Virtual Element Methods for HPC Applications

Industrial Training

Submitted in partial fulfillment of the requirements of

Masters of Science in
Computational Mechanics

by

Sanath Keshav

Under the supervision of

Dr.Mariano Vazquez
Dr.Guillaume Houzeaux

Computer Applications in Science and Engineering
Barcelona Supercomputing Center BSC - CNS

universitat politècnica de catalunya

Barcelona

August 2018

Acknowledgements

I would like to sincerely thank Dr. Mariano Vazquez, High Performance Computational Mechanics Group Manager, Computer Applications in Science and Engineering CASE, Barcelona Supercomputing Center BSC for believing in me and providing this opportunity to work on this Internship with him. A special thanks goes out to Dr. Guillaume Houzeaux , who provided me with direction, technical support, many encouraging conversations and fruitful discussions that helped me overcome the faltering steps encountered during the course of the internship. The guidance of Guillaume and Mariano, whose multidimensional and seemingly infinite expertise seems to fall short of time in this world to express itself, and whose enthusiasm and ability are unparalleled - have been instrumental to my learning at BSC. I greatly appreciate their patience, support and motivation throughout this period. Their endless guidance kept me pushing in the intent of broadening my limited knowledge and understanding of this field.

I owe a debt of gratitude also to the institute itself - a quiet dignified glass building in the corner of Barcelona - and everyone who is a part of it for inspiring me and providing such a conducive research oriented learning atmosphere which motivated me everyday.

And finally, to Barcelona, where the heart is.

Sanath Keshav

Abstract

Virtual Element Method (VEM) is a recent numerical technique which is a generalisation of the Finite Element Method on polygons and polyhedrons. Among other things, general shaped elements allow the very efficient meshing of complicated domains, such as those arising many engineering problems such as in poromechanics, fluid-structure interaction, crack propagation and wave scattering to name a few. While the conventional finite element method still enjoys its status as the method of choice for engineers, the handling of complicated geometries remains one of its difficulties for such problems. In response to this, several methods have been recently introduced which allow for very general polygonal elements, including the polygonal finite element method (PFEM), discontinuous Galerkin method and the Virtual element method. The difference between the various Finite Element Methods and the Virtual Element Method is that VEM admits non-polynomial basis functions which are not required to be computed in practice. Instead, the degrees of freedom are chosen so that the stiffness matrix is computed exactly without explicitly knowing the basis functions. This enables us to work with more general meshes than that of the Finite Element Method.

The project focuses mainly on the theory and implementation of the Virtual Element Method, to a variety of problems. Preliminary apriori error analysis was studied for the poisson problem. A hybrid combination of Finite element method and Virtual element method was implemented in order to test compatibility when there exists traditional finite elements and non traditional virtual elements in the same mesh to reduce computational costs in large scale problems. A toolbox was developed in matlab to generate efficient and sufficiently high order gauss quadrature rules for numerical integration on arbitrary polygons with least number of gauss points. A number of problems were solved in 2D including the fourth order nonlinear Rosenau equation with a mixed approach. A 3D virtual element solver was implemented for generalized polyhedrons with any number of faces and any number of sides on any face to solve the poisson problem and an order of convergence analysis was performed. Alternative polyhedral mesh generation methods were explored because of the absence of robust commercial polyhedral mesh generators by finding the dual of a simply connected graph (Tetrahedral mesh) using METIS (METIS_PartMeshDual) and openFOAM (polyDualMesh). XML ascii vtk paraview format was used for visualization of general polyhedrons. Tools like MATLAB, Fortran90 are used in conducting all the numerical experiments on the Virtual Element Methods.

Contents

Chapter 1

Basic Principles of Virtual Element method

In this report we introduce Virtual element method for some simple elliptic and parabolic problems and study the basic properties of the method. The novelty of the virtual element method is to take the space and degrees of freedom in such a way that the elementary stiffness matrices can be computed without actually computing these non-polynomial functions, but just using the degrees of freedom. In doing so, we can easily deal with complicated element geometries and higher order continuity requirements.

The virtual element method can be viewed as an extension of Finite element methods to general polygonal and polyhedral elements. The strongest aspects in favour of the virtual element method are its firm mathematical foundations, simplicity in implementation, and efficiency and accuracy in computations. In particular, the virtual element method permits the analysis to be performed without using any numerical quadrature formulas. It also admits the decomposition of the domain into non overlapping elements that can be of very general shape(convex or non convex polygons).

1.1 Two Dimensional Elliptic Problem

Consider the model problem of the poisson equation in two dimensions:

$$\begin{aligned} -\Delta u &= f \qquad in\ \Omega \\ u &= 0 \qquad on\ \Gamma \end{aligned}$$

where Ω is a polygonal domain in $\mathbb{R}^2$ and $f \in L^2(\Omega)$.

1.1.1 Continuous Problem

The weak formulation is given by

$$find\ u \in V := H_0^1(\Omega)\ \ such\ that$$
$$a(u,v) = \int_\Omega \nabla u.\nabla v\ d\Omega = \int_\Omega fv\ d\Omega = (f,v)$$

The weak form follows these properties.

- $a(u,v)$ is symmetric *i.e* $a(u,v) = a(v,u)$
- $a(u,v)$ is continuous/bounded *i.e*

$$\begin{aligned} a(u,v) = \int_\Omega \nabla u.\nabla v &\leq \Big(\int_\Omega |\nabla u|^2)^{1/2}\Big)\Big(\int_\Omega |\nabla v|^2)^{1/2}\Big) \\ &\leq \gamma |u|_1 |v|_1 \end{aligned}$$

- $a(u,v)$ is coervive *i.e*

$$a(v,v) = \int_\Omega |\nabla v|^2 = |v|_1^2 \geq \alpha \|v\|_1^2$$

- (f,v) is bounded

$$(f,v) \leq c\|v\|_1$$

By lax Milligram theorem, it is well known that the problem has a unique solution.

1.1.2 Discrete Problem

Let τ_h be a decomposition of Ω into elements K, and ϵ_h be the set of edges e of τ_h. Let h be the maximum of all diameters of the elements of τ_h. The bilinear form $a(u,v)$ and norm can be written as

$$\begin{aligned} a(u,v) &= \sum_{K\in\tau_h} a^K(u,v) \quad \forall u,v \ \in V \\ |v|_1 &= \Big(\sum_{K\in\tau_h} |v|_{1,K}^2\Big)^{1/2} \quad \forall v \ \in V \end{aligned}$$

The broken sobolev space and the broken semi-norm can be defined as

$$\begin{aligned} H'(\tau_h) &= \{v \in L^2(\Omega) : v|_K \in H^1(K), \forall K \in \tau_h\} \\ |v|_{1,h} &= \Big(\sum_K \|\nabla v\|_{0,K}^2\Big)^{1/2} \end{aligned}$$

For each h, we assume

- $V_h \subset V = H_0^1(\Omega)$
- A symmetric bilinear form a_h such that $a_h(u,v) = \sum_K a_h^K(u,v) \ \ \forall u,v \in V_h$
- $f_h \in V_h'$ where V_h' is the dual space of V_h and contains all the bounded linear functions. This implies that f_h is bounded.

The discrete problem is

$$\begin{aligned} &find\ u_h \in V_h \ \ such\ that \\ a_h(u_h, v_h) = &\ \langle f_h, v_h \rangle \ \ \forall v_h \in V_h \end{aligned}$$

such that good approximation properties also hold *i.e*

$$\|u - u_h\|_r \leq c h^{min(k+1,s)-r} |u|_s \tag{1.1}$$

We assume certain properties on $a_h(u_h, v_h)$ for the convergence.

- k - consistency : For all $p \in \mathbf{P}_k(K)$ and for all $v_h \in V_{h|K}$, we have

$$a_h^K(p, v_h) = a^K(p, v_h) \tag{1.2}$$

- Stability: There exists constants α^* and α_* such that $\forall v_h \in V_{h|K}$

$$\alpha_* a^K(v_h, v_h) \leq a_h^K(v_h, v_h) \leq \alpha^* a^K(v_h, v_h) \tag{1.3}$$

Proposition 1.1.1 *Continuity of the discrete bilinear form.*

$$a_h^K(u_h, v_h) \leq |u_h|_{1,K} |v_h|_{1,K} \quad \forall u_h, v_h \in V_{h|K} \tag{1.4}$$

Proof: Let $u, v \in V_{h|K}$. Now for any $\lambda \in \mathbf{R}$ we have

$$\begin{aligned} a_h^K(u - \lambda v, u - \lambda v) &= a_h^K(u,u) - 2\lambda a_h^K(u,v) + \lambda^2 a_h^K(v,v) \\ Set \quad \lambda &= \frac{a_h^K(u,v)}{a_h^K(v,v)} \\ &\Longrightarrow \quad a_h^K(u,u) - \frac{a_h^K(u,v)^2}{a_h^K(v,v)} \\ 0 \leq a_h^K(u - \lambda v, u - \lambda v) &= a_h^K(u,u) - \frac{a_h^K(u,v)^2}{a_h^K(v,v)} \\ \Longrightarrow \quad &\leq \sqrt{a_h^K(u,u)} \sqrt{a_h^K(v,v)} \\ &\leq \sqrt{\alpha^* a^K(u,u)} \sqrt{\alpha^* a^K(v,v)} \\ &= \alpha^* |u_h|_{1,K} |v_h|_{1,K} \\ \Longrightarrow \ a_h^K(u_h, v_h) &\leq |u_h|_{1,K} |v_h|_{1,K} \end{aligned}$$

Theorem 1.1.1 *Under the assumptions of symmetry, stability, k- consistency, continuity of $a_h(u_h, v_h)$ and boundedness of f_h, there exixts a unique $u_h \in V_h$ satisfying the discrete variational problem. Further, for the approximation u_I and u_π, there exists a constant c such that*

$$|u - u_h|_1 \leq c\big(|u - u_I|_1 + |u - u_\pi|_{1,h} + \delta_h\big) \tag{1.5}$$

where δ_h satisfies

$$(f, v) - \langle f_h, v_h \rangle \leq \delta_h |v_h|_1 \tag{1.6}$$

Proof: Let $\theta = u_h - u_I$

$$\begin{aligned}
\alpha_*|\theta|_1^2 = \alpha_* a(\theta,\theta) \leq a_h(\theta,\theta) &= \sum_K a_h^K(\theta,\theta) \\
&= \sum_K a_h^K(uj,\theta) - a_h^K(u_I,\theta) \\
&= \langle f_h,\theta\rangle - \sum_K a_h^K(u_I,\theta)
\end{aligned}$$

$$\begin{aligned}
\implies \quad & \langle f_h,\theta\rangle - \sum_K a_h^K(u_I,\theta) - a_h^K(u_\pi,\theta) + a_h^K(u_\pi,\theta) \\
= \quad & \langle f_h,\theta\rangle - \sum_K a_h^K(u_I-u_\pi,\theta) + a^K(u_\pi,\theta) \\
= \quad & \langle f_h,\theta\rangle - \sum_K a_h^K(u_I-u_\pi,\theta) + a^K(u_\pi,\theta) - a^K(u,\theta) + a^K(u,\theta) \\
= \quad & \langle f_h,\theta\rangle - (f,\theta) - \sum_K a_h^K(u_I-u_\pi,\theta) + a^K(u_\pi-u,\theta) \\
\leq \quad & \delta_h|\theta|_1 - \sum_K c\big(|u_I-u_\pi|_{1,K} + |u_\pi-u|_{1,K}\big)|\theta|_{1,K} \\
\alpha_*|\theta|_1^2 \quad \leq \quad & \big(\delta_h + \alpha^*|u_I-u_\pi|_{1,h} + \alpha^*|u_\pi-u|_{1,h}\big)|\theta|_1 \\
|\theta|_1 \quad \leq \quad & c\big(|u_I-u_\pi+u-u|_{1,h} + |u_\pi-u|_{1,h} + \delta_h\big) \\
|\theta|_1 \quad \leq \quad & c\big(|u_I-u|_1 + |u_\pi-u|_{1,h} + \delta_h\big) \\
|u-u_h|_1 \quad \leq \quad & c\big(|u_I-u|_1 + |u_\pi-u|_{1,h} + \delta_h\big)
\end{aligned}$$

1.1.3 The Virtual element space

Let a simple polygon K with n edges , Barycenter $\vec{x_K}$ and diameter h_K. Let us define for $k \geq 1$

$$B^k(\partial K) = \{v \in C^0(\partial K) : v|_e \in \mathbf{P}_k(e), \forall e \in \partial K\} \tag{1.7}$$

Now, the finite dimensional space $V_{K,k}$ is defined as

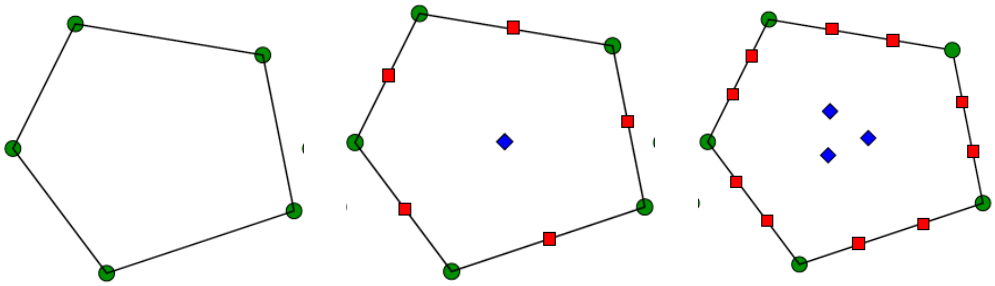
$$V_{K,k} = \{v \in H^1(K); v \in B^k(\partial K), \Delta v \in \mathbf{P}_{k-2}(K)\} \tag{1.8}$$

For $k = 1$

$$v|_e \in \mathbf{P}_1(e) \forall e \in \partial K$$

$dim(V_{K,k=1})$ = number of vertices of the element K.
For $k = 2$

$$\begin{aligned}
v|_e \in \mathbf{P}_2(e) \forall e \in \partial K \\
\Delta v \in \mathbf{P}_0 \\
i.e \;\; \Delta v = c \;\; in \; K
\end{aligned}$$



(a) k = 1 (b) k = 2 (c) k = 3

$dim(V_{K,k=2})$ = number of vertices of the element K + number of midpoints on edges of element K + one volume moment of element K.

For general k

$$v|_e \in \mathbf{P}_k(e) \forall e \in \partial K$$
$$\Delta v \in \mathbf{P}_{k-2}$$
$$dim(V_{K,k}) = N_v + (k-1)N_v + \frac{k(k-1)}{2}$$

where N_v is the number of vertices of the element K.
The degrees of freedom of $V_{K,k}$ are given by

- The value of v_h at the vertices of K
- On each edge e, the value of v_h at the $k-1$ internal points of the $(k+1)$ -point Gauss-Lobatto quadrature rule on e
- The moments up to order $k-2$ of v_h in K

  $$\frac{1}{|K|}\int_K v_h m_\alpha, \quad \alpha = 1 \dots n_{k-2}$$

  where $n_{k-2} = dim(\mathbf{P}_{k-2}(K))$ and the scaled monomials m_α are defined as

  $$m_\alpha := \left(\frac{\vec{x} - \vec{x_k}}{h_k}\right)^\alpha$$

  where $|\alpha| = \alpha_1 + \alpha_2$.

Under the assumption that the domain Ω is partitioned into non-overlapping polygons, there exists a γ such that for all h_k, there exists a ball of radius $\geq \gamma h_k$ which can be inscribed in the polygon. For any interpolant $u_I \in V_{K,k}$ and $u_\pi \in \mathbf{P}_k$ the following error estimates can be proved as shown in [5].

$$\|u - u_\pi\|_K + h_k|u - u_\pi|_{1,K} \leq ch_k^s|u|_{s,k} \quad some\ 1 \leq s \leq k+1 \tag{1.9}$$
$$\|u - u_I\|_K + h_k|u - u_I|_{1,K} \leq ch_k^s|u|_{s,k} \quad some\ 1 \leq s \leq k+1 \tag{1.10}$$

The global virtual element space V_h can be constructed as

$$V_h = \{v \in V | v \in B^k(\partial K) \;\; and \;\; \Delta v|_K \in \mathbf{P}_{k-2}(K) \forall K \in \tau_h\} \tag{1.11}$$

$$dim(V_h) = N_V + N_E(k-1) + N_P \frac{k(k-1)}{2} \tag{1.12}$$

where N_V is the number of vertices, N_E is the number of edges and N_P is the number of elements in the domain Ω. The global degrees of freedom are the values of v_h at the vertices, the values at the midpoint of the edges and the volume moments for each element in the domain.

1.1.4 Construction of the Approximate Bilinear form

Consider the bilinear form

$$a^K(p,v) = \int_K \nabla p.\nabla v dK = -\int_K \Delta p v dK + \int_{\partial K} \frac{\partial p}{\partial n} v ds \quad p \in \mathbf{P}_k(K), v \in V_{K,k} \tag{1.13}$$

$$\Delta p \in \mathbf{P}_{k-2}(K)$$
$$\frac{\partial p}{\partial n} \in \mathbf{P}_{k-1}(e)$$

which can be computed from the degrees of freedom and from the gauss lobotto quadrature points on the edge respectively. Hence, the integral can be computed.
To compute the stiffness matrix, we must define a suitable projector $\Pi^\nabla : V_h \to \mathbf{P}_k$.
Let us define a projector $\Pi^\nabla_{K,k} : V_{K,k} \to \mathbf{P}_k(K)$ as a solution of

$$a^K(\Pi^\nabla_{K,k} v_h, p) = a^K(v, p \quad \forall p \in \mathbf{P}_k(K)) \tag{1.14}$$

$$P_0(\Pi^\nabla_{K,k} v_h) = P_0(v_h) \tag{1.15}$$

where

$$\begin{aligned} P_0(\varphi) &= \frac{1}{N_v} \sum_{i=1}^{N_v} \varphi(V_i) \quad for\ k = 1 \\ P_0(\varphi) &= \frac{\int_K \varphi dK}{|K|} \quad for\ k \geq 2 \end{aligned}$$

We can easily observe that $\Pi^\nabla_{K,k} p_k = p_k$ when $p_k \in \mathbf{P}_k(K)$. We know that, $a^K(u,v) = (\nabla u, \nabla v)_K$. Then

$$\begin{aligned} u &= \Pi^\nabla_{K,k} u + (I - \Pi^\nabla_{K,k}) u \\ \nabla u &= \nabla \Pi^\nabla_{K,k} u + \nabla (I - \Pi^\nabla_{K,k}) u \end{aligned}$$

$$\begin{aligned} (\nabla u, \nabla v)_K &= (\nabla \Pi^\nabla_{K,k} u, \nabla \Pi^\nabla_{K,k} v) + (\nabla (I - \Pi^\nabla_{K,k}) u, \nabla \Pi^\nabla_{K,k} v) \\ &+ (\nabla \Pi^\nabla_{K,k} u, \nabla (I - \Pi^\nabla_{K,k}) v) + (\nabla (I - \Pi^\nabla_{K,k}) u, \nabla (I - \Pi^\nabla_{K,k}) v) \end{aligned}$$

The second and third term in the above equation is equal to 0 from the defination of the $\Pi^{\nabla}_{K,k}$ operator. Hence,

$$(\nabla u, \nabla v)_K = (\nabla \Pi^{\nabla}_{K,k} u, \nabla \Pi^{\nabla}_{K,k} v) + (\nabla (I - \Pi^{\nabla}_{K,k}) u, \nabla (I - \Pi^{\nabla}_{K,k}) v) \tag{1.16}$$

The first term of the above equation is called the consistency term and the second term is called the stability term.

$$a^K_h = [consistency\ term] + [stability\ term]$$

The consitency term can be computed exacty while the stability term has to be approximated with a suitable symmetric positive definite bilinear form $S^K(u, v)$ such that the k-consitency and the stability assumptions of $a^k_h(u, v)$ is satisfied.
k - consistency :

$$a^K_h(p, v) = a^K(p, v) \quad \forall p \in \mathbf{P}_k(K) \tag{1.17}$$

Proof:

$$\begin{aligned}
a^K_h(p, v) &= a^k(\Pi^{\nabla}_{K,k} p, \Pi^{\nabla}_{K,k} v) + S^K(p, v) \\
S^K(p, v) &= 0 \qquad\qquad since\ \Pi^{\nabla}_{K,k} p = p \\
a^K_h(p, v) &= a^K(p, \Pi^{\nabla}_{K,k} v) \\
&= a^K(p, v) \qquad By\ the\ defination\ of\ \Pi^{\nabla}_{K,k}
\end{aligned}$$

Stability :

$$\alpha_* a^K(v_h, v_h) \leq a^K_h(v_h, v_h) \leq \alpha^* a^K(v_h, v_h) \tag{1.18}$$

Proof: Let then $S^K(u, v)$ be any positive definite symmetric bilinear form to be chosen to verify

$$c_0 a^K(v, v) \leq S^K(v, v) \leq c_1 a^K(v, v) \ \ \forall v \in V_{K,k}\ with\ \Pi^{\nabla}_{K,k} v = 0 \tag{1.19}$$

$$\begin{aligned}
a^K_h(u, v) &= a^K(\Pi^{\nabla}_{K,k} u, \Pi^{\nabla}_{K,k} v) + S^K(u - \Pi^{\nabla}_{K,k} u, v - \Pi^{\nabla}_{K,k} v) \quad \forall u, v \in V_{K,k} \\
a^K(u, v) &= a^K(\Pi^{\nabla}_{K,k} u, \Pi^{\nabla}_{K,k} v) + a^K(u - \Pi^{\nabla}_{K,k} u, v - \Pi^{\nabla}_{K,k} v) \quad \forall u, v \in V_{K,k} \\
a^K_h(v, v) &\leq a^K(\Pi^{\nabla}_{K,k} v, \Pi^{\nabla}_{K,k} v) + c_1 a^K(u - \Pi^{\nabla}_{K,k} v, v - \Pi^{\nabla}_{K,k} v) \\
&\leq max\{1, c_1\}(a^K(\Pi^{\nabla}_{K,k} v, \Pi^{\nabla}_{K,k} v) + a^K(u - \Pi^{\nabla}_{K,k} v, v - \Pi^{\nabla}_{K,k} v)) \\
&= \alpha^* a^k(v, v)
\end{aligned}$$

Similarly,

$$\begin{aligned}
a^K_h(v, v) &\geq min\{1, c_0\}(a^K(\Pi^{\nabla}_{K,k} v, \Pi^{\nabla}_{K,k} v) + a^K(u - \Pi^{\nabla}_{K,k} v, v - \Pi^{\nabla}_{K,k} v)) \\
&= \alpha_* a^K(v, v)
\end{aligned}$$

In order to compute the local stiffness matrix, we need to compute the stability and the consitency term numerically. The key aspect in this process is to estimate the projector Π^{∇} given by the orthogonality condition,

$$\left(\nabla p_k, \nabla\left(\Pi^{\nabla} v_h - v_h\right)\right)_{0,K} = 0 \quad \forall p_k \in \mathbf{P}_k(K) \tag{1.20}$$

It can be seen that the above condition defines $\Pi^{\nabla} v_h$ only upto a constant. Hence another projector P_0 is applied onto $\Pi^{\nabla} v_h$. $P_0 : V_k(K) \to \mathbf{P}_0(K)$

$$P_0(\Pi^{\nabla} v_h - v_h) = 0 \tag{1.21}$$

Since m_α is a basis for $\mathbf{P}_k(K)$, substituting m_α in (1.20) we get

$$\left(\nabla m_\alpha, \nabla(\Pi^{\nabla} v_h - v_h)\right)_{0,K} = 0 \quad \alpha = 1 \dots n_k \tag{1.22}$$

Since $\Pi^{\nabla} v_h$ belongs to $\mathbf{P}_k$, it can be written as

$$\Pi^{\nabla} v_h = \sum_{\beta=1}^{n_k} s^\beta m_\beta \tag{1.23}$$

Substituting (1.23) in (1.20) we get,

$$\sum_{\beta=1}^{n_k} s^\beta (\nabla m_\alpha, \nabla m_\beta)_{0,E} = (\nabla m_\alpha, \nabla v_h)_{0,E}, \quad \alpha = 1 \dots n_k \tag{1.24}$$

To eliminate the indeterminacy of the above equation

$$\sum_{\beta=1}^{n_k} s^\beta P_0 m_\beta = P_0 v_h \tag{1.25}$$

Together the equations (1.24) and (1.25) can be written in a compact form as

$$G\, s = b \tag{1.26}$$

For each basis function φ_i, we define s_i^α as the coefficients of $\Pi^{\nabla}\varphi_i$ in the basis of m_α

$$\Pi^{\nabla}\varphi_i = \sum_{\alpha=1}^{n_k} s_i^\alpha m_\alpha, \quad i = 1 \dots N^{dof} \tag{1.27}$$

The above equation in the compact form can be written as

$$s^{(i)} = G^{-1} b^{(i)} \tag{1.28}$$

In the matrix representation Π_*^{∇} of the operator Π^{∇} is given by $(\Pi_*^{\nabla})_{\alpha i} = s_i^\alpha$, that is ,

$$\Pi_*^{\nabla} = G^{-1} B \tag{1.29}$$

Let

$$\begin{aligned}
\Pi^{\nabla}\varphi_i &= \sum_{j=1}^{N^{dof}} \pi_i^j \varphi_j \quad i = 1 \dots N^{dof} && (1.30)\\
\pi_i^j &= dof_j(\Pi^{\nabla}\varphi_i) && (1.31)\\
\pi_i^j &= \sum_{\alpha=1}^{n_k} s_i^\alpha dof_j(m_\alpha) && (1.32)\\
D_{i\alpha} &:= dof_i(m_\alpha) && (1.33)\\
\pi_i^j &= \sum_{\alpha=1}^{n_k} (G^{-1}B)_{\alpha i} D_{j\alpha} = (DG^{-1}B)_{ji} && (1.34)\\
\implies \Pi^{\nabla} &= DG^{-1}B = D\Pi_*^{\nabla} && (1.35)
\end{aligned}$$

The Virtual element method local stiffness matric K_K^h for a polygon K can be written as

$$(K_K)_{ij} = (\nabla\Pi_{K,k}^{\nabla}\varphi_i, \nabla\Pi_{K,k}^{\nabla}\varphi_j) + (\nabla(I-\Pi_{K,k}^{\nabla})\varphi_i, \nabla(I-\Pi_{K,k}^{\nabla})\varphi_j) \tag{1.36}$$

$$(\nabla(I-\Pi_{K,k}^{\nabla})\varphi_i, \nabla(I-\Pi_{K,k}^{\nabla})\varphi_j)_{0,K} \approx \sum_{r=1}^{N^{dof}} dof_r((I-\Pi^{\nabla})\varphi_i) dof_r((I-\Pi^{\nabla})\varphi_j) \tag{1.37}$$

The matrix expression for the VEM local stiffness matix is given by

$$K_K^h = (\Pi_*^{\nabla})^T \tilde{G}(\Pi_*^{\nabla}) + (I-\Pi^{\nabla})^T(I-\Pi^{\nabla}) \tag{1.38}$$

1.1.5 Construction of the Load term

The load term is approximated by an L^2 projector. Let,

$$\langle f_h, v_h\rangle = \sum_K \int_K f_h v_h dK \tag{1.39}$$

$$Let \quad f_h = P_{k-2}^K f \quad where \;\; (P_{k-2}^K f, p_{k-2}) = (f, p_{k-2}) \; \forall p_{k-2} \in \mathbf{P}_{k-2} \tag{1.40}$$

$$\begin{aligned}
\implies \langle f_h, v_h\rangle = \sum_K \int_K P_{k-2}^K f v_h &= \sum_K \int_K P_{k-2}^K f P_{k-2}^K v_h \\
&= \sum_K \int_K f P_{k-2}^K v_h \\
\langle f_h, v_h\rangle - (f, v_h) &= \sum_K \int_K f_h v_h - \int_K f v_h \\
&= \sum_K \sum_K P_{k-2}^K f v_h - \int_K f v_h + \int_K P_{k-2}^K f P_0^K v_h - \int_K P_{k-2}^K f P_0^K v_h \\
&= \sum_K \int_K (P_{k-2}^K f - f)(v_h - P_0^K v_h) \\
&\leq \sum_K \|P_{k-2}^K f - f\|_{0,K} \|P_0^K v_h - v_h\|_{0,K} \\
&\leq \sum_K c h_K^k - 1 |f|_{k-1,K} c h_K |v_h|_{1,K} \\
&\leq c h_K^k |f|_{k-1,K} |v_h|_{1,K} \\
&\leq c h^k \Big(\sum_K |f|_{k-1,K} \Big) |v_h|_1 \\
&\leq \delta |v_h|_1
\end{aligned}$$

1.2 Two Dimensional Parabolic Problem

Consider the model problem of a parabolic heat equation in two dimensions given by

$$\begin{aligned} u_t - \Delta u &= f \qquad in\ \Omega \ \ for\ t \in (0,T) \\ u &= 0 \qquad on\ \Gamma = \partial\Omega \ \ for\ \ \in (0,T) \\ u(0) &= u_0 \qquad in\ \Omega \end{aligned}$$

where Ω is a polygonal domain in $\mathbb{R}^2$, $f \in L^2(\Omega \times (0,T))$ represents a source term and $u_0 \in L^2(\Omega)$ is the initial data. u represents the unknown variable of interest and u_t denotes its time derivative.

The weak formulation of the problem is given by

$$\begin{aligned} &find\ u \in L^2(0,T,H_0^1(\Omega))\ with\ u_t \in L^2(0,T,H^{-1}(\Omega)), such\ that \\ &(u_t(t),v) + a(u(t,v)) = \langle f(t),v\rangle \ \ \forall v \in H_0^1(\Omega),\ t\ in\ (0,T) \end{aligned}$$

The solution to the parabolic problem is given by

$$u(t) = \sum_{i=1}^{N^{dof}} \alpha_i(t)\varphi_i \tag{1.41}$$

Substituting (1.41) into the weak form we get

$$[M]\{\frac{d\alpha}{dt}\} + [K]\{\alpha\} = \{F\} \tag{1.42}$$

where

$$[K]_{ij} \quad = \quad (\nabla\varphi_i, \nabla\varphi_j) = (\nabla\Pi^{\nabla}_{K,k}\varphi_i, \nabla\Pi^{\nabla}_{K,k}\varphi_j) + (\nabla(I-\Pi^{\nabla}_{K,k})\varphi_i, \nabla(I-\Pi^{\nabla}_{K,k})\varphi_j)$$

$$[M]_{ij} \quad = \quad (\varphi_i, \varphi_j) = (\Pi^0_{K,k}\varphi_i, \Pi^0_{K,k}\varphi_j) + ((I-\Pi^0_{K,k})\varphi_i, (I-\Pi^0_{K,k})\varphi_j)$$

A backward euler scheme is used to discretize the temporal term, we get

$$[M]\{\frac{\alpha^{n+1}-\alpha^n}{\Delta t}\} + [K]\{\alpha^{n+1}\} = \Delta t\{F^{n+1}\} \tag{1.43}$$

$$\begin{aligned} [M]\{\alpha^{n+1}\} - [M]\{\alpha^n\} + \Delta t[K]\{\alpha^{n+1}\} &= \Delta t\{F^{n+1}\} \\ ([M] + \Delta t[K])\{\alpha^{n+1}\} &= [M]\{\alpha^n\} + \Delta t\{F^{n+1}\} \\ \{\alpha^{n+1}\} &= ([M] + \Delta t[K])^{-1}([M]\{\alpha^n\} + \Delta t\{F^{n+1}\}) \end{aligned} \tag{1.44}$$

The analysis of the parabolic problem, virtual element space construction follows similarly to the elliptic problem discussed earlier in this chapter.

1.2.1 Construction of the Mass Matrix

As seen in the construction of the load term, the L^2 projector is defined as

$$(p_k, \Pi^0 v_h - v_h)_{0,K} = 0 \quad p \in \mathbf{P}_k(K) \; and \; v_h \in V_k(K) \tag{1.45}$$

Proceeding as before, we define the matrix

$$H_{\alpha\beta} := (m_\alpha, m_\beta)_{0,K} \quad \alpha, \beta = 1 \dots n_k \tag{1.46}$$

$$\Pi^0 v_h = \sum_{\alpha=1}^{n_k} t^\alpha m_\alpha \tag{1.47}$$

Then, mimicking what we did for the operator Π^∇, we define the n_k vectors t and c with components respectively

$$c^\alpha := (m_\alpha, v_h)_{0,K} \tag{1.48}$$

Hence, we have

$$\begin{aligned} Ht &= c \\ t &= H^{-1}c \end{aligned}$$

Since (1.48) is not computable for the monomials m_α of degree k and $k-1$, we replace it by

$$c^\alpha := (m_\alpha, \Pi^\nabla v_h)_{0,K} \tag{1.49}$$

Now we have

$$[M_K]_{ij} = \int_K \varphi_i \varphi_j = \int_K \Pi^0 \varphi_i \Pi^0 \varphi_j + \int_K (I - \Pi^0)\varphi_i (I - \Pi^0)\varphi_j \tag{1.50}$$

As before, the first term ensures consistency and must be computed exactly, while the second one guarantees stability and can be spproximated as,

$$\int_K (I - \Pi^0)\varphi_i (I - \Pi^0)\varphi_j \approx |K| \sum_{r=1}^{N^{dof}} dof_r((I - \Pi^0)\varphi_i) dof_r((I - \Pi^0)\varphi_j) \tag{1.51}$$

The final formula for the local VEM matrix is given by

$$M_K^h = C^T H^{-1} C + |K|(I - \Pi^0)^T (I - \Pi^0) \tag{1.52}$$

Chapter 2

Numerical Experiments

2.1 2D Elliptic Problem - Poisson Equation

We consider the Poisson equation with homogeneous boundary condition

$$-\Delta u = f \quad \text{in } \Omega \tag{2.1}$$
$$u = 0 \quad \text{on } \Gamma \tag{2.2}$$

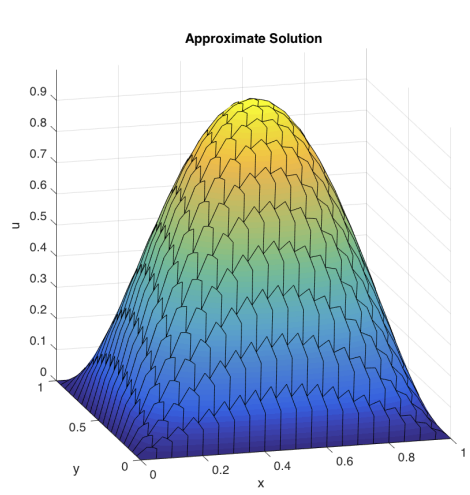
We take a unit square $\Omega = (0,1) \times (0,1)$ and $f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$. The exact solution of this equation is given by $u(x,y) = \sin(\pi x)\sin(\pi y)$. The solution of the Poisson equation is shown in Figure 2.2. The voronoi meshes are generated using PolyMesher[2]. The non-convex and the triangular meshes were a part of the code "The Virtual Element Method in 50 lines of MATLAB" by Oliver Sutton [7].
As we see that the solution is obtained on much general meshes. Optimal error estimates in the H^1 and L^2 norm are proved for the Poisson equation in [5], that is

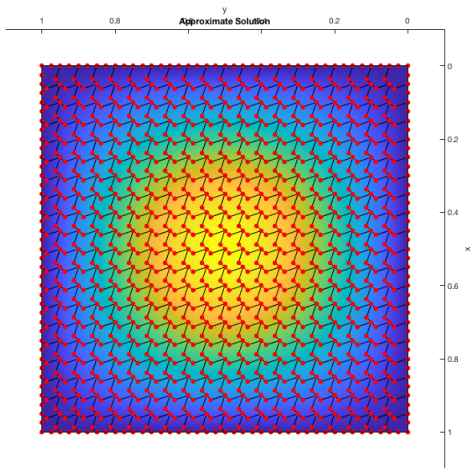
$$\|u - u^h\| \leq Ch^{k+1}|u|_{k+1} \tag{2.3}$$
$$\|u - u^h\|_1 \leq Ch^{k}|u|_{k+1} \tag{2.4}$$

To this end, we mesh the domain using squares and conduct numerical experiments for $k = 2$, although much general meshes can work as well. The results are tabulated below.

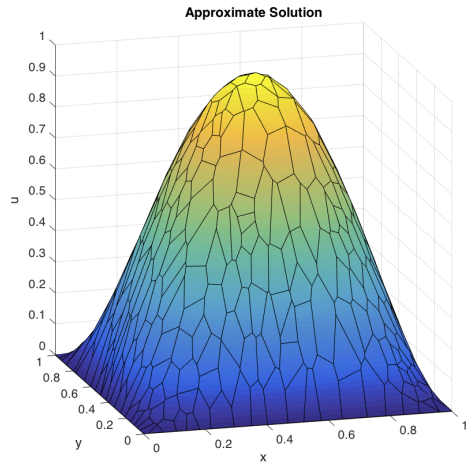
h	$\|u - u_h\|_{L^\infty}$	p
0.10000	0.0083340	-
0.05000	0.0027232	1.6137
0.02500	0.0007053	1.9484
0.01250	0.0002030	1.7966

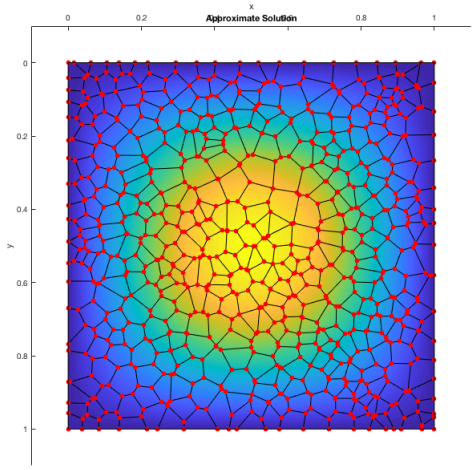
Table 2.1: Table illustrating the order of convergence(≈ 2) of VEM with $k = 1$ on smoothed voronoi meshes for poisson equation.



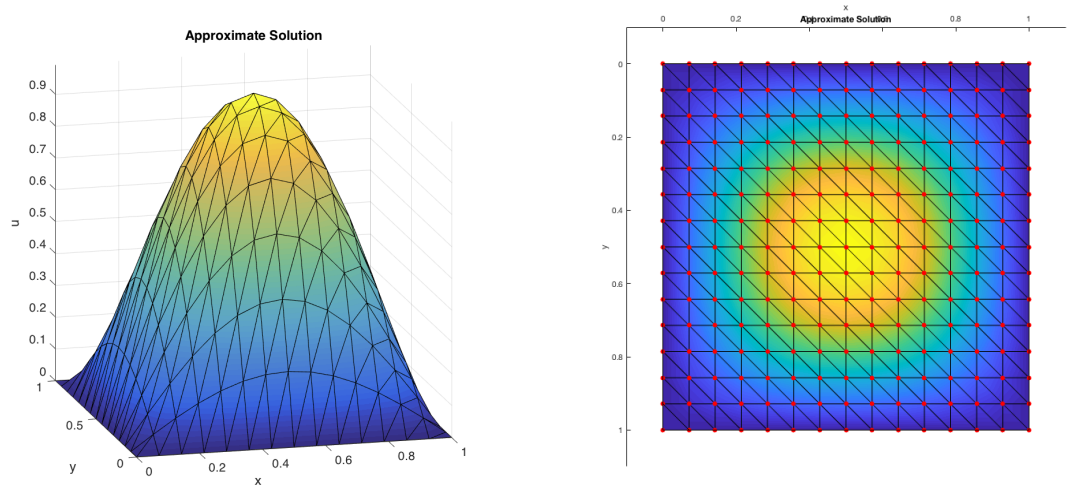


(a) Non-convex mesh





(b) Voronoi mesh



(a) Regular triangular mesh

Figure 2.2: Approximate solution of the Poisson equation on different meshes for $k = 1$

h	$\|u - u_h\|_{L^\infty}$	p
0.10000	0.0029135	-
0.06667	0.0005799	3.9809
0.05000	0.0001866	3.9405
0.04000	0.0000763	4.0053

Table 2.2: Table illustrating the order of convergence(≈ 4) of VEM with $k = 2$ for poisson equation.

Expected optimal order of 2 of the L infinity norm of the error was obtained by order of convergence analysis of the first order virtual element method of successfully refined smoothed voronoi polygonal meshes. Super convergent order of 4 was obtained of the L infinity norm of the error for the second order virtual element method of successfully refined square meshes.

2.2 2D Parabolic Problem - transient heat conduction

In this subsection, we quickly summarize the results of Virtual Element Method for Parabolic type PDE.

$$\begin{aligned} u_t - \Delta u &= f(x, y, t) \quad \text{in } \Omega \times (0, T] \\ u(x, 0) &= u_0(x) \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma \end{aligned}$$

A detailed analysis of the method including optimal error estimates have been proved in [3]. Here, we mesh the unit square domain with square elements. The source function $f(x, y, t)$ is chosen so

that the exact solution is given by $u(x,y,t) = e^{-t}\sin(\pi x)\sin(\pi y)$. A backward Euler method was employed to discretize the temporal direction. The results are summarized in table 2.3.

h	$\|u-u_h\|_{L^\infty}$	p
0.25	0.039097	-
0.125	0.005303	2.8822
0.0625	0.0006655	2.9943

Table 2.3: Order of convergence analysis for VEM with $k = 2$ for transient heat conduction equation. $\Delta t \approx h^3$ and $T = 1s$

2.3 2D Non-linear 4^{th} order problem

The Rosenau equation is an example of a nonlinear partial differential equation, which governs the dynamics of dense discrete systems and models wave propagation in nonlinear dispersive media.

We consider the two dimensional Rosenau equation,

$$\begin{aligned} u_t + \Delta^2 u_t &= \nabla.\vec{f}(u) \quad \text{in } (x,y,t) \in \Omega \times (0,T] \\ u(x,y,0) &= u_0(x,y) \\ u &= 0 \qquad \text{on } \Gamma \\ \Delta u &= 0 \qquad \text{on } \Gamma \end{aligned}$$

where $\Omega \subset \mathbb{R}^2$ and $\vec{f}(u) = (f_1(u), f_2(u))$ is a sufficiently smooth function from $\mathbb{R}$ to $\mathbb{R}^2$. Generally

$$f_j(u) = \sum_{i=1}^{N} \frac{c_i u^{p_i+1}}{p_i+1},$$

$c_i \in \mathbb{R}$ and p_i is a positive real number. We first split the problem into two second order equations as follows.
Setting $-\Delta u = v$, and using the boundary conditions, we obtain the problem :

$$\begin{aligned} u_t - \Delta v_t &= \nabla.\vec{f}(u) \quad \text{in } (x,y,t) \in \Omega \times (0,T] \\ -\Delta u &= v \\ u(x,y,0) &= u_0(x,y) \\ u &= 0 \qquad \text{on } \Gamma \\ v &= 0 \qquad \text{on } \Gamma \end{aligned}$$

Then we apply C^0 - piecewise elements for approximating both u and v. The weak formulation of the problem reads:

Find $\{u, v\} : (0,T] \to H_0^1(\Omega) \times H_0^1(\Omega)$ such that

$$(u_t, \chi) + a\,(v_t, \chi) = (\nabla.\vec{f}(u), \chi), \quad \text{for all } \chi \in H_0^1(\Omega), \tag{2.5}$$

$$a(u, \varphi) = (v, \varphi), \quad \text{for all } \varphi \in H_0^1(\Omega), \tag{2.6}$$

$$u(0) = u_0(x). \tag{2.7}$$

where $(.,.)$ denotes the standard L^2-innerproduct and $a(u,v) = (\nabla u, \nabla v)$.
We rewrite the convective term as

$$\int_\Omega \nabla.\vec{f}(u)\chi \, dx = \int_\Omega (\vec{\beta}(u).\nabla u)\, \chi \, dx,$$

where $\beta_i(u) = f_i'(u)$. Defining the local component of the convective term

$$b^K(u, \chi;\, w) = \int_K (\vec{\beta}(w).\nabla u)\chi \, dx.$$

We define an approximation b_h^K of the convective part b^K as

$$b_h^K(u, \chi;\, w) = \int_K \left(\vec{\beta}(\Pi^0_{K,k} w).\nabla(\Pi^\nabla_{K,k} u)\right) (\Pi^0_{K,k}\chi) \, dx$$

It can be shown that b_h^K is stable and k-consistent to an order which is more than what we require for optimality.
We consider the following Model Problem

$$\begin{aligned} u_t + \Delta^2 u_t &= \nabla.\vec{f}(u) + g(x,y,t) \quad \text{in } \Omega \times (0,T] \\ u(x,y,0) &= 0 \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned}$$

where $\vec{f}(u) = (u^2 - u, u^2 - u)$ and $g(x,t)$ is such that the exact solution is given by $u(x,y,t) = t\sin(\pi\, x)\sin(\pi\, y)$.

(Rosenau - u) (Rosenau - v)

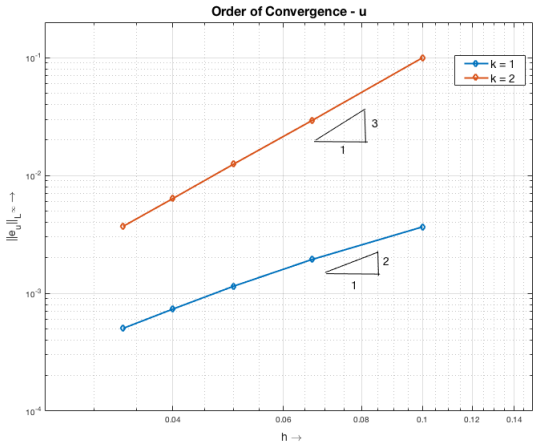
$k = 1$

N	h	$\|u - u^h\|_{L^\infty}$	$\|v - v^h\|_{L^\infty}$	ooc u	ooc v
10	0.10000	0.00366033	0.1455987	-	-
15	0.06667	0.00193649	0.0745702	1.5702	1.6502
20	0.05000	0.00114848	0.0440943	1.8160	1.8263
25	0.04000	0.00073454	0.0283913	2.0029	1.9729
30	0.03333	0.00050403	0.0197122	2.0656	2.0010

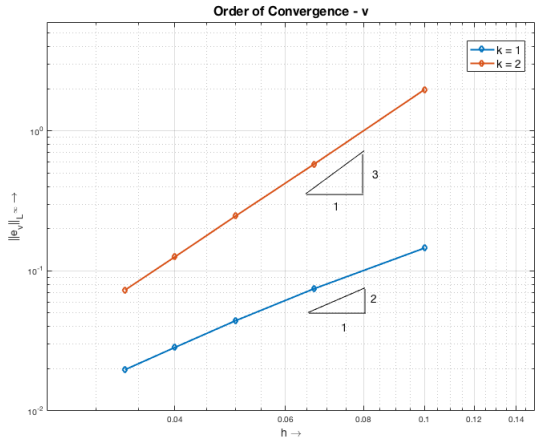
$k = 2$

N	h	$\|u - u^h\|_{L^\infty}$	$\|v - v^h\|_{L^\infty}$	oocu	ooc v
10	0.10000	0.09989844	1.9736070	-	-
15	0.06667	0.02929570	0.5784882	3.0254	3.0266
20	0.05000	0.01249712	0.2467182	2.9614	2.9621
25	0.04000	0.00637409	0.1258414	3.0171	3.0170
30	0.03333	0.00370326	0.0731038	2.9783	2.9790

Table 2.4: Order of convergence analysis for VEM for Rosenau equation on square meshes.



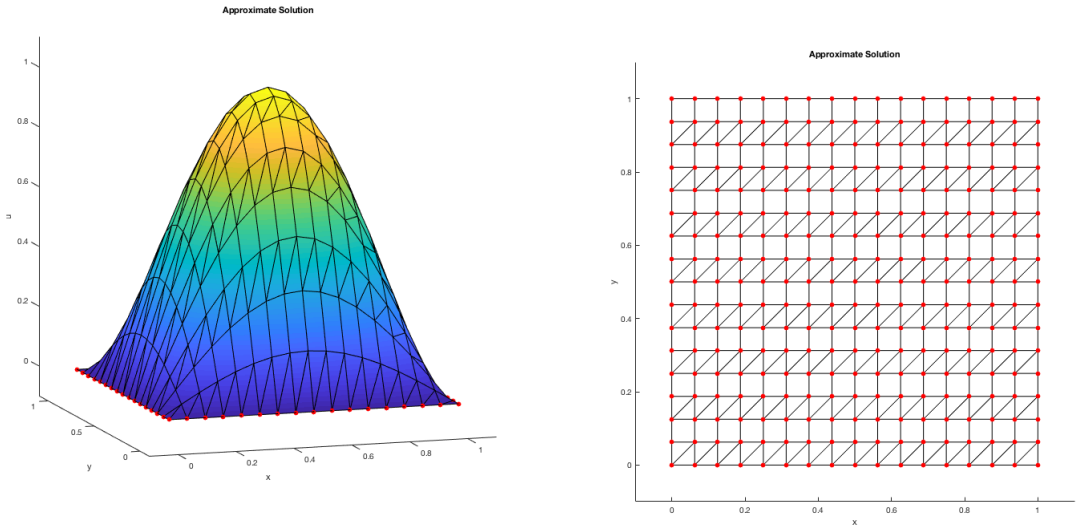
Order of convergence of u



Order of convergence of v

2.4 2D Hybrid Finite element - Virtual element approach

It is clear although the virtual element method has its clear advantages over the traditional finite element method, it is considerably more expensive to compute the element stiffness matrices in Virtual element method compared to finite element method. A hybrid combination of Finite element method and Virtual element method was implemented in order to test compatibility when there exists traditional finite elements and non traditional virtual elements in the same mesh to reduce computational costs in large scale problems. In simpler terms, problems were solved on meshes with



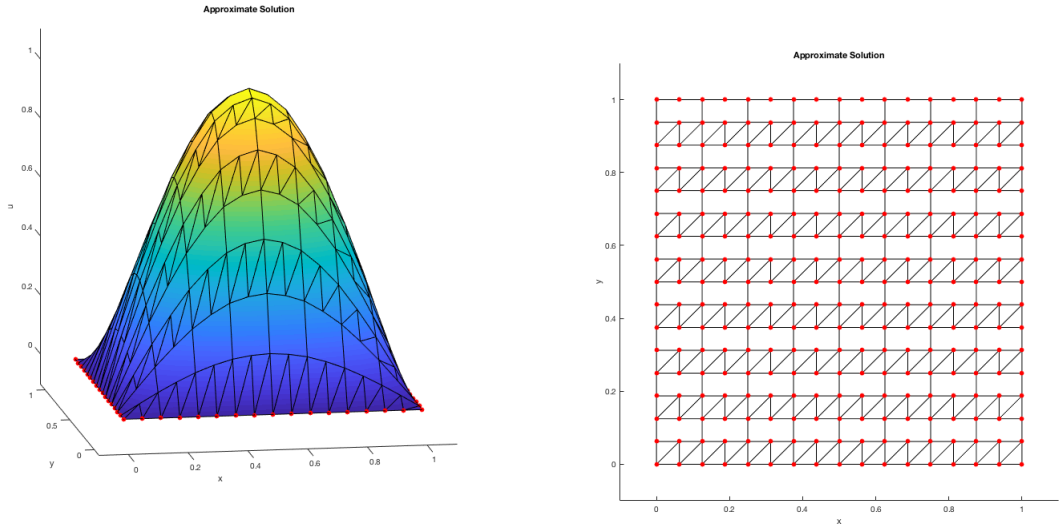
(a) Triangle - Square Hybrid

Figure 2.3: Approximate solution of the Poisson equation on Triangle and 4 noded element mesh using Hybrid approach for $k = 1$

different kinds of elements. The elemental matrices of simpler traditional finite element type elements were computed using the finite element method and the elemental matrices of more complex polygonal elements were calculated by virtual element method. This reduces the computational costs greatly in large scale problems.

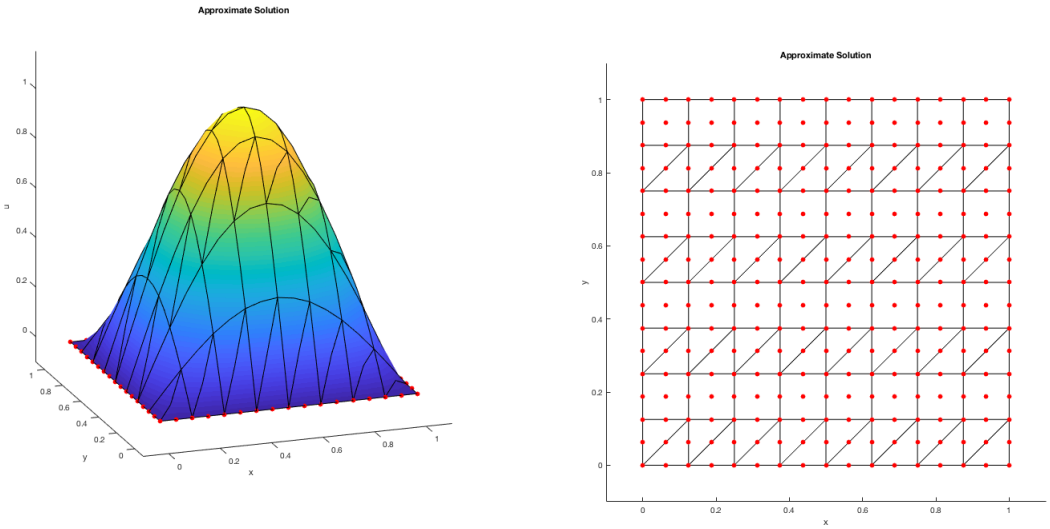
This is possible because of the way the virtual element space has been constructed. Since we are using conforming version of virtual element method and force elemental continuity over edges and faces and choosing the test function in such a way that they are polynomials on the edges allows us to seamlessly use finite element method and virtual element method in elements that are neighboring each other. The error estimates are also optimal because of individual optimal estimates for both FEM and VEM in corresponding elements.

In the figures, the elemental matrices of the triangular elements were computed using finite element formulation and square (4 noded) and rectangular (6 noded) elements were computed using virtual element formulation. Successful refinement and an order of convergence study for both the meshes were done and optimal error order was obtained. Note that this improvement allows us to handle hanging nodes which is impossible with traditional conforming finite element methods.



(a) Triangle - Rectangle Hybrid

Figure 2.4: Approximate solution of the Poisson equation on Triangle and 6 noded element mesh using Hybrid approach for $k = 1$



(a) Triangle - Square Hybrid

Figure 2.5: Approximate solution of the Poisson equation on different meshes using Hybrid approach for $k = 2$

2.5 2D PolyGauss - Numerical Quadrature on polygons

The most expensive part of computing the elemental matrices using the Virtual element formulation is the repeated numerical integration on arbitrary polygons. Numerical integration over arbiraty polygons can be done easily by decomposing the polygon into several triangles about its centroid or any interior point inside the polygon in case of integration over non-convex polygons. This method is accurate but inefficient because of the use of too many gauss points on each decomposed triangle of the polygon. Hence, it is useful to calculate quadrature points for polygons.

A quadrature is a formula of the form

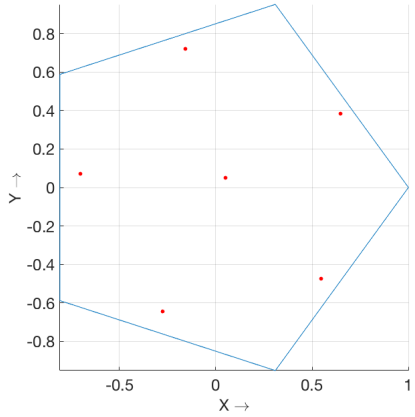
$$\int_{\Omega} w(x)f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where Ω is the designated integration region, f is an integrand defined on Ω and w is the weight function. w(x) = 1 is used for the quadratures designed here. The points x_i are called quadrature nodes, and w_i are the quadrature weights. Typically, quadratures are designed so that the above equation is exact for all functions in a pre-selected set. Classical choices of the pre-selected set of functions include polynomials up to a certain degree, trigonometric functions, and basis functions of a particular function space defined on Ω.

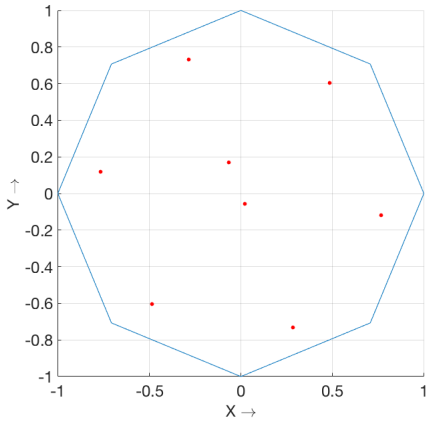
$$\begin{bmatrix} \int_{\Omega} w(x)\Phi_1(x)dx \\ \int_{\Omega} w(x)\Phi_2(x)dx \\ \vdots \\ \int_{\Omega} w(x)\Phi_m(x)dx \end{bmatrix} = \begin{bmatrix} \Phi_1(x_1) & \Phi_1(x_2) & \dots & \Phi_1(x_n) \\ \Phi_2(x_1) & \Phi_2(x_2) & \dots & \Phi_2(x_n) \\ \dots & \dots & \dots & \dots \\ \Phi_m(x_1) & \Phi_m(x_2) & \dots & \Phi_m(x_n) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

The above system is a nonlinear set of equations where the unknowns are x_i and w_i. Since the number of unknowns are more than the number of equations, we have infinitely many solutions. A random initial guess is assumed with a sufficiently high number of gauss points which ensures the determinant of the coefficient matrix is nonzero and the nonlinear system of equations is solved using newton raphson to obtain a set of quadratures and weights that can numerically integrate a function over Ω exactly. The process is continued by reducing the number of gauss points incrementally until the newton raphson algorithm does not converge to obtain the minimum number of gauss points and weights that are required to perform the designed numerical integration exactly. Additional care has to be taken to ensure the positivity of the weights and containment of the quadrature points inside the domain of integration.

Numerical examples are presented and the obtained quadrature rules are verified and validated by integrating a function over Ω using the computed quadrature rules and by decomposing the polygon into triangles and integrating the function over each triangle individually and summing it and obtained machine level accuracy using the computed quadrature rules compared to the other.



Computed 4th order accurate quadratures over Pentagon



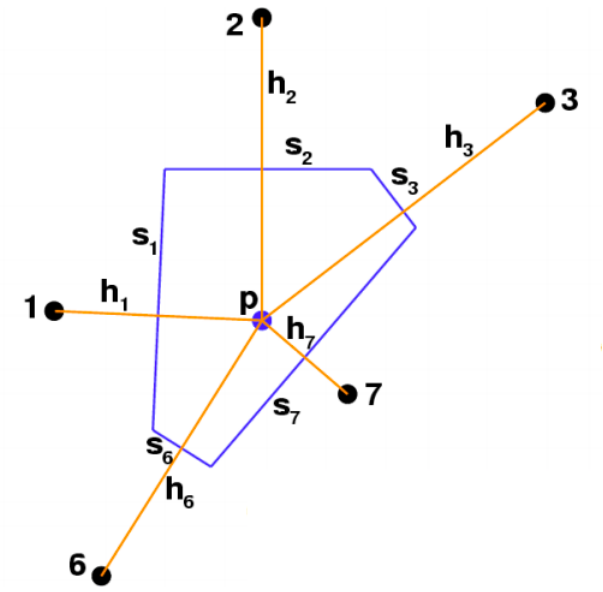
Computed 5th order accurate quadratures over Octagon

QX	QY	QW
-0.704116115086675	0.070445154019182	0.374359080509096
0.049414054264412	0.051464490426036	0.633600865904257
-0.157663165781741	0.720439873859606	0.328029687181632
0.646181681650431	0.383983744766850	0.307742387922124
0.545836566910750	-0.473729370418703	0.350315047021378
-0.276517677822754	-0.645262438215422	0.383594222199333

QX	QY	QW
0.021920717237543	-0.056527396417683	0.540722020857064
0.283065699617759	-0.729947237272195	0.341651510259588
0.767373098082583	-0.118853810715858	0.352727586032046
0.486616968464928	0.605137691770278	0.352727586012839
-0.767373098082227	0.118853810716746	0.352727586032825
-0.065780725464756	0.169630085760780	0.198879274074143
-0.283944824120177	0.732214253380688	0.336263975462505
-0.486616968464002	-0.605137691769517	0.352727586015152

The quadrature rules can be computed efficiently using the above algorithm but cannot be isogeometrically parameterized on polygons of the same number of sides because the shape functions on the polygon are not polynomials. The shape functions on arbitrary polygons can be given by laplace interpolants ϕ_i where

$$\begin{aligned} \phi_i(p) &= \frac{\alpha_i(p)}{\sum_j \alpha_j(p)} \\ \alpha_i(p) &= \frac{s_i(p)}{h_i(p)} \end{aligned}$$



Laplace interpolants

Because the shape functions are not polynomials but are asymptotic functions, the jacobians of the transformation is also an asymptotic function which cannot be represented as a linear combination of the preselected set basis of the quadrature. Hence it is not possible to compute the quadrature on the master polygon and transform it. Instead for very computationally intensive problems, it makes sense to compute the quadrature of every polygonal element in the domain beforehand as a pre-process and use the computed quadrature for every element during every numerical integration.

2.6 3D Elliptic Problem - Poisson equation

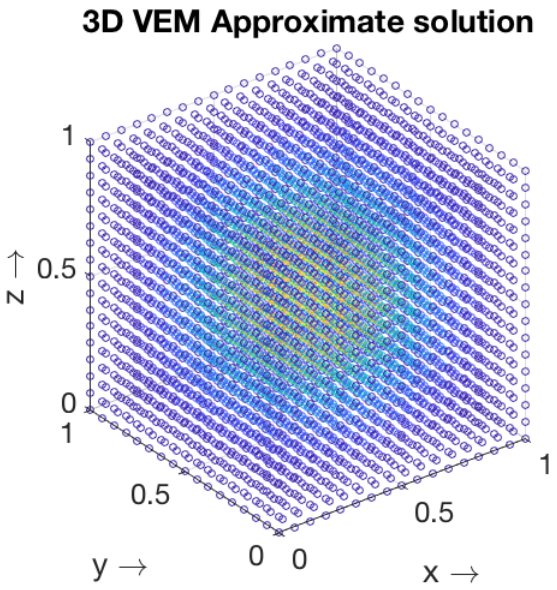
We go back to the model poisson problem where this time Ω is a convex polyhedron. We suppose that we have a decomposition of Ω in a rather general polyhedra. Almost every polyhedron will be acceptable in the decomposition. The theory discussed in chapter 1 holds for the 3D case. The 3D projectors are computed by individually considering the projectors of individual faces of the polyhedral. The individual projectors of each face can be computed in 2 ways. The first way is by assuming a coordinate transformation and computing the 2D projectors of the face by assuming it as a 2D element and the second way is by computing the face projectors in 3D themselves. Both methods were explored. The volume and centroid of arbitrary polyhedral elements were computed numerically by

$$\begin{aligned} V_E &= \int_{E\in\Omega} d\mathbf{x} \\ x_d &= \frac{1}{V_E}\int_{E\in\Omega} \mathbf{x}\ d\mathbf{x} \end{aligned}$$

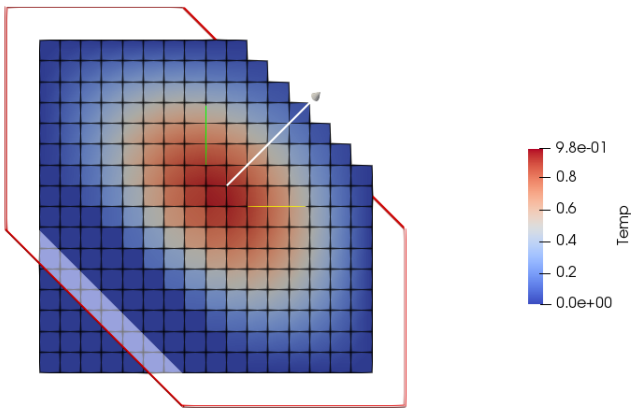
We consider the Poisson equation in 3 dimensions with homogeneous boundary condition

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \Gamma \end{aligned}$$

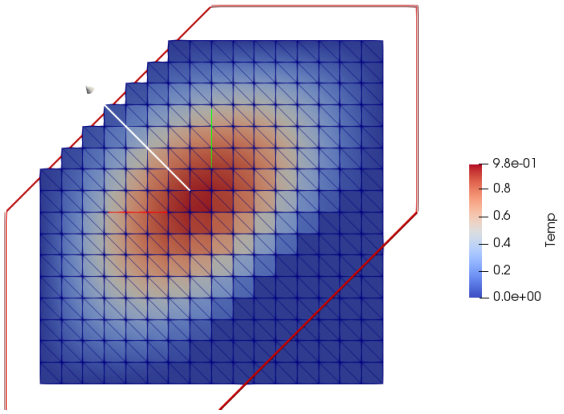
We take a unit cube $\Omega = (0,1) \times (0,1) \times (0,1)$ and $f(x,y,z) = 3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z)$. The exact solution of this equation is given by $u(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$.



3D VEM Solution - Scatter plot



Mesh with Cube elements



Mesh with Wedge elements

An order of convergence analysis was performed by successive refinement on cubic mesh and on the wedge mesh and expected optimal order was obtained.

$k = 1$

N	h	$\|u - u^h\|_{L^\infty}$	$\|u - u^h\|_{L^2}$	ooc L^∞	ooc L^2
2	0.86602	0.5876485	0.1038825	-	-
4	0.43301	0.2457649	0.0741661	1.2576	0.4861
8	0.21650	0.0756436	0.0257261	1.6999	1.5275
16	0.10825	0.0201026	0.0070390	1.9118	1.8697
32	0.05412	0.0051066	0.0018011	1.9769	1.9664

Table 2.5: Order of convergence analysis for 3D VEM for Poisson equation on cube mesh.

Chapter 3

Conclusion

The newly developed Virtual Element method was implemented to solve several problems. Theoretical results including the consistency of the bilinear forms was discussed. Using this result optimal error estimates were established. The numerical experiments were carried out primarily in MATLAB. To generate the Voronoi type meshes in 2D, a software named Polymesher was used. In addition, custom meshes were written for 2D and 3D cases. A hybrid combination of Finite element method and Virtual element method was implemented in order to test compatibility when there exists traditional finite elements and non traditional virtual elements in the same mesh to reduce computational costs in large scale problems. A toolbox was developed in matlab to generate efficient and sufficiently high order gauss quadrature rules for numerical integration on arbitrary polygons with least number of gauss points. A number of problems were solved in 2D including the fourth order nonlinear Rosenau equation with a mixed approach. A 3D virtual element solver was implemented for generalized polyhedrons with any number of faces and any number of sides on any face to solve the poisson problem and an order of convergence analysis was performed. Virtual Element method although more mathematically and computationally intensive compared to the Finite element method has its advantages and could be used in tandem with FEM to solve the problem better.

Bibliography

[1] Beirao Da veiga, F Brezzi, A. C. G. M. D. M. A. R. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences 23*, 1 (2013), 199–214.

[2] Cameron Talischi, Glaucio H. Paulino, A. P. I. F. M. Polymesher: a general-purpose mesh generator for polygonal elements written in matlab. *Journal Structural and Multidisciplinary Optimization 45*, 3 (feb 2012), 309–328.

[3] Giuseppe Vacca, B. D. V. Virtual element methods for parabolic problems on polygonal meshes. *Numerical Methods for Partial Differential Equations* (mar 2015), 199–214.

[4] L. BEIRÃO DA VEIGA, F. BREZZI, L. M. A. R. The hitchhiker's guide to the virtual element method.

[5] Russo, A. A. B. M. Equivalent projectors for virtual element methods. *Computers and Mathematics with Applications* (2013).

[6] Susanne Brenner, R. S. *The Mathematical Theory of Finite Element Methods*. Springer, 2014.

[7] Sutton, O. J. The virtual element method in 50 lines of matlab.