Discontinuous Galerkin reduced basis approximation for direct numerical simulation of the Navier Stokes equation: Master Thesis Report

Nirav Vasant Shah,
M.Sc. Candidate, Water Resources Engineering and Management,
University of Stuttgart,
Stuttgart, Deutschland

Supervisor: Prof. Dr. Bernard Haasdonk,
Institute of Applied Analysis and Numerical Simulation,
Universitity of Stuttgart,
Stuttgart, Deutschalnd

Co-advisor: Prof. Gianluigi Rozza,
Scuola Internazionale Superiore di Studi Avanzati,
Trieste, Italy

Co-advisor: Dr. Martin Hess,
Scuola Internazionale Superiore di Studi Avanzati,
Trieste, Italy

December 12, 2017

Contents

Chapter 1

Introduction

The topic of the thesis deals with the reduced basis approach for the numerical solution of Navier Stokes Equation which is the core of Computational Fluid Dynamics. The thesis performs numerical simulation of Navier Stokes equation through Discontinuos Galerkin method and subsequently the parametrization of equation and reduced basis approach though Proper Orthogonal Decomposition.

The goal of thesis is to derive, discretize and implement the Discontinuous Galerkin weak form of Navier Stokes equation and perform numerical simulation of Navier Stokes equation. In the course of numerical solution we measure the simulation efforts.

Subsequently the equation is parametrized for parameters such as fluid property, geometry of domain, boundary condition. This allows the approximation of numerical solution, with approximation being with respect to parameter space. The solution of the parametrized form is stored for reduced basis evaluations.

The Proper Orthogonal Decompostion than sorts and extracts the stored solutions based on eigen value. This extraction with parametrised form is used for prediction of full numerical solution. In this process the evaluations are made faster but with increase in induction of error. We compare the time saving vs. induced error.

In chapter 2, Navier Stokes equation is introduces. We first derive the Navier Stokes equation from conservation equation or Reynolds Transport Theoreom. We discuss about the flow classification and important issues related to numerical simulation of Navier Stokes equation. At the end of the chapter we discuss boundary conditions and its consequences on well posedness of the problem.

In chapter 3, we bring the problem from infinite dimensional domain to finite dimensional problem by introducing discretisation. We first introduce grid, constituents of elements and transformation between local and global geometry. Further the function spaces over the grid are discussed during which, the interpolation function, need for different function spaces for pressure and velocity is discussed. We than discuss the advantages of Discontinuous Galerkin method over Finite Volume method and develop more detailed understanding of the interpolation function used over the element. We further introduce the weak form and discrete formulation of Navier Stokes equation.

In chapter 4, We discuss implementation of the discrete form of Navier Stokes equation in RBMATLAB, an opensource software developed by University of Stuttgart and University of Mnster, for numerical simulation. During the chap-

ter we discuss matrix assembly, dimension of assembled matrices, solver performance issues, boundary treatment and basis function formation in RBMATLAB.

In chapter 5, we discuss some properties and tests useful for examination of assembled matrices. We further present results from numerical experiment and some case studies.

In chapter 6, we interpret and explain the outcomes of numerical experiments.

In chapter 7, we introduce parametric space, discuss parametrization and empirical interpolation method. We than perform reduced basis approximation and proper orthogonal decomposition method.

In chapter 8, we perform the numerical experiment related to reduced order modelling and discuss, interpret and assess the results form reduced basis method.

In appendix, supplementary theoretical background is provided which is expected to help the readers to better comprehend the thesis.

Chapter 2

Engineering perspective and mathematical formulation

The subject of mathematical applications in fluid mechanics starts with one of the variants of Navier Stokes equation. Almost all processes of fluid mechanics require considerations related to Navier Stokes equation. Hence the importance of Navier Stokes equation is impossible to be ignored as far as mathematical approaches in fluid mechanics are concerned. The numerical method for incompressible equation is far more simple as compared to compressible Navier Stokes equation. This is due to removal of dependence on equation of state.

2.1 Derivation of Navier Stokes equation

Before deriving Navier Stokes equation we introduce some notations. The domain is denoted by $\Omega \subseteq \mathbb{R}^d$. The domain boundary is denoted by Γ. The domain boundary is divided into Dirichlet boundary Γ_D and Neumann boundary Γ_N i.e. $\Gamma_D \cup \Gamma_N = \Gamma$.

The governing equations for incompressible stokes flow are conservation equations: Mass conservation and momentum conservation. The cnoservations equations are derived based on concept of control volume and control surface. Control volume is the volume, fixed or moving with constant velocity in space, through which fluid moves. Control surface is the surface enclosing control volume. All equations can be derived from Reynold's transport equation:

$$\frac{dB}{dt}|_{system} = \frac{d}{dt}\int_{cv} b\rho dV + \int_{cs} b\rho u.dA \tag{2.1}$$

cv = Control volume
cs = Control surface
B = Extensive property under consideration
b = Intensive property corressponding to B
ρ = Density of fluid
u = Velocity of fluid at the control surface

If in the above equation B is substituted as momentum P, correspondingly b as velocity u, we receive change in momentum which as per Newton's Second law of Motion is equal to sum of external forces acting on the system.

$$F = \frac{dP}{dt} = \frac{d}{dt}\int_{cv} u\rho dV + \int_{cs} u\rho u.dA \tag{2.2}$$

This sum of forces arises from stresses σ (shear stresses and normal stresses) and external force f such as weight, Electromagnetic force.

$$F = \int_{cs} \sigma \cdot dA + \int_{cv} \rho f dV \tag{2.3}$$

Where,
σ = Stress
f = External force per unit volume

Equating external force with change in momentum i.e. equating (2.2) and (2.3) and with the application of Gauss divergence theoreom we derive Navier Stokes equation,

$$-2\nabla \cdot (\nu \nabla^s u) + (1/\rho)\nabla p + (u \cdot \nabla)u = f \quad \text{in} \quad \Omega \tag{2.4}$$

The incompressible mass conservation equation can be written as

$$\nabla \cdot u = 0 \quad \text{in} \quad \Omega \tag{2.5}$$

The boundary conditions can be expressed as,
Dirichlet boundary:

$$u = u_D \quad \text{on} \quad \Gamma_D \tag{2.6}$$

Neumann boundary:

$$-pn + 2\nu(n \cdot \nabla^s)u = g \quad \text{on} \quad \Gamma_N \tag{2.7}$$

Where,
u = flow velocity and $u : \Omega \to \mathbb{R}^d$
p = pressure and $p : \Omega \to \mathbb{R}$
ν and ρ = kinematic viscocity and density (fluid properties) and $\nu : \Omega \to \mathbb{R}$ and $\rho : \Omega \to \mathbb{R}$
f = external force $f : \Omega \to \mathbb{R}^d$
u_D = specified flow velocity at Dirichlet boundary $u_D : \partial\Omega_D \to \mathbb{R}^d$
n = normal unit vector $n : \partial\Omega \to \mathbb{R}^d$
g = specified Neumann flux $g : \partial\Omega_N \to \mathbb{R}^d$

The (2.4) is known as Strong form.

It can be seen that the steady state Navier Stokes equation is non linear and has two unknown variables, pressure P and velocity u. The additional equation added by continuity equation is hence necessary in order to sufficient number of equation for sufficient number of unknowns.

We also introduce dimensionless number Reynolds number,Re which is the most characteristic of the flow. Reynolds number is defined as the ratio of Inertial force to the viscous force and is measured by,

$$Re = uL/\nu \tag{2.8}$$

Here, L = Characteristic geometrical dimension, for example Diameter of Pipe in case of pipe flow or Span of wing of Aircraft in case of flow over aircraft wing

2.1.1 Direct numerical simulation

We now differentiate between the type of flows, Laminar and Turbulent.
Laminar flow is characterised by well defined velocity and pressure field and low Reynolds number. This flow has very low velocity and pressure fluctuations. The viscous force is balanced by pressure force and the flow has negligible inertial force. Mathematically, the non linear term in (2.4) is no longer present. Such equation is known as Stokes equation.

In contrast, the Turbulent flow is characterised by fluctuations in velocity and pressure field and high Reynolds number. The flow has high velocity and inertial force is present in addition to viscous and pressure force. This inertial force makes the equation (2.4) non linear. The fluctuations of velocity and pressure are of the order of Kolmogrov scale.

The method used for solving equation (2.4) numerically is known as Direct Numerical Simulation, abbreviated as DNS. The direct numerical simulation could be computationally very expensive especially in applications such as turbulent flows as the time and space grid size is of the order of Kolmogrov scale but the time period or space dimension over which the simulation is carried out are very large. In order to avoid such computational expenses alternate models are used replacing original model. However, the alternate model can not explain completely the flow physics. The prediction of accurate flow physics description requires economical numerical solution of (2.4). It is to be noted that simulation of turbulent flow is always a compromise between required flow physics prediction and computational efforts.

2.2 Wellposedness of strong form of Navier Stokes equation

Chapter 3

Discretization and function spaces

3.1 Grid geometry

In numerical anlysis the problem is solved into finite dimensional domain. This finite dimensional domain is constructed by projecting actual domain onto a subdomain called as grid through a grid function. If the original domain is denoted by Ω we denote the grid by Ω_h and define grid function by $G : \Omega \to \Omega_h$ and we note that $\Omega_h \subset \Omega$. In present case we use triangular grid and denote each triangle as τ_h. We also note that $\Omega_h = \cup_{h=1}^{nel} \tau_h$ where nel is the total number of elements in the grid. Each triangle is an 'Element'of the grid. The boundary betwen elements i.e. interelement boundary is denoted by Γ. In case of grid the boundaries comprises domain boundaries and interelement boundaries i.e. $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma$. During discussion on jump operator and average we denote the element under consideration as τ_h^- and neighbouring element as τ_h^+. We also denote normal pointing from element itself towards neighbouring element as n^+ and normal pointing from neighbouring element towards element itself as n^-. Corresspondingly every quantity from element itself is denoted by superscript $+$ and from neighbouring element is denoted by $-$.

In case of 2-dimensional domain the grid could be triangular grid or rectangular grid. The triangular grids are useful for irregular geometry and also on regular geometry if flow physics, and correspondigly the solution, is expected to be complex. This flexibility requires additional efforts to define the grid accurately. That is, unlike structured grid, an unstructured grid needs to define connectivity of vertices, which form an edge, which in turn forms a face. These faces in case of 3-dimensional grid constitute a tetrahedral elements. In case of 2-dimensional grids we have only faces which are 2-dimensional entity, edges which are 1-dimensional entity and points or vertices which are 0-dimensional entities.

For triangular grids we also consider Barycentric co-ordinate system. In Barycentric co-ordinate system any point within the triangle is reprensented in terms of vertices forming the triangle. Any point r within triangle is expressed in terms of vertices forming the triangle r_1, r_2, r_3. This is represented as,

$$r = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3 \tag{3.1}$$

where, $\lambda_1, \lambda_2, \lambda_3$ are weightages which represent the distance of point r from respective vertices. The weightages satisfy the criteria,

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{3.2}$$

Hence we only need to satisfy 2 values in 2-dimensional plane in order to fully define the position of point.

For example, the centroid of triangle will have $\lambda_1 = 1/3, \lambda_2 = 1/3$. By (3.2) we have $\lambda_3 = 1/3$ i.e. a point which is equidistant to all vertices.

3.2 Grid parameters

We refer "Grid parameters" as the geometrical parmeters which are dependent only on the geometry of the problem or the grid or both. These parameters do not depend upon the mathematical formulation but are supplementry to the mathematical formulation. On the triangular grid we have 3 entities faces, edges, vertices as explained above. From faces we have the area (equivalent to volume of element in case of 3-dimensional grid) and Jacobian. As explained later in the weak form of Navier Stokes Compact Discontinuous Galerkin and transformation between local and global geometry the area of element is useful for volume intergral terms and the Jacobian is useful for gradient transformation. From edges we derive the edge length which is useful for boundary integral terms and normal vector which is useful for flux calculation. The normal vector is considered positive when pointing outward from the element negative when pointing inward to the element. Every element has 3 neighbouring element and the element shares each of his edge with one of its neighbour. From vertices we derive the vertex index which helps to define the connectivity of the vertices which is useful especially in case of unstructured grid.

In order to give clear visualization of domain or precisely, continuous domain and grid or discretiozed domain we introduce a unit square with circular obstacle with center of circle coinciding with center of square.

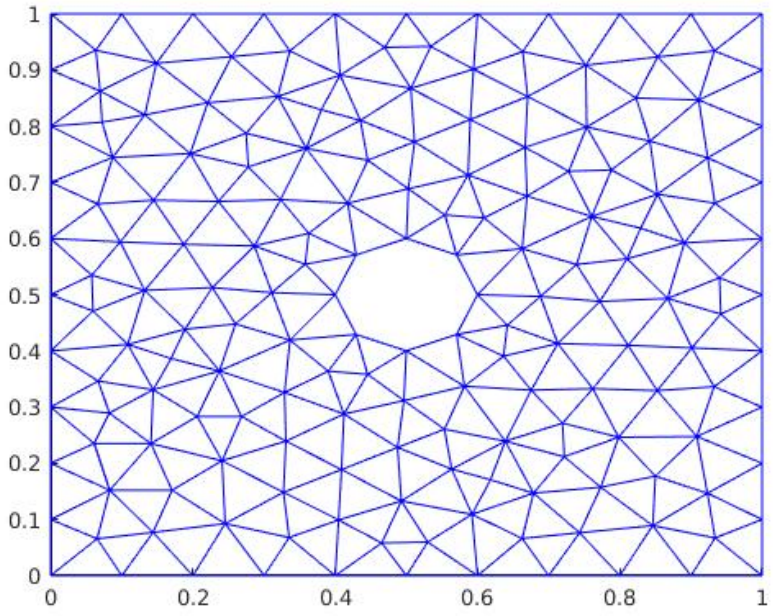


Figure 3.1: Discretized domain or grid

3.3 Global and local co-ordinate system

The integral terms are evaluated on a reference triangle instead of element itself. Accordingly, a co-ordinate transformation is performed for every element to reference triangle for evaluating integrals. The co-ordinate system in which reference triangle lies is called reference or local co-ordinate system and the co-ordinate system in which element itself lies is called global co-ordinate system. The reference triangle has vertices $(0,0),(1,0),(0,1)$ in order. The element is defined by vertex indices in order forming triangle. The mapping from reference triangle $\hat{T}$ to each element τ_k is defined by mapping,

$$F_k : \hat{T} \rightarrow \tau_k \tag{3.3}$$

This mapping function is defined as,

$$F_k : X = J_k \hat{X} + C \tag{3.4}$$

Here,

J_k = Jacobian matrix of element k or Rotational matrix for transformation from local co-ordinate system to global co-ordinate system

C = Translational matrix for transformation from local co-ordinate system to global co-ordinate system

X = Co-ordinates of vertices of element in Global co-ordinate system

$\hat{X}$ = Co-ordinates of vertices of reference triangle in local co-ordinate system

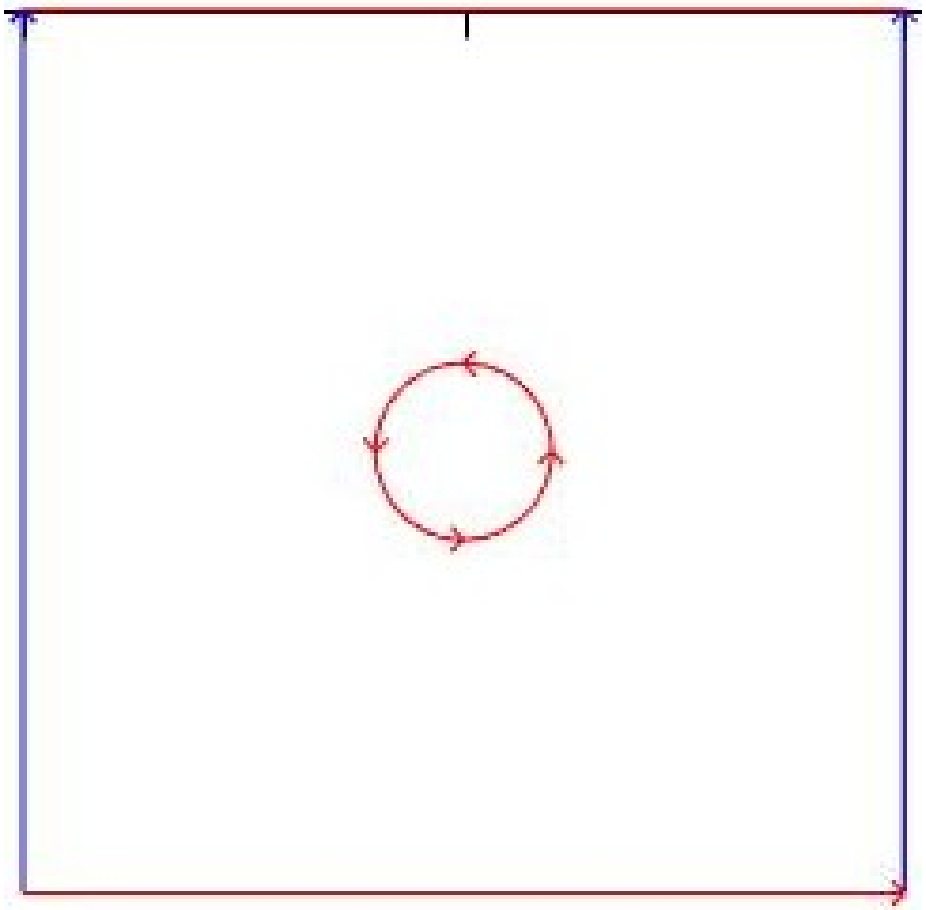
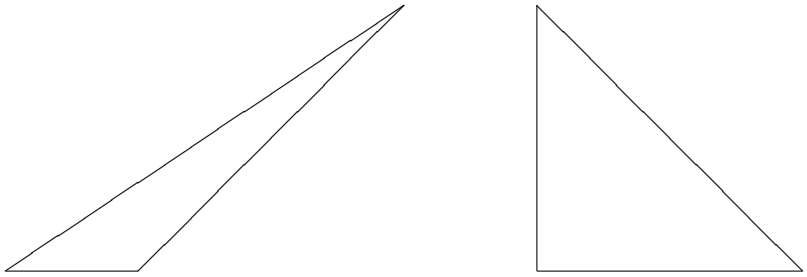


Figure 3.2: Continuous domain

We consider image of global function space on reference triangle and is represented by superscript $\hat{}$. This function space is known as local basis function space.



Global geometry (left) to Local geometry (right)

The volume integral of function $f(x)$ in global geometry is related to volume integral on local geometry as

$$\int_{\Omega} f(x)dx = \sum_{k=1}^{nel} \int_{\tau_k} f(x)dx = \sum_{k=1}^{nel} \int_{\hat{T}} f(\hat{x}) \det(J_k)d\hat{x} \tag{3.5}$$

The linear boundary integral of function $f(x)$ on global geometry is related to boundary integral on local geometry as,

$$\int_{\partial\Omega} f(x)ds = \int_{\partial\hat{\Omega}} f(\hat{x})ld\hat{s} \tag{3.6}$$

Also, the following holds,

$$\nabla g = JIT * \hat{\nabla}\hat{g} \tag{3.7}$$

where,

l = length of boundary on global geometry
g = A function in global co-ordinate system
$\hat{g}$ = A function in local co-ordinate system corresponding to function in g in global co-rodinate system

Here g and $\hat{g}$ satisfy,

$$g(x) = \hat{g}(\hat{x}) \tag{3.8}$$

3.3.1 Jump operator

The jump operator of quantity u at an internal boundary is defined as,

$$[u] = u^+ \cdot n^+ + u^- \cdot n^- \tag{3.9}$$

where n is normal to the cell boundary pointing outwards from the cell.

As pointed out by [1] this jump has two disadvantages.

1. The function space of quantity itself and the function space of jump are diferrent that is, jump of vector is scalar and jump of scalar is vector.
2. The use of this definition camoulages the presence of normal.

To overcome this disadvantages [1] used the jump definitions as ,

(here, Subscript i referes to the elements itself and subscript j refers to the neoghbouring element, [[]] refers to jump and n refers to normal vector.)

1.
$[[pn]] = p^+n^+ + p^-n^-$ on Γ
$[[pn]] = pn$ on Γ_D
where p is scalar.

2.
$[[n \otimes v]] = n^+ \otimes v^+ + n^- \otimes v^-$ on Γ
$[[n \otimes v]] = n \otimes v$ on Γ_D
or
$[[n \cdot v]] = n^+ \cdot v^+ + n^- \cdot v^-$ on Γ
$[[n \cdot v]] = n \cdot v$ on Γ_D
where v is vector

3.
$[[n \cdot \sigma]] = n^+ \cdot \sigma^+ + n^- \cdot \sigma^-$ on Γ
$[[n \cdot \sigma]] = n \cdot \sigma$ on Γ_D
where σ is Second order vector

3.3.2 Average operator

Similarly the average operator is defined as,

$$\{u\} = \frac{u^+ + u^-}{2} \tag{3.10}$$

3.4 Discontinuous Galerkin Method

In the context of discontinuous galerkin method we introduce function space $V(\tau_h)$ and $Q(\tau_h)$ for analytical solution of velocity and analytical solution of pressure respectively. The space containing high fidelity solution is called truth space denoted by V_h. The dimension of V_h is denoted as N_h. (The subscript h hereafter refers to truth space.)

$$V = \{\phi \in L^2(\Omega) | v|_k \in P^D(K) \forall K \in \tau_h\} \tag{3.11}$$

$$Q = \{\psi \in L^2(\Omega) | q|_k \in P^{D-1}(K) \forall K \in \tau_h\} \tag{3.12}$$

Here, P^D denotes space of polynomials of degree at most D over Ω.

We apply similar procedure as in Finite element method i.e. multiplying the partial differential equation by test function and intergration by parts. However, we note that our test function is not continuous on the interface. Hence, we require flux approximations and jumps at the interface. These requirements have given rise to different discontinuous Galerkin methods. For explanation of each method we refer to literatures such as [4] for local discontinuous galerkin and [2] for Compact discontinuous Galerkin and Interior penalty method.

Discontinuous Galerkin methods for Navier Stokes equation were compared by [2]. The local discontinuous Galerkin(LDG) method extends the computational stencil beyond immediate neighbours whereas compact discontinuous Galerkin(CDG) and interior penalty method(IPM) only connect to neighbouring elements. The CDG method provides more flexibility with respect to stabilisation constant at the cost of additional simulation effort related to computation of lifting operator, while the IPM method requires restrictions on penalty parameter in order to maintain coercivity of bilinear form. Both methods, CDG and IPM, have almost similar convergence rates.

3.4.1 Nodal basis function and Orthonormal basis function

The "Basis function" are also known as "Interpolation function" or "Ansatz function". There are two kinds of basis function which are used in the application of Finite element or Discontinuous Galerkin Method. Before, explaining the diferrence between two types of basis functions, we understand the procedure to define the function completely.

In order to define a polynomial of given degree completely, we need to calculate its co-efficients. A polynomial of degree n in 1-dimensional domain has $n+1$ coefficients. In case of 2-dimensional domain the number of co-efficients become $(n+1)(n+2)/2$. Two define these co-efficients the known values of function at number of points equal to the number of co-efficients is required. These points are known as nodes. In case of triangular element, the nodes located as,

1. For polynomial of degree 1 i.e. $p = 1$, the nodes are selected as vertices of element.
2. For polynomial of degree 2 i.e. $p = 2$, the nodes are selected as vertices of element and mid point of edges.

Similarly for higher order polynomial the nodes are selected as shown in 3.4.

3.4.2 Nodal Basis Function

Nodal basis function is also known as "Shape function". This basis function has value of 1 at its respetive node and 0 at other nodes. At all other points it is interpolated based on the degree of the basis function.

3.4.3 Orthonormal Basis Function

Orthonormal basis functions are the basis functions defined in such a way that all basis functions are orthonormal to each other. The number of orthonormal basis functions for a given element is same as number of Nodal basis functions. In fact, in present analysis the orthonormal basis functions are derived from Nodal basis functions.

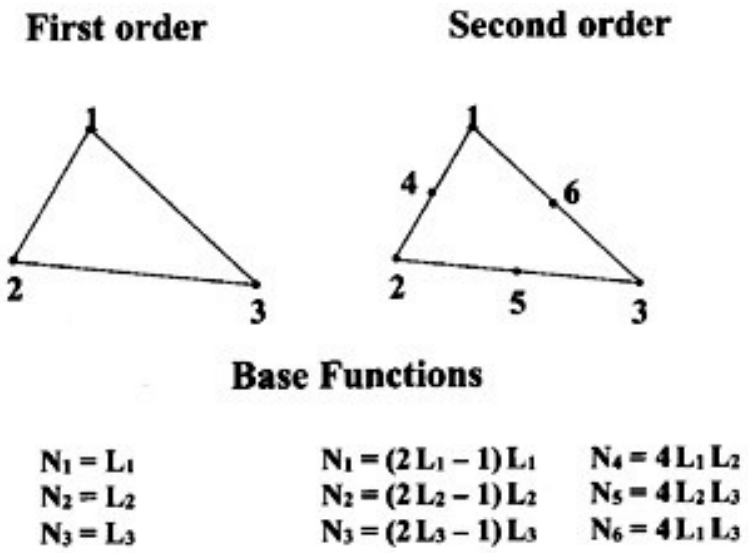


Figure 3.3: Finite Element nodes on triangle for $p = 1$ and $p = 2$ [3]

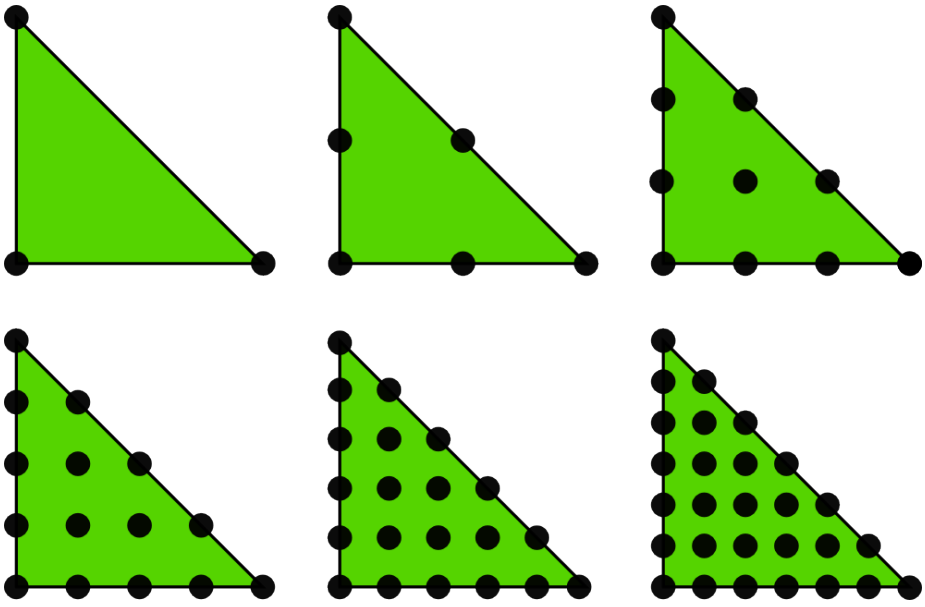


Figure 3.4: Finite Element nodes on triangle for polynomials of different degrees http://hplgit.github.io/INF5620/doc/pub/sphinx-fem/._main_fem009.html

3.5 Weak and discrete form of Navier Stokes equation

Following the approach presented by [2] and [1] we arrive at weak form of Navier Stokes interior penalty approximation as,

$$a_{IP}(u,\phi) + c(u:u,\phi) + b(\phi,p) + (p,[n\cdot\phi])_{\Gamma\cup\Gamma_D} = l_{IP}(\phi) \tag{3.13}$$

$$a_{IP} = (2\nu\nabla^s u, \nabla^s\phi) + C_{11}([n\otimes u],[n\otimes u])_{\Gamma\cup\Gamma_D} - (2\nu\nabla^s u,[n\otimes\phi])_{\Gamma\cup\Gamma_D} - (2\nu[n\otimes u],\nabla^s\phi)_{\Gamma\cup\Gamma_D} \tag{3.14}$$

$$l_{IP}(\phi) = (f,\phi) + (t,\phi)_{\Gamma_N} + C_{11}(u_D,\phi)_{\Gamma_D} - (n\otimes u_D, 2\nu\nabla^s\phi)_{\Gamma_D} \tag{3.15}$$

$$b(\phi,p) = -\int_\Omega \psi\nabla\cdot\phi \tag{3.16}$$

Here, $c(u:u,\phi)$ is non linear term which we discuss later.

Following discretization of the continuous form we arrive at discrete form as below. We refer hereafter, throughout thesis, subscript h for function in discrete form.

$$AU + BP = F \tag{3.17}$$

where,
A and B = Stiffness matrix
U = Global velocity degree of freedom vector
P = Global pressure degree of freedom vector
F = Right hand side vector

Chapter 4

Implementation aspects

We discuss now the implementation of discrete formulation of Navier Stokes Discontinuous Galerkin weak formulation in RBMATLAB, A MATLAB library containing all our reduced simulation approaches for linear and nonlinear, affine or arbitrarily parameter dependent evolution problems with finite element, finite volume or local discontinuous Galerkin discretizations.

Before we discuss details of implementation it is important to understand some frequently used terminologies and the data type of Basis Function and derivative of basis function in RBMATLAB.

4.1 Terminologies

A. *params* and *paramsP* : *params* and *paramsP* are structures corressponding to velocity and pressure respective containing following fields

1. *dimrange* : We refer *dimrange* as dimension of range or solution. In present analysis this means Velocity has *dimrange* of 2 and Pressure has *dimrange* of 1.

2. *pdeg* : We refer *pdeg* as the degree of polynomial used as interpolation polynomial.

3. $ndofs\_per\_element$: We refer $ndofs\_per\_element$ as the number of degrees of freedom over an element.

B. *grid* : *grid* is the structure containing following fields

4. *nelements* : *nelements* is total number of elements in grid.

5. NBI : $NBI(i, j)$ represents element number of jth neighbour of element i. $1 \leq i \leq nelements$ and $1 \leq j \leq 3$.

6. NX and NY : $NX(i, j)$ and $NY(i, j)$ represent x component and y component of normal from element i to j numbered neighbour.

7. JIT : $[JIT(k,:,1)', JIT(k,:,2)']$ represents Jacobian Inverse Transpose of element k.

8. A : $A(k)$ represents area of kth element.

9. EL : $EL(i,j)$ represents length of edge between element i and jth neighbour of Element i. The notation EL is used when i refers to superscript $+$ and j refers to element with superscript $-$.

We also use some other local variables as below:
10. k : k represents the element number and $1 \leq k \leq nelements$.

11. ids : $ids(i,:)$ represents the indices of degree of freedom of element i where $1 \leq i \leq nelements$ in global degree of fredom vector. Size of $ids(i,:)$ is $ndofs\_per\_element$. $ids\_velocity$ corressponds to indices corressponding to velocity and $ids\_pressure$ corressponds to indices corressponding to pressure. Further notations such as $ids\_velocity\_neighbour$ should be read as ids of velocity degree of freedom of neighbouring element.

Basis Function ϕ_h in RBMATLAB :

The Basis functions in RBMATLAB is generated by routine called ldg_evaluate_basis.m.

We interpolate the solution from solution at known degrees of freedom. In matrix formulation this means basis function is matrix of the size,

$$\phi_h \epsilon \mathbb{R}^{ndofs\_per\_element X dimrange} \tag{4.1}$$

where each row of ϕ_h denotes
This representation creates many zeros in matrix, however, it does not require new evaluation for each component of vector quantity in dimrange of solution.
Due to the zeros in the basis function, zeros in derivative of basis functions are produced.
The derivative of basis function is generated by routine ldg_evaluate_basis_derivative. The derivative of basis function $(\phi_h)_i$, where $1 \leq i \leq nodfs\_per\_element$ is a cell containing matrix $\nabla(\phi_h)_i$ of size,

$$\nabla(\phi_h)_i \epsilon \mathbb{R}^{dimrange X 2} \tag{4.2}$$

where the colmuns of matrix corresspond to $\nabla_x(\phi_h)_i$ and $\nabla_y(\phi_h)_i$.

4.2 Assembly of average operator

Average of quantity, A_h in discrete form is assembled as,

$$A_h = (A_h^+ + A_h^-)/2 \tag{4.3}$$

The assembly of average operator is relatively simple as compared to jump operator which is explained in section 4.3

4.3 Jump operator

Jump of quantity, A_h in discrete form is assembled as,

$$[A_h] = A_h^+ n^+ + A_h^- n^- \tag{4.4}$$

In case of terms such as $[A_h], [B_h]$ we assemble matrices as,
For internal edges Γ,

$$[A_h], [B_h] = A_h^+ n^+ B_h^+ n^+ + A_h^+ n^+ B_h^- n^- + A_h^- n^- B_h^+ n^+ + A_h^- n^- B_h^- n^- \tag{4.5}$$

and for dirichlet edges Γ_D

$$[A_h], [B_h] = A_h^+ n^+ B_h^+ n^+ \tag{4.6}$$

4.3.1 Matrix assemblies

We repeat here, for conveniece, again notations of L^2 scalar product from weak form

Here, (p, q) refers to,
If p and q are scalars,

$$(p, q) = \int_\Omega pq d\Omega \tag{4.7}$$

If p and q are vectors,

$$(p, q) = \int_\Omega p \cdot q d\Omega \tag{4.8}$$

If p and q are tensors,

$$(p, q) = \int_\Omega p : q d\Omega \tag{4.9}$$

i.e. (.,.) denotes L_2 inner product.

The matrices from weak form of Navier Stokes equation have been assembled in 3 steps,

1. Evaluating function at vertex of local element and transform to global geometry
2. Performing integral of function over local element and transform to global geometry(if not done in step 1)
3. Performing a loop over all elements and allocate integral at position in global matrix(for bilinear terms)/global vector(for linear terms) according to index of element degree of freedom in global degree of freedom vector.

We also perform numerical integration over domain Ω as,

$$\int_\Omega f(x) = \Sigma_{i=1}^{nop} f(x_i) * w_i \tag{4.10}$$

Where,

x_i = Location of function evaluation w_i = Weight at corressponding point nop = Number of points

The location of function evaluation, number of points and weights are based on Gaussian quadrature rule.

Also the determinant of jacobian is twice the area of triangle.

$$J(k) = 2 * A(k) \tag{4.11}$$

With this preliminary informations we discuss now the assembly of matrices.
1. $(\nabla\phi, \nabla\phi)$:

Step 1: Evalauation of $(\nabla\phi, \nabla\phi)$,
We first evaluate derivative of $\hat{\phi}$ and $JIT(k,:,:)$ through ldg_evaluate_basis_derivative and grid structure. We also perform elementary operation so as to rceiv one global basis function in each row. The matrix transformation from local derivative to global derivative is based on the formula 3.7.
We than assemble matrix $res_1[i,j] = \phi_i * \phi_j^T$ for $1 \leq i, j \leq ndofs\_per\_element$

Step 2: Performing integration in global co-ordinate system,
We perform the numerical integration as per 4.10
$res_2 = \int_{\hat{\Omega}} res_1 * 2 * grid.A(k) d\hat{\Omega}$

Step 3: Looping over each element and performing following operation in each loop $res_3[ids\_velocity, ids\_velocity] = res_2$

2. $([n \otimes \phi_h], [n \otimes \phi_h])_{\Gamma \cup \Gamma_D}$:
Step 1: On local element following functions are evaluated,

$res_1^{++} = [n \otimes \hat{\phi_h}]^+ [n \otimes \hat{\phi_h}]^+$
$res_1^{+-} = [n \otimes \hat{\phi_h}]^+ [n \otimes \hat{\phi_h}]^-$
$res_1^{-+} = [n \otimes \hat{\phi_h}]^- [n \otimes \hat{\phi_h}]^+$
$res_1^{--} = [n \otimes \hat{\phi_h}]^- [n \otimes \hat{\phi_h}]^-$

Please note that $\hat{\phi_h}$ is evaluated at local coordinate corressponding to global coordinate.

Step 2: In step 2 we perform following integration, We perform the numerical integration as per 3.6
$res_2^{++} = \int_\Gamma res_1^{++} * grid.EL$
$res_2^{+-} = \int_\Gamma res_1^{+-} * grid.EL$
$res_2^{-+} = \int_\Gamma res_1^{-+} * grid.EL$
$res_2^{--} = \int_\Gamma res_1^{--} * grid.EL$

Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,
$res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] = res_2^{++}$
$res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] = res_2^{+-}$
$res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] = res_2^{-+}$
$res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] = res_2^{--}$

Finally,
$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$

It is to be noted that on dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.

3. $(\nabla u_h, [n \otimes \phi_h])_{\Gamma \cup \Gamma_D}$:

Step 1: On local element following functions are evaluated,

$$
\begin{aligned}
res_1^{++} &= \nabla {u_h}^{+} [n \otimes \hat{\phi_h}]^{+} \\
res_1^{+-} &= \nabla {u_h}^{+} [n \otimes \hat{\phi_h}]^{-} \\
res_1^{-+} &= \nabla {u_h}^{-} [n \otimes \hat{\phi_h}]^{+} \\
res_1^{--} &= \nabla {u_h}^{-} [n \otimes \hat{\phi_h}]^{-}
\end{aligned}
$$

Please note that $\hat{\phi_h}$ is evaluated at local coordinate corresponding to global coordinate in accordance with 3.8.

Step 2: In step 2 we perform following integration, We perform the numerical integration as per 3.6

$$
\begin{aligned}
res_2^{++} &= \int_\Gamma res_1^{++} * grid.EL \\
res_2^{+-} &= \int_\Gamma res_1^{+-} * grid.EL \\
res_2^{-+} &= \int_\Gamma res_1^{-+} * grid.EL \\
res_2^{--} &= \int_\Gamma res_1^{--} * grid.EL
\end{aligned}
$$

Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,

$$
\begin{aligned}
res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] &= res_2^{++} \\
res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] &= res_2^{+-} \\
res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] &= res_2^{-+} \\
res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] &= res_2^{--}
\end{aligned}
$$

Finally,

$$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$$

It is to be noted that on dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.

The same routine is also used in case of $([n \otimes \phi], \nabla u)_{\Gamma \cup \Gamma_D}$

4. $(\psi_h, [n \cdot \phi_h])_{\Gamma \cup \Gamma_D}$:

Step 1: On local element following functions are evaluated,

$$
\begin{aligned}
res_1^{++} &= {\psi_h}^{+} [n \cdot \hat{\phi_h}]^{+} \\
res_1^{+-} &= {\psi_h}^{+} [n \cdot \hat{\phi_h}]^{-} \\
res_1^{-+} &= {\psi_h}^{-} [n \cdot \hat{\phi_h}]^{+} \\
res_1^{--} &= {\psi_h}^{-} [n \cdot \hat{\phi_h}]^{-}
\end{aligned}
$$

Please note that $\hat{\phi}$ is evaluated at local coordinate corresponding to global coordinate in accordance with 3.8.

Step 2: In step 2 we perform following integration, We perform the numerical integration as per 3.6

$$
\begin{aligned}
res_2^{++} &= \int_\Gamma res_1^{++} * grid.EL \\
res_2^{+-} &= \int_\Gamma res_1^{+-} * grid.EL \\
res_2^{-+} &= \int_\Gamma res_1^{-+} * grid.EL \\
res_2^{--} &= \int_\Gamma res_1^{--} * grid.EL
\end{aligned}
$$

Step 3: Loop over all elements, define the global assembly matrix as zero matrix and perform following operation in each loop,
$res_3^{++}[ids\_velocity\_self, ids\_velocity\_self] = res_2^{++}$
$res_3^{+-}[ids\_velocity\_self, ids\_velocity\_neighbour] = res_2^{+-}$
$res_3^{-+}[ids\_velocity\_neighbour, ids\_velocity\_self] = res_2^{-+}$
$res_3^{--}[ids\_velocity\_neighbour, ids\_velocity\_neighbour] = res_2^{--}$

Finally,
$res_3 = res_3^{++} + res_3^{+-} + res_3^{-+} + res_3^{--}$

It is to be noted that on dirichlet boundary only $res_1^{++}, res_2^{++}, res_3^{++}$ is evaluated as all other terms are zero.

5. $-\int_\Omega \psi \nabla \cdot \phi$ We note that, In accordance with 3.7

$$\nabla\phi = JIT\hat{\nabla}\hat{\phi} \tag{4.12}$$

and in accordance with 3.8

$$\psi(x) = \hat{\psi}(\hat{x}) \tag{4.13}$$

Step 1 : We first evaluate JIT, $\hat{\nabla}\hat{\phi}$ and ψ and assemble following local matrix
$res_1 = \hat{\psi}_i \hat{\nabla} \cdot \hat{\phi}_j$

Step 2 : We integrate the function over domain
$res_2 = -\int_{\hat{\Omega}} res_1 * 2 * A(k)$

Step 3: Assemble the global matrix
$res_3[ids\_presure, ids\_velocity] = res_2$

We use the same routine for $-\int_\Omega \nabla \cdot \phi_i \psi_j$

6. $(t, \phi)_{\Gamma_N}$:

Step 1: On local element following function is evaluated $res_1 = \hat{\phi}_i \cdot t$ in accordance with 3.8
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_N} res_1 EL$
Step 3: Loop over element having Neumann boundary and perform following operation in each loop $res_3[ids] = res_2$

7. $(u_D, \phi)_{\Gamma_D}$:

Step 1: On local element following function is evaluated $res_1 = \hat{\phi}_i * u_D$ in accordance with 3.8
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 EL$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$

8. $(\psi, n \cdot u_D)_{\Gamma_D}$:

Step 1: On local element following function is evaluated $res_1 = \hat{\psi} n \cdot u_D$ in accordance with 3.8

Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 EL$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$

9. (f, ϕ):

Step 1: On local element following function is evaluated $res_1 = \hat{\phi} f$ in accordance with 3.8
Step 2: An integral is performed over an element $res_2 = \int_{\hat{\Omega}} res_1 2 * A(k)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$

10. $(n \otimes u_D, \nabla\phi)_{\Gamma_D}$:

Step 1: On local element following function is evaluated $res_1 = n \otimes u_D \nabla\phi$ in accordance with 3.8
Step 2: An integral is performed over an element $res_2 = \int_{\Gamma_D} res_1 2 * A(k)$
Step 3: Loop over element having Dirichlet boundary and perform following operation in each loop $res_3[ids] = res_2$

Bibliography

[1] A. Montlaur, S. Fernandez-Mendez, and A. Huerta. Discontinuous galerkin methods for the stokes equations using divergence-free approximations. *International Journal for Numerical Methods in Fluids*, 57(9):1071–1092, 2008.

[2] A. Montlaur, S. Fernandez-Mendez, J. Peraire, and A. Huerta. Discontinuous galerkin methods for the navierstokes equations using solenoidal approximations. *International Journal for Numerical Methods in Fluids*, 64(5):549–564, 2010.

[3] A. C. J. Paes, N. M. Abe, V. A. Serrão, and A. Passaro. Simulations of Plasmas with Electrostatic PIC Models Using the Finite Element Method. *Brazilian Journal of Physics*, 33:411–417, June 2003.

[4] P.-O. Persson, J. Bonet, and J. Peraire. Discontinuous galerkin solution of the navierstokes equations on deformable domains. *Computer Methods in Applied Mechanics and Engineering*, 198(17):1585 – 1595, 2009.