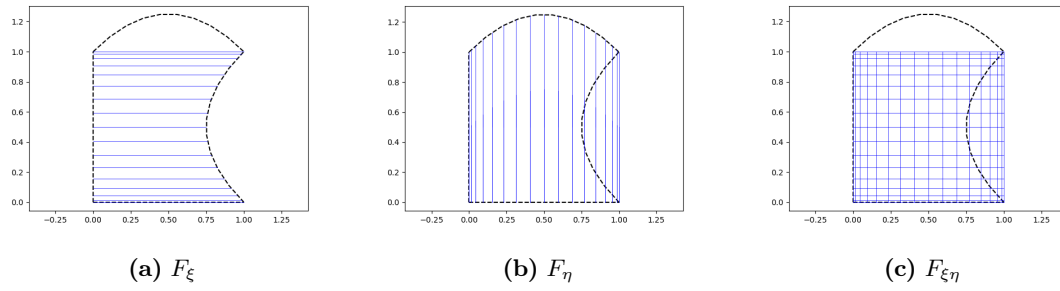


Figure 2.10: Transfinite interpolation components

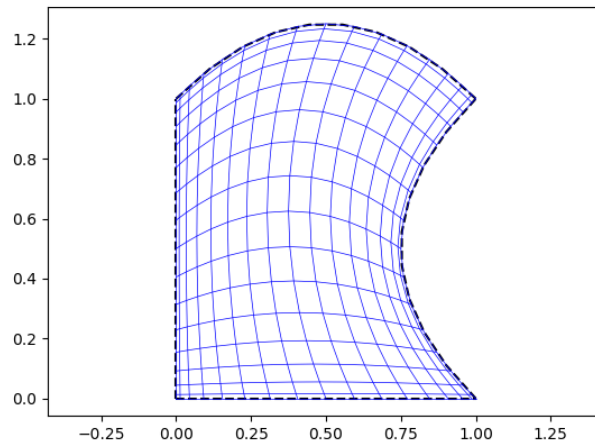


Figure 2.11: Generated transfinite mesh

After implementing all the transfinite interpolation for each element, it can be assembled for the test case. Figure 2.12 is one of the test cases developed. In the figure, there are 32 elements in total and the interpolation node is Gauss-Lobatto-Legendre nodes with polynomial degree set to 3.

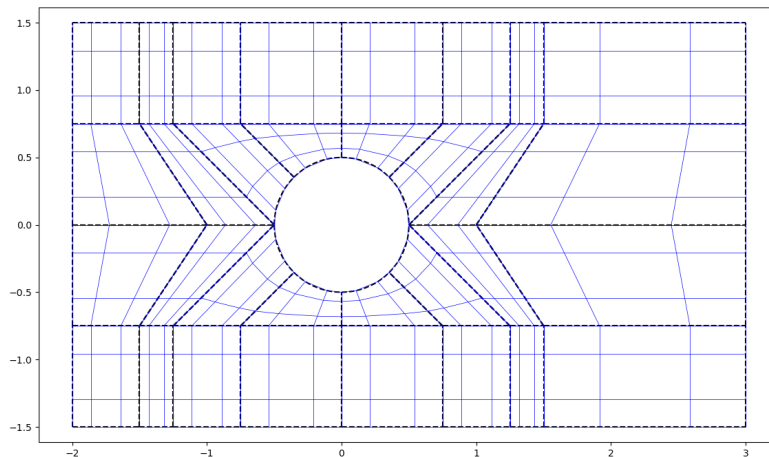


Figure 2.12: Transfinite mesh for one of the test cases

After establishing the transfinite mapping, the derivative with respect to ξ and η in the Jacobian matrix can be derived, i.e. $\frac{dx}{d\xi}$, $\frac{dx}{d\eta}$ and $\frac{dy}{d\xi}$, $\frac{dy}{d\eta}$, thus the pullback operator is found.

3 | Mimetic Spectral Implementation

In the last chapter, it has been shown that the concepts and operations in vector calculus can be transformed into the differential geometry which can be further represented by algebraic cochains. In this chapter, the transformation between differential continuous settings to the discrete counterpart is introduced. This is achieved by the projection operator.

The partial derivative equations of interest lie in an infinite dimensional space. Here, a projection π is implemented to map the differential form in infinite dimensional space to finite dimensional space – a finite function space ($\pi: \Lambda^k \rightarrow \Lambda_h^k$).

The projection contains two operations: $\pi = \mathcal{I} \circ \mathcal{R}$. First, a transformation from differential form to cochain is done by the reduction operator $\mathcal{R}$. Later, a reconstruction operator $\mathcal{I}$ is used to reconstruct the differential form using cochain.

3.1 Reduction Operator

The reduction operator is a map from p -forms to p -cochains, $\mathcal{R}: \Lambda^p(\Omega) \rightarrow C^p(D)$, where Ω is a manifold, D is a cell complex. It is defined as:

$$\langle \mathcal{R}a^{(p)}, C_p \rangle := \int_{C_p} a^{(p)}, \quad (3.1)$$

where $\mathcal{R}a^{(p)}$ is a p -cochain that acts on a p -chain C_p .

The effect of the reduction operator can be demonstrated as following. Consider a cell complex consisting x_N number 0-cells and x_{N-1} number of 1-cells at interval of $[-1, 1]$. Suppose a 0-form $a^{(0)}$ on 1D space, the reduction operation $\mathcal{R}a$ is equivalent to sampling a scalar function at x_N points, i.e. the cochain values are

$$[a(x_1) \ a(x_2) \ \dots \ a(x_N)]. \quad (3.2)$$

If we apply the coboundary operator, the resultant 1-cochain is

$$\delta[a(x_1) \ a(x_2) \ \dots \ a(x_N)] = [a(x_2) - a(x_1) \ \dots \ a(x_N) - a(x_{N-1})]. \quad (3.3)$$

It can be seen that the coboundary operator assigns values to the line segments. Such operation can also be done by taking the exterior derivative first to the 0-form $a^{(0)}$ and then the 1-form $da^{(0)}$ is applied by the reduction operator. It can be proved that it is equivalent to the previous operation, which leads to an important property of reduction operator that the sequence of applying reduction and exterior derivative operation has no effect to the final result, $\mathcal{R} \circ d = \delta \circ \mathcal{R}$. This rule applies to any differential form as the following diagram commutes.

$$\begin{array}{ccc} \Lambda^k & \xrightarrow{d} & \Lambda^{k+1} \\ \downarrow \mathcal{R} & & \downarrow \mathcal{R} \\ C^k & \xrightarrow{\delta} & C^{k+1} \end{array}$$

It should be noted that since the reduction operation reduces the dimensionality of the function space, there will be a loss of information in cochains.

3.2 Reconstruction Operator

The reconstruction operator $\mathcal{I}$ acts the opposite direction of $\mathcal{R}$ – it maps $\mathcal{I}:C^p(D) \rightarrow \Lambda^p(\Omega)$, which takes the cochain and gives differential form. The reconstruction is done by interpolation of cochains using basis function, so there are multiple choices for reconstruction, whether it can be Lagrangian polynomial, or B-spline, etc. Although it is free to choose reconstruction method, a certain number of requirements needs to be fulfilled [21]. One is that the operator is the right inverse of $\mathcal{R}$, so it returns identity

$$\mathcal{R} \circ \mathcal{I} = \mathbb{I} , \quad (3.4)$$

and its left inverse (which is π) must be an approximation close to identity with an error, which varies due to the approximated function and the polynomial degree that is used.

$$\mathcal{I} \circ \mathcal{R} = \mathbb{I} + e . \quad (3.5)$$

This is easy to figure out, as the reduction of a reconstruction of cochain will return the original cochain while the second equation dropped a certain amount of information to get the cochain from differential form which for sure leads to an inevitable error. In addition, the operator has to satisfy $d \circ \mathcal{I} = \mathcal{I} \circ \delta$, which means that the sequence of taking exterior derivative/coboundary operator and reconstruction does not affect the final result, just like the commuting diagram of the reduction operator.

In the following sections, the reconstruction operation used in the thesis will be put in detail.

3.3 Basis Function

The difference between the traditional Finite Element Method and the Spectral Element Method is that the SEM uses high degree polynomials as basis functions, while FEM generally uses linear or quadratic function as basis function. In this section, we discuss two types of basis functions used to approximate (reconstruct) the differential forms. They are Lagrange nodal function and edge function.

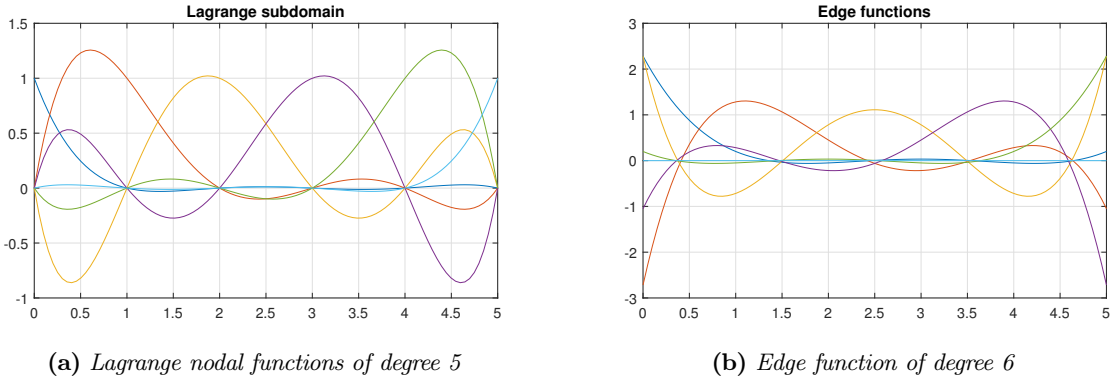


Figure 3.1: Sample of basis functions to be used in this study

3.3.1 Nodal Function

The first type of basis functions is the nodal basis function, which is widely used in spectral element method [22, 23].

For 0-form $a^{(0)}(x)$, the basis function is a set of normalized Lagrange polynomials to reconstruct 0-cochains. Take a 1D case for example, a 0-form is defined in an interval of $[-1, 1]$. Suppose after reduction we have $N+1$ number of nodal values x_i ($1 \leq i \leq N+1$) (The node position may be uniformly distributed but also can be Gauss-Lobatto-Legendre nodes), which sample the $a(x_i)$ that gives the discrete values a_i ($1 \leq i \leq N+1$). The Lagrange polynomial, $h_i(x)$, of degree N is given by

$$h_i(x) = \prod_{j=0, j \neq i}^{N+1} \left(\frac{x-x_j}{x_i-x_j} \right). \quad (3.6)$$

These functions have the property that they become zero in all points x_j except x_i which is equal to $h_i(x_i) = 1$.

$$h_i(x_j) = \begin{cases} 1 & i=j, \\ 0 & i \neq j. \end{cases} \quad (3.7)$$

The 0-form can be reduced and then interpolated/reconstructed as

$$a_h^{(0)}(x) = \mathcal{I} \circ \mathcal{R}(a^{(0)}(x)) = \sum_{i=1}^{N+1} a(x_i) h_i(x) = \sum_{i=1}^{N+1} a_i h_i(x). \quad (3.8)$$

The function is approximated by a Lagrange polynomial of degree N and the subscript h just indicates that currently the reconstructed form stays in a finite dimensional space. Figure 3.2 shows an example projection of a 0-form/function using Lagrange polynomials. The original 0-form is shown in blue line, $u = e^{0.5x}(\sin(0.5\pi x) + 2\pi \cos(0.5\pi x))$. First, the 0-form is reduced to some nodal (for example equidistant or GLL nodes) values denoted as the blue points. Then these nodes are used to reconstruct a finite dimensional function where the Lagrange polynomials are used. The reconstructed 0-form u_h is denoted as the red line. With the increase of order, the interpolation will increasingly converge to the original 0-form.

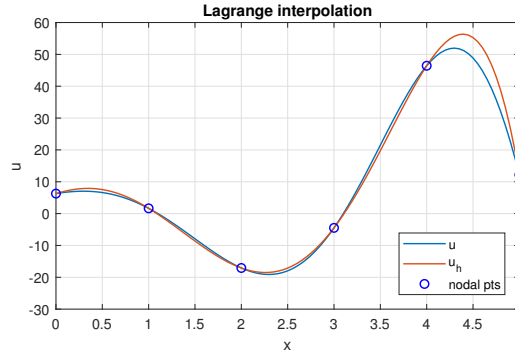


Figure 3.2: Nodal reconstruction of $u(x)$ using Lagrange polynomials

It is important that one can evaluate the integral of the reconstructed function by defined weights and the following equations holds,

$$w_i = \int_{-1}^1 h_i(x) dx, \quad (3.9)$$

$$\int_{-1}^1 a_h^{(0)}(x) dx = \sum_{i=1}^{N+1} w_i a(x_i). \quad (3.10)$$

That leads to the numerical integration or quadrature which is a method to calculate the integration as weighted summation of evaluations of the function at a finite set of points. The position and number of x_i in $a(x_i)$ and the weight w_i differs according to different method and accuracy. In our code, the Gauss quadrature or Gauss-Lobatto-Legendre quadrature is used. The main difference is that the Gauss quadrature points x_i do not include start and ending points, namely -1 and 1 while Gauss-Lobatto-Legendre includes.

We will discuss the quadrature in detail later.

3.3.2 Edge Function

For 1-form, the edge function $e_i(x)$ is developed [24, 25] to reconstruct the 1-cochain to 1-form. The integral of edge function satisfies:

$$\int_{L_j} e_i(x) = \begin{cases} 1 & i=j, \\ 0 & i \neq j. \end{cases} \quad (3.11)$$

which indicates that at each section of line segment L_j , the integration of the function is one if $i=j$, instead of nodal values being one like nodal function. Its definition is given by

$$e_i(x) = -\sum_{k=1}^{i-1} dh_k(x) \quad i=1,2,\dots,N, \quad (3.12)$$

where $h_k(x)$ is the Lagrange polynomials mentioned before, and the $dh_k(x)$ is the derivative of it. Note the different number of summation that nodal function and edge function have. If there are $N+1$ number of 0-cochains, then there will be N number of 1-cochains.

For a 1-form $u^{(1)} = u(x)dx$, the reduction operation gives the corresponding 1-cochain C^1 , which is the integral values $u_1, u_2, \dots, u_N$,

$$\mathcal{R}u^{(1)} = C^1 = \sum_{i=1}^N u_i, \quad (3.13)$$

where

$$u_i = \int_{x_i}^{x_{i+1}} u^{(1)} = \int_{x_i}^{x_{i+1}} u(x)dx \quad i=1,2,\dots,N. \quad (3.14)$$

The 1-cochains can also be derived by applying coboundary operator to the 0-cochains. After reduction, we can reconstruct the 1-form using edge function defined before:

$$u_h^{(1)}(x) = \sum_{i=1}^N u_i e_i(x). \quad (3.15)$$

The reconstruction operation is can also be derived by the exterior derivative of $a_h^{(0)}$ [1]:

$$\begin{aligned} u_h^{(1)} &= da_h^{(0)}(x) = d\left(\sum_{i=1}^{N+1} a(x_i)h_i(x)\right) = \sum_{i=1}^{N+1} a(x_i)dh_i(x) \\ &= \sum_{i=1}^{N+1} a(x_i)\left(\sum_{k=1}^i dh_k(x) - \sum_{k=1}^{i-1} dh_k(x)\right) \\ &= \sum_{i=1}^{N+1} a(x_i)(-e_{i+1}(x) + e_i(x)) \\ &= \sum_{i=1}^N (a(x_{i+1}) - a(x_i))e_i(x) \\ &= \sum_{i=1}^N (\delta a(x_i))e_i(x), \end{aligned} \quad (3.16)$$

where the $\delta a(x_i)$ is exactly the discrete version of u_i . Now we can say that the edge function validates the commuting relationship of $\mathcal{I}\delta = d\delta$, as shown,

$$\begin{array}{ccc} C^k & \xrightarrow{\delta} & C^{k+1} \\ \downarrow \mathcal{I} & & \downarrow \mathcal{I} \\ \Lambda_h^k & \xrightarrow{d} & \Lambda_h^{k+1} \end{array}$$

Either, we get the reconstructed 1-form from 1-chain or by applying the exterior derivative to the reconstructed 0-form. With the properties of $\mathcal{R}d = \delta\mathcal{R}$ in the last section, it is obvious to derive the relationship: $\pi d = d\pi$

$$\begin{array}{ccc} \Lambda^k & \xrightarrow{\delta} & \Lambda^{k+1} \\ \downarrow \pi & & \downarrow \pi \\ \Lambda_h^k & \xrightarrow{d} & \Lambda_h^{k+1} \end{array}$$

Figure 3.3 shows an example of the edge function interpolation in 1D space. The manufactured function $u(x)$ is denoted as the blue line. The reconstructed function $u_h(x)$ is calculated as

$$u_h(x) = \sum_{i=1}^N \left(\int_{x_i}^{x_{i+1}} u(x_i) dx \right) e_i(x), \quad (3.17)$$

and is denoted as the red line. Unlike the nodal function that every nodal value equals exactly to the original function point, it is the integral over the interval of each section of the interpolation function $\int_L u_h(x) dx$ equals to that of the original one $\int_L u(x) dx$.

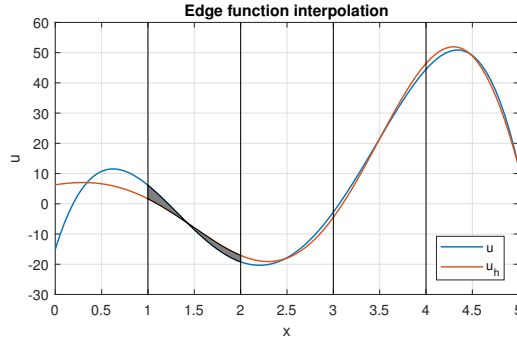


Figure 3.3: Reconstruction using edge function

3.4 Quadrature

Since the inner product is defined as an integration operation, we need a technique to evaluate the integrals of the differential forms (reconstructed as polynomials), within each element domain. Therefore, we use numerical integration or quadrature to numerically calculate the integration. It was introduced briefly in the Nodal Function section, the fundamental building block is a summation over a finite set of points:

$$\int_{-1}^1 u(\xi) d\xi \approx \sum_{i=1}^n w_i u(\xi_i), \quad (3.18)$$

where w_i are specified constants or weights and ξ_i represents n number of distinct points in the interval $-1 \leq \xi_i \leq 1$, also known as integration points. Although there are many different types of numerical integration that gives ξ_i and w_i , we restrict our attention to Gaussian quadrature.

There are also variants of Gauss quadrature as well, such as Gauss, Gauss-Radau, and Gauss-Lobatto-Legendre, etc. As mentioned before, in our code, we can use Gauss quadrature as well as Gauss-Lobatto-Legendre quadrature. These two quadratures are also the commonly used in the many SEMs.

For Gauss quadrature, the integration points are the roots of the Legendre polynomials $P_n(x)$, which is defined as

$$P_n(x) = 2^n \sum_{j=0}^n \binom{n}{j} \left(\frac{n+j-1}{n} \right), \quad (3.19)$$

where the i -th Gauss integration node, x_i , is the i -th root of P_n . The roots can be solved by any root solving algorithms. The corresponding weight is given by

$$w_i = \frac{2}{(1-x_i^2)P_n'(x_i)^2}. \quad (3.20)$$

Due to the involvement of Legendre polynomials, the method is usually known as Gauss-Legendre quadrature. It should be noted that the number of integration points n is not necessarily equal to the order of the approximated polynomials. However, the quadrature integration equation has $2n$ number of parameters: n points and n weights. Hence we can hope to make it exact for all polynomials of degree $2n-1$ that have $2n$ number of coefficients.

The Gauss-Lobatto-Legendre quadrature is a slight variant of Gauss quadrature, as it takes the start point and end point into account. The points are calculated as

$$x_i = \begin{cases} -1 & i=1, \\ \text{i-th zeros of } P_n'(x) & i=2,3,\dots,n-1, \\ 1 & i=n. \end{cases} \quad (3.21)$$

The weights for the GLL-node are calculated using:

$$w_i = \frac{2}{n(n+1)P_n^2(x_i)}. \quad (3.22)$$

Given the order of quadrature n , the GLL integration is exact for polynomials order up to $2n-3$, smaller than Gauss quadrature due to the extra start/end points. Figure 3.4 shows some typical nodes used in Gaussian quadrature.

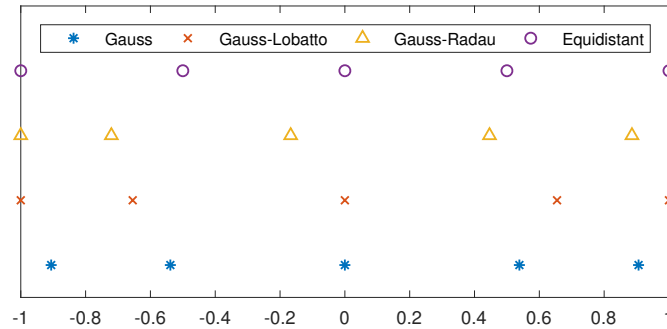


Figure 3.4: Comparison of different nodes

3.5 Multi-dimensional Case

In the previous sections, the basis functions for 0-cochain and 1-cochain interpolation were introduced. We can extend them into 2D or 3D by tensor product. In this section, a 2D scenario is illustrated for 0,1,2-form

reduction and reconstruction. In general, they are reconstructed the same way as in the 1D case, but with help of tensor product.

The 0-form in 2D are just multi-variate functions $a^{(0)} = a(x, y)$, thus the reduction is the same as the 1D case – sampling the function at a set of points. Suppose we reduce the 2D 0-form on a Cartesian coordinate, and denote the nodal values as $a(x_i, y_j)$ ($i, j = 0, 1, \dots, N$) and there are $(N+1) \times (N+1)$ number of points in two directions, then we can reconstruct the 0-cochain using tensor product as shown in the Equation 3.23.

$$a_h^{(0)} = a_h(x, y) = \sum_{i=0}^N \sum_{j=0}^N a(x_i, y_j) h_i(x) h_j(y) . \quad (3.23)$$

As for 1-form, the reconstruction can be illustrated by an example. Consider a 1-form given by $u^{(1)} = \alpha(x, y)dx + \beta(x, y)dy$. Reduce it by integrating the $u^{(1)}$ over the two directions, which results,

$$\alpha_{i,j} := \int_{x_{i-1}}^{x_i} \alpha(x, y_j) dx \quad i = 1, 2, \dots, N \quad j = 0, 1, \dots, N , \quad (3.24)$$

$$\beta_{i,j} := \int_{y_{j-1}}^{y_j} \beta(x_i, y) dy \quad i = 0, 1, \dots, N \quad j = 1, 2, \dots, N . \quad (3.25)$$

The reconstruction is shown as

$$u_h^{(1)} = u_h(x, y) = \sum_{i=1}^N \sum_{j=0}^N \alpha_{i,j} e_i(x) h_j(y) + \sum_{i=0}^N \sum_{j=1}^N \beta_{i,j} h_i(x) e_j(y) . \quad (3.26)$$

It can also be derived from exterior derivative of 0-form. If the 0-form is $a_h^{(0)} = \sum_{i=0}^N \sum_{j=0}^N a_{i,j} h_i(x) h_j(y)$, the exterior derivative is

$$u_h^{(1)} = da_h^{(0)} = \sum_{i=1}^N \sum_{j=0}^N (a_{i,j} - a_{i-1,j}) e_i(x) h_j(y) + \sum_{i=0}^N \sum_{j=1}^N (a_{i,j} - a_{i,j-1}) h_i(x) e_j(y) . \quad (3.27)$$

For 2-form, consider a 2-form, $\phi^{(2)} = \phi(x, y)dx \wedge dy$. The reduction are 2-cochains:

$$\phi_{i,j} := \int_{x_{i-1}}^{x_i} \int_{y_{j-1}}^{y_j} \phi(x, y) dx dy \quad i, j = 1, 2, \dots, N . \quad (3.28)$$

Therefore the reconstructed 2-form is

$$\phi_h^{(2)} = \phi_h(x, y) = \sum_{i=1}^N \sum_{j=1}^N \phi_{i,j} e_i(x) e_j(y) . \quad (3.29)$$

Similarly, the reconstructed $\mathcal{IR}\phi^{(2)}$ can be derived as the exterior derivative of 1-form, $d\mathcal{IR}u^{(1)}$,

$$\phi_h^{(2)} = du_h^{(1)} = \sum_{i=1}^N \sum_{j=1}^N (\alpha_{i,j} - \alpha_{i-1,j} + \beta_{i,j} - \beta_{i,j-1}) e_i(x) e_j(y) . \quad (3.30)$$

3.6 Pullback Operator

For simplicity, the discussion above does not include transformation between reference domain and computation domain. In fact the pullback happens during the reduction operation. After reconstruction, the inverse pullback is applied to bring them back to the physical domain.

Suppose the ξ, η being the reference domain with x, y is the physical one, the reduction of 0-form, in 2D

$$C^0 = \mathcal{R}\Phi^* a^{(0)}(x, y) \quad (3.31)$$

$$= \mathcal{R}a^{(0)}(\xi, \eta) \quad (3.32)$$

$$= \sum_{i=0}^N \sum_{j=0}^N a(\xi_i, \eta_j) , \quad (3.33)$$

where ξ_i is the coordinate of the i -th 0-cell on the domain. Then the reconstruction of the 0-cochain is done where the inverse of pullback operator Φ^{-*} (see Equation 2.77) is applied.

$$a_h^{(0)} = \Phi^{-*} \mathcal{I}C^0 \quad (3.34)$$

$$= \Phi^{-*} \sum_{i=0}^N \sum_{j=0}^N a_{ij} h_i(\xi) h_j(\eta) \quad (3.35)$$

$$= \sum_{i=0}^N \sum_{j=0}^N a_{ij} h_i(\xi(x, y)) h_j(\eta(x, y)) . \quad (3.36)$$

From the equation, we can conclude that the pullback operator is merely a change of variables thus applying pullback operator does not change the general equations structure whether it is with the coordinate of (x, y) or (ξ, η) . Therefore, we omit such operation in the following derivations in order to make the equation clearer to see.

3.7 Weak Form and Test Function

In this section, we talk about the weak form which is the essential for the spectral element method. Normally, equations should hold absolutely pointwise while the weak formulation does not have this requirement. In the weak formulation, the equation is multiplied with test functions and integrated over a domain. We require such integration equation hold. It means that the our method does not approximate the original equation, but the weak form of the original equation.

Here we use a simple Poisson equation to show the general methodology for spectral element method. The equation is shown as

$$\begin{aligned} -\Delta\phi &= f , \\ \text{with B.C.:} \\ \phi &= 0 \text{ at } \partial\Omega . \end{aligned} \quad (3.37)$$

The above equation is referred to strong form in contrast to the weak form. A test function v is multiplied to the equation and the resulting equation is integrated over the domain, shown as

$$-\int_{\Omega} (\Delta\phi) v dx = \int_{\Omega} f v dx \quad \forall v . \quad (3.38)$$

Here ϕ is a second order derivative thus we use integration by parts to reduce the order of derivatives. By doing this, we actually weaken the requirement on the smoothness of the solution.

$$-\int_{\Omega} (\Delta\phi)v dx = \int_{\Omega} \nabla\phi \cdot \nabla v dx - \int_{\partial\Omega} \frac{\partial\phi}{\partial n} v ds \quad \forall v, \quad (3.39)$$

where $\frac{\partial\phi}{\partial n} = \nabla\phi \cdot n$ is the derivative of ϕ to the normal direction n . An important property of weak formulations is that the test function is supposed to be zero on the parts of the boundary where the solution ϕ is known. In our example, $v=0$ at $\partial\Omega$, so the last term vanishes, and the resulting equation to be solved is:

$$\int_{\Omega} \nabla\phi \cdot \nabla v dx = \int_{\Omega} f v dx \quad \forall v. \quad (3.40)$$

If the original strong form holds then this equation holds, but we can not say this vice versa. That's why we called it weak form, which we only need to make the integration at both sides equal to hold the equation, rather than make every point in the domain equal to satisfy the equation.

The test function is approximated by basis functions, thus $v \approx v_h = \sum_{i=1}^N v_i h_i(x)$. The option to choose type of basis function is open, it can be something else, such as hat function, other than ones talked in the last section. The unknown function (often named as trial function in Finite Element Method) is also approximated using the same way: $\phi \approx \phi_h = \sum_{i=1}^N \phi_i h_i(x)$.

The problem now has become a weak form: find ϕ_h that for all v_h

$$\int_{\Omega} \nabla\phi_h \cdot \nabla v_h dx = \int_{\Omega} f v_h dx, \quad (3.41)$$

$$\text{i.e. } (\nabla\phi_h, \nabla v_h) = (f, v_h). \quad (3.42)$$

What's missing is the space that v and ϕ stays. There are restrictions over the test function and the solution but that is beyond the scope of this thesis. In MSEM, ϕ and v belongs to a Hilbert space. Detailed discussions can be found in [22, 16]. Since the basis functions $h_i(x)$ are known, then the ϕ_h to be solved is completely determined once the coefficients ϕ_i have been found.

Insert the summation expansion of ϕ_h and v_h to the equation above, we have

$$\int_{\Omega} \left(\frac{d}{dx} \left(\sum_{j=1}^N \phi_j h_j(x) \right) \frac{d}{dx} \left(\sum_{i=1}^N v_i h_i(x) \right) \right) dx = \int_{\Omega} \left(f \sum_{i=1}^N v_i h_i(x) \right) dx, \quad (3.43)$$

which can be factored as

$$\begin{aligned} \sum_{i=1}^N v_i \left(\sum_{j=1}^N \left(\int_{\Omega} (h'_i(x) h'_j(x)) dx \right) \phi_j - \int_{\Omega} f h_i(x) dx \right) &= 0, \\ \sum_{i=1}^N v_i \left(\sum_{j=1}^N K_{ij} \phi_j - F_i \right) &= 0. \end{aligned} \quad (3.44)$$

Since v_i is arbitrary, so we can choose v_i for each equation, a straightforward case is make the first equation $v_1=1$ and any $v_n=0$ ($n \neq 1$), the second equation $v_2=1$ and any $v_n=0$ ($n \neq 2$), etc. Therefore there are N number of equations for N number of unknowns in total and is a matrix form of $K\phi=F$ type that we can solve.

The matrix K is the integral of the product of derivative of two basis functions, and because the locality of the basis function (being non-zero at specific region), that ensures the sparsity of the matrix since as long as they are not neighbors, the product is zero.

$$\begin{bmatrix} K_{11} & K_{12} & \dots \\ \vdots & \ddots & \\ K_{i1} & & K_{ij} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_j \end{bmatrix} - \begin{bmatrix} F_1 \\ \vdots \\ F_i \end{bmatrix} = 0$$

After applying the boundary condition, we can solve the equation to get the coefficient ϕ_i and the PDE is solved.

3.8 About Python

In this essay, the implementation of the mimetic spectral element method is done by using the Python library named `MimeticFEM`. Here is a short introduction to the package.

The general architecture is, at first, the mesh information should be given, which in our case, the four boundaries of the element is defined as a function of reference coordinates. The four boundary functions will be used as parameters to instantiate a class of transfinite mesh, which plays the role of mesh generator. The resulting object contains mappings and metric tensors. After that, a function space class is used, with the defined mesh object being the parameter. The resulted object serves as a “warehouse” that stores information such as mesh, nodes type, polynomial degree, etc. This is where the differential form basis are defined from. The basis is created as a new object under a subclass of specific function space parent class. Since there are 0-form, 1-form and 2-form, three basis form objects are defined in the code and each of them is associated with certain methods based on what type of nodes it is, such as Gauss points or GLL points.

Up to now, the basis form object contains coordinates of the grid in the reference domain at which the basis functions are evaluated. Therefore, the function of inner product can be applied to get the mass matrix for certain differential forms. If there is more than one element, an assembly function is required to assemble the mass matrix of each element to an integrated mass matrix. For cases with complex geometry, such assembly should be customized by users. This is where gathering matrix is used.

With mass matrices, the system equation is established where the boundary condition is applied. After solving the equation, the solution, which is cochain, are assigned to specific differential forms class attribute. The reconstruction then is implemented by the “reconstruct” method in the basis form class. Finally an export function is called to assign the values to the physical domain to plot results. In addition to the plotting, the package includes error function which can be used to compare the numerical results with the exact solution if it exists and compute the error for error analysis.

In general, the package provides a complete and thorough toolbox and framework for numerical solution with mimetic spectral element method. The package is very flexible as users can choose different settings with a simple parameter change. In addition, the package framework is adjustable and open as well, as the use of abstract class enables users to add new features, such as a new mesh generator, to the package without heavily modifying the code.

4 | Poisson and Laplace Equation

The Poisson equation is a good way to test the code. Here we use the method of manufactured solutions for performance evaluation. The idea is to generate an artificial equation setting that have an exact solution. Since the solution is exact, then the numerical error, if the scheme is implemented correctly, should converge to the machine precision level, thus this method is a good and common way to test the developed code. After the manufactured solution, we will implement the method to solve Laplace equation for flow around cylinder problem and investigate the results.

4.1 Reference domain

As mentioned earlier, most computation schemes have reference domain for computational convenience. Here in this thesis, the reference domain is a 2-dimensional Cartesian plane, a square ranging from -1 to 1 with its center being at the origin point. As usual, we will denote this plane with its coordinates with ξ and η .

The continuous domain is discretized with grid which contains points, lines, and surfaces. These objects are associated with corresponding k-cochains which are produced by the reduction of k-forms.

The grid position can be customized, the most common case is the equally spaced nodes, nevertheless, references have shown that if the nodes are GLL nodes, the convergence rate is better than the one with uniformly distributed nodes grid. In addition, GLL quadrature is used and the integral can be approximated by

$$\int_{-1}^1 \int_{-1}^1 f(\xi, \eta) d\xi d\eta = \sum_{i=1}^N \sum_{j=1}^N f(\xi_i, \eta_j) w_i w_j . \quad (4.1)$$

Note this is a 2D case, so there are w_i and w_j derived for two directions.

For the deformed physical domain, mapping between the physical domain and the reference domain will be done. For multiple-element case, additional matrix assembly will be applied. Both of them are additional equation transformation thus do not effect the equations derived here in the next section.

4.2 Equation in Matrix Form

The original 2D Poisson equation is

$$u = -\nabla \cdot \phi , \quad (4.2)$$

$$\nabla \cdot u = f . \quad (4.3)$$

Here we use the mixed formulation of Poisson equation. Compared with the one (Equation 3.37) just introduced in the last section, now we have two unknowns, namely u and ϕ . The reason of using two-field mixed formulation is that we wish to prescribe the normal flux $\nabla \phi$ along the boundary strongly and eliminate those degree of freedoms, which the original equation can only prescribe the flux weakly. Another reason is that when f is zero, then the second equation behaves like conservation of mass. That is to say, this formulation can be extended to the Stokes equation or Navier-Stokes equations. So this formulation is the first step towards those PDEs.

The potential is a scalar, thus it could either be a 0-form or 2-form in 2D. Here we set the the potential to 2-form.

It is natural to associate the u as 1-form, and since we define the 1-form velocity in 2D space, it has two components, $u^{(1)} = u_\xi d\eta - u_\eta d\xi$, which if integrated, is actually the flux cross the specific line.

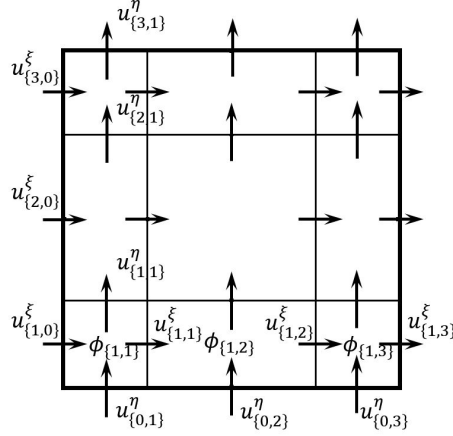


Figure 4.1: Reference domain and differential forms defined

Rewrite the Poisson equation into differential form, shown as

$$u^{(1)} = -d\star\phi^{(2)} , \quad (4.4)$$

$$du^{(1)} = f^{(2)} . \quad (4.5)$$

The weak form of equations above are given by

$$(\alpha^{(1)}, u^{(1)}) + (\alpha^{(1)}, d\star\phi^{(2)}) = 0 , \quad (4.6)$$

$$(\beta^{(2)}, du^{(1)}) = (\beta^{(2)}, f^{(2)}) , \quad (4.7)$$

where α and β are test functions.

Now we can apply Equation 2.85 to the second term in the first equation, and finally the weak form is : Find $u^{(1)} \in \Lambda^1(\Omega)$ and $\phi^{(2)} \in \Lambda^2(\Omega)$ such that $\forall \alpha^{(1)} \in \Lambda^1(\Omega)$ and $\forall \beta^{(2)} \in \Lambda^2(\Omega)$ that

$$(\alpha^{(1)}, u^{(1)}) + (d\alpha^{(1)}, \phi^{(2)}) - \int_{\partial\Omega} tr\alpha^{(1)} \wedge tr\star\phi^{(2)} = 0 , \quad (4.8)$$

$$(\beta^{(2)}, du^{(1)}) = (\beta^{(2)}, f^{(2)}) . \quad (4.9)$$

Next, apply the mimetic projection operator π_h to reduce the equations to finite dimensional space: Find $u_h^{(1)} \in \Lambda_h^1$ and $\phi_h^{(2)} \in \Lambda_h^2$ such that $\forall \alpha_h^{(1)} \in \Lambda_h^1$ and $\beta_h^{(2)} \in \Lambda_h^2$ we have

$$(\alpha_h^{(1)}, u_h^{(1)}) + (d\alpha_h^{(1)}, \phi_h^{(2)}) - \int_{\partial\Omega} tr\alpha_h^{(1)} \wedge tr\star\phi_h^{(2)} = 0 , \quad (4.10)$$

$$(\beta_h^{(2)}, du_h^{(1)}) = (\beta_h^{(2)}, f_h^{(2)}) . \quad (4.11)$$

The differential forms defined can be illustrated as shown in the Figure 4.1.

Since the u is 1-form, so they are the edges in the domain, ϕ and f are 2-form which are the surfaces in the domain. The orientation for 1-form is outer-oriented and the positive orientation is the from left to right and from bottom to top, and for 2-form, the positive direction is a source.

4.2.1 Derivation of $u^{(1)} = -d\star\phi^{(2)}$

In 2D space, $u^{(1)}$ has two components, $u^{(1)} = u_\xi d\eta - u_\eta d\xi$, where the reduction is applied thus

$$u_{i,j}^\xi = \int_{\eta_{i-1}}^{\eta_i} u_\xi(\xi_j, \eta) d\eta \quad i=1,2,3,\dots,N \quad j=0,1,2,\dots,N, \quad (4.12)$$

$$u_{i,j}^\eta = \int_{\xi_{j-1}}^{\xi_j} u_\eta(\xi, \eta_i) d\xi \quad i=0,1,2,\dots,N \quad j=1,2,3,\dots,N. \quad (4.13)$$

The reconstruction is done by edge functions and nodal functions,

$$u^{(1)} = \mathcal{IR}u^{(1)}(\xi, \eta) = \sum_{i=1}^N \sum_{j=0}^N u_{i,j}^\xi e_i(\xi) h_j(\eta) - \sum_{i=0}^N \sum_{j=1}^N u_{i,j}^\eta h_i(\xi) e_j(\eta). \quad (4.14)$$

The test function can be expanded similarly resulting in Equation 4.16.

$$\alpha^{(1)} = \alpha_\xi d\eta - \alpha_\eta d\xi \quad (4.15)$$

$$= \sum_{i=1}^N \sum_{j=0}^N \alpha_{i,j}^\xi e_i(\xi) h_j(\eta) - \sum_{i=0}^N \sum_{j=1}^N \alpha_{i,j}^\eta h_i(\xi) e_j(\eta), \quad (4.16)$$

Note that the α^ξ/α^η has the same structure as u^ξ/u^η . Now the inner product can be written, with help of Equation 2.79 and the Hodge operation properties, as

$$(\alpha_h^{(1)}, u_h^{(1)}) = \int_{\Omega} \alpha^{(1)} \wedge \star u^{(1)} \quad (4.17)$$

$$\begin{aligned} &= \int_{\Omega} (\alpha_\xi d\eta - \alpha_\eta d\xi) \wedge \star (u_\xi d\eta - u_\eta d\xi) \\ &= \int_{\Omega} (\alpha_\xi d\eta - \alpha_\eta d\xi) \wedge (-u_\xi d\xi - u_\eta d\eta) \\ &= \int_{\Omega} (-\alpha_\xi u_\xi d\eta \wedge d\xi + \alpha_\eta u_\eta d\xi \wedge d\eta) \\ &= \int_{\Omega} (\alpha_\xi u_\xi + \alpha_\eta u_\eta) d\xi \wedge d\eta \\ &= \int_{\Omega} \left(\sum_{i=1}^N \sum_{j=0}^N \sum_{k=1}^N \sum_{l=0}^N \alpha_{k,l}^\xi u_{i,j}^\xi e_i(\xi) h_j(\eta) e_k(\xi) h_l(\eta) + \dots \right. \\ &\quad \left. \sum_{i=0}^N \sum_{j=1}^N \sum_{k=0}^N \sum_{l=1}^N \alpha_{k,l}^\eta u_{i,j}^\eta h_i(\xi) e_j(\eta) h_k(\xi) e_l(\eta) \right). \end{aligned} \quad (4.18)$$

The expression inside the integration bracket can be written in matrix form, it becomes

$$\begin{bmatrix} \alpha_{1,0}^\xi & \dots & \alpha_{k,l}^\xi & \alpha_{0,1}^\eta & \dots & \alpha_{k,l}^\eta \end{bmatrix} \begin{bmatrix} e_1(\xi) h_0(\eta) \\ e_1(\xi) h_1(\eta) \\ \vdots \\ e_k(\xi) h_l(\eta) \\ h_0(\xi) e_1(\eta) \\ \vdots \\ h_k(\xi) e_l(\eta) \end{bmatrix} \begin{bmatrix} e_1(\xi) h_0(\eta) & \dots & e_i(\xi) h_j(\eta) & h_0(\xi) e_1(\eta) & \dots & h_i(\xi) e_j(\eta) \end{bmatrix} \begin{bmatrix} u_{1,0}^\xi \\ u_{1,1}^\xi \\ \vdots \\ u_{i,j}^\xi \\ u_{0,1}^\eta \\ \vdots \\ u_{i,j}^\eta \end{bmatrix}$$

where the second term and the third term constitute the mass matrix, denoted as $\mathbb{M}^{(1)}$. This is a $2N(N+1) \times 2N(N+1)$ matrix. The upper script (1) indicates that the mass matrix is associated with the 1-forms. As a result, the inner-product is derived by integrating the above expression, which is actually integrating each element in the mass matrix. This is also where quadrature steps in.

For the second term $(d\alpha^{(1)}, \phi^{(2)})$, the exterior derivative is

$$d\alpha^{(1)} = d(\alpha_\xi d\eta - \alpha_\eta d\xi) \quad (4.19)$$

$$= d\left(\sum_{i=0}^N \sum_{j=1}^N \alpha_{i,j}^\xi e_i(\xi) h_j(\eta)\right) - d\left(\sum_{i=1}^N \sum_{j=0}^N \alpha_{i,j}^\eta h_i(\eta) e_j(\xi)\right) \quad (4.20)$$

$$= \sum_{i=1}^N \sum_{j=1}^N (\alpha_{i,j}^\xi - \alpha_{i-1,j}^\xi) e_i(\xi) e_j(\eta) (d\xi \wedge d\eta) - \sum_{i=1}^N \sum_{j=1}^N (\alpha_{i,j}^\eta - \alpha_{i,j-1}^\eta) e_j(\eta) e_i(\xi) (d\eta \wedge d\xi) \quad (4.21)$$

$$= \sum_{i=1}^N \sum_{j=1}^N (\alpha_{i,j}^\xi - \alpha_{i-1,j}^\xi + \alpha_{i,j}^\eta - \alpha_{i,j-1}^\eta) e_i(\xi) e_j(\eta) . \quad (4.22)$$

Note that the $d\xi$ and $d\eta$ are included in the α^η/u^η and α^ξ/u^ξ . Writing them out in bracket is for clarity.

The 2-form $f^{(2)}$, $\phi^{(2)}$ and $\beta^{(2)}$ in the Poisson equations are derived with the same way, so we only mention the derivation of $\phi^{(2)}$, the reduction and the reconstruction are

$$\phi_{i,j} = \int_{\xi_{i-1}}^{\xi_i} \int_{\eta_{j-1}}^{\eta_j} \phi(\xi, \eta) d\xi d\eta , \quad \phi^{(2)} = \sum_{i=1}^N \sum_{j=1}^N \phi_{i,j} e_i(\xi) e_j(\eta) .$$

With the expression of $\phi^{(2)}$ and $d\alpha^{(1)}$ the inner product is derived as

$$(d\alpha^{(1)}, \phi^{(2)}) = \int_{\Omega} d\alpha^{(1)} \wedge \star \phi^{(2)} \quad (4.23)$$

$$\begin{aligned} &= \sum_{i=1}^N \sum_{j=1}^N (\alpha_{i,j}^\xi - \alpha_{i-1,j}^\xi + \alpha_{i,j}^\eta - \alpha_{i,j-1}^\eta) e_i(\xi) e_j(\eta) \wedge \star \sum_{k=1}^N \sum_{l=1}^N \phi_{k,l} e_k(\xi) e_l(\eta) \\ &= \int_{\Omega} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N \phi_{k,l} (\alpha_{i,j}^\xi - \alpha_{i-1,j}^\xi + \alpha_{i,j}^\eta - \alpha_{i,j-1}^\eta) e_i(\xi) e_j(\eta) e_k(\xi) e_l(\eta) (d\xi \wedge d\eta) . \end{aligned} \quad (4.24)$$

In matrix form, it can be written as

$$\begin{bmatrix} \alpha_{0,1}^\xi & \dots & \alpha_{i,j}^\eta \end{bmatrix} \begin{bmatrix} 0 & 1 & \dots \\ \vdots & \ddots & \\ 1 & & 0 \end{bmatrix}^T \begin{bmatrix} e_1(\xi) e_1(\eta) \\ e_1(\xi) e_2(\eta) \\ \vdots \\ e_k(\xi) e_l(\eta) \end{bmatrix} \begin{bmatrix} e_1(\xi) e_1(\eta) & e_1(\xi) e_2(\eta) & \dots & e_i(\xi) e_j(\eta) \end{bmatrix} \begin{bmatrix} \phi_{1,1} \\ \phi_{1,2} \\ \vdots \\ \phi_{k,l} \end{bmatrix}$$

where the second term is the transpose of the incidence matrix, denoted as $\mathbb{E}^{(2,1)}$. It is a $N^2 \times 2N(N+1)$ matrix. The third term and the fourth term is the mass matrix, denoted as $\mathbb{M}^{(2)}$, which is a $N^2 \times N^2$ matrix.

4.2.2 Derivation of $du^{(1)} = f^{(2)}$

As for the second equation, $du^{(1)} = f^{(2)}$. The test function $\beta^{(2)}$ and the exterior derivative of $u^{(1)}$ are given by

$$\beta^{(2)} = \sum_{k=1}^N \sum_{l=1}^N \beta_{k,l} e_k(\xi) e_l(\eta) (d\xi \wedge d\eta) , \quad (4.25)$$

$$du^{(1)} = d(u_\xi d\eta - u_\eta d\xi) \quad (4.26)$$

$$= \sum_{i=1}^N \sum_{j=1}^N (u_{i+1,j}^\xi - u_{i,j}^\xi) e_i(\xi) e_j(\eta) (d\xi \wedge d\eta) - \sum_{i=1}^N \sum_{j=1}^N (u_{i,j+1}^\eta - u_{i,j}^\eta) e_j(\eta) e_i(\xi) (d\eta \wedge d\xi) \quad (4.27)$$

$$= \sum_{i=1}^N \sum_{j=1}^N (u_{i+1,j}^\xi - u_{i,j}^\xi + u_{i,j+1}^\eta - u_{i,j}^\eta) e_i(\xi) e_j(\eta) . \quad (4.28)$$

So for the first term in the equation, the inner product

$$(\beta^{(1)}, du^{(1)}) = \int_{\Omega} \beta^{(1)} \wedge \star du^{(1)} \quad (4.29)$$

$$= \int_{\Omega} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N \beta_{k,l} (u_{i,j}^\xi - u_{i,j-1}^\xi + u_{i,j}^\eta - u_{i-1,j}^\eta) e_i(\xi) e_j(\eta) e_k(\xi) e_l(\eta) (d\xi \wedge d\eta) . \quad (4.30)$$

Then rewrite it in matrix form it becomes

$$\begin{bmatrix} \beta_{1,1} & \dots & \beta_{k,l} \end{bmatrix} \begin{bmatrix} e_1(\xi) e_1(\eta) \\ e_1(\xi) e_2(\eta) \\ \vdots \\ e_k(\xi) e_l(\eta) \end{bmatrix} \begin{bmatrix} e_1(\xi) e_1(\eta) & e_1(\xi) e_2(\eta) & \dots & e_i(\xi) e_j(\eta) \end{bmatrix} \begin{bmatrix} 0 & 1 & \dots \\ \vdots & \ddots & \\ 1 & & 0 \end{bmatrix} \begin{bmatrix} u_{1,0}^\xi \\ u_{1,1}^\xi \\ \vdots \\ u_{i,j}^\xi \\ u_{0,1}^\eta \\ \vdots \\ u_{i,j}^\eta \end{bmatrix}$$

where the second term and the third term is the mass matrix $\mathbb{M}^{(2)}$, which is identical to the mass matrix defined in the first equation. The incidence matrix $\mathbb{E}^{(2,1)}$ is also the same as the one defined before.

For the source term,

$$(\beta^{(2)}, f^{(2)}) = \int_{\Omega} \beta^{(2)} \wedge \star f^{(2)} \quad (4.31)$$

$$= \int_{\Omega} \sum_{i=1}^N \sum_{j=1}^N \beta_{i,j} e_i(\xi) e_j(\eta) \wedge \star \sum_{k=1}^N \sum_{l=1}^N f_{k,l} e_k(\xi) e_l(\eta) \quad (4.32)$$

$$= \int_{\Omega} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N \beta_{i,j} f_{k,l} e_i(\xi) e_j(\eta) e_k(\xi) e_l(\eta) . \quad (4.33)$$

The summation equation is written in matrix form,

$$\begin{bmatrix} \beta_{1,1} & \dots & \beta_{i,j} \end{bmatrix} \begin{bmatrix} e_1(\xi) e_1(\eta) \\ e_1(\xi) e_2(\eta) \\ \vdots \\ e_k(\xi) e_l(\eta) \end{bmatrix} \begin{bmatrix} e_1(\xi) e_1(\eta) & e_1(\xi) e_2(\eta) & \dots & e_i(\xi) e_j(\eta) \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{1,2} \\ \vdots \\ f_{k,l} \end{bmatrix}$$

Up to now we have derived all the equations in detail. Note that there are test functions on both sides and since in weak formulation the equation should hold for all α and β , we can remove the degree of freedom of the test function term. Finally in matrix form, the equation is shown as

$$\mathbb{M}^{(1)}u^{(1)} + \mathbb{E}^{(2,1)T}\mathbb{M}^{(2)}\phi^{(2)} = tr \star \phi^{(2)} , \quad (4.34)$$

$$\mathbb{M}^{(2)}\mathbb{E}^{(2,1)}u^{(1)} = f^{(2)} , \quad (4.35)$$

$$\begin{bmatrix} \mathbb{M}^{(1)} & \mathbb{E}^{(2,1)T}\mathbb{M}^{(2)} \\ \mathbb{M}^{(2)}\mathbb{E}^{(2,1)} & 0 \end{bmatrix} \begin{bmatrix} u^{(1)} \\ \phi^{(2)} \end{bmatrix} = \begin{bmatrix} tr \star \phi^{(2)} \\ \mathbb{M}^{(2)}f^{(2)} \end{bmatrix} . \quad (4.36)$$

The system matrix will be solved after imposing boundary conditions. In the post-processing stage, the cochains are reconstructed. We have discussed about the reconstruction before, but that is just to show how the equations are derived. The reconstruction created polynomial expressions of unknowns, which can be used to plot over the domain.

4.3 Multiple Elements

The domain of interest is divided into multiple elements, where each element is meshed by transfinite mesh method with Gauss-Lobatto-Legendre nodes. To implement it on the computer, one has to be wisely label all the edges and surfaces inside the element. Therefore, there is a process to map the local numbering of a single element to the global numbering of the multiple elements. Such mapping is called the degree of freedom map or the gathering matrix. Later, the element will be assembled together. That is to say the mass matrix that represents every element will be assembled to one general matrix, which is the very one that is going to be solved.

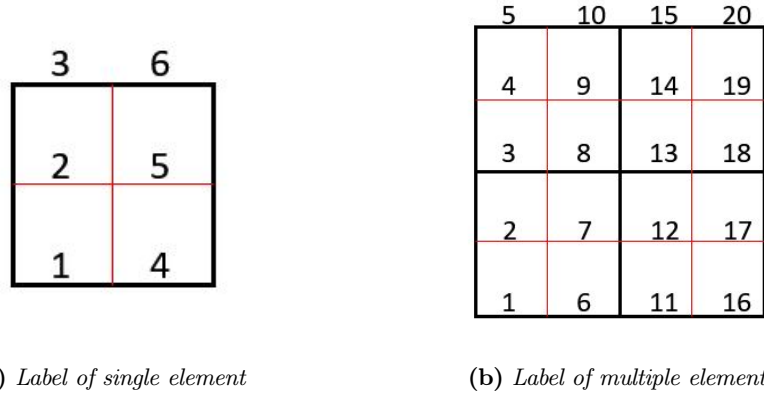


Figure 4.2: Local labeling and global labeling

The idea behind the gathering matrix is the connection between the local labeling and the global labeling. The labeling of objects in the thesis is to count from the bottom to the top, and from the left to the right. As shown in the Figure 4.2a, for a single element with a polynomial degree of 2, the labeling of 1-forms (lines) starts from the bottom “1” to the top “3”, and continue to count from the first column to the right one. The issue raised when the multiple elements are introduced, where there are conflicts between the global labeling and the local labeling.

The general system matrix that is going to be solved requires every object in it to be uniquely labeled. Therefore the 1-forms will be labeled as Figure 4.2b by gathering matrix. As a result, the gathering matrix can be considered as a function that takes the element number and local object number as input and gives out the global number in the general matrix, like

$$F_{gather\ matrix}(element\ number\ \#, local\ number\ \#) = global\ number\ \# .$$

Thus assembling small elements to one big element, one only needs to look up the gathering matrix to get the correct global number of every object in every element. Consider the multiple element case in Figure 4.2b, the gathering matrix should be as

$$F_{gather\ matrix} = \begin{pmatrix} 1 & 2 & 3 & 6 & 7 & 8 \\ 3 & 4 & 5 & 8 & 9 & 10 \\ 11 & 12 & 13 & 16 & 17 & 18 \\ 13 & 14 & 15 & 18 & 19 & 20 \end{pmatrix}, \quad (4.37)$$

where the row represents the element index and the column represents local numbering and the element value is the global number. Note in 2D, the gathering matrix for 2-forms is much simpler than for 1-forms, since they are associated to surfaces that do not have naming conflict.

4.4 Manufactured solution

Here we use a manufactured solution Equation 4.38 to test the code.

$$\phi^{(2)} = \sin(\pi x) \cos(\pi y). \quad (4.38)$$

The velocity is

$$u = \text{grad } \phi = -\pi \cos(\pi x) \sin(\pi y) \mathbf{i} + \sin(\pi x) \cos(\pi y) \mathbf{j}. \quad (4.39)$$

The source term $f^{(2)}$ is derived as

$$f = \frac{d^2 \phi}{dx^2} + \frac{d^2 \phi}{dy^2} = -\pi^2 \sin(\pi x) \sin(\pi y) - \pi^2 \sin(\pi x) \sin(\pi y). \quad (4.40)$$

For boundary conditions, both Dirichlet boundary condition and Neumann boundary condition were used. The boundary conditions are given as:

$$\begin{cases} -\nabla^2 \phi = f & \text{on } \Omega, \\ \phi = g & \text{on } \Gamma_1, \\ u = h & \text{on } \Gamma_2, \end{cases} \quad (4.41)$$

where g is calculated using Equation 4.38; h is calculated using Equation 4.39; The boundaries satisfy $\partial\Omega = \Gamma_1 \cup \Gamma_2$, $\Gamma_1 \cap \Gamma_2 = \emptyset$.

The test domain contains 8 elements, as shown in the Figure 4.3, where the boundary values are applied to the edge of the domain and the cylinder surface.

At right hand side of equation, the boundary integral is calculated using quadrature,

$$\int_{\partial\Omega} \text{tr} \alpha^{(1)} \wedge \text{tr} \star \phi^{(2)} = \int_{\partial\Omega} \text{tr} \alpha^{(1)} \wedge \text{tr} \phi^{(0)} \quad (4.42)$$

$$= \int_{\partial\Omega} (\alpha_\xi d\eta - \alpha_\eta d\xi) \wedge \phi(\xi, \eta) \quad (4.43)$$

$$= \int_{\partial\Omega} \sum_{i=1}^N -\alpha_i^\xi e_i(\xi) \phi(\xi, \eta_b) \quad (4.44)$$

$$= \sum_{i=1}^N \sum_{j=0}^N -\alpha_i^\xi \phi(\xi_j) e_i(\xi_j) w_j. \quad (4.45)$$

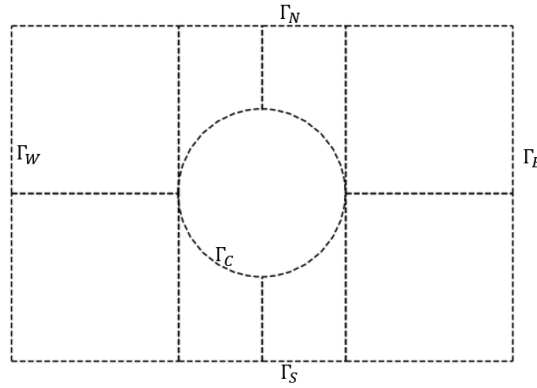


Figure 4.3: Physical domain and boundaries for the test

In the equation, the $\alpha_\xi d\eta$ is canceled, because at boundary there is only one component and the other one is zero. Note that the calculation is done in reference domain.

So there will be N number of boundary values being assigned corresponding to the polynomial order. Removing the test function term in the equation, we will get the right hand side $tr \star \phi_h^{(0)}$. That's the boundary integral in x-direction, for y-direction, the calculation is similar. It is worthwhile to point out that if the ϕ at boundary is constant, then we can move the ϕ term out of integration and use the property of edge function that $\int e(x) = 1$, so the right hand side can be directly assigned.

For the manufactured solution case, the boundary conditions are set that the ϕ values are applied to the left and right side and around the cylinder boundaries, so $\Gamma_1 = \Gamma_W \cup \Gamma_E \cup \Gamma_C$. The u values are applied to the rest of the boundaries, namely top and bottom side of the domain so $\Gamma_2 = \Gamma_N \cup \Gamma_S$. The given manufactured solution ϕ is designed to equal to zero along the exterior edges of the domain, except the cylinder, thus we only need to add boundary integral values along the cylinder at the right hand side term.

If it is the Dirichlet boundary condition where the top and bottom velocity is enforced to the boundary, then for every u cochain term along the boundary, the corresponding equation in the system is taken out and assigned to the boundary values at right hand side of the equation.

4.4.1 Results

Solutions with polynomial order $N=9$ are shown in Figure 4.4a and 4.4b, for ϕ and u respectively. It can be seen that the numerical results are in a good agreement with the manufactured solutions, which can be confirmed by the L2-norm error¹. In this case, L2 error of both ϕ and u has reached to the magnitude of order 10^{-5} .

By increasing the polynomial degree, the L2-norm error continues to decrease until it converges to the machine precision level, which means that the exact solution and the numerical result are treated as equal. The L2 plots are shown in the Figure 4.5a - 4.5b, for polynomial degrees verses ϕ error and u error respectively.

In the two semi-log plots 4.5a and 4.5b, we noted that the L2 error decreases exponentially with increasing polynomial order. So we have an exponential convergence rate here, which is the fastest convergence rate we can get. Such convergence rate reflects the advantage of the spectral element method. It is faster than any classical high-order finite element method. The exponential convergence rate of the method can be confirmed by [16, 26].

It should be mentioned that this numerical results solution is never equal to the exact solution regardless the tiny difference with the exact solution, it merely means that the computer has reached its limit to handle small numbers, and that is why the convergence plot flattened at certain polynomial degrees. It can be seen that when the polynomial degree of N is 21, the error (blue line) has reached the machine precision level. If the element size is smaller (the case $r=2$ or 3 in the figures, meaning half or one third of the original size), the N could be smaller to converge to the machine precision level.

¹The L2-norm error is the square root of the summation of the squared difference between the calculated value and the exact value, $\|u_{exact} - u_h\|_{L^2} = \sqrt{\int_{\Omega} |u_{exact} - u_h|^2 d\Omega}$

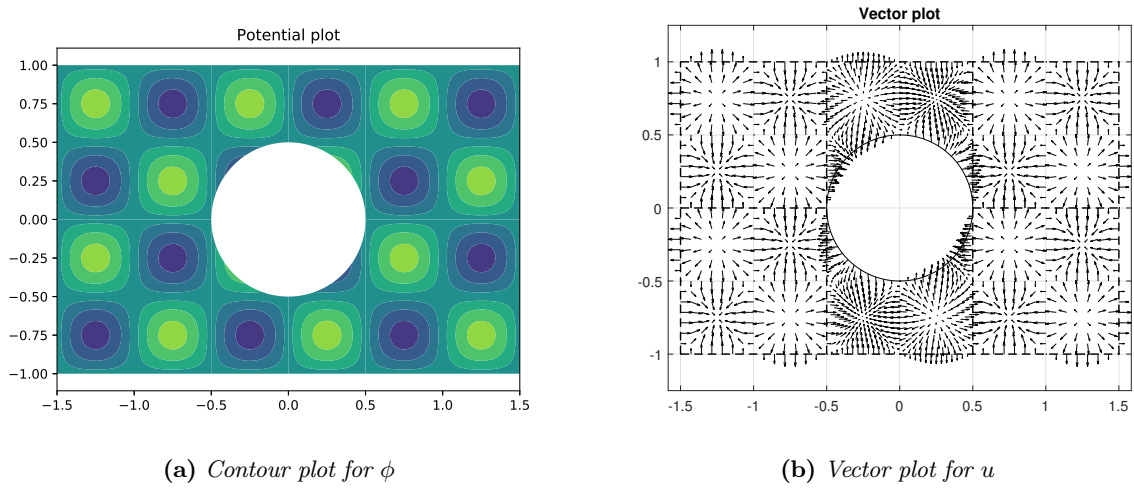


Figure 4.4: Numerical solution for the manufactured case

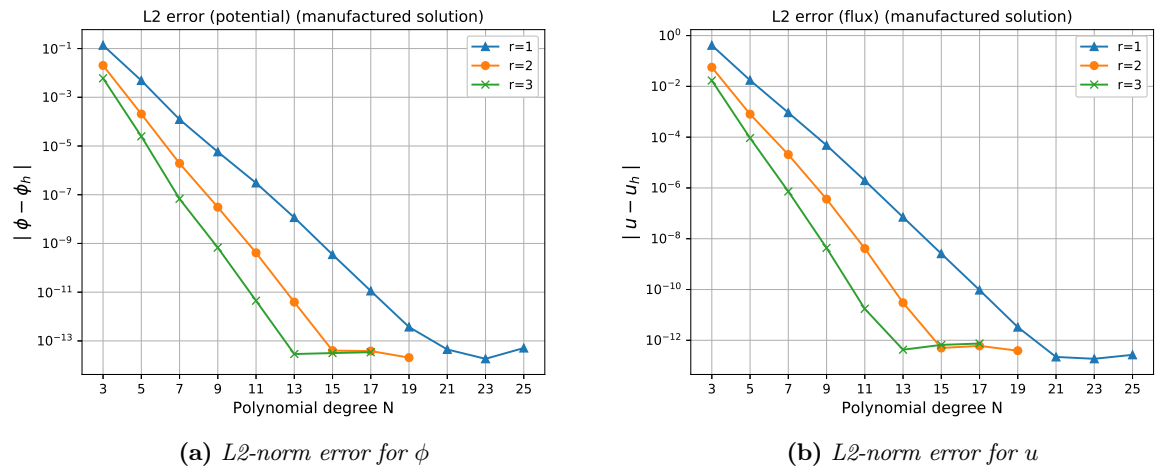


Figure 4.5: L_2 -norm error for the manufactured case (polynomials order vs. error)

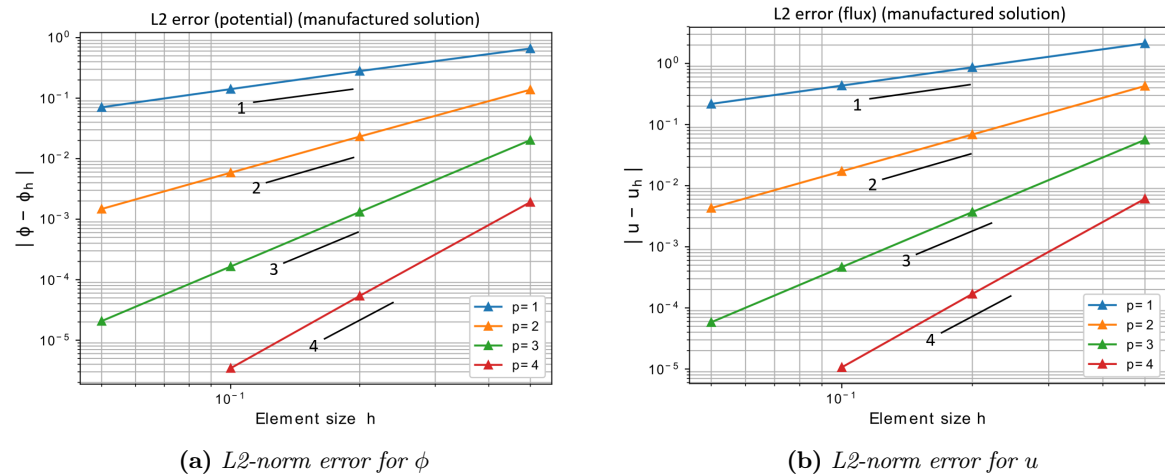


Figure 4.6: L_2 -norm error for the manufactured case (element size vs. error)

The initial 8-element domain can be refined by splitting the original element by half or $r \times r$ (in the code it is called element layout), thus one can have smaller element size. If the element layout is $r=3$, it means the original element is divided into nine smaller elements equally. That is what we call the h-refinement in computational physics. We have already shown that the h-refinement is helpful to the convergence, in the Figure 4.5a and 4.5b, where $r=1,2,3$ are plotted.

We can also express the effect of h-refinement in an element size verses error plot, as shown in the Figure 4.6a and 4.6b. At the same polynomial order, the L2-norm error decreases with smaller element size. The plots also confirm that the convergence rate for each polynomial order case is stable, and the slope of the convergence is very close to the theoretical convergence rate [26] which is equal to the degree of the basis functions $Slope=1,2,3,4$.

If the manufactured solution is simpler, for example, lower order manufactured solution, it is easier to converge as the convergence plot in the Figure 4.7 indicates, where each line represents an individual manufactured solution. The boundary conditions applied are the same as the previous case.

It should be highlighted that if the domain is an orthogonal shape, such as a square unit, it is also easier to converge. Particularly for the low order manufactured solution, it will converge to the machine precision level instantly. For $\phi=x$ case, the numerical solution is already at machine precision level error at polynomial degree 1 and the $\phi=(3-x)y(2-y)$ will converge to the machine precision level at degree of 3. But that instant convergence does not happen in this flow around cylinder domain.

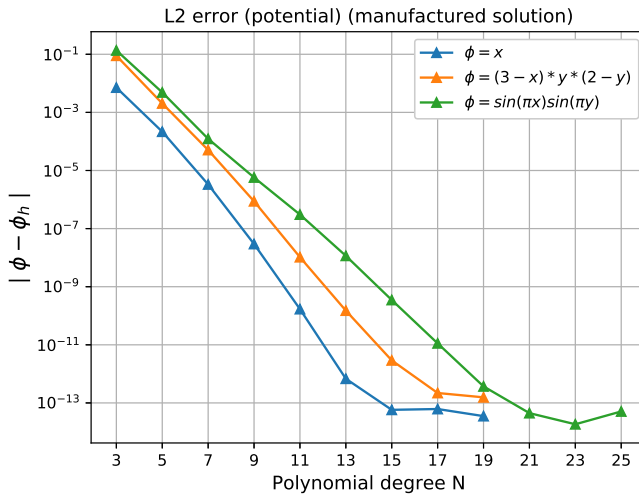


Figure 4.7: Convergence plot for simple manufactured solution

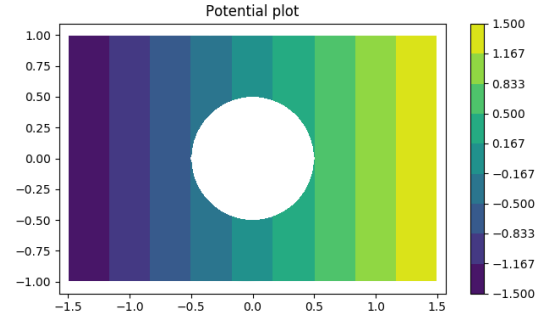


Figure 4.8: Potential contour of $\phi=x$

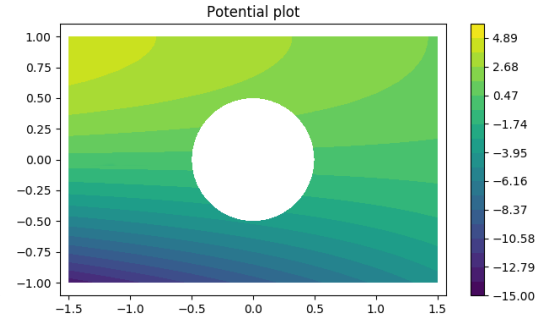


Figure 4.9: Potential contour of $\phi=(3-x)y(2-y)$

It should be mentioned that for all Dirichlet Boundary condition, the results do not show a significant difference. The polynomial degree that reached the machine precision is the same with the case mentioned above.

4.5 Laplace Equation

Without the source term f , the Poisson equation becomes the Laplace equation, which in fluid dynamics, the potential flow follows and analytical solutions exist. Rewrite the Laplace equation into differential form, shown as:

$$u^{(1)} - d^* \phi^{(2)} = 0, \quad (4.46)$$

$$du^{(1)} = 0. \quad (4.47)$$

The following steps is the same with the case of Poisson equation. The weak form of the above equation is given by inner product, and reduced by mimetic projection π_h . Finally, the matrix form ends at

$$\begin{bmatrix} \mathbb{M}^{(1)} & \mathbb{E}^{(2,1)T} \mathbb{M}^{(2)} \\ \mathbb{M}^{(2)} \mathbb{E}^{(2,1)} & 0 \end{bmatrix} \begin{bmatrix} u^{(1)} \\ \phi^{(2)} \end{bmatrix} = \begin{bmatrix} tr \star \phi^{(2)} \\ 0 \end{bmatrix}. \quad (4.48)$$

4.5.1 Results

Case 1

In this section, there are two test cases for potential flow around the cylinder. The first case is a typical channel flow, where the boundary condition is set as shown in the Figure 4.10.

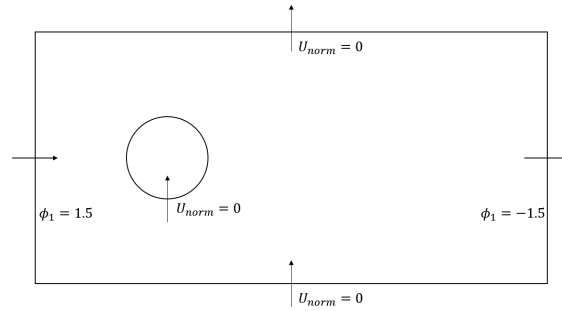


Figure 4.10: Boundary conditions for the case 1

The results for the velocity vector plot and potential plot is shown as Figure 4.11a and 4.11b respectively. Based on the figures, it can be said that at polynomial degree of 8, the solution has already been in a good agreement with how a channel flow should behave.

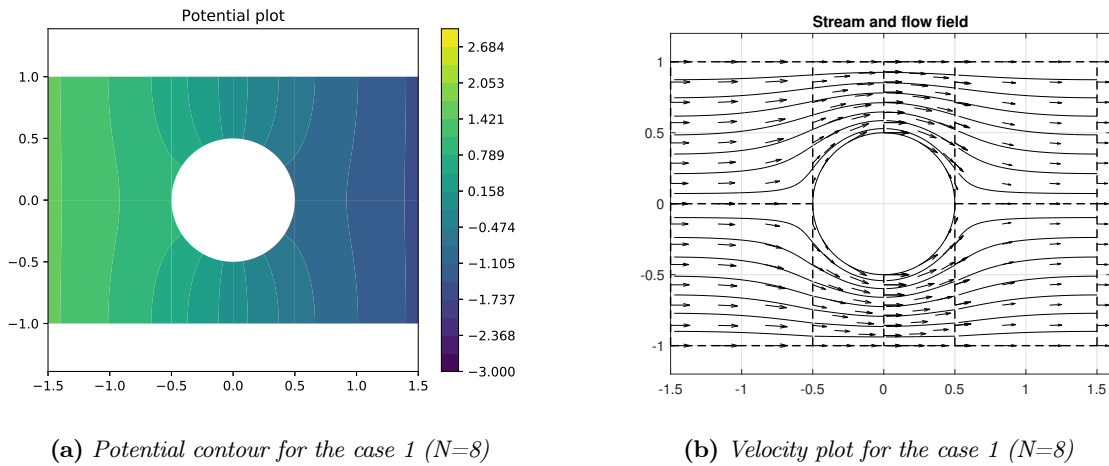


Figure 4.11: Results for channel flow solution case

With such boundary conditions, one cannot validate the results quantitatively since there is no analytical solution. But we can prove that the second equation, the continuity equation is satisfied by looking at the sectional flux. Since the normal velocity along the wall is set to zero, the flux from the left side section must equal to the right end section, as well as the section over the cylinder position. The calculation is made and it has been proved that the difference of flux has already been a machine precision values to the magnitude of 10^{-16} at $N=1$.

It should be noted that under low polynomial degrees e.g. $N=3$, there are some discrepancies (velocity vectors have different directions) along the conjunctions between elements. This is because along the boundary, it is the normal velocities that are calculated instead of the tangential one. As a result, under coarse mesh,

the difference in the tangential component is large to be visible. Such deviations will vanish with the increase of polynomial degrees.

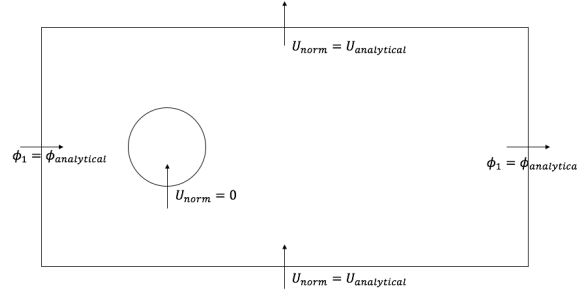


Figure 4.12: Boundary conditions for the case 2

Case 2

The second case is aimed for comparing the analytical solution to the numerical results, so we can obtain convergence plot. The potential flow has a classic analytical solution when the doublet and uniform flow are overlapped. Therefore, the boundary condition in the code is set to be consistent with the analytical solution, shown as the Figure 4.12, where $\phi_{analytical}$ and $u_{analytical}$ are values in the analytical solution.

The numerical solution, which depicts the potential ϕ and velocity u of polynomial order $N=8$ are shown in the Figure 4.13a and 4.13b. The velocity plot and potential plot shows a slight difference with the first channel flow case. But it can be noted that the normal velocity along the lower and upper wall is no longer zero.

Analytical Solution of Flow Around Cylinder

The detailed derivation of the analytical solution of flow around cylinder can be found in the [27]. Here we have a brief introduction to the inviscid, incompressible potential solution of flow over cylinder. In general, the solution is the superposition of uniform flow and a doublet.

So the velocity potential is given the Equation 4.49. Here, the polar coordinate used for convenience.

$$\phi = U_{\text{inf}} r \cos \theta + \frac{K \cos \theta}{r}, \quad (4.49)$$

where U_{inf} is the uniform flow velocity, a is the cylinder radius, $K = U_{\text{inf}} r^2$ is the strength of the doublet at the origin.

The velocity is derived by

$$v_r = \frac{\partial \phi}{\partial r} = U_{\text{inf}} \left(1 - \frac{a^2}{r^2}\right) \cos \theta, \quad (4.50)$$

$$v_\theta = \frac{1}{r} \frac{\partial \phi}{\partial \theta} = -U_{\text{inf}} \left(1 + \frac{a^2}{r^2}\right) \sin \theta, \quad (4.51)$$

where $a^2 = \frac{K}{U_{\text{inf}}}$ ■

After the numerical solution is obtained, we chose a series of sectional data, such as the sectional velocity across the center of the cylinder, for comparison. The results are as shown in the Figure 4.14. Clearly, the numerical solution fits perfectly to the analytical solution, which validates the correctness of the solution.

The overall L2 error are also calculated, as shown in the Figure 4.15a and 4.15b. The situation is similar to the Poisson equation case. With the increase of element number and polynomial degree, we get an exponential convergence rate. As for the h-refinement, they are shown in the Figure 4.16a - 4.16b, the slope of convergence rate is as the same as the manufactured solution case, which is equal to order of the method.

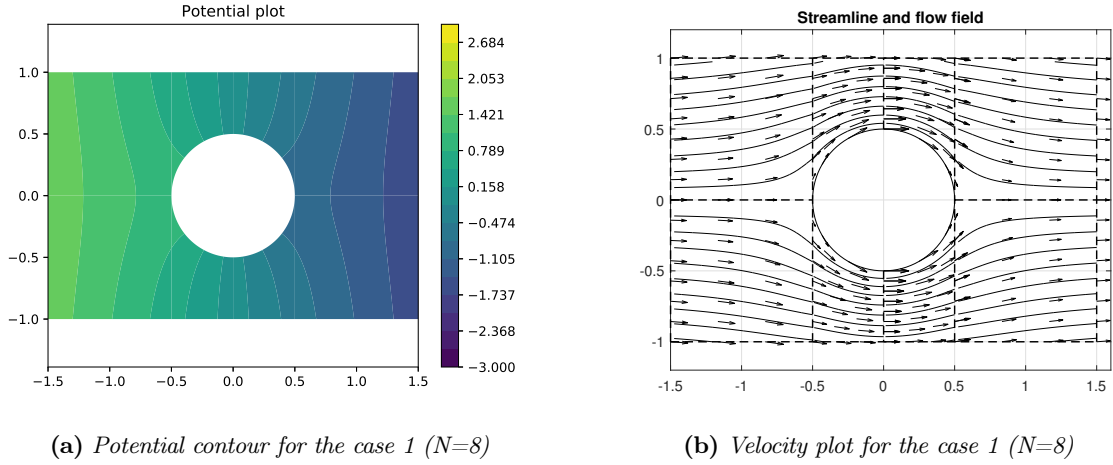


Figure 4.13: Results for analytical solution case

It can be seen that the L2 error decreases much quicker in the p-convergence plot than that in the h-convergence plot. So one may choose to do p-refinement in terms of L2 error performance. But in terms of the p-refinement vs. h-refinement, one should also consider factors such the number of degree of freedom of the system, CPU lapse time, etc. Here in our code, we made a quick comparison by taking the two cases with similar L2 error level. That are case 1 (element layout 1-by-1, polynomial degree 11) and case 2 (element layout 10-by-10, polynomial degree 4). The result shows that, in terms of number of degree of freedom, case 1 has 2860 number of unknowns while case 2 has 38240 unknowns. For the CPU lapse time, the case 1 spent 29.20 seconds to finish the computation while case 2 costs 75.47 seconds. So from the point of view of matrix size and time consumption, the p-refinement outperforms the h-refinement.

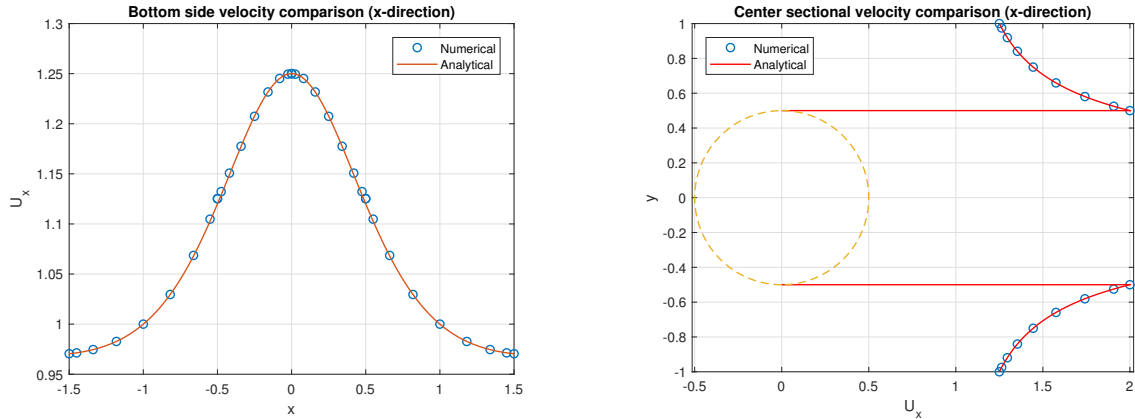
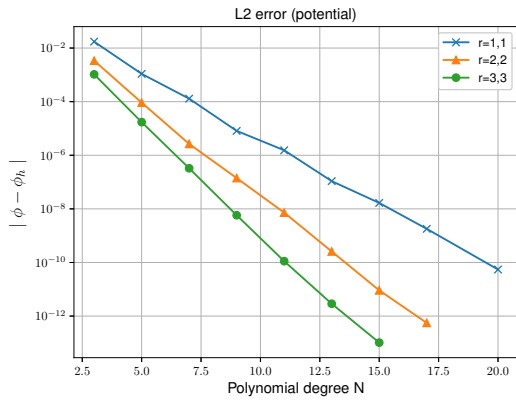


Figure 4.14: Comparison between numerical results and exact solution

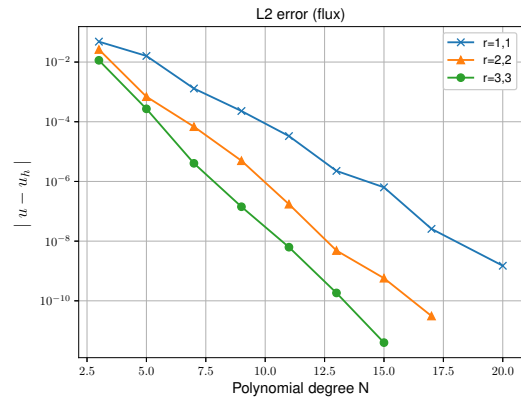
In addition, it is also possible to alter the shape of the cylinder to other shape for practical application, such as an airfoil. As the typical NACA airfoils have polynomial definitions, the transfinite mesh made it easy to embed the airfoil definition into the code, while the system equation remains the same. Here an example case is given, where a NACA0012 airfoil is made. The potential contour is shown in the Figure 4.17a.

The boundary condition is set that at left and right boundary, Γ_W and Γ_E are prescribed with ϕ while at the top Γ_N , bottom Γ_S and the airfoil surface Γ_C , the velocity u is prescribed. After the reconstruction, we can obtain the velocity over the airfoil surface and calculate the pressure coefficient C_p by Bernoulli equation, $C_p = 1 - (\frac{u}{U_\infty})^2$. The pressure distribution plot is shown in the Figure 4.17b. We use a panel method solution [28] for comparison. As we can see, the numerical solution is in a good agreement with the panel method results. It should be highlighted that if the boundary condition is prescribing velocity, then the Kutta condition should be applied to eliminate the velocity jump at trailing edge.

On the other hand, we can remain the shape but alter the governing equations, for example the Stokes flow, where the ϕ will play the role of pressure and an additional variable, the vorticity will be added. The

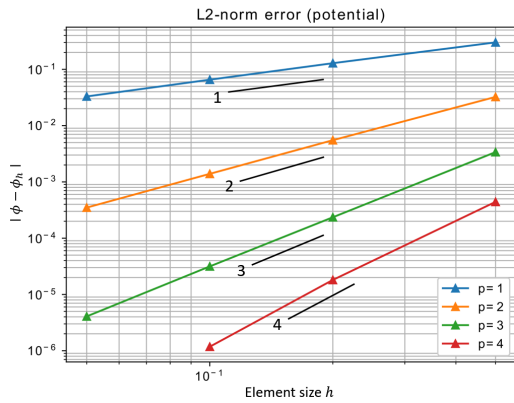


(a) L_2 -norm error for ϕ

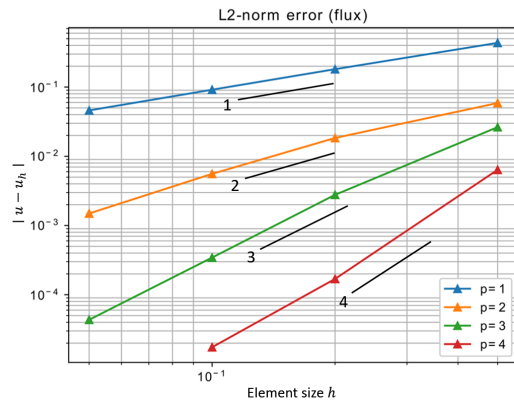


(b) L_2 -norm error for u

Figure 4.15: L_2 -norm error for the analytical case (polynomials order vs. error)



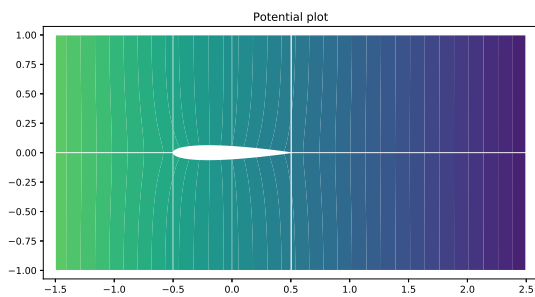
(a) L_2 -norm error for ϕ



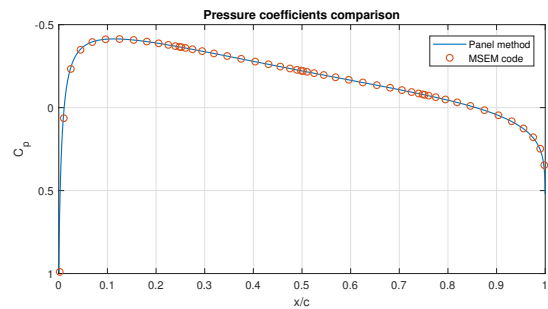
(b) L_2 -norm error for u

Figure 4.16: L_2 -norm error for the analytical case (element size vs. error)

detailed governing equation can be found in [16].



(a) Polynomial degree 10, element layout 2×2



(b) Pressure coefficients over the airfoil (AOA=0)

Figure 4.17: Mimetic spectral element method for airfoil

5 | Conclusion and Recommendations

Conclusion

In this thesis, we applied a numerical computation scheme called mimetic spectral element method to solve the PDEs. The test case of flow around cylinder requires the mesh to fit the body so the curvilinear mesh is used. Judging from the manufactured solution, the scheme is concluded as a success that both discretization and computation is feasible.

One of the advantages of the spectral element method is the exponential convergence rate. In addition, the fact that the vector calculus operators are replaced with a single exterior derivative makes the derivation of equation less complex. The generalized Stokes theorem enables the algebraic topology to do the discretization of differential equations. It has been shown that it is very natural to discrete the differential forms and is compatible to numerical computation.

With help of transfinite interpolation, the physical domain can be accurately meshed without any approximation. This is a significant advantage over the regular finite element method, where the curved surface is usually approximated by polygons. As long as one obtains the parametric description of the boundaries, one can easily generate a mesh that exactly fits to the body.

The Poisson equation is a good example problem to show how the scheme goes and how the different steps are involved when deriving mimetic spectral element method. The error analysis for Poisson equation shows that the computation converges to the machine level precision where we realized a exponential p-convergence rate (Figure 4.5a, 4.5b) and an optimal h-convergence rate (Figure 4.6a, 4.6b), thus reflect the effectiveness of the method over the deformed domain. The Laplace equation for flow around cylinder shows the same accuracy level and convergence rate as the Poisson equation.

The Python library `MimeticFEM` has provided various functions and methods to construct the system equations with differential forms. Though the package is not mature to release and has some sign convention problem, it has provided correct results to the thesis. Besides it has a series of handy functions that can save a lot of time and avoid trivial errors. Based on the computation time consumption, we have demonstrated that in our code, the p-refinement outperformed the h-refinement.

Recommendations

A straightforward future research goal is to implement another governing equation to test the mesh and the code. As mentioned, the Stokes flow is a reasonable choice since it has manufactured solution and practical meaning in fluid dynamics. One of the initial goals of the thesis is to realize a potential flow together with Stokes flow. With the introduction of viscosity, it is possible to apply the no-slip boundary condition. Nevertheless, some problems were encountered for the Stokes equation. Only one test case of lid-driven problems was developed, and due to the time limit, the rest is dropped. Based on Stokes flow, one can continue to proceed to the Navier-Stokes equations to realize a completed fluid simulation.

For mesh, the transfinite interpolation is successfully implemented in the code, but it should be tested on more different shapes. In this study, we have tested the mesh with one cylinder arc being tangential to its neighboring edges. Thus it is cautiously optimistic to say the transfinite mesh can handle the steep curves or bends.

The transfinite interpolation method is actually developed for CAD applications. Then it is worthwhile to explore if the CAD modeling and the physical modeling can be combined. For example, the complex shape is modelled by transfinite interpolation grid and then use the same grid points to solve the governing equations over the domain. In the field of finite element method, there is a similar approach called isogeometric analysis, which integrate NURBS geometry into finite element analysis, so the models can be designed and tested using the same data set [29].

For the implementation, there is also room for further improvement. One is the multi-element support. In the thesis, there are 8 basic elements which can be divided into smaller ones. However these basic elements are made manually. Thus a mesh generator is recommended, that could automatically generate the mapping of grids so one does not need to write them for every new case. Another one is to extend the code into 3D, so a lot of problems can be used either as a test or as practical calculation. For wind energy, the flow around the rotor blade can not use 2D solver as it has the third component when the wind blade rotates.

If one to use it in practical, it is worthwhile to test the computation efficiency of the code. Although, we have made a comparison between the p-refinement and the h-refinement, more broad and rigorous reviews can be extended. With the increase of computation capabilities of computer chips, having fewer elements to save computing time might become less attractive. One potential research subject may be a benchmark to assess the time lapse of mimetic spectral element method to reach the same accuracy by other methods.

Acknowledgement

No good result is obtained alone, I would like to give my honest thanks to those involved in my thesis.

My supervisor Dr. ir. Marc Gerritsma, offered an excellent opportunity to explore the world of scientific computation. Marc is the lecturer of the CFD courses in Aerodynamics. During the master study, I obtained a lot of working knowledge and picked up various expertise tools thanks to Marc's lectures and thesis. I will appreciate and cherish this experience, because hardly there will be any chance that one can sit down and learn things dedicatedly after graduation. I thank Marc for the guidance of my thesis.

I would also like to thank to Ir. Varun Jain who offers tremendous and extraordinary support during my final thesis study. I appreciate his patience in dealing with some difficult subjects and proofreading my thesis draft. His expertise and knowledge is amazing. Anyway, it is an honor to work with him.

The thank list does not end here. I was helped by a lot of people, from lecturers to PhD fellows, from friends to classmates to people that I only know once. Getting a diploma is not easy, every support counts and I do appreciate.

Thank you!

Shuming Liu

Delft, the Netherlands

Bibliography

- [1] Gerritsma, M., “An introduction to a compatible spectral discretization method,” Mechanics of Advanced Materials and Structures, Vol. 19, No. 1-3, 2012, pp. 48–67.
- [2] Rogers, S. M., Optimal Control of Nonholonomic Mechanical Systems, Ph.D. thesis, University of Alberta, 2017.
- [3] Palha, A., Rebelo, P. P., Hiemstra, R., Kreeft, J., and Gerritsma, M., “Physics-compatible discretization techniques on single and dual grids, with application to the Poisson equation of volume forms,” Journal of Computational Physics, Vol. 257, 2014, pp. 1394–1422.
- [4] Tonti, E., “On the mathematical structure of a large class of physical theories,” Tech. rep., Dipartimento di Matematica - Politecnico di Milano, 1971.
- [5] Tonti, E., “Why starting from differential equations for computational physics?” Journal of Computational Physics, Vol. 257, 2014, pp. 1260–1290.
- [6] Frankel, T., The geometry of physics: an introduction, Cambridge University Press, 2011.
- [7] Bott, R. and Tu, L. W., Differential forms in algebraic topology, Vol. 82, Springer Science & Business Media, 2013.
- [8] Wikipedia, “Manifold — Wikipedia, The Free Encyclopedia,” 2017, [Online; accessed 31-October-2017].
- [9] Burke, W. L., Applied differential geometry, Cambridge University Press, 1985.
- [10] Misner, C. W., Thorne, K. S., and Wheeler, J. A., “Gravitation,” 1973.
- [11] Wilson, S., “Differential forms, fluids, and finite models,” Proceedings of the American Mathematical Society, Vol. 139, No. 7, 2011, pp. 2597–2604.
- [12] Morita, S., Geometry of differential forms, Vol. 201, American Mathematical Soc., 2001.
- [13] Vassiliou, P. J. and Lisle, I. G., Geometric approaches to differential equations, Vol. 15, Cambridge University Press, 2000.
- [14] Loomis, L. H. and Sternberg, S., Advanced Calculus: Revised, World Scientific Publishing Company, 2014.
- [15] Mattiussi, C., “The finite volume, finite element, and finite difference methods as numerical methods for physical field problems,” Advances in Imaging and electron physics, Vol. 113, Elsevier, 2000, pp. 1–146.
- [16] Kreeft, J. and Gerritsma, M., “Mixed mimetic spectral element method for Stokes flow: A pointwise divergence-free solution,” Journal of Computational Physics, Vol. 240, 2013, pp. 284–309.
- [17] De Doncker, P., “The use of transfinite elements in the method of moments applied to electromagnetic scattering by dielectric cylinders,” Antennas and Propagation, 1999. IEE National Conference on., IET, 1999, pp. 41–44.
- [18] Gordon, W. J. and Hall, C. A., “Construction of curvilinear co-ordinate systems and applications to mesh generation,” International Journal for Numerical Methods in Engineering, Vol. 7, No. 4, 1973, pp. 461–477.
- [19] Thompson, J. F., Soni, B. K., and Weatherill, N. P., Handbook of grid generation, CRC press, 1998.
- [20] Steinberg, S., Fundamentals of grid generation, CRC press, 1993.
- [21] Kreeft, J. J., Palha, A., and Gerritsma, M. I., “Mimetic spectral element method for generalized convection-diffusion problems,” Proceedings of ECCOMAS2010, this issue, 2010.
- [22] Karniadakis, G. E. and Sherwin, S. J., “Spectral/hp element methods for CFD,” 1999.
- [23] Proot, M. M. and Gerritsma, M. I., “A least-squares spectral element formulation for the Stokes problem,” Journal of Scientific Computing, Vol. 17, No. 1-4, 2002, pp. 285–296.

BIBLIOGRAPHY

- [24] Gerritsma, M., “Edge functions for spectral element methods,” Spectral and High Order Methods for Partial Differential Equations, Vol. 76, 2011, pp. 199–207.
- [25] Robidoux, N., “Polynomial histopolation, superconvergent degrees of freedom, and pseudospectral discrete hodge operators,” 2008.
- [26] Palha, A., Koren, B., and Felici, F., “A mimetic spectral element solver for the Grad–Shafranov equation,” Journal of Computational Physics, Vol. 316, 2016, pp. 63–93.
- [27] Anderson Jr, J. D., Fundamentals of aerodynamics, Tata McGraw-Hill Education, 2010.
- [28] Katz, J. and Plotkin, A., Low-speed aerodynamics, Vol. 13, Cambridge university press, 2001.
- [29] Cottrell, J. A., Hughes, T. J., and Bazilevs, Y., Isogeometric analysis: toward integration of CAD and FEA, John Wiley & Sons, 2009.

