Thesis - Nodal Derivatives in the Finite Element Method as Applied to Geodynamic Modelling

Tom Weir

July 1, 2019

Contents

Abstract

A range of methods for calculating nodal derivatives were evaluated for the Finite Element Method in the context of geodynamics. For stress and heat flux of surface nodes, the Consistent Boundary Flux method of Zhong et al [13] was found to be more accurate than the other methods examined. For the stress tensor on internal nodes, the most accurate of the methods tested was found to be the Superconvergent Patch Recovery method of Zienkiewicz and Zhu [14] [15].

Chapter 1

Introduction

The Finite Element Method (FEM) is used extensively in computational modelling in many fields, including in geodynamics, such as in the codes CITCOMCU [10], ASPECT [1] and CONMAN [8]. In the case of viscous geodynamic models the primary variables of the model are the velocity vector v_i, the pressure p and the temperature T, and these are known to a high degree of accuracy on the nodes. The derivatives of these variables are known to a usually lower degree of accuracy on the nodes. However, some of these derivatives are of geophysical interest, are related to other quantities of geophysical interest, or are useful in deriving other quantities of geophysical interest. The derivatives in question are the strain rate tensor $e_{ij} = \frac{1}{2}(\partial_i v_j + \partial_j v_i)$ [1] and the temperature flux vector $\partial_i T$. Also of interest is the stress tensor $\sigma_{ij} = 2\mu e_{ij} - \delta_{ij} p$ [2], which is function of the strain rate tensor.

The stress tensor on the surface is required for calculating the dynamic topography, which is an important output and constraint for mantle convection models. [4] [12]. The temperature flux at the Earth's surface is of significant importance in the thermal evolution of the Earth [5], as is the heat flux between the core and the mantle [9] The strain rate inside the domain is useful, for example in the case of viscoelastoplasticity [3].

In this thesis I examine, implement and analyse various methods of calculating these quantities on the nodes, both on the surface of the domain and in the domain itself, and evaluate which method produces the most accurate results. I start with surface derivatives, examining the surface stress and temperature flux, and then examine the strain on the internal nodes. These are all implemented using the code Fieldstone [11].

[1] Where $\partial_i = \frac{\partial}{\partial x_i}$.

[2] where μ is the viscosity and δ_{ij} is the Kronecker delta.

Chapter 2

Surface Derivatives

2.1 Surface Strain

We start with an investigation of the into the strain (or traction) on the surfaces of the computational domain. There are three methods used and compared in this work, the centre-to-node method (CN), the least squares method (LS), and the consistent boundary flux method (CBF). The details of these methods are described below.

2.1.1 Methods

Centre-to-Node (CN)

In this method, the relevant derivative is calculated using a standard finite element method for each element. The value of the derivative on each node is then calculated as an unweighted average of each element adjacent to the element. With regular quadrilateral elements and a rectangular domain, as used in this paper, this leads to interior nodes receiving the average of four elements, edge nodes two elements, and corner nodes one element. This is also referred to as the pressure smoothing method in other works.

Least Squares (LS)

This method is related to the Superconvergent Patch Recovery Method of Zienkiewicz and Zhu [14] [15], which is explained in greater detail in Section 3.1.3. In short, it takes a patch of four elements, and computes the value of the derivative at the centre of each of these nodes. It then takes a polynomial and fits it to these values using a least squares method, and uses this to compute the value of the derivative at the node.

Consistent Boundary Flux (CBF)

The CBF is the most complex of the three methods being detailed here. It is obtained by rearanging the Stokes equation into a form that allows for the boundary fluxes to be calculated directly. Starting with a standard form of the Stokes

Equation:

$$\partial_j \sigma_{ij} = b_i$$

Where σ_{ij} is the stress tensor, and b_i is the body force vector. We then premultiply by an arbitrary test function N and integrate over the entire domain to get the weak form of the equation:

$$\int_\Omega N \partial_j \sigma_{ij} dV = \int_\Omega N b_i dV$$

We then use the chain rule and divergence theorem to rewrite this equation:

$$\int_\Gamma N \sigma_{ij} n_j \; dS - \int_\Omega \partial_j N \sigma_{ij} \; dV = \int_\Omega N b_i dV$$

We note that the traction vector is defined by:

$$t_i = \sigma_{ij} n_j$$

and use this to rearrange the equation to give us the following:

$$\int_\Gamma N t_i dS = \int_\Omega \partial_j N \sigma_{ij} dV + \int_\Omega N b_i dV$$

This equation can then be solved over the domain using a finite element method, to give us the tractions. It should be noted that the definition of the domain Ω in the above equations is general: as such the domain can be defined as consisting only of the boundary elements.

I begin by discussing the discretisation of the lefthand side of the equation. Firstly, the domain is divided into elements, and the equation is integrated over each element. The elemental equations are then assembled into a full domain matrix.

Now, we consider the tractions to be unknowns on the nodes, in standard FEM style. For Q1 elements:

$$t_x = \sum_{i=1}^{2} t_{x_i} N_i \qquad t_y = \sum_{i=1}^{2} t_{y_i} N_i \tag{2.1.1}$$

where N_i is the ith basis function, and t_{x_i} is the x-traction on the ith node.

We can use this to expand the equation:

$$\begin{bmatrix} \int_\Gamma N_0 t_{x_0} dS \\ \int_\Gamma N_0 t_{y_0} dS \\ \int_\Gamma N_1 t_{x_1} dS \\ \int_\Gamma N_1 t_{y_1} dS \\ \int_\Gamma N_2 t_{x_2} dS \\ \int_\Gamma N_2 t_{y_2} dS \\ \int_\Gamma N_3 t_{x_3} dS \\ \int_\Gamma N_3 t_{y_3} dS \end{bmatrix} = \begin{bmatrix} \int_\Omega \partial_i N_0 \sigma_{x i_0} dV \\ \int_\Omega \partial_i N_0 \sigma_{y i_0} dV \\ \int_\Omega \partial_i N_1 \sigma_{x i_1} dV \\ \int_\Omega \partial_i N_1 \sigma_{y i_1} dV \\ \int_\Omega \partial_i N_2 \sigma_{x i_2} dV \\ \int_\Omega \partial_i N_2 \sigma_{y i_2} dV \\ \int_\Omega \partial_i N_3 \sigma_{x i_3} dV \\ \int_\Omega \partial_i N_3 \sigma_{y i_3} dV \end{bmatrix} + \begin{bmatrix} \int_\Omega N_0 b_{x_0} dV \\ \int_\Omega N_0 b_{y_0} dV \\ \int_\Omega N_1 b_{x_1} dV \\ \int_\Omega N_1 b_{y_1} dV \\ \int_\Omega N_2 b_{x_2} dV \\ \int_\Omega N_2 b_{y_2} dV \\ \int_\Omega N_3 b_{x_3} dV \\ \int_\Omega N_3 b_{y_3} dV \end{bmatrix} \tag{2.1.2}$$

We now consider the above element, which is on the lower boundary of the domain. The tractions do not exist on nodes 2 and 3, and as such the lower half of the vector equation above may be discarded[1]. Further, as the x and y components

[1] N.B. This is mathematically dubious and difficult to rigorously justify. However, it works.

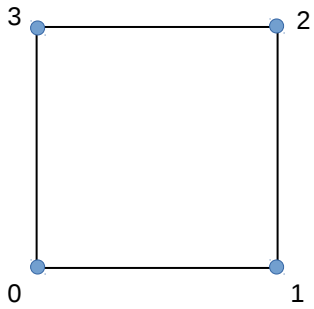


Figure 2.1: Diagram of the Q_1P_0 element showing element numbering used.

of the equation are independent of each other, they can be separated. For the x-tractions, this gives us:

$$\begin{bmatrix} \int_\Gamma N_0 t_{x_0} dS \\ \int_\Gamma N_1 t_{x_1} dS \end{bmatrix} = \begin{bmatrix} \int_\Omega \partial_i N_0 \sigma_{x i_0} dV \\ \int_\Omega \partial_i N_1 \sigma_{x i_1} dV \end{bmatrix} + \begin{bmatrix} \int_\Omega N_0 b_{x_0} dV \\ \int_\Omega N_1 b_{x_1} dV \end{bmatrix} \tag{2.1.3}$$

We integrate the left hand side over only one edge, in this case the 0-1 edge:

$$\int_\Gamma N_i t_x dS = \int_{\Gamma_{0-1}} N_i t_x dS \tag{2.1.4}$$

$$\int_{\Gamma_{0-1}} N_0 t_x dS = \int_{\Gamma_{0-1}} N_0 (N_0^\Gamma t_{x0} + N_1^\Gamma t_{x1}) dS \tag{2.1.5}$$

$$\int_{\Gamma_{0-1}} N_1 t_x dS = \int_{\Gamma_{0-1}} N_1 (N_0^\Gamma t_{x0} + N_1^\Gamma t_{x1}) dS \tag{2.1.6}$$

These can either be integrated exactly [2], or using a trapezoidal integration method[3]. In the analytical case, we have the following results:

$$\int_{\Gamma_{0-1}} N_0 N_0^\Gamma dS = \int_{\Gamma_{0-1}} N_1 N_1^\Gamma dS = \frac{h}{2}\frac{2}{3} \tag{2.1.7}$$

$$\int_{\Gamma_{0-1}} N_1 N_0^\Gamma dS = \int_{\Gamma_{0-1}} N_0 N_1^\Gamma dS = \frac{h}{2}\frac{1}{3} \tag{2.1.8}$$

where h is the distance between the nodes on the edge being integrated over. This leads to a left hand side of the form:

$$\frac{h}{2}\begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} t_{x0} \\ t_{x1} \end{bmatrix} \tag{2.1.9}$$

[2]This can either be an analytical solution, or an exact solution calculated using Gauss-Legendre Quadrature.

[3]N.B. In the context of CBF, this trapezoidal integration method is also referred to as Gauss-Lobatto Quadrature, as trapezoidal integration is equivalent to the lower limit of Gauss-Lobatto Quadrature, in the case that the number of integration points is two.

For the trapezoidal integration, we get the following:

$$\int_{\Gamma_{0-1}} N_0 N_0^{\Gamma} dS = \int_{\Gamma_{0-1}} N_1 N_1^{\Gamma} dS = \frac{h}{2} \tag{2.1.10}$$

$$\int_{\Gamma_{0-1}} N_1 N_0^{\Gamma} dS = \int_{\Gamma_{0-1}} N_0 N_1^{\Gamma} dS = 0 \tag{2.1.11}$$

This leads to a left hand side of the form:

$$\frac{h}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} t_{x0} \\ t_{x1} \end{bmatrix} \tag{2.1.12}$$

This second method of integration is also known as mass lumping (ML), and leads to a few consequences that are worth noting. Firstly, it leads to a diagonal left hand side matrix, which makes the solving step trivial and computationally very cheap. Secondly, it means each element is decoupled from each other element. That is, the traction at each node for one dimension is dependent exclusively on the right hand side value at that node and that dimension. This latter point become relevant later in the discussion and analysis of what I have termed "corner weirdness."

The algorithm proceeds as follows. All elements are looped over. For each element, the right hand side vector of Equation 2.1.2 is formed, in the same manner as in the main Stokes solver of the code. Each boundary of the domain is then looped over, and if the element lies on the boundary, the relevent components of the elemental right hand side vector are extracted, and assembled into the global right hand side vector, in standard FEM manner. The elemental left hand side matrix, as computed using either an exact or mass lumping technique as detailed above, is assembled into the global left hand side matrix. Note that as each boundary is looped over for each element, corner elements are assembled over twice, as they lie on two boundaries. After all elements are looped over, the left and right hand sides of the equation are solved, to give the tractions on all boundary nodes.

This algorithm can be extended to Q_2Q_1 and Q_3Q_2 elements, by applying the same principles. This is not done in detail here. However, note that the mass lumping in these cases uses a higher order Gauus-Lobatto integration, with 3 integration points for Q_2Q_1 and 4 points for Q_3Q_2.

2.1.2 Benchmarks

These methods are tested against several analytical benchmarks to measure their accuracy compared to one another, and how this changes as a function of mesh resolution.

Zhong, Gurnis and Hulbert

In their paper, primarily concerned with introducing and detailing the CBF, Zhong et al [13] introduce a benchmark to test their implementation of the CBF. This benchmark consists of a 1×1 box, with free slip boundary conditions on all boundaries. The density is uniformly zero throughout the domain, except at a depth y_0, where it is given by $\rho = \rho_0 \alpha T(x)$. The temperature is defined by $T(x) = \cos(kx)$, where k is the wavenumber, defined in terms of the wavelength λ as $k = \frac{2\pi}{\lambda}$. They use a Greens function method to derive a analytical solution for the y-traction on the upper surface, given by $t_y = \frac{\cos(kx)}{\sinh(k)}(k(1-y_0)\sinh(k)\cosh(ky_0) - k\sinh(k(1-y_0)) + \sinh(k)\sinh(ky_0))$. In their paper, Zhong et al. solve this benchmark for a 64×64 grid, for the cases that $y_0 = 63/64$, $y_0 = 62/64$ and $y_0 = 59/64$[4] Replication of these results is shown in Table 2.1.1. I also present a modified version of this benchmark in which the value of y_0 is kept constant at 0.5, as this allows the grid resolution to be varied while avoiding a situation where the density is zero on all nodes. This therefore allows the accuracy of the various methods to be examined as a function of grid spacing, to allow a more complete analysis. Further to this, in the paper Zhong et al only examine the accuracy of the traction at the top left of the domain. In addition to this I calculate L_1 and L_2 norms of the errors over the whole upper surface, as a function of grid resolution.

Table 2.1.1: Top left corner tractions.

Method	$y_0 = 63/64$	$y_0 = 62/64$	$y_0 = 59/64$
Analytical	0.995476	0.983053	0.912506
Pressure Smoothing (Zhong et al.)	1.15974	1.06498	0.911109
CBF (Zhong et al.)	0.994236	0.982116	0.912157
CBF (Fieldstone)	0.994236	0.982116	0.912157

Firstly, I present plots of the tractions, as calculated using the CBF and CN methods, along with the analytical solution, for a range of grid resolutions. These are shown in Figure 2.2. As expected, both methods show increasing accuracy with increasing resolution. Two notes should be made about the benchmark at this stage. Firstly, the benchmark was assessed using two codes, using two different formulations: one using a penalty method, one using a full pressure method. These codes produced results differing only by a fraction of a percent, and therefore only one set of results is shown. Mass lumping similarly produces very little difference to the results. Secondly, the value of $\rho_0\alpha$ must be equal to the number of elements along each axis to obtain results which match the analytical solution.

[4]Note that due to a typo in the paper of Zhong et al., they erroneously state that they use $y_0 = 60/64$ instead of $y_0 = 59/64$.

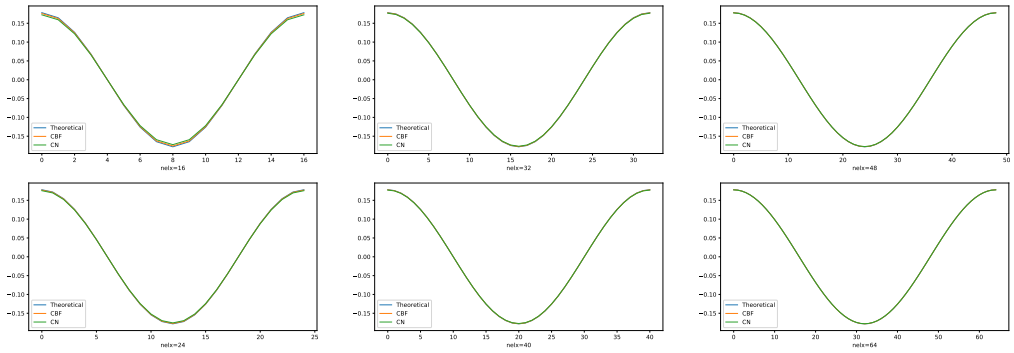


Figure 2.2: Plot of the CBF, CN and analytical tractions on the upper surface for the benchmark of Zhong, Gurnis and Hulbert, for multiple values of the number of elements.

Table 2.1.2: Convergence Rates of the L_1 and L_2 norms for the benchmark of Zhong et al., for each of the three methods.

Method	L_1	L_2
LS	2.00	2.00
CN	2.05	2.05
CBF	2.06	2.06

Next we examine the error norms. The log-log graph of these norms as a function of the grid resolution is shown in Figure 2.3. As can be seen, the relationship shown in this graph is linear, indicating that the relationship between these norms and grid resolution is of the form

$$||e_t||_1 = Ch^{r_t} \tag{2.1.13}$$

where e_t is the norm of the traction t, h is the grid spacing, C is a constant and r_t is the convergence rate. This can then be rearranged to:

$$\log(||e_t||_1) = \log(C) + r_t \log(h) \tag{2.1.14}$$

Regression analysis can be carried out to calculate the value of the gradient and intercept of the lines. From these, it is possible to calculate that in this case the CBF produces results equivalent in accuracy to the CN method with an approximately 31% higher grid resolution, and the LS method with a 63% increase in grid resolution.

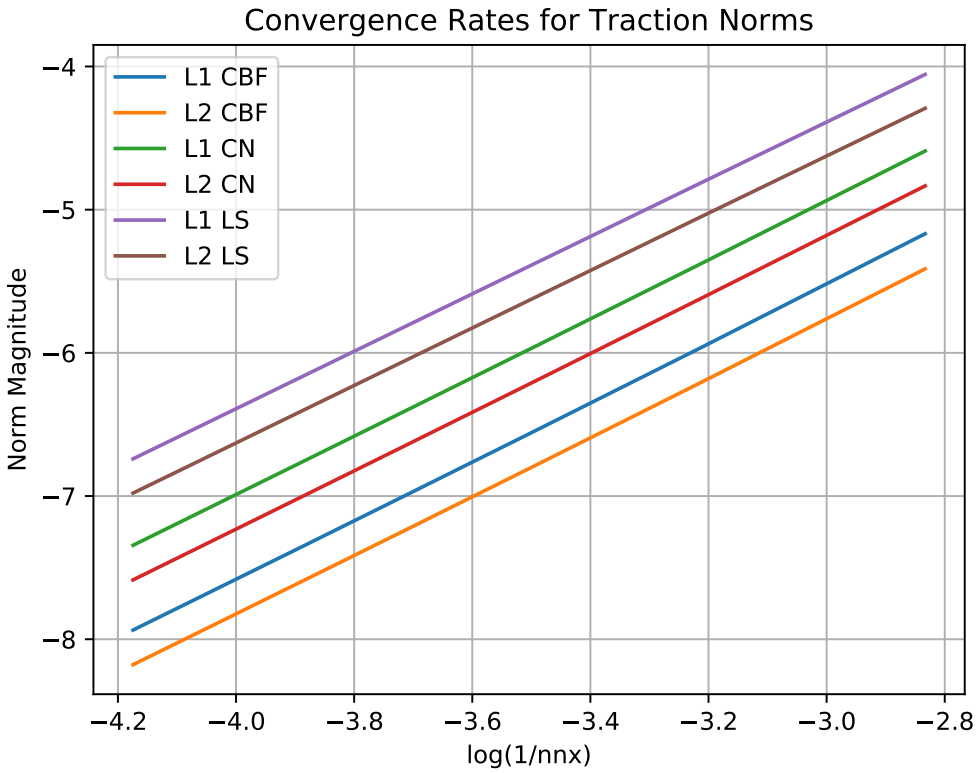


Figure 2.3: Convergence rates for the benchmark of Zhong, Gurnis and Hulbert.

Donea and Huerta

The second benchmark we use is that of Donea and Huerta, taken from [2]. This once again uses a 1×1 box, however this time all boundary conditions are no-slip. The benchmark uses the following prescribed body forces:

$$\begin{aligned} b_x &= (12-24y)x^4 + (-24+48y)x^3 + (-48y+72y^2-48y^3+12)x^2 \\ &\quad +(-2+24y-72y^2+48y^3)x + 1 - 4y + 12y^2 - 8y^3 \\ b_y &= (8-48y+48y^2)x^3 + (-12+72y-72y^2)x^2 \\ &\quad +(4-24y+48y^2-48y^3+24y^4)x - 12y^2 + 24y^3 - 12y^4 \end{aligned}$$

This is then solved to give:

$$\begin{aligned} u(x,y) &= x^2(1-x)^2(2y-6y^2+4y^3) \\ v(x,y) &= -y^2(1-y)^2(2x-6x^2+4x^3) \\ p(x,y) &= x(1-x) - 1/6 \end{aligned}$$

where the constant for the pressure is chosen such that the total pressure over the domain is zero.

Unlike the previous benchmark, this benchmark has no-slip boundary conditions on all boundaries, and as such both t_x and t_y can be computed on all four boundaries, and as we have the full equation for both velocity and pressure we can calculate analytical solutions for both of these on all boundaries.

Certain solutions of the Donea and Huerta benchmark produce a checkerboard pressure mode. As the surface traction is a function of the pressure (for the "outward" tractions), this means that we get a zigzag pattern in the tractions, as is shown in Figure 2.4. This behaviour only occurs for the full pressure version of the code, and not for the penalty version. As such, further analysis is done solely on the results of the penalty version.

The second matter to be discussed is the "corner weirdness" problem. The value of the tractions on the corner nodes is not properly defined in a square box, as there is no definition of the normal at a corner. The method is implemented in a manner such that each corner node is assembled over twice. This leads to the result that, on a corner node, $t_y = \sigma_{yy} + \sigma_{xy}$ and $t_x = \sigma_{xx} + \sigma_{xy}$.

This has few consequences with free slip boundary conditions, as $\sigma_{xy} = 0$ on the boundaries and therefore it does not impact the non-zero tractions. With no slip boundary conditions however, it leads to anomalous values for the tractions at the corner nodes. As is shown in Figures 2.6 and 2.5, if mass lumping is not used, then this anomalous value at the corner can propogate into the domain. In the absence of a better solution, the best mechanism for minimising the impact of this phenomenon is to use mass lumping and to ignore the traction values on the corner nodes in any analysis.

As before, we examine the convergence rates of the L_1 and L_2 norms of the various measures of the tractions, as a function of grid spacing. We use the results of the mass lumped code, and ignore the values of the corner nodes, in calculating these norms, so as to avoid the effects of the corner weirdness. Unlike the previous benchmark, these norms were calculated for all four surfaces of the domain, and for both t_x and t_y, using three methods, giving 24 in total.

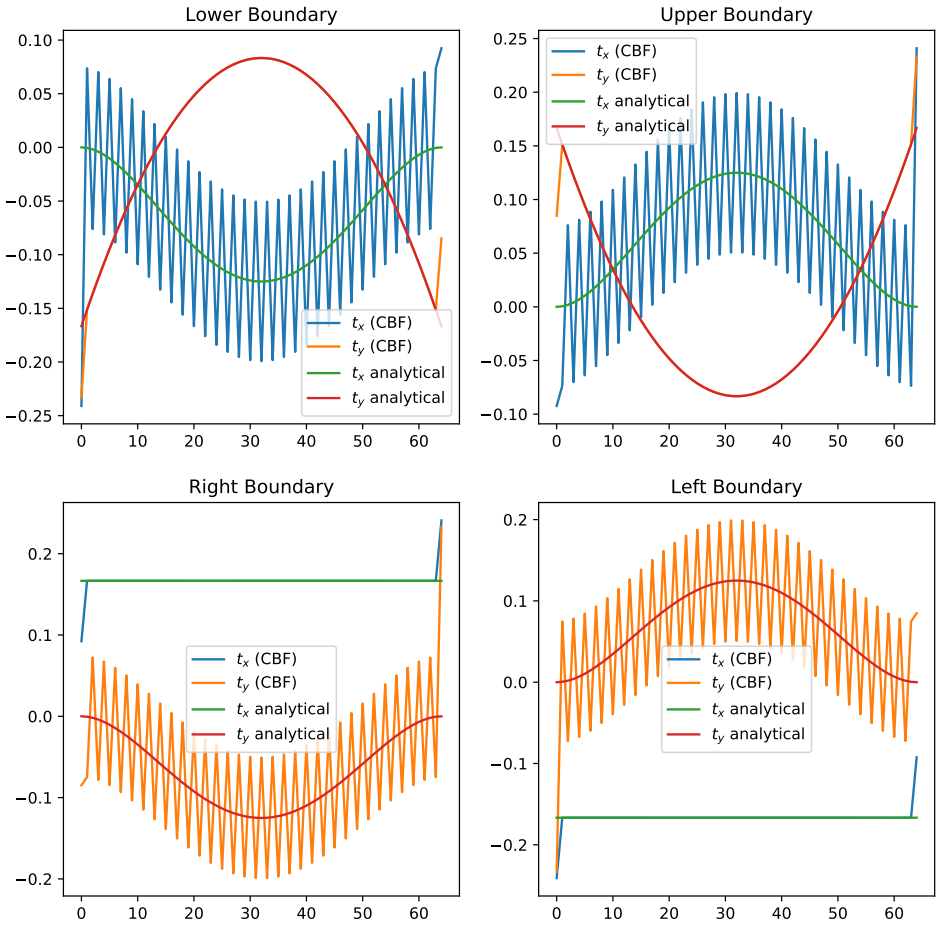


Figure 2.4: Surface Tractions for the benchmark of Donea and Huerta, showing analytical values and CBF values, calculated using a "full" solution of the velocity-pressure system. Note the effects of checkerboard pressure, causing a zigzag pattern in the CBF tractions.

Table 2.1.3: L_2 norm convergences for the Q_1P_0 Donea and Huerta benchmark.

Method	Upper		Lower		Sides	
	t_x	t_y	t_x	t_y	t_x	t_y
LS	1.90	1.99	1.90	1.99	2.00	1.90
CN	1.01	0.92	1.01	0.92	0.96	1.01
CBF	1.76	2.03	1.76	2.03	1.89	1.76

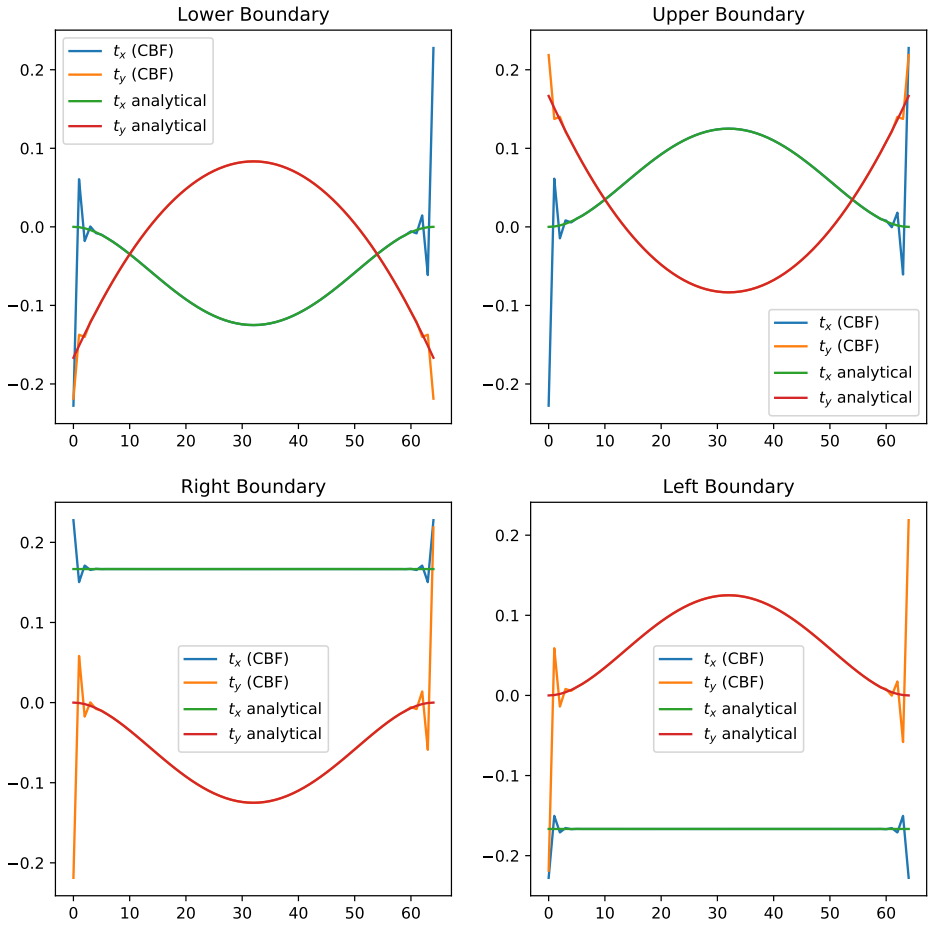


Figure 2.5: Surface Tractions for the benchmark of Donea and Huerta, showing analytical values and CBF values, calculated using a penalty method, with no mass lumping. Note the "corner weirdness" propogating into the domain.

Table 2.1.4: L_1 norm convergences for the Q_1P_0 Donea and Huerta benchmark.

Method	Upper		Lower		Sides	
	t_x	t_y	t_x	t_y	t_x	t_y
LS	1.86	1.95	1.86	1.95	1.95	1.86
CN	1.01	0.89	1.01	0.89	0.94	1.01
CBF	1.82	1.99	1.82	1.99	1.86	1.82

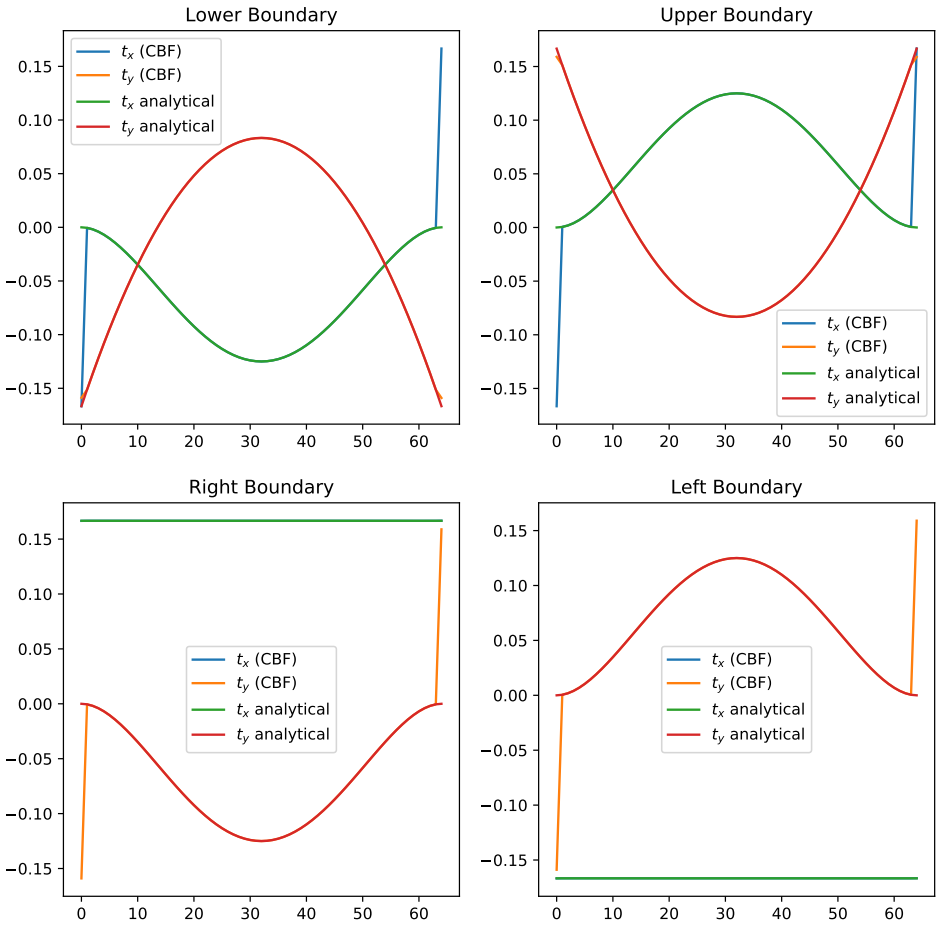


Figure 2.6: Surface Tractions for the benchmark of Donea and Huerta, showing analytical values and CBF values, calculated using a penalty method, with mass lumping. Note the "corner weirdness" does not propogate into the domain.

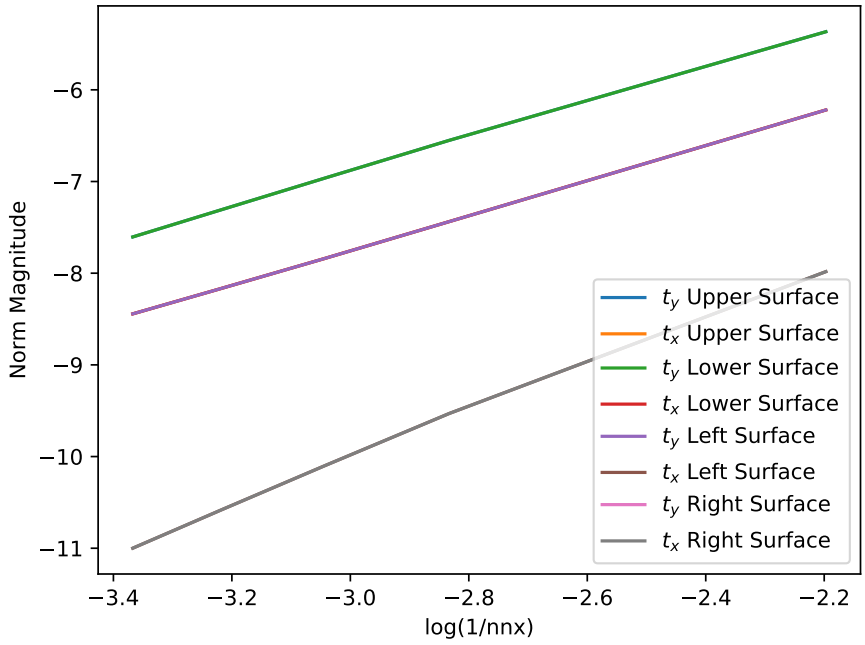


Figure 2.7: Convergence for Q_2Q_1.

Donea and Huerta Q_2Q_1

The Consistent Boundary Flux method was extended to Q_2Q_1 elements, and tested on the benchmark of Donea and Huerta. As previously, the corner weirdness phenomenon is observed, and propogates into the domain unless the mass lumping method is used. This is shown in Figures 2.8 and 2.9. The convergence of the norms is shown in Figure 2.7.

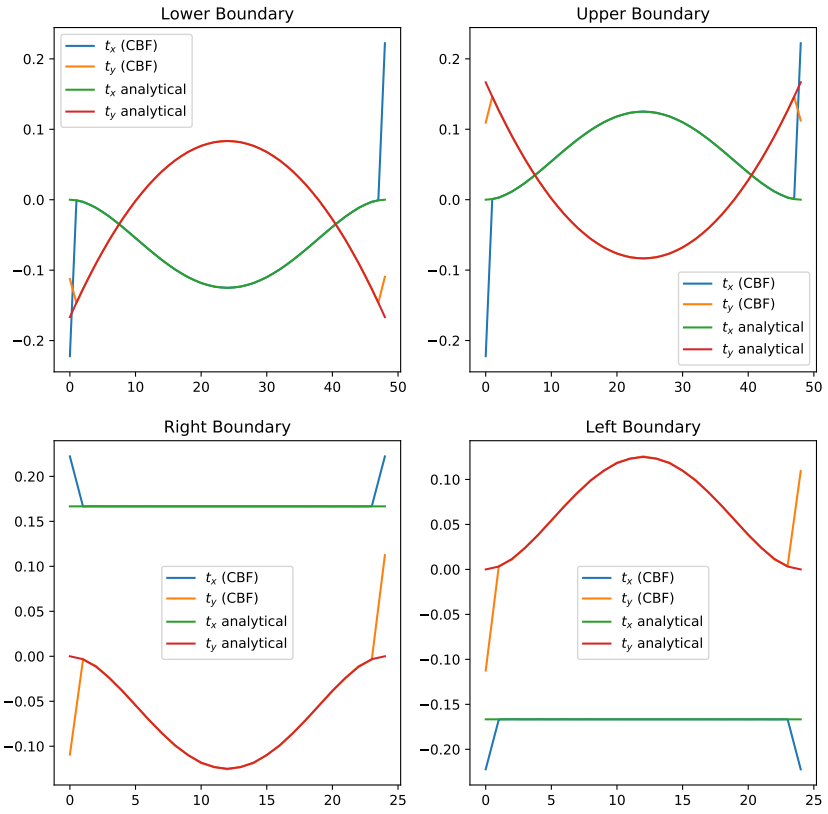


Figure 2.8: Tractions for the CBF Donea and Huerta Benchmark using Q_2Q_1 elements with Mass Lumping.

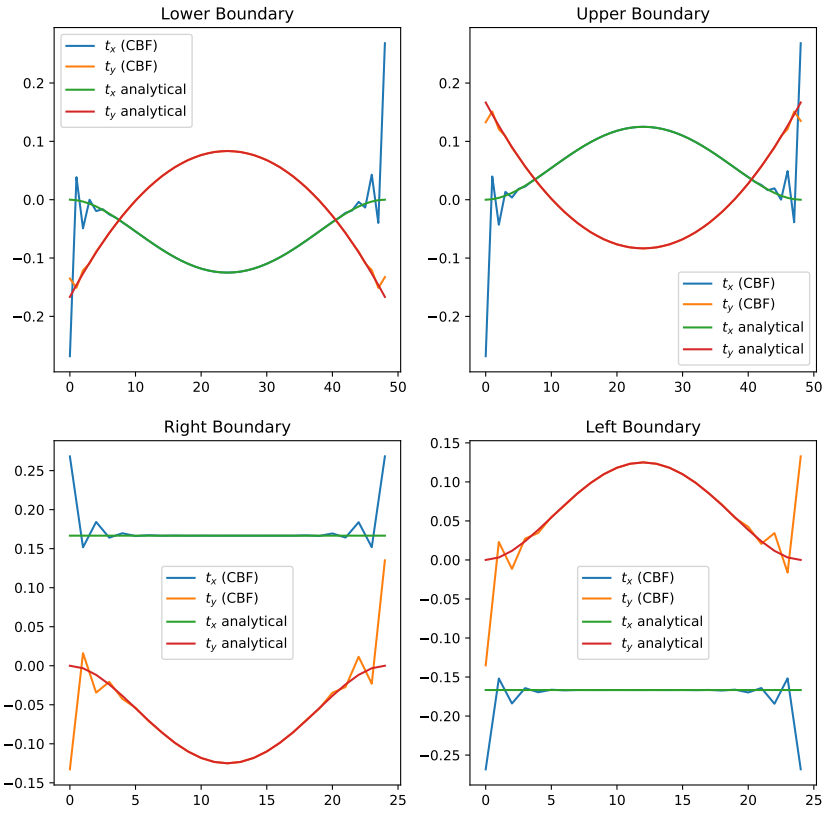


Figure 2.9: Tractions for the CBF Donea and Huerta Benchmark using Q_2Q_1 elements without Mass Lumping.

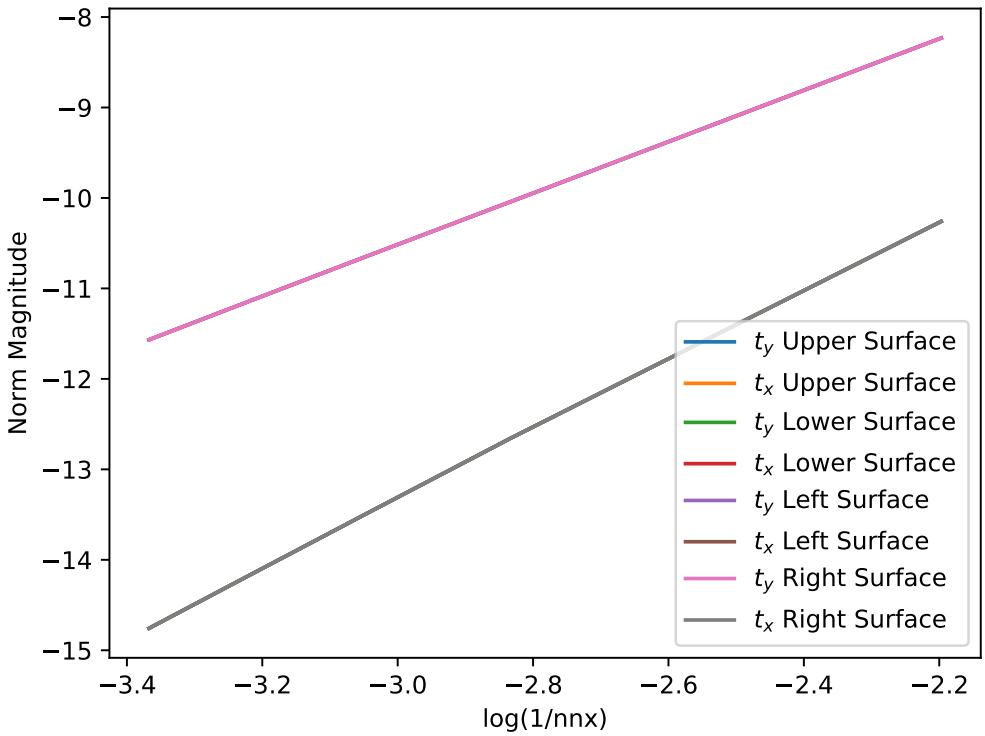


Figure 2.10: Convergence for Q_3Q_2.

Donea and Huerta Q_3Q_2

The Consistent Boundary Flux method was extended to Q_3Q_2 elements, and tested on the benchmark of Donea and Huerta. As previously, the corner weirdness phenomenon is observed, and propogates into the domain unless the mass lumping method is used. This is shown in Figures 2.11 and 2.12. The convergence of the norms is shown in Figure 2.10.

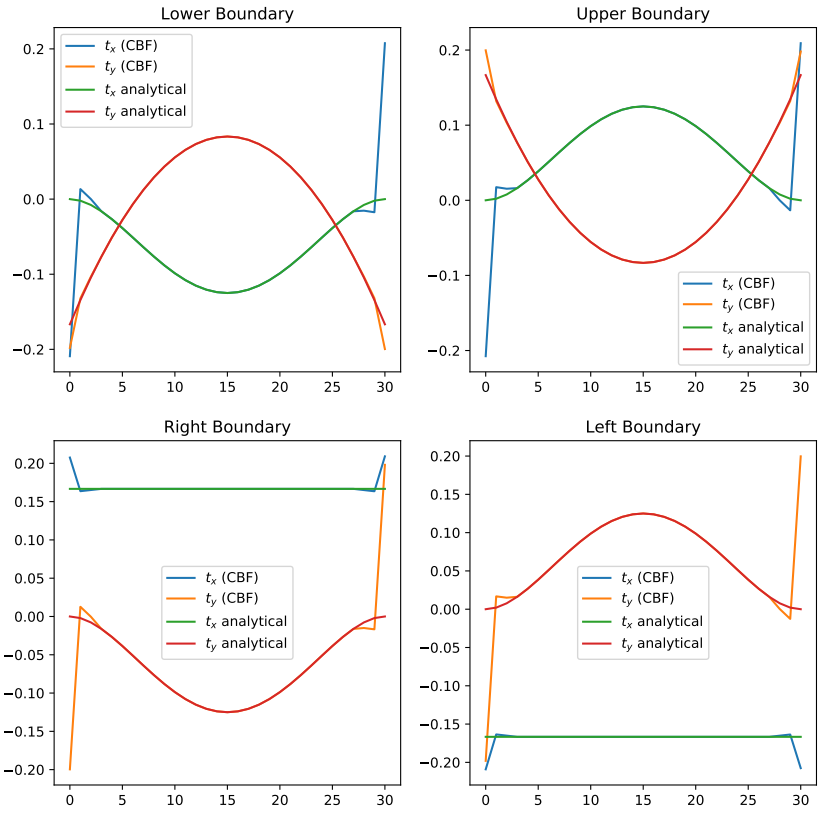


Figure 2.11: Tractions for the CBF Donea and Huerta Benchmark using Q_3Q_2 elements with Mass Lumping.

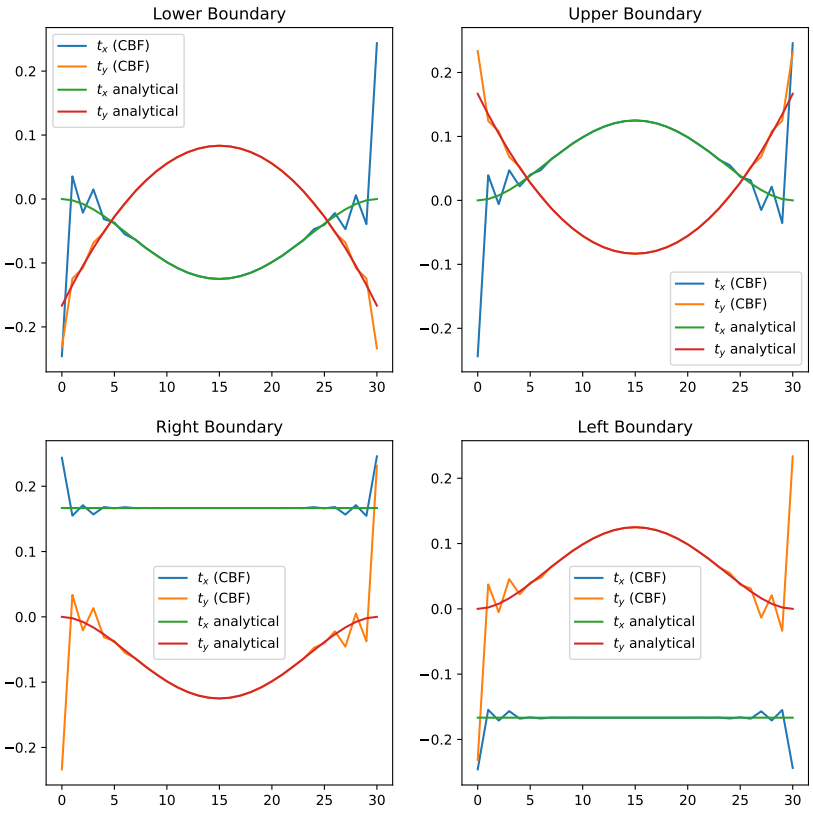


Figure 2.12: Tractions for the CBF Donea and Huerta Benchmark using Q_3Q_2 elements without Mass Lumping.

Annulus Benchmark

The CBF was implemented in an annulus benchmark taken from the fieldstone manual [11].

This is an analytical benchmark, where the velocity, pressure and density fields are defined by the following:

$$v_r(r,\theta) = g(r)k\sin(k\theta), \tag{2.1.15}$$

$$v_\theta(r,\theta) = f(r)\cos(k\theta), \tag{2.1.16}$$

$$p(r,\theta) = kh(r)\sin(k\theta), \tag{2.1.17}$$

$$\rho(r,\theta) = \aleph(r)k\sin(k\theta), \tag{2.1.18}$$

with

$$f(r) = Ar + B/r, \tag{2.1.19}$$

$$g(r) = \frac{A}{2}r + \frac{B}{r}\ln r + \frac{C}{r}, \tag{2.1.20}$$

$$h(r) = \frac{2g(r) - f(r)}{r}, \tag{2.1.21}$$

$$\aleph(r) = \partial_r\partial_r g - \frac{\partial_r g}{r} - \frac{g}{r^2}(k^2-1) + \frac{f}{r^2} + \frac{\partial_r f}{r}, \tag{2.1.22}$$

$$A = -C\frac{2(\ln R_1 - \ln R_2)}{R_2^2\ln R_1 - R_1^2\ln R_2}, \tag{2.1.23}$$

$$B = -C\frac{R_2^2 - R_1^2}{R_2^2\ln R_1 - R_1^2\ln R_2}. \tag{2.1.24}$$

The values of the constants are as follows:

$$R1 = 1 \tag{2.1.25}$$

$$R2 = 2 \tag{2.1.26}$$

$$C = -1 \tag{2.1.27}$$

$$k = 4 \tag{2.1.28}$$

The values of R1 and R2 ensure that the velocity is tangential to the inner and outer surfaces.

The velocity and pressure fields are solved for using a pre-existing fieldstone code

The x and y tractions are then calculated on the inner and outer surfaces using both the CBF and Centre-to-Node methods, and are shown along with the analytical values in Figure 2.13. The L_1 and L_2 norms were calculated as a function of grid spacing for each of these tractions, and these are shown in Figure 2.14. Note that the x and y tractions have identical norm magnitudes.

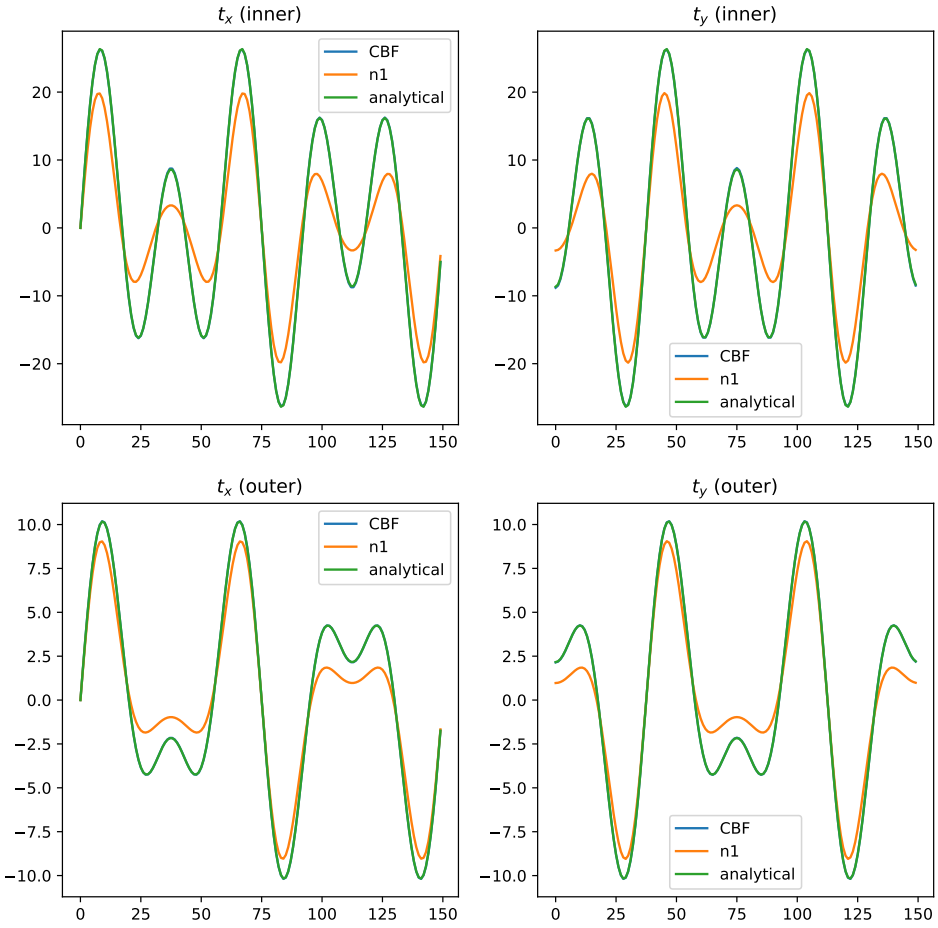


Figure 2.13: Tractions for the annulus benchmark.

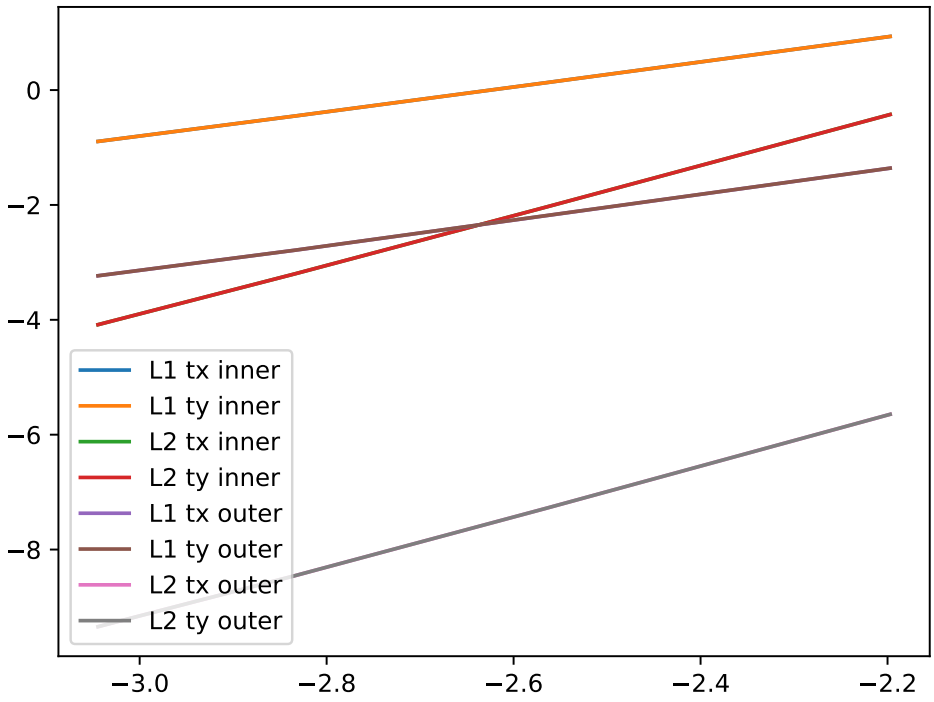


Figure 2.14: Norm Convergence for the annulus benchmark.

2.2 Temperature Flux

We next move on to an examination of the temperature flux at the surface of the domain. There are two methods examined in this piece, the elemental integration and the Consistent Boundary Flux method.

2.2.1 Methods

Elemental Integration Method

This method consists of calculating the flux for each surface element, and then summing them to get the total flux/Nusselt Number.

CBF

This method is similar to the surface traction CBF method detailed in Section 2.1.1. It is derived by rearranging the heat equation to calculate the fluxes directly. We start from a conventional form of the heat equation:

$$\rho c_p(\partial_t T + u_i \partial_i T) = \partial_i k \partial_i T \tag{2.2.1}$$

We then move to the weak form:

$$\int_\Omega \rho c_p(\partial_t T + u_i \partial_i T) d\Omega = \int_\Omega \partial_i k \partial_i T d\Omega \tag{2.2.2}$$

As previously, we integrate by parts and apply the divergence theorem:

$$\int_\Omega N \rho c_p(\partial_t T + u_i \partial_i T) d\Omega = \int_\Gamma N \partial_i k T n_i d\Gamma - \int_\Omega \partial_i N k \partial_i T d\Omega \tag{2.2.3}$$

What we care about is the heat flux $q_i = -\partial_i kT$, so we can substitute this in and rearrange, to give:

$$\int_\Gamma N q_i d\Gamma = -\int_\Omega N \rho c_p \partial_t T d\Omega - \int_\Omega N \rho c_p u_i \partial_i T d\Omega - \int_\Omega \partial_i N k \partial_i T d\Omega \tag{2.2.4}$$

This is then discretised in a manner equivalent to the discretisation of the strain CBF discussed above in Section 2.1.1.

2.2.2 Benchmarks

I present here a set of analytical benchmarks of increasing complexity to test the two methods. On each benchmark the Nusselt number is calculated as a function of grid spacing, and the fractional error of this is calculated. The Nusselt number is defined as

$$Nu = \int_{x=x_0}^{x=x_1} \partial_y (kT(y = y_1))dx \tag{2.2.5}$$

where the domain is $[x_0, x_1] \times [y_0, y_1]$. These are tested a version of Fieldstone [11] with Q_1 temperature elements and an instantaneous temperature solver.

Conduction Analytical Benchmark

The two methods are first tested against a pure conduction benchmark. This is a steady state pure conduction benchmark with no heating, solving the simplest form of the heat equation:

$$\partial_i k \partial_i T = 0 \tag{2.2.6}$$

This is solved over the domain [0,1]×[0,1]. The temperature is constrained to be 0 on the lower boundary and 1 on the upper boundary, which leads to the temperature being defined by the equation $T = y$, and an analytical Nusselt number of 1. The temperature profile is shown in Fig 2.16. This benchmark is taken from the fielstone manual [11].

As can be seen in Figure 2.15, the CBF converges and becomes more accurate with higher resolution. However, the elemental method is extremely accurate due to the simple nature of the benchmark, and is thus accurate to within machine precision for all resolutions.

Conduction Analytical Benchmark Heating

The next benchmark is related to the above benchmark, but with the addition of a constant heating term, such that the heat equation is now of the form:

$$\partial_i k \partial_i T + H = 0 \tag{2.2.7}$$

This is solved with the same constraints and domain as above, with the heating factor $H = 1$ and heat conductivity $k = 1$. This gives a temperature defined by the equation:

$$T = -\frac{1}{2}y^2 + \frac{1}{2}y \tag{2.2.8}$$

and thus an analytical Nusselt number of $\frac{1}{2}$. This benchmark is taken from the fielstone manual [11]. The temperature profile is shown in Figure 2.18, and the error convergence in Figure 2.17. In this more complicated case, the CBF can be seen to outperform the elemental method.

Convection-Conduction Analytical Benchmark

An analytical benchmark was created as part of this thesis for the purposes of testing these two methods of calculating temperature flux. It is a steady state benchmark, taking into account only advection and diffusion. It is therefore a

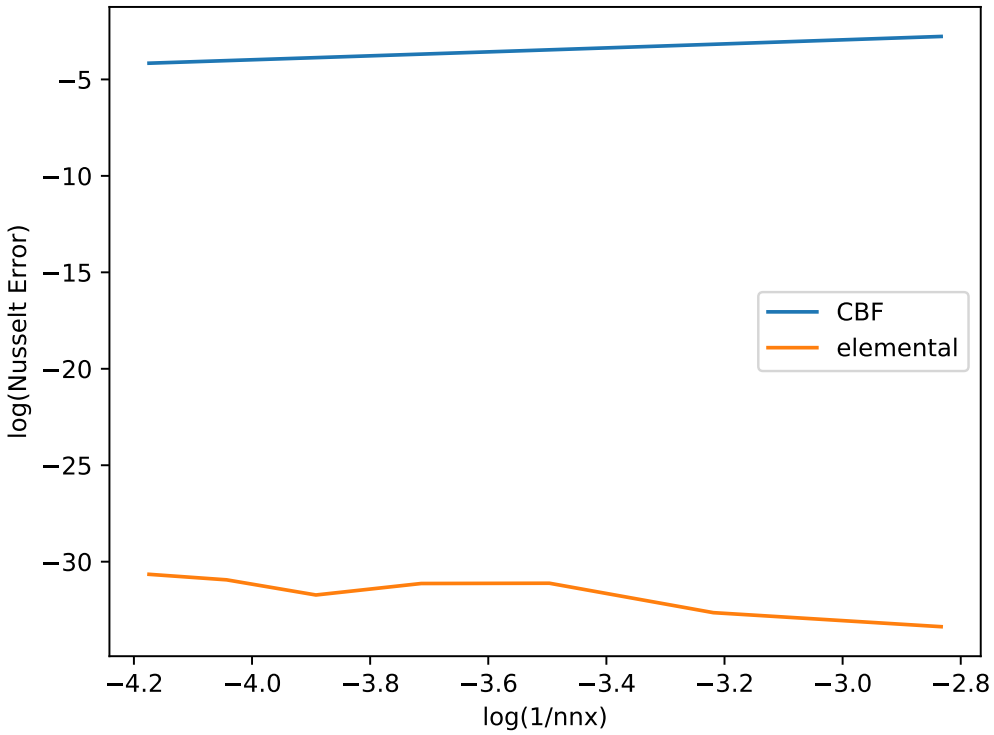


Figure 2.15: Convergence of temperature flux for the conduction benchmark without heating.

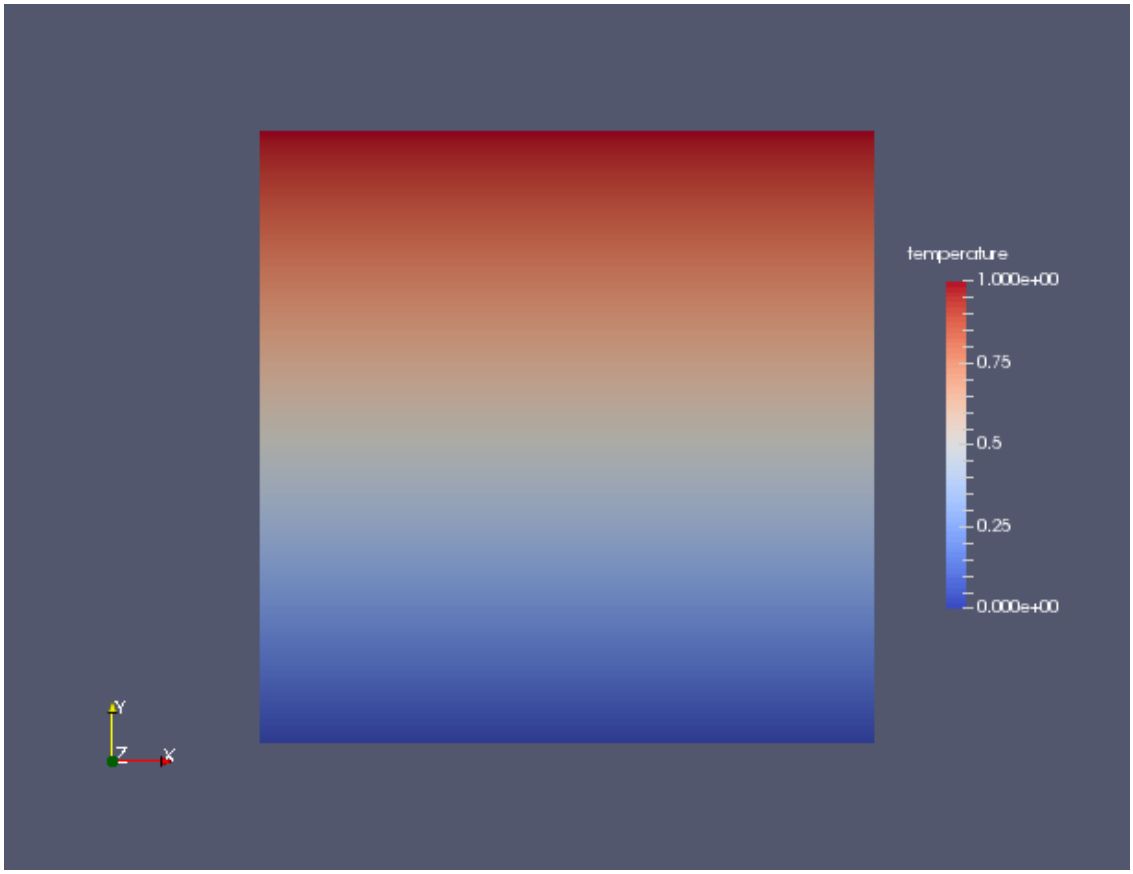


Figure 2.16: Temperature Profile for the Conduction Benchmark.

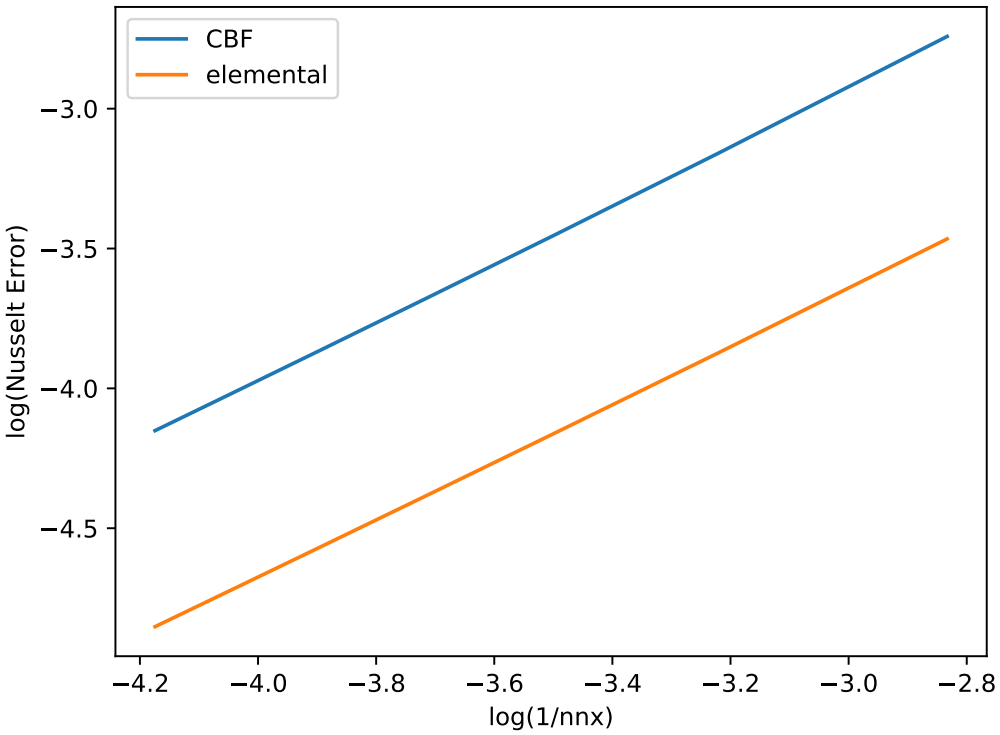


Figure 2.17: Convergence of temperature flux for the conduction benchmark with heating.

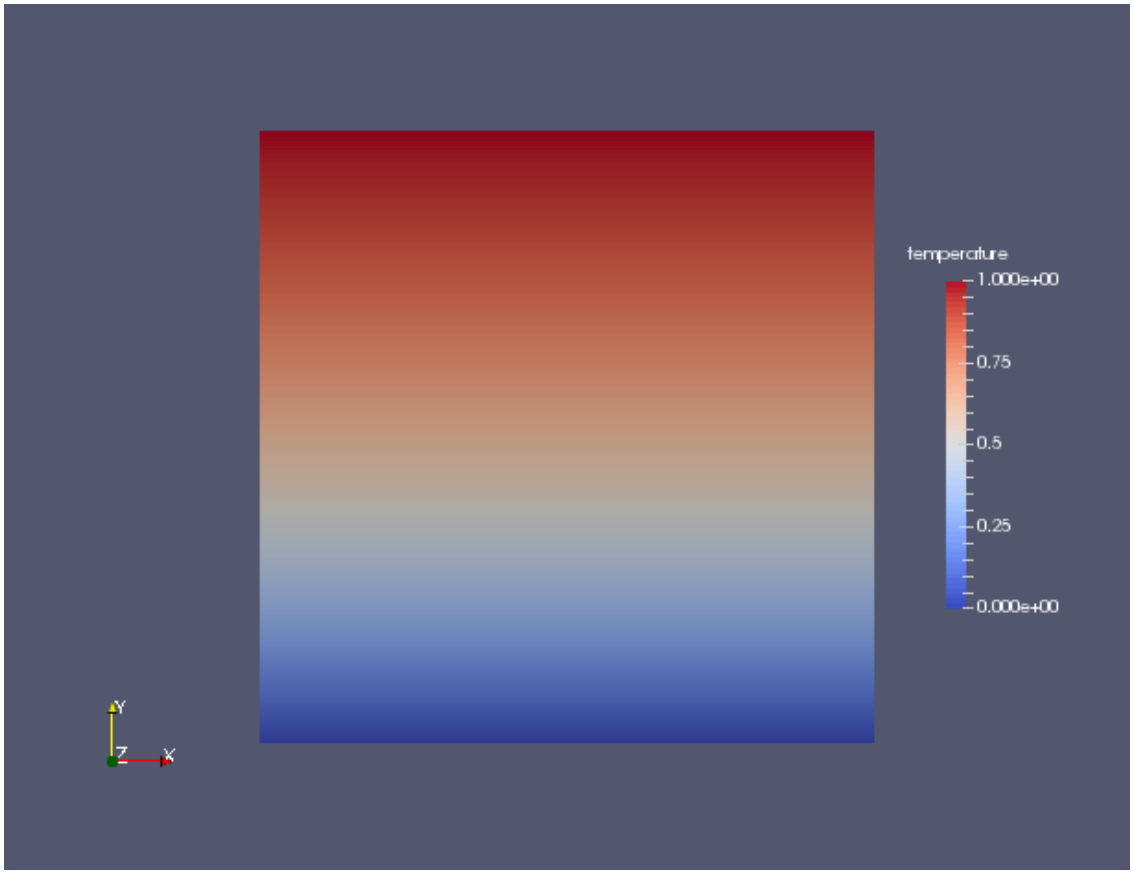


Figure 2.18: Temperature Profile for the Conduction-Heating Benchmark.

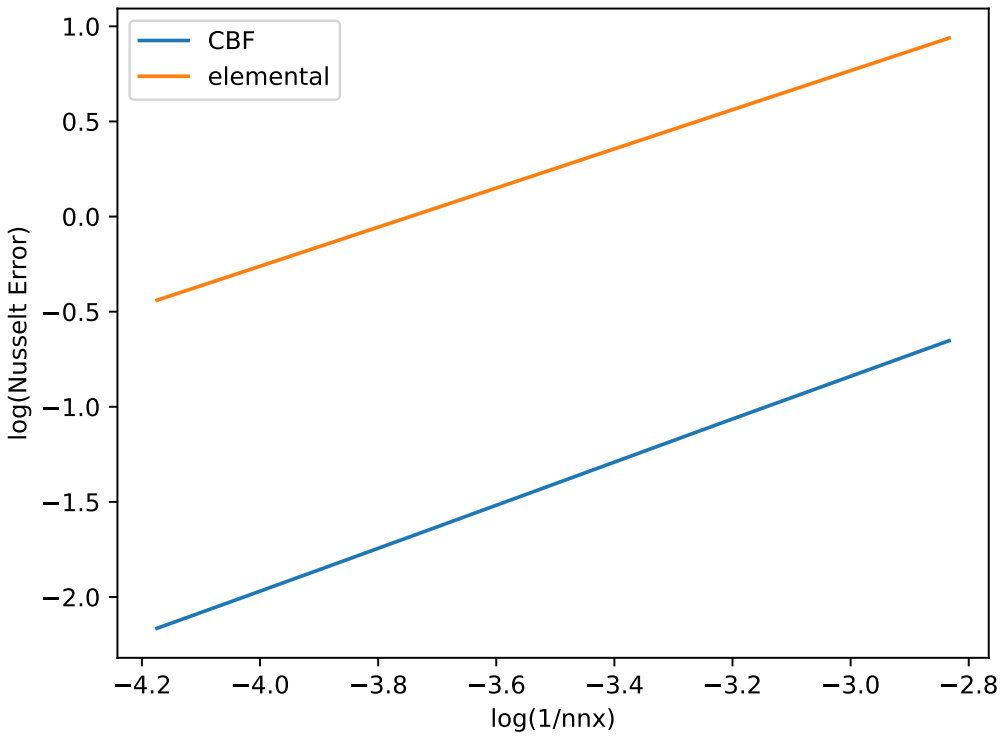


Figure 2.19: Convergence of temperature flux for the Convection-Conduction Analytical Benchmark.

solution to the following form of the temperature equation:

$$\rho c_p u_i \partial_i T - \partial_i k \partial_i T = 0 \tag{2.2.9}$$

The solution is the following:

$$T = x^2(y - y^3) \tag{2.2.10}$$

$$u_x = 0 \tag{2.2.11}$$

$$u_y = \frac{2y - 2y^3 - 6x^2 y}{x^2 - 3x^2 y^2} \tag{2.2.12}$$

As this velocity has singularities in it, the domain must be chosen to avoid these. The domain therefore chosen is [3,4]×[0.5,1]. The velocity and temperature fields are shown in Figures 2.21 and 2.20 respectively.

We calculate the temperature flux over the upper boundary using both methods, and compare it to the analytical value.

As was the case for benchmarking the surface strain, this benchmark is repeatedly run for a range of resolutions, to measure the accuracy of the two methods. As in the previous benchmark, the CBF outperforms the elemental method.

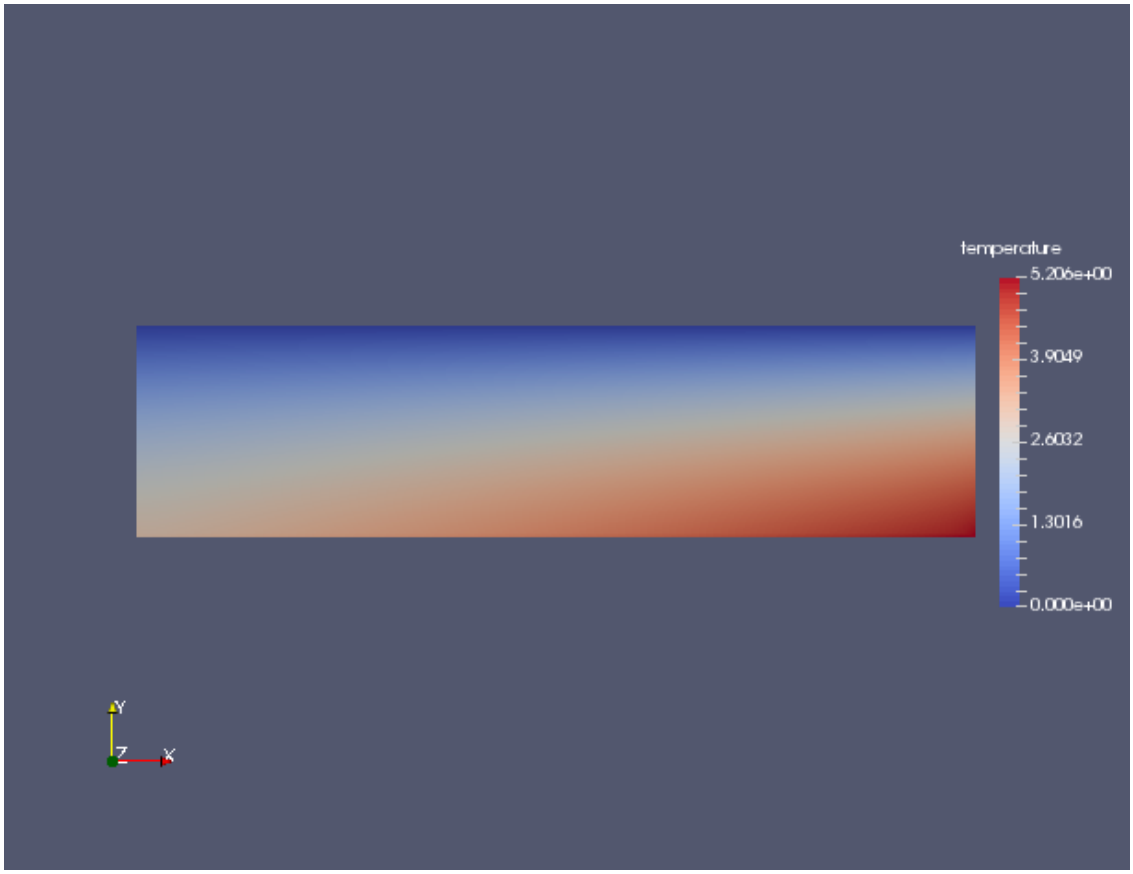


Figure 2.20: Temperature Profile for the Conduction-Convection Benchmark.

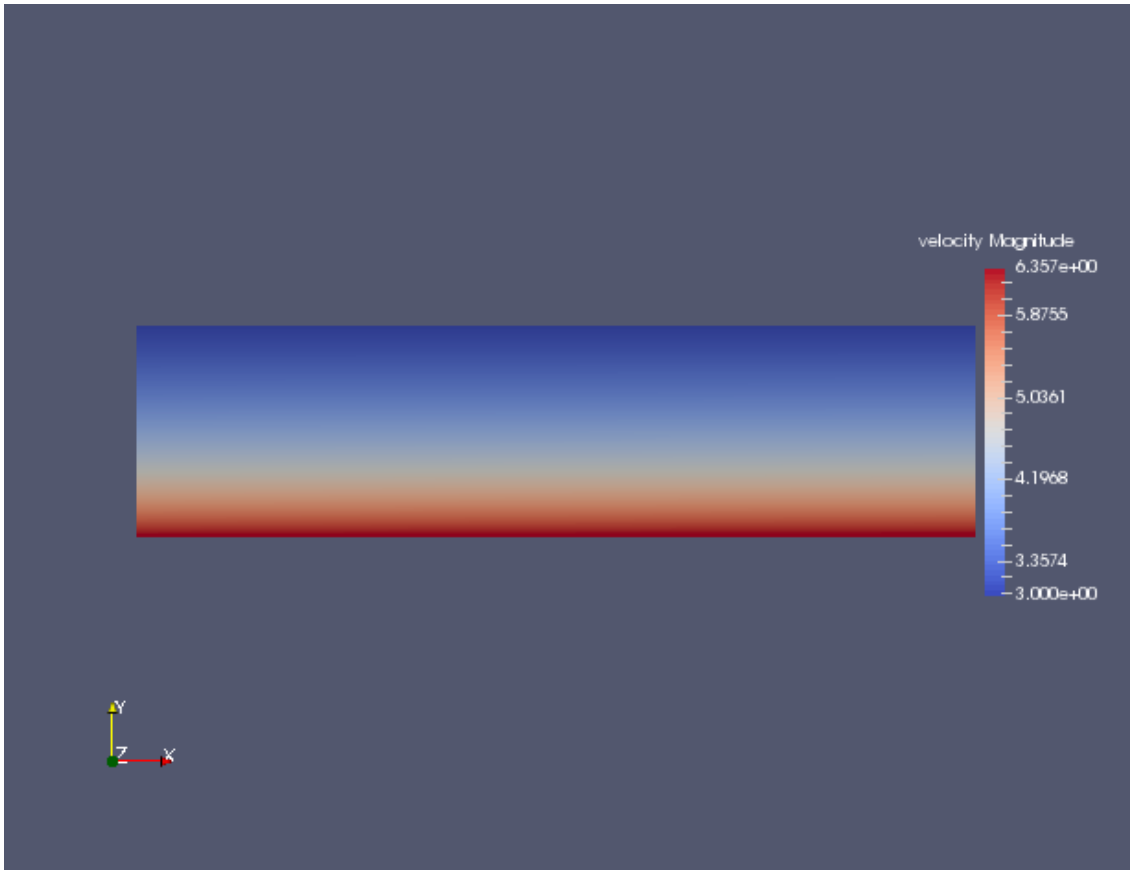


Figure 2.21: Velocity Profile for the Conduction-Convection Benchmark.

Viscous Dissipation Analytical Benchmark

This benchmark features viscous dissipation/frictional heating, solving the following form of the energy conservation equation:

$$\rho c_p u_i \partial_i T - \partial_i k \partial_i T = \Phi \tag{2.2.13}$$

where

$$\Phi = \tau_{ij} \partial_i u_j \tag{2.2.14}$$

This benchmark is based on the horizontal laminar flow of a fluid through an infinitely long pipe (Poiseuille Flow). The domain is [-1,1]×[-1,1]. The temperature is constrained to be 0 on the upper and lower boundaries, and the velocity is given by:

$$v_x = u_0(1 - y^2) \tag{2.2.15}$$

$$v_y = 0 \tag{2.2.16}$$

This gives a temperature profile of

$$T = \frac{1}{3}\frac{\eta u_0^2}{k}(1 - y^4) \tag{2.2.17}$$

The convergence of the Nusselt number for this benchmark (using an analytical value for the frictional heating) is shown in Figure 2.22.

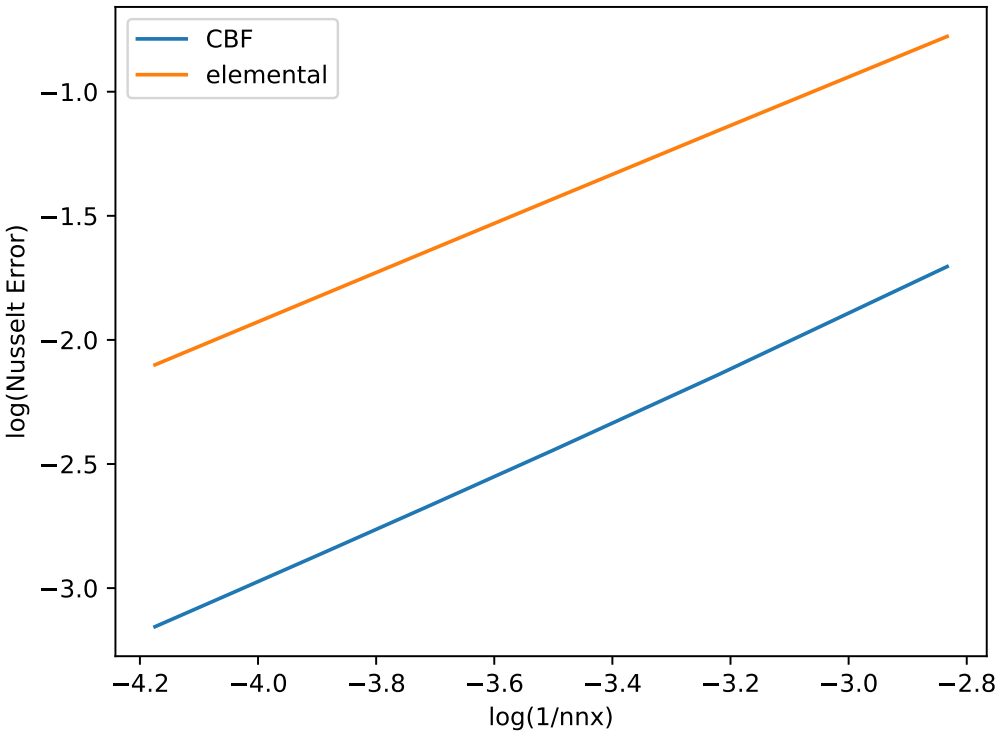


Figure 2.22: Convergence of temperature flux for the Viscous Dissipation Benchmark.

Chapter 3

Internal Nodes

3.1 Methods

Several methods of calculating the internal nodal derivatives of the velocity field in a Navier-Stokes system were examined. In this thesis I examine three methods: the centre-to-node, corner-to-node and superconvergent patch recovery method. Further methods not examined in detail in this thesis include the Finite Nodal Displacement Method developed by Ilinca and Pelletier [6] [7]. They report their method to be the most accurate, however attempts to implement it were not successful.

3.1.1 Centre to Node

For this method, we first compute the value of the derivative at the centre of each element. Then, for each node, the nodal value is taken to be the mean of the values at centre of each element adjacent to the node. For nodes inside an element, the nodal value is take to be equal to the elemental value.

3.1.2 Corner to Node

For each element, the value of the derivative is calculated at the position of each node adjacent to the element, and this is taken as the nodal value. For nodes that are adjacent to multiple nodes, the nodal value is taken to be the mean of the values computed for each adjacent element.

3.1.3 Superconvergent Patch Recovery

This method is based on that of Zienkiewicz and Zhu [14] [15]. Within each element, there are points at which the derivatives converge faster than elsewhere. These points are located at the roots of a Legendre Polynomial, and are thus at the same points used for Gauss-Legendre Quadrature. For this reason they are also referred to as Gauss Points. For Q_1 elements, the single superconvergent point is located at the centre of the element, and thus at the root of the 0th order

Legendre Polynomial. For Q_2 there are 4 points, located at $(\pm\sqrt{3}, \pm\sqrt{3})$; these are the roots of the 1st order Legendre Polynomial. The reasons for the superconvergence at these points, and the reasons for their location, are beyond the scope of this thesis. In this method we take a patch of 4 elements, and first calculate the derivatives at their superconvergent points. We then fit a polynomial to these points, and use this polynomial to calculate the value of the derivatives at the internal nodes. This can be done in two ways, and for Q_1 it proceeds as follows:

For Q_1 we wish to fit a 4 term polynomial of the following form: $g = a+bx+cy+dxy$, where g is the relevant derivative. The first method is the Least Squares method. This method minimises the sum $J = \frac{1}{2}\sum_{i=1}^{4}[a+bx_i+cy_i+dx_iy_i-g^h(x_i,y_i)]^2$ with respect to the variables a,b, c and d. We define here that g^h is the derivative as calculated using the shape functions etc, and (x_i, y_i) are the coordinates of the ith Gaussian Point. The function is minimised when $\frac{\partial J}{\partial a} = \frac{\partial J}{\partial b} = \frac{\partial J}{\partial c} = \frac{\partial J}{\partial d} = 0$. Imposing these constraints lead to the following linear system:

$$\begin{bmatrix} \sum_{i=1}^4 1 & \sum_{i=1}^4 x_i & \sum_{i=1}^4 y_i & \sum_{i=1}^4 x_iy_i \\ \sum_{i=1}^4 x_i & \sum_{i=1}^4 x_i^2 & \sum_{i=1}^4 y_ix_i & \sum_{i=1}^4 x_i^2y_i \\ \sum_{i=1}^4 y_i & \sum_{i=1}^4 y_ix_i & \sum_{i=1}^4 y_i^2 & \sum_{i=1}^4 x_iy_i^2 \\ \sum_{i=1}^4 x_iy_i & \sum_{i=1}^4 x_i^2y_i & \sum_{i=1}^4 x_iy_i^2 & \sum_{i=1}^4 x_i^2y_i^2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^4 g^h(x_i,y_i) \\ \sum_{i=1}^4 x_ig^h(x_i,y_i) \\ \sum_{i=1}^4 y_ig^h(x_i,y_i) \\ \sum_{i=1}^4 x_iy_ig^h(x_i,y_i) \end{bmatrix} \tag{3.1.1}$$

This system can then be solved for the values of a,b, c and d. The value of the derivative at the node can then be calculated using $g^{node} = a + bx_{node} + cy_{node} + dx_{node}y_{node}$.

The second method is to hold the equation to be true at each Gauss point seperately, thus leading to the following system:

$$\begin{bmatrix} 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \\ 1 & x_4 & y_4 & x_4y_4 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} g^h(x_1,y_1) \\ g^h(x_2,y_2) \\ g^h(x_3,y_3) \\ g^h(x_4,y_4) \end{bmatrix} \tag{3.1.2}$$

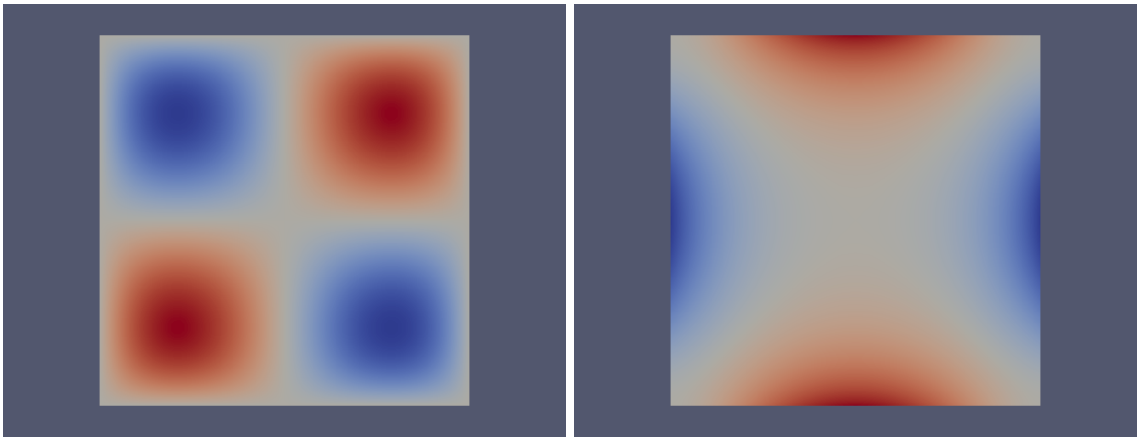
This can again be solved for a,b, c and d, and these used to calculate g^{node}.

For Q_2, both of these methods can be applied, using either a 9 term polynomial or a 16 term polynomial. The 9 term polynomial is of the form $g = a + bx + cy + dxy + ex^2 + fy^2 + gx^2y + hxy^2 + ix^2y^2$, and the 16 term polynomial is of the form $g = a + bx + cy + dxy + ex^2 + fy^2 + gx^2y + hxy^2 + ix^2y^2 + jx^3 + ky^3 + lx^3y + my^3x + nx^3y^2 + oy^3x^2 + px^3y^3$. Note that for the second method using 9 terms, the left hand matrix is not square; the system must therefore be solved using a least squares method, such that if the equation is specified as $Ax = b$, the solution is $x = (A^TA)^{-1}A^Tb$.

In the benchmarks section I refer to these methods with numbers. Method 1 refers to the 9 term polynomial with the first assembly method, method 2 to the 9 term method with the second assembly method, method 3 to the 16 term polynomial with the first assembly method, and finally method 4 to the 16 term polynomial with the second assembly method.

3.2 Benchmark - Donea and Huerta

As in the surface strain section, we use the benchmark of Donea and Huerta [2] to test these methods.



(a) e_{xx} (b) e_{xy}

Figure 3.1: Stresses for the Donea and Huerta Benchmark.

$$\begin{aligned} b_x &= (12-24y)x^4+(-24+48y)x^3+(-48y+72y^2-48y^3+12)x^2 \\ &\quad +(-2+24y-72y^2+48y^3)x+1-4y+12y^2-8y^3 \\ b_y &= (8-48y+48y^2)x^3+(-12+72y-72y^2)x^2 \\ &\quad +(4-24y+48y^2-48y^3+24y^4)x-12y^2+24y^3-12y^4 \end{aligned}$$

This is then solved to give:

$$\begin{aligned} u(x,y) &= x^2(1-x)^2(2y-6y^2+4y^3) \\ v(x,y) &= -y^2(1-y)^2(2x-6x^2+4x^3) \\ p(x,y) &= x(1-x)-1/6 \end{aligned}$$

These give the following strains:

$$\begin{aligned} e_{xx} &= 4xy(1-x)(1-2x)(1-y)(1-2y) \\ e_{xy} &= x^2(1-x)^2(1-6y+6y^2)-y^2(1-y)^2(1-6x+6x^2) \\ e_{yy} &= -4xy(1-x)(1-2x)(1-y)(1-2y)=-e_{xx} \end{aligned}$$

These stresses are shown in Figure 3.1.

This benchmark is run using the fieldstone code. As the results for the Q_1P_0 element have already been examined by Thieulot in an unpublished paper, here I present only the results for the Q_2Q_1 element. Three variations of this element are examined. The first is with the elements as regular quadrilaterals. The second is the semi-random element, in which the corner nodes are displaced randomly, and the other nodes are placed such that the element remains quadrilateral. The third method is the fully random method, in which all elements are displaced randomly from their quadrilateral positions.

These displacements have a maximum amplitude of $\frac{h}{10}$, where h is the relevant dimension of the element. The results of the regular and semi-random elements are indistinguishable, and are therefore presented together.

Three measures of the error are calculated for each independent element of the strain rate tensor. The first is the L_2 error, defined as

$$err_{L_2} = \sqrt{\sum_{elements} \int_{\Omega_{element}} (e_{ij}^{analytical} - e_{ij}^{numerical})^2 d\Omega} \tag{3.2.1}$$

The second and third are mean nodal errors, the first on the internal nodes and the second on the edge nodes. They are defined as:

$$err_{internal} = \sqrt{\frac{\sum_{internalnodes} (e_{ij}^{analytical}(x_{node}, y_{node}) - e_{ij}^{numerical}(x_{node}, y_{node}))^2}{n_{internalnodes}}} \tag{3.2.2}$$

$$err_{edge} = \sqrt{\frac{\sum_{edgenodes} (e_{ij}^{analytical}(x_{node}, y_{node}) - e_{ij}^{numerical}(x_{node}, y_{node}))^2}{n_{edgenodes}}} \tag{3.2.3}$$

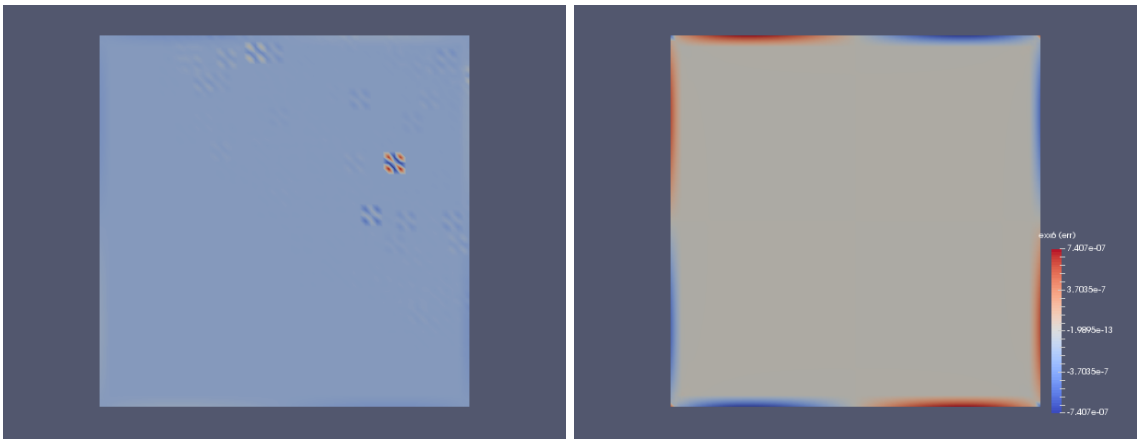
As $e_{xx} = -eyy$, the norms for these two terms are identical and presented together. These norms are calculated for a range of resolutions i.e. resolution studies are carried out.

3.2.1 Q2Q1

Regular Grid

The graphs of the norm convergences are shown in Figure 3.5, and the convergence rates are shown in Table 3.2.1. It can be seen that only one method of SPR converges regularly; the other methods have irregular convergence patterns. The reason for this can be seen in Figure 3.2, which shows the errors in e_{xx} for each of the methods at a given resolution. It can be seen that the errors for the first three methods are not regular, but are dominated by individual elements with errors of a much larger magnitude than that of other elements. The reasons for this are not conclusively proven. However, the two methods using the method that of minimising the least squares function J have a high condition number, and method 3 is overfitted and solved using a least squares method.

The velocity is shown in Figure 3.4 to converge at a rate of 3. The centre to node and corner to node converge at a rate of around 2, or 1.5 on the edges. The SPR exhibits superconvergence, with a convergence rate of 3. For the internal nodes, a rate of 3.5 is observed. This is lower than the rate of 4 observed by Zienkiewicz and Zhu, but is still faster than the rate of velocity convergence and as such can be termed ultraconvergence. It is not entirely clear why my results do not replicate the rate of convergence that Zienkiewicz and Zhu acheive. It may be a consequence of the use of a different benchmark, and the relative simplicity of the benchmark they used compared to the benchmark of Donea and Huerta implemented here.



(a) e_{xx} (b) e_{xy}

Figure 3.2: Stress errors for two methods of the Superconvergent Patch Recovery Method.

Table 3.2.1: Top left corner tractions.

Method	Convergence Rate					
	L_2		Interal		Edge	
	e_{xx}	e_{xy}	e_{xx}	e_{xy}	e_{xx}	e_{xy}
Centre to Node	1.90	1.86	1.97	1.95	1.46	1.46
Corner to Node	1.89	1.96	1.98	2.00	1.46	1.46
Superconvergent Patch Recovery	3.07	2.88	3.51	3.46	2.99	2.99

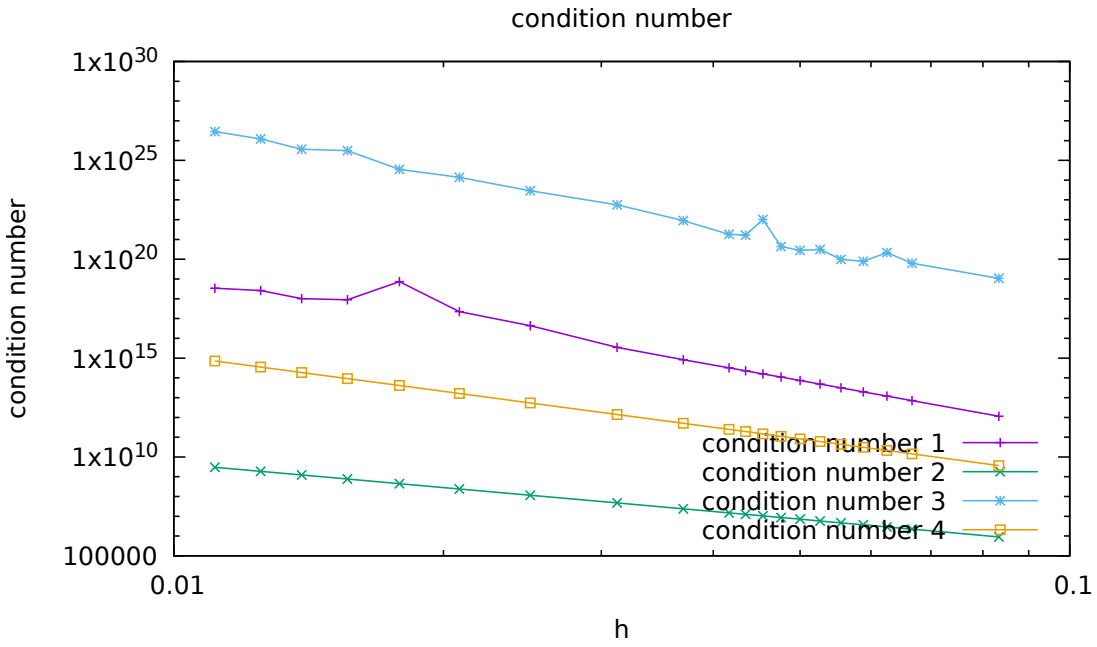


Figure 3.3: Mean condition number of the left hand side matrix for each of the methods of the Superconvergent Patch Recovery method, as a function of grid spacing, for a Q_2Q_1 regular mesh.

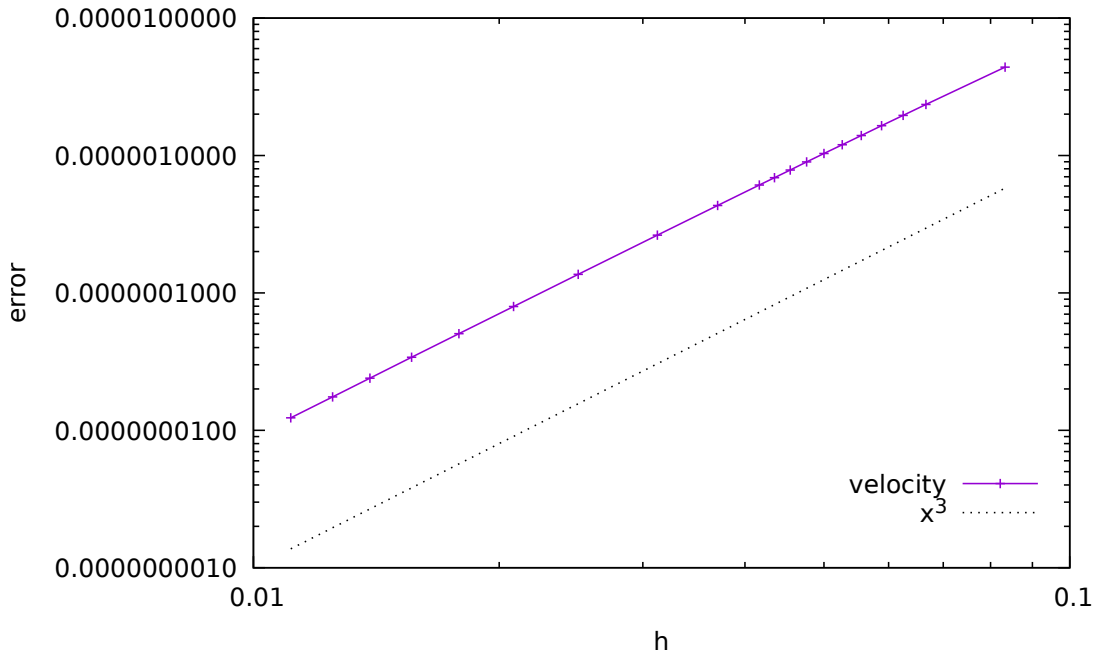
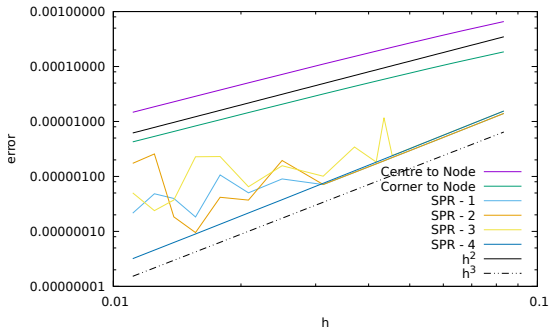
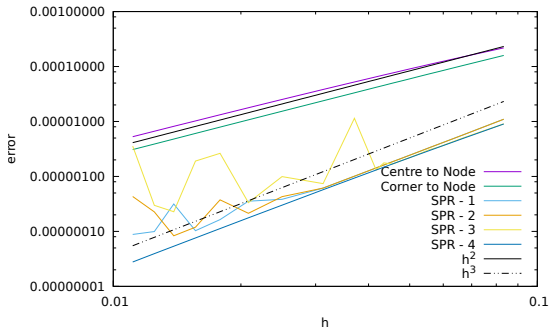


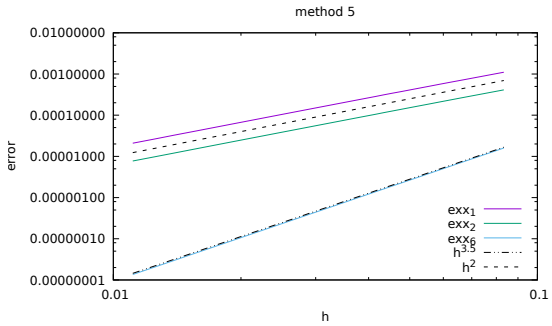
Figure 3.4: Convergence of the velocity for a Q_2Q_1 regular mesh.



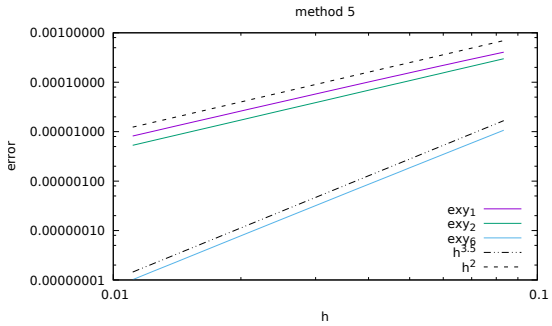
(a) L_2 errors e_{xx}



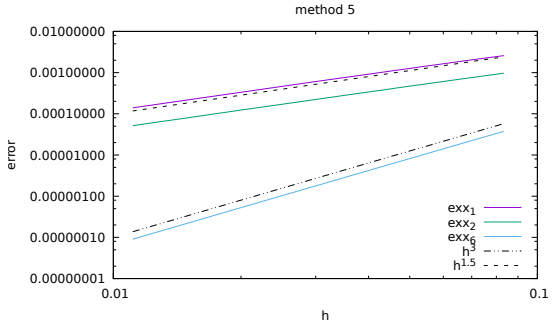
(b) L_2 errors e_{xy}



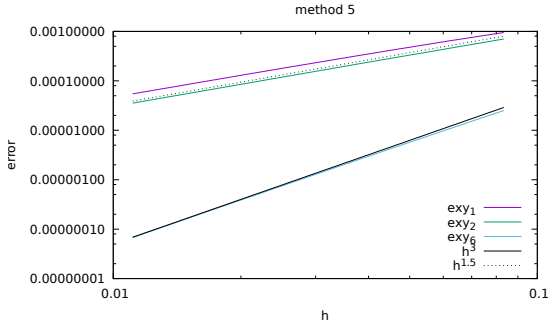
(c) Nodal internal errors e_{xx}



(d) Nodal internal errors e_{xy}



(e) Nodal edge errors e_{xx}



(f) Nodal edge errors e_{xy}

Figure 3.5: Convergence of several norms as a function of grid resolution for the Q_2Q_1 regular mesh.

Table 3.2.2: Top left corner tractions.

Method	Convergence Rate					
	L_2		Interal		Edge	
	e_{xx}	e_{xy}	e_{xx}	e_{xy}	e_{xx}	e_{xy}
Centre to Node	1.30	0.96	1.44	1.10	0.92	0.92
Corner to Node	0.93	0.93	1.05	1.01	0.53	0.53
Superconvergent Patch Recovery	0.91	0.86	0.99	0.95	0.47	0.47

Randomised Grid

As is shown in Figure 3.6, in this instance the convergence rate of the velocity is now 2. As Table 3.2.2 shows, no method converges as well as in the regular grid case, and no method is superconvergent. The best convergence is obtained by the centre to node, which converges at rates close to 1.5.

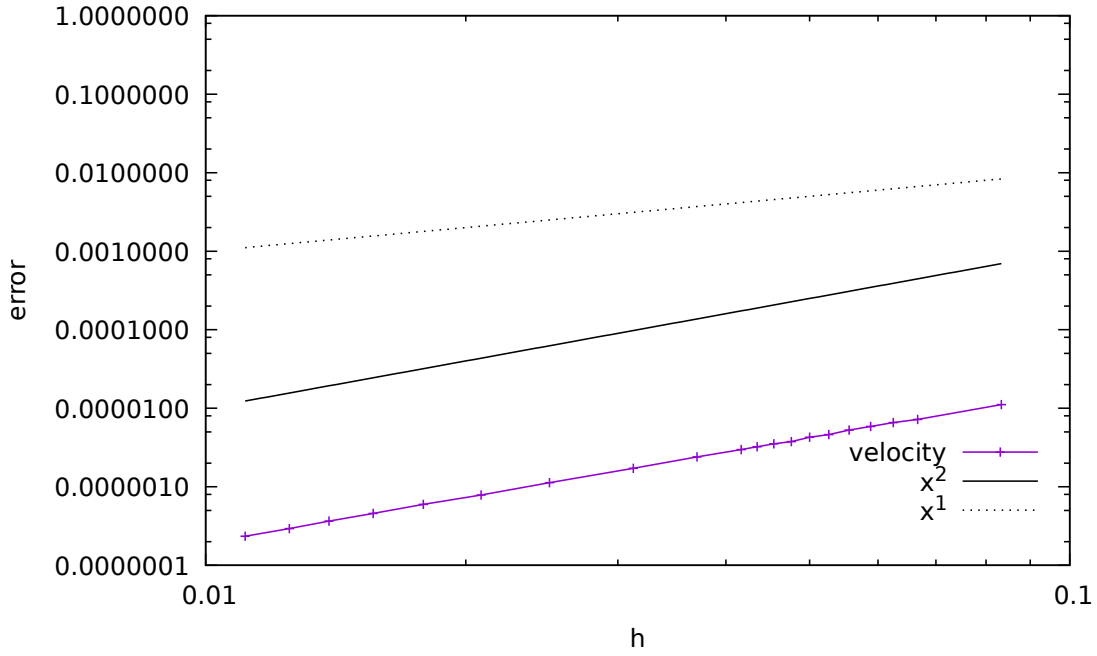
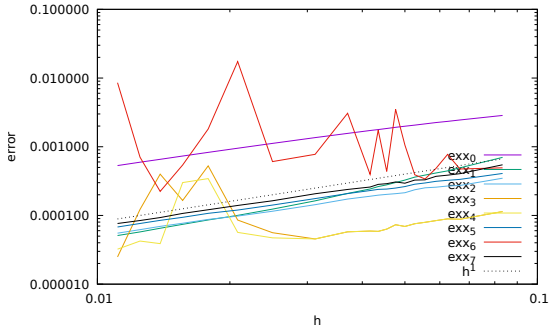
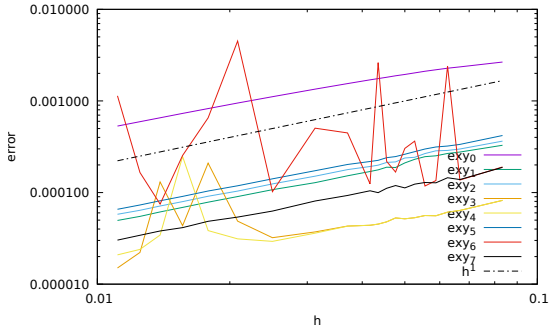


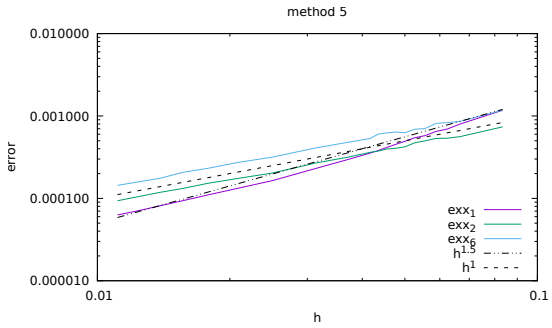
Figure 3.6: Convergence of the velocity for a Q_2Q_1 regular mesh.



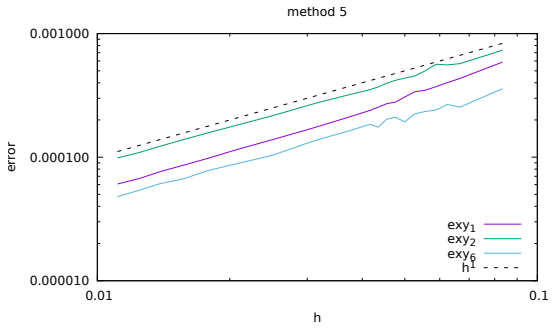
(a) L_2 errors e_{xx}



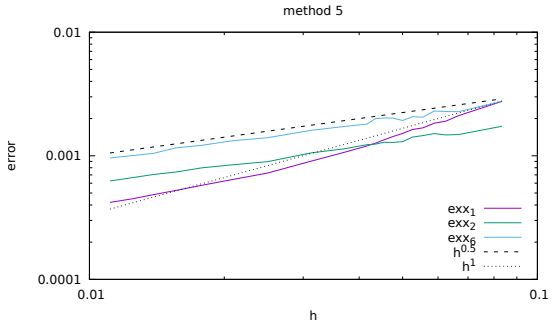
(b) L_2 errors e_{xy}



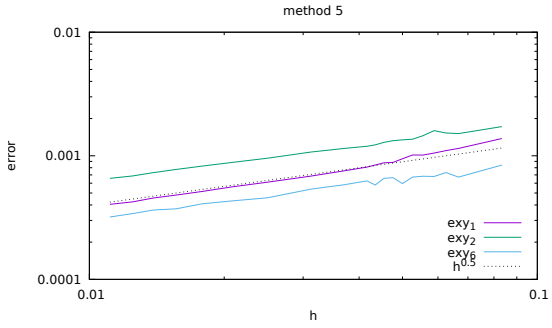
(c) Nodal internal errors e_{xx}



(d) Nodal internal errors e_{xy}



(e) Nodal edge errors e_{xx}



(f) Nodal edge errors e_{xy}

Chapter 4

Conclusion

The Consistent Boundary Flux method of Zhong et al [13] was replicated, and compared to other methods of computing stresses on the surfaces of the domain for Q_1P_0 elements. The results of their benchmark (with free-slip boundary conditions) were replicated, and convergence analysis of the results was carried out. It was then extended to the benchmark of Donea and Huerta [2], with no-slip boundary conditions. On both benchmarks it outperformed all other methods. It was also extended to an annulus benchmark, where it again outperformed other methods. It was further extended to Q_2Q_1 and Q_3Q_2 elements for the benchmark of Donea and Huerta.

The Consistent Boundary Flux method was then extended to the calculation of surface temperature flux, and tested against a range of benchmarks. It outperformed an elemental integration method of flux calculation for all but the simplest benchmarks.

Several methods of calculating strain on internal nodes were implemented on Q_2Q_1 elements, and tested against the benchmark of Donea and Huerta [2]. For regular and semi-randomised grid the superconvergent patch recovery method of Zienkiewicz and Zhu [14] [15] was found to be the most accurate, whereas for a randomised grid a centre-to-node method was found to be the best. However, Ilinca and Pelletier [6] [7] report that their finite nodal displacement mechanism outperforms the superconvergent patch recovery method, although attempts to replicate this method for this thesis were unsuccessful.

Bibliography

[1] W. Bangerth, J. Dannberg, R. Gassmöller, and T. Heister. Aspect v2.0.1 [software], 2018.

[2] Jean Donea and Antonio Huerta. *Finite Element Methods for Flow Problems*. John Wiley & Sons, 2003.

[3] Thibault Duretz, Alban Souche, Rene De Borst, and Laetitia Le Pourhiet. The benefits of using a consistent tangent operator for viscoelastoplastic computations in geodynamics. *Geochemistry, Geophysics, Geosystems*, 19(12):4904–4924, 2018.

[4] BH Hager and MA Richards. Long-wavelength variations in earth's geoid: physical models and dynamical implications. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 328(1599):309–327, 1989.

[5] AM Hofmeister and RE Criss. Earth's heat flux revised and linked to chemistry. *Tectonophysics*, 395(3-4):159–177, 2005.

[6] Florin Ilinca and Dominique Pelletier. Computation of accurate nodal derivatives of finite element solutions: The finite node displacement method. *International journal for numerical methods in engineering*, 71(10):1181–1207, 2007.

[7] Florin Ilinca and Dominique Pelletier. Nodal velocity derivatives of finite element solutions: The find method for incompressible navier–stokes equations. *International journal for numerical methods in engineering*, 76(4):455–481, 2008.

[8] S.D. King. Conman v2.0.0 [software], 2008.

[9] Thorne Lay, John Hernlund, and Bruce A Buffett. Core–mantle boundary heat flow. *Nature geoscience*, 1(1):25, 2008.

[10] L. Moresi, S. Zhong, M. Gurnis, L. Armendariz, E. Tan, and T. Becker. Citcomcu v1.0.3 [software], 2009.

[11] C. Thieulot. Fieldstone, 2019.

[12] S. Zhang and U. Christensen. Some effects of lateral viscosity variations on geoid and surface velocities induced by density anomalies in the mantle. *Geophy. J. Int.*, 114:531–547, 1993.

[13] S. Zhong, M. Gurnis, and G. Hulbert. Accurate determination of surface normal stress in viscous flow from a consistent boundary flux method. *Phys. Earth. Planet. Inter.*, 78:1–8, 1993.

[14] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992.

[15] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992.