CHALMERS
UNIVERSITY OF TECHNOLOGY

Implementation and Evaluation of a New Numerical Scheme for Diffusive Fluxes using Finite Volume Method

Master's thesis in Applied Mechanics

SRIKANTH CHOWDADENAHALLI SATHYANARAYANA

Department of Applied Mechanics
Division of Fluid Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:82

Implementation and Evaluation of a New Numerical Scheme for Diffusive Fluxes using Finite Volume Method

SRIKANTH CHOWDADENAHALLI SATHYANARAYANA

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2016

Implementation and Evaluation of a New Numerical Scheme for Diffusive Fluxes using Finite Volume Method

Supervisor and Examiner: Niklas Andersson, Applied Mechanics

Master's Thesis 2016:82
ISSN 1652-8557
Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2016

Abstract

Grid distortion is usually one of the major problems in the field of Computational Fluid dynamics (CFD). The quality of the results obtained from the numerically modeled simulations hugely depends on grid quality. There are many instances in modeling a fluid dynamics problem where grid distortion is inevitable. When this occurs the corresponding Numerical Scheme used to approximate the fluxes must account for these opposing changes.

A new numerical scheme called the Preferred Direction Diffusion Scheme for the calculation of diffusive fluxes is presented in this thesis. The numerical scheme proposed here is less sensitive to grid quality, therefore, the transformation of the grids is expected to be more accurate compared to the traditional transformation techniques like central differencing.

Initially, the scheme is implemented in MATLAB to study its Mathematical behavior. It is further applied to an Unsteady heat Conduction equation in 3D. It is tested on a simple case-a Square duct with adverse grid conditions. The obtained results are compared with the results obtained from an existing numerical scheme with conventional transformation technique. Later, both the results are compared to an analytical solution and conclusions are drawn based on that.

The implemented scheme is further evaluated for its robustness and accuracy through appropriate code verification techniques. Code verification usually involves error evaluation of the numerical schemes for known benchmark results. The obvious choice for a benchmark solution is the analytical solution but it's usually impossible to obtain them with a sufficient solution structure. The method of Manufactured solutions (MMS) plays a useful role in this situation. This method is fairly straightforward and purely a mathematical procedure.

MMS is implemented in CALC++ for an unsteady heat equation in 3D and Navier-stokes equation. CALC++ is a massively parallel incompressible flow solver (written in C++) developed at the division of Fluid Dynamics, Chalmers University of Technology. The scheme is evaluated by subjecting it to systematic grid tests. Its performance is judged by studying the rate of reduction of the discretization error with increasing grid resolution.

Keywords: CFD, Numerical scheme, Diffusive fluxes, PDS, CD, MMS, CALC++, MATLAB .

Acknowledgements

Firstly, I would like to thank my parents and brother for supporting me in every way possible during my stay in Sweden. I would like to thank my supervisor Niklas Andersson for giving me an opportunity to work on this thesis. My understanding of numerical methods for fluid dynamics and programming vastly improved during the past year through this thesis. I am very grateful to him for that. I would also like to thank Ragnar Lárusson and Huadong Yao for helping me out with CALC++. Finally, I want to thank all my friends here in Gothenburg and back home.

Srikanth, Gothenburg, November 2016

List of Symbols

T	temperature
t	time
α	Thermal diffusivity
(r, Θ)	radial coordinates
V	volume
$\overline{T}$	cell averaged temperature
A	cell face area
(x_1, x_2, x_3)	Cartesian coordinates
(ξ, η, ζ)	Computational coordinates
Δt	time step
$\mathcal{V}$	cell properties
Ψ	Inverse components of preferred direction
φ	Flow variable
ρ	density
ν	kinematic viscocity
P	Pressure
Q	unsteady term vector
F	Flux vector

Abbreviations

CFD	Computational Fluid Dynamics
PDS	Preferred direction Diffusion Scheme
CD	Central Differencing
MMS	Method of Manufacured solutions
OOA	Observed order of accuracy

Contents

1
Introduction

The application of finite volume method for a domain in curvilinear coordinates usually involves a transformation from the physical space to Computational space. This Process itself is an approximation and involves a one to one mapping between the two coordinate systems (1.1).

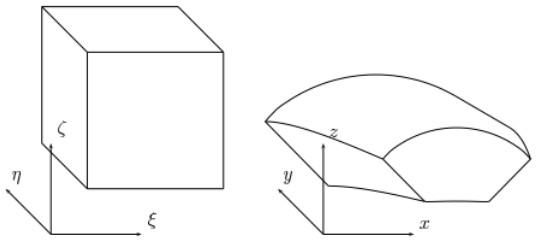


Figure 1.1: Representation of Physical and computational space

The curvilinear grid will, therefore, be transformed into a uniform grid, thereby, making it easier to solve the finite volume equations in the physical space. If the computational domain is discretized in space using a grid with unfavourable features such as non-orthogonality, skewness,etc., the applied numerical scheme must account for these adverse conditions failing to do so will result in inaccurate solutions.

1.1 New Numerical Scheme

A new numerical scheme for the calculation of diffusive fluxes which is based on detailed metrics and reconstructed flow quantities is used in this project. The proposed numerical scheme is aimed to establish a new method to transform complex boundary-fitted grids from physical space to computational space at a higher accuracy. This scheme will account for the above-mentioned conditions, therefore, it is expected to be less sensitive to mesh quality which would affect the diffusive flux calculations. In addition, this new scheme incorporates a higher order transformation technique but still follows second order accuracy for spatial discretization like the central differencing scheme. This finite volume scheme is based on cell centered, structured grids with the gradients being solved at the face centers, see figure (3.1). This scheme could be, for example, useful in the process of shape optimization of a turbine blade during which a mesh topology is initially defined for the blade which results in the generation of a boundary fitted grid. Based on the given design constraints in each iteration of the procedure, reshaping of the blade takes place continuously with constant mesh topology. This leads to a continuous change in the mesh configuration which might result in the production of unfavorable cells in the

domain. Also, in general, CFD problems where grid skewness is predominant, the new numerical scheme is presumed to account for it, therefore, the results obtained could be more accurate in these circumstances.

1.2 Mesh Quality

During the calculation of diffusive fluxes some of the main mesh quality parameters considered are cell aspect ratio, skewness, stretching and curvilinear cells [3]. The diffusive gradient calculations are hugely affected by cell skewness due to non-orthogonality. Therefore special treatments are to be conducted on the gradient terms, this aspect is elucidated in [3]. The major mesh parameters studied in this thesis are cell skewness and curvilinearity (1.2).

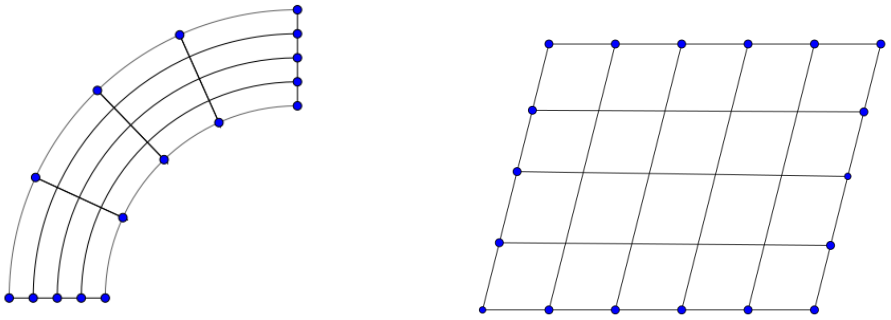


Figure 1.2: Curvilinear and skewed grids

1.3 Code Verification

A numerical scheme is usually tested for robustness and accuracy once it is implemented in the code. This process is roughly known as code verification. In any numerical assessment, three important processes are required to be understood and conducted in order to better establish the accuracy of the obtained solutions. These processes are verification of codes, verification of calculations and validation [4]. Verification of calculation involves error estimation and verification of code involved error evaluation from a known solution. Both these procedures are purely mathematical. The process of validation involves corroborating physical laws with the obtained numerical results.
Code Verification is performed to make sure there are no mistakes in a CFD code and the solution algorithm [6]. It is, therefore, useful to check if the implemented numerical method captures a physical phenomenon accurately. For a numerical scheme to be stable, the errors must not develop in the direction of solution calculation and the discretized equations must approach the original governing equations as the spatial and time limits tend to zero. This difference between the discretized equations and the original partial differential equations is called discretization error.

$$DE_i = f_i - f_{exact} \tag{1.1}$$

Here i represents the mesh level. In this thesis, only discretization errors are studied and the errors affecting it, e.g. round-off and iterative convergence errors, are ignored.

1.3.1 Order of accuracy (OOA)

One of the meticulous code verification tests is the order of accuracy test, which basically determines if the discretization error is reduced at an expected rate with increasing grid refinement. This test is, therefore, useful to check if the formal order of accuracy of the numerical scheme is attained in the obtained solutions. The formal order of accuracy is usually established through the truncation error of the numerical scheme used. For example, a central differencing scheme is formally second order accurate, which can be derived using the Taylor series analysis. On the other hand, the observed order of accuracy is obtained through the output of the code from a simulation. This accuracy is very sensitive to coding mistakes, smoothness of the solution, incorrect numerical algorithms, and solutions which are not in asymptotic convergence range, which basically means that the lower order truncation error terms dominate [6]. Consider a series expansion of the discretization error in terms of h_i, which is a measure of an element size on some mesh level i,

$$DE_i = f_i - f_{exact} = g_p h_i^p + H.O.T \tag{1.2}$$

Here, p, g_p, h are the observed order of accuracy, coefficient of the leading order term, reference grid spacing respectively. Using the assumption that the $H.O.T$ are negligible and equation (1.2), two discretization errors for two mesh levels can be represented as,

$$\begin{aligned} DE_1 &= f_1 - f_{exact} = g_p h_1^p \\ DE_2 &= f_2 - f_{exact} = g_p h_2^p \end{aligned} \tag{1.3}$$

Combining the two equations,

$$\frac{DE_2}{DE_1} = \frac{g_p h_2^p}{g_p h_1^p} = \Big(\frac{h_2}{h_1}\Big)^p \tag{1.4}$$

Taking natural log on both sides, the observed order of accuracy is obtained as,

$$p = \frac{ln\Big(\frac{DE_2}{DE_1}\Big)}{ln(r)} \tag{1.5}$$

where r is grid refinement factor given by, $r = h_2/h_1$. When an exact solution is known, the observed order of accuracy of the numerical scheme can be obtained. When using this method, special care has to be taken to make sure the iterative and round-off error don't affect the calculations. This can be achieved by setting very high tolerance levels during the simulations.

2

Governing Equations and Solution Methods

2.1 Governing Equations

This section provides a detailed description of the governing equations used in this project. The Continuity and momentum equations for incompressible, laminar, 3D flow in the absence of body forces are presented below.

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\nu \frac{\partial u_i}{\partial x_j}) \tag{2.2}$$

Further, the unsteady heat equation in 3D is given below,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x_j^2} \tag{2.3}$$

Here Thermal Diffusivity α is a constant and is set to $1.9 \times 10^{-}5\ \ m^2/s$.The equation used in obtaining the analytical solution is the 2D unsteady conduction equation in cylindrical coordinate system,

$$\frac{\partial T}{\partial t} = \alpha (\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \Theta^2}) \tag{2.4}$$

2.2 Numerical Method

The governing equations can be in general represented as,

$$\frac{\partial Q}{\partial t} + \frac{\partial F_j}{\partial x_j} = 0 \tag{2.5}$$

For incompressible flow equations,

$$Q = \begin{bmatrix} 0 \\ u_i \end{bmatrix}, F_j = \begin{bmatrix} u_j \\ u_i u_j + \frac{1}{\rho} p \delta_{ij} - \nu \frac{\partial u_i}{\partial x_j} \end{bmatrix} \tag{2.6}$$

For unsteady heat equation,

$$Q = \begin{bmatrix} T \end{bmatrix}, F_j = \begin{bmatrix} -\alpha \frac{\partial T}{\partial x_j} \end{bmatrix} \tag{2.7}$$

2.2.1 Finite Volume Method

Equation (2.5) is integrated over an arbitrary Control Volume V,

$$\int_V \frac{\partial Q}{\partial t}\, dV + \int_V \frac{\partial F_j}{\partial x_j}\, dV = 0 \tag{2.8}$$

Further Gauss theorem is applied to the flux part thereby solving it over the Control surface and cell average $\overline{Q}$ is used to represent the unsteady term.

$$\frac{\partial \overline{Q}}{\partial t} V + \int_S F_j . \, dA_j = 0 \tag{2.9}$$

The surface integral in equation (2.9) can be further approximated using the areas of the control surfaces and the average face flux.

$$\frac{\partial \overline{Q}}{\partial t} V + \sum_{i=1}^{faces} F_j^i . \, A_j^i = 0 \qquad A_j = n_j A \tag{2.10}$$

The following sections will describe how to discretize the unsteady and flux terms in equation (2.10).

2.2.2 Spatial Discretization

2.2.2.1 Convective Fluxes

The convective fluxes in CALC++ environment are solved using second order upwind scheme.

2.2.2.2 Diffusive Fluxes

The diffusive fluxes in both the MATLAB implementation and CALC++ environment are calculated using Preferred direction diffusion scheme (PDS). Further, an implementation of Centered Difference method was done in the MATLAB script to compare with the results obtained from PDS. A detailed explanation of PDS is provided in chapter 3. This section gives a brief description of the Centered difference scheme. The diffusion gradients are obtained in the computational domain and then converted into physical space using chain rule relation given by,

$$\frac{\partial \varphi}{\partial x_i} = \frac{\partial \xi}{\partial x_i}\frac{\partial \varphi}{\partial \xi} + \frac{\partial \eta}{\partial x_i}\frac{\partial \varphi}{\partial \eta} + \frac{\partial \zeta}{\partial x_i}\frac{\partial \varphi}{\partial \zeta} \qquad i = 1, 2, 3 \tag{2.11}$$

The derivatives of the flow variables are further obtained using the centered difference approach (also see figure 3.1).

$$\begin{aligned} \frac{\partial \varphi}{\partial \xi} &= \overline{\varphi}_{i+1,j,k} - \overline{\varphi}_{i,j,k} \\ \frac{\partial \varphi}{\partial \eta} &= \frac{1}{4}(\overline{\varphi}_{i,j+1,k} - \overline{\varphi}_{i,j-1,k}) + \frac{1}{4}(\overline{\varphi}_{i+1,j+1,k} - \overline{\varphi}_{i+1,j-1,k}) \\ \frac{\partial \varphi}{\partial \zeta} &= \frac{1}{4}(\overline{\varphi}_{i,j,k+1} - \overline{\varphi}_{i,j,k-1}) + \frac{1}{4}(\overline{\varphi}_{i+1,j,k+1} - \overline{\varphi}_{i+1,j,k}) \end{aligned} \tag{2.12}$$

Further, the derivatives of physical space with respect to computational space is obtained using the Transformation matrix. This process is further clearly explained in [1].

2.2.3 Temporal Discretization

A detailed description of the temporal discretization of the heat equation implemented in the MATLAB script is provided in this section. The equation (2.10) is further integrated over time t,

$$\int_t \frac{\partial \overline{\varphi}}{\partial t} V \, dt = \alpha \int_V \sum_{i=1}^{faces} \frac{\partial \varphi}{\partial x_j}^i A_j^i \, dt \tag{2.13}$$

The equation is averaged over time,

$$(\overline{\varphi}^{n+1} - \overline{\varphi}^n)V = \alpha \int_V \sum_{i=1}^{faces} \frac{\partial \varphi}{\partial x_j}^i A_j^i \, dt \tag{2.14}$$

Further first order Euler explicit method which is a forward time marching scheme is used as the time scheme,

$$\overline{\varphi}^{n+1} = \overline{\varphi}^n + \frac{\alpha \Delta t}{V} \sum_{i=1}^{faces} (\frac{\partial \varphi}{\partial x_j}^i)^n A_j^i \tag{2.15}$$

The time step Δt is set to 0.2 sec during the simulations. During the CALC++ simulations a time step of 10^{-5} is used.

2.2.4 Solving Method

For the MATLAB implementation, the equation (2.15) after its reduction to algebraic form is solved using Gauss-Seidel Iteration method. The Tolerance level of the iterations is set to $1 \times 10^{-}3$. Further, the CALC++ enviroment utilizes solvers available in PETSc library.

2.3 Analytical Method

The equation (2.4) is solved analytically using the separation of the variables technique of solving Partial Differential Equations. After applying the appropriate boundary and initial conditions, temperature T is obtained as a function in space and time as,

$$T(r, \Theta, t) = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} C_{nm} sin(\nu_n \Theta) e^{(-\alpha \lambda_{nm}^2 t)} [Y_\nu(\lambda_{nm} a) J_\nu(\lambda_{nm} r) - J_\nu(\lambda_{nm} a) Y_\nu(\lambda_{nm} r)] \tag{2.16}$$

Since the governing equation is a parabolic equation it will tend to become elliptic with increasing time (become steady). This would be computationally expensive to achieve. Therefore, eigenvalues comparable to the solution matrix for a particular grid size is obtained and the solutions are obtained for smaller time periods. Also, the fact that the unsteady term is exponential decaying could be exploited in order to obtain a stable analytical solution. A comprehensive mathematical description of the derivation is provided in [2]. A detailed derivation of the equation for the presented case is provided in Appendix B.

2.4 Initial and boundary conditions

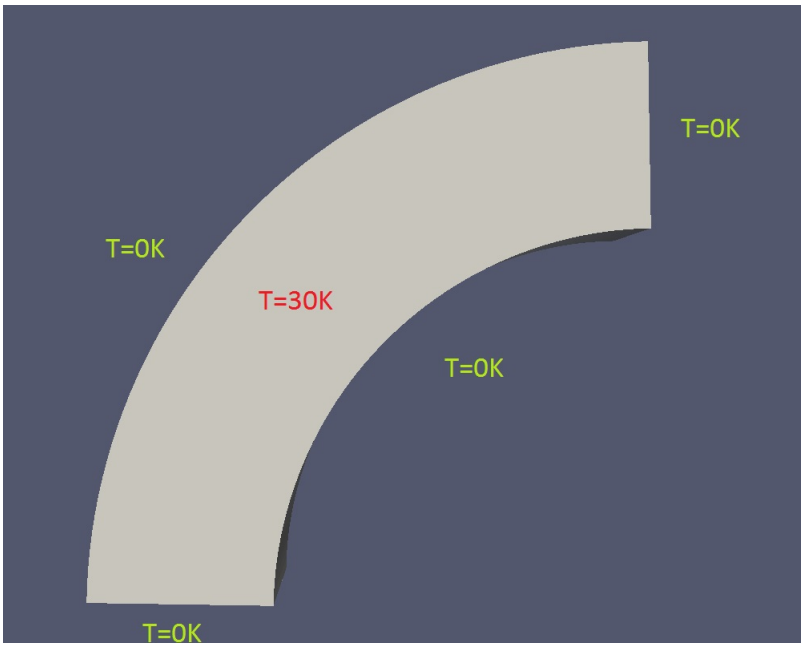


Figure 2.1: Initial and Boundary conditions

For the MATLAB implementation, the initial temperature in the domain is set to a uniform value of $30K$. For the numerical analysis, the boundary conditions are prescribed by creating ghost cells and further implementing it using Central differencing. The temperature values used for the boundary conditions are shown in figure (2.1). The boundary conditions on the other two faces in the direction normal to the plane were set to zero gradient.

3

Preferred Direction Diffusion Scheme

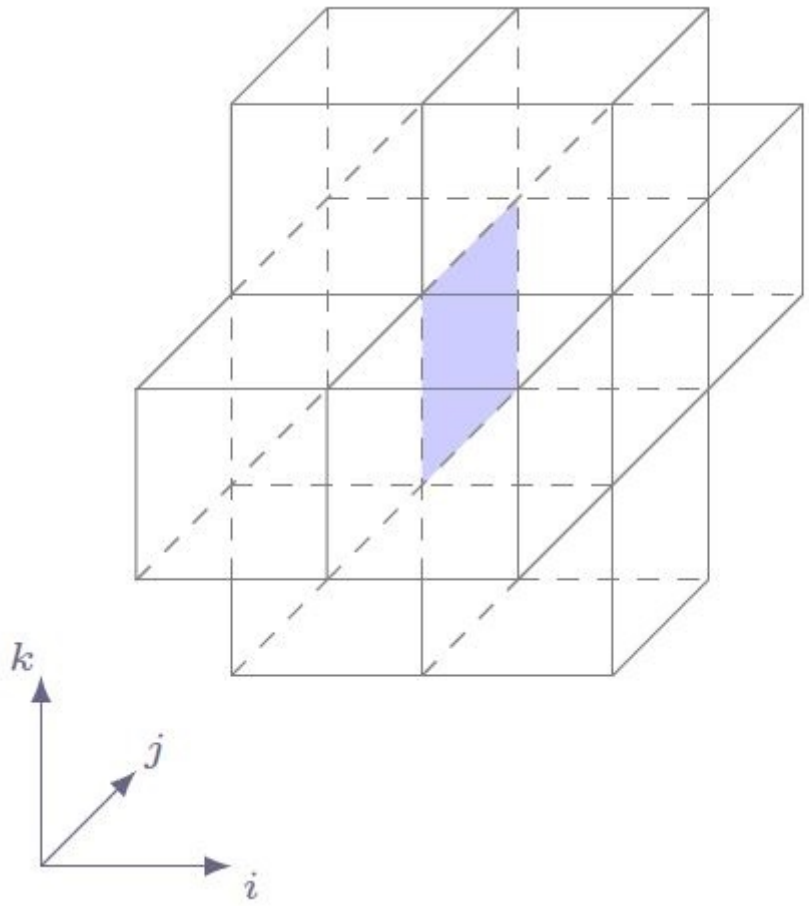


Figure 3.1: Representation of Flux Molecule in Computational Space

In this new numerical scheme, the diffusion gradients on the faces are evaluated using the cell averages obtained from the surrounding ten cells as indicated in figure (3.1). Note that, since the faces near domain boundaries are not surrounded by ten cells, they have to be treated differently. Twenty cell properties including volume of the cell are calculated for each of the considered ten cells per face in the physical space (A). These volume integrals are calculated using Gauss-point Quadrature (A).

3.1 Preferred Direction

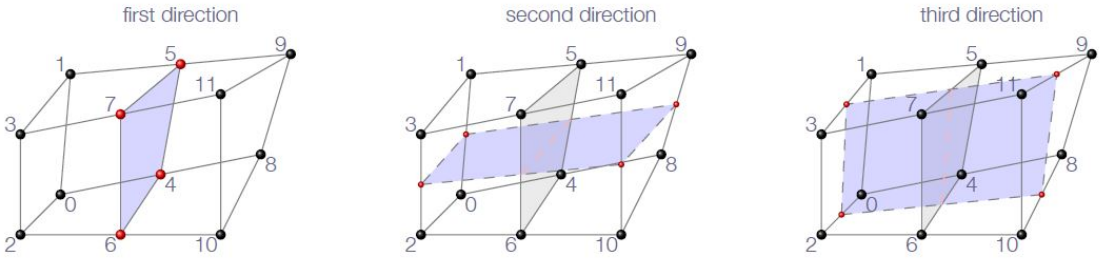


Figure 3.2: Preferred Directions of a face

In order to calculate the gradient coefficients for each face, three unique directions for each face inside the domain is calculated. Three planes are defined using the grid nodes. These directions are referred to as preferred directions, see figure (3.2)

The preferred directions for each face of the flux molecule are calculated using the following relations,

$$\begin{aligned} \text{First direction,} \qquad & n1 = \frac{(r_7-r_4)\times(r_5-r_6)}{|(r_7-r_4)\times(r_5-r_6)|} \\ \text{Second direction,} \qquad & n2 = \frac{(r_{10}-r_0)\times(r_8-r_2)+(r_{11}-r_1)\times(r_9-r_3)}{|(r_{10}-r_0)\times(r_8-r_2)+(r_{11}-r_1)\times(r_9-r_3)|} \\ \text{Third direction,} \qquad & n3 = \frac{(r_9-r_0)\times(r_8-r_1)+(r_{11}-r_2)\times(r_{10}-r_3)}{|(r_9-r_0)\times(r_8-r_1)+(r_{11}-r_2)\times(r_{10}-r_3)|} \end{aligned} \tag{3.1}$$

The directions $n1$, $n2$ and $n3$ are not in general normal to each other.

3.2 Local Co-ordinates

A local co-ordinate system is now established for each face using the directions obtained in equation (3.1). This co-ordinate system relates the global co-ordinate system, the preferred directions and the reference point (x_o, y_o, z_o) located at the face center.

$$\begin{gathered} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} + \xi \begin{bmatrix} n1_x \\ n1_y \\ n1_z \end{bmatrix} + \eta \begin{bmatrix} n2_x \\ n2_y \\ n2_z \end{bmatrix} + \zeta \begin{bmatrix} n3_x \\ n3_y \\ n3_z \end{bmatrix} \\ \begin{bmatrix} n1_x & n2_x & n3_x \\ n1_y & n2_y & n3_y \\ n1_z & n2_z & n3_z \end{bmatrix} \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = N \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = \begin{bmatrix} x - x_o \\ y - y_o \\ z - z_0 \end{bmatrix} \\ \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = N^{-1} \begin{bmatrix} x - x_o \\ y - y_o \\ z - z_0 \end{bmatrix} \\ \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} & \Psi_{13} \\ \Psi_{21} & \Psi_{22} & \Psi_{23} \\ \Psi_{31} & \Psi_{32} & \Psi_{33} \end{bmatrix} \begin{bmatrix} x - x_o \\ y - y_o \\ z - z_0 \end{bmatrix} \end{gathered} \tag{3.2}$$

3.3 Extraction of Gradient Coefficients

The variation of the flow variable φ in the computational space, (ξ, η, ζ) is assumed to be,

$$\varphi(\xi,\eta,\zeta) = C_0 + C_1\xi + C_2\eta + C_3\zeta + C_4\xi\eta + C_5\xi\zeta + C_6\eta^2 + C_7\zeta^2 + C_8\xi\eta^2 + C_9\xi\zeta^2 \quad (3.3)$$

The unknown coefficients $[C_0....C_9]$ are obtained by solving a system of equations obtained by integrating the equation (3.3) over the ten cell volumes of a Flux molecule (see figure 3.1).

$$\begin{bmatrix}\overline{\varphi}_0 V_0 & . & . & . & \overline{\varphi}_9 V_9\end{bmatrix} = \begin{bmatrix}C_0 & . & . & . & C_9\end{bmatrix} \begin{bmatrix} \int\int\int_{\Omega_0} d\Omega & \cdots & \int\int\int_{\Omega_0} \xi\zeta^2 d\,\Omega \\ . & & . \\ . & & . \\ . & & . \\ \int\int\int_{\Omega_9} d\Omega & \cdots & \int\int\int_{\Omega_9} \xi\zeta^2 d\,\Omega \end{bmatrix} \quad (3.4)$$

Or

$$\Phi = \begin{bmatrix}C_0 & . & . & . & C_9\end{bmatrix} A$$

In equation (3.4), Φ represents the degrees of freedom in the CFD solver (cell-averages of the considered solver variables). Since the vector Φ is known the matrix A can be calculated from the known metric data obtained in equations (A.1 and 3.2). Thus, the vector Φ can be represented as,

$$\begin{bmatrix}C_0 & . & . & . & C_9\end{bmatrix} = \Phi A^{-1} \quad (3.5)$$

A detailed derivation of the integrals in the A matrix is provided in Appendix A. Also, the procedure for obtaining the integrals in the domain boundaries is provided.

3.4 Face Gradients

Further, differentiating equation (3.3),

$$\begin{aligned} d\varphi = C_1 d\xi + C_2 d\eta + C_3 d\zeta + C_4\xi d\eta + C_4\eta d\xi + C_5\xi d\zeta + C_5\zeta d\xi + \\ 2C_6\eta d\eta + 2C_7\zeta d\zeta + C_8\xi\eta d\eta + C_8\eta^2 d\xi + 2C_9\xi\zeta d\zeta + C_9\zeta^2 d\xi \end{aligned} \quad (3.6)$$

Since the local coordinate system is centered at the face center, equation (3.6) results in,

$$\begin{aligned} (\xi,\eta,\zeta) &= (0,0,0) \\ d\varphi &= C_1 d\xi + C_2 d\eta + C_3 d\zeta \end{aligned} \quad (3.7)$$

By Differentiating the second equation in (3.2), the relationship between the derivatives in (x,y,z) space and (ξ,η,ζ) space is obtained as,

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} n1_x & n2_x & n3_x \\ n1_y & n2_y & n3_y \\ n1_z & n2_z & n3_z \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} = N \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} \quad (3.8)$$

Finally, the equations in (3.7 and 3.8) are combined to obtain the coefficients required for calculating the derivatives of φ in the Cartesian coordinate system (x,y,z) at the face

center.

$$\begin{aligned}
\begin{bmatrix} Cx_0 \\ . \\ . \\ . \\ Cx_9 \end{bmatrix} &= \begin{bmatrix} (n1_x A_{10}^{-1} + n2_x A_{20}^{-1} + n3_x A_{30}^{-1}) V_0 \\ . \\ . \\ . \\ (n1_x A_{19}^{-1} + n2_x A_{29}^{-1} + n3_x A_{39}^{-1}) V_9 \end{bmatrix} \\
\begin{bmatrix} Cy_0 \\ . \\ . \\ . \\ Cy_9 \end{bmatrix} &= \begin{bmatrix} (n1_y A_{10}^{-1} + n2_y A_{20}^{-1} + n3_y A_{30}^{-1}) V_0 \\ . \\ . \\ . \\ (n1_y A_{19}^{-1} + n2_y A_{29}^{-1} + n3_y A_{39}^{-1}) V_9 \end{bmatrix} \\
\begin{bmatrix} Cz_0 \\ . \\ . \\ . \\ Cz_9 \end{bmatrix} &= \begin{bmatrix} (n1_z A_{10}^{-1} + n2_z A_{20}^{-1} + n3_z A_{30}^{-1}) V_0 \\ . \\ . \\ . \\ (n1_z A_{19}^{-1} + n2_z A_{29}^{-1} + n3_z A_{39}^{-1}) V_9 \end{bmatrix}
\end{aligned} \tag{3.9}$$

$$\frac{\partial \varphi}{\partial x} = \begin{bmatrix} Cx_0 .... Cx_9 \end{bmatrix} \begin{bmatrix} \overline{\varphi_0} \\ . \\ . \\ . \\ \overline{\varphi_9} \end{bmatrix}, \frac{\partial \varphi}{\partial y} = \begin{bmatrix} Cy_0 .... Cy_9 \end{bmatrix} \begin{bmatrix} \overline{\varphi_0} \\ . \\ . \\ . \\ \overline{\varphi_9} \end{bmatrix}, \frac{\partial \varphi}{\partial z} = \begin{bmatrix} Cz_0 .... Cz_9 \end{bmatrix} \begin{bmatrix} \overline{\varphi_0} \\ . \\ . \\ . \\ \overline{\varphi_9} \end{bmatrix}$$

4

Method of Manufactured Solutions (MMS)

Method of Manufactured solutions is one of the most sophisticated code verification methods available in the field of computational science and engineering [4]. The main idea behind this method is to manufacture a solution for a flow variable using a continuous function instead of obtaining an exact solution. This manufactured solution need not be physically realistic since code verification is purely a mathematical procedure. Using the manufactured solution a source term is derived for the governing equation. Now, this governing equation is solved using numerical methods and compared with the existing known manufactured solution. Therefore, this method could also be seen as a problem solved backward.
For example, consider a function,

$$F(a) = 0 \tag{4.1}$$

Now, say $a = u$ in (4.1),

$$F(u) \neq 0 \tag{4.2}$$

It is not necessary for u to be a solution for (4.1), therefore consider (4.2) to be equal to a source term Q,

$$F(u) = Q \tag{4.3}$$

$$F(a) = Q \tag{4.4}$$

Now, $a = u$ is a solution for (4.4). This is the fundamental idea behind the method of manufactured solutions.

4.1 General procedure of MMS

The following steps are used to obtain the observed order of accuracy required to verify the code,

- Determine the governing equations.
- Construct a manufactured solution.
- Use the solution to compute the source term to modify the governing equations.
- Obtain initial and boundary conditions using the manufactured solution.
- Setup a case by obtaining a grid to the domain and define suitable parameters like time step etc.
- Perform simulations for the modified equations on multiple mesh levels.
- Obtain the global discretization error of the numerical solutions.
- Conduct the order of accuracy test using the solution to determine if the observed order of accuracy matches the formal order of accuracy.

An example of MMS using 1D unsteady heat equation is provided in appendix C.

4.2 Manufactured Solutions

To ensure the accuracy of the code verification procedure special care has to be taken to obtain the manufactured solution. Some of them are mentioned below,

- The considered manufactured solution should contain smooth analytic functions which ensure that the obtained solution will match the formal order of accuracy.
- The solution should have a sufficient number of non-trivial derivatives. For example, in momentum equation, if the manufactured solution for velocity is linear and if the diffusion term is solved using a second-order method, it would lead to incorrect predictions of the observed order of accuracy.
- The manufactured solution derivatives should be bounded by a constant to ensure that the solution is not varying strongly in space and time. Also, this ensures that the solution does not contain any singularities.
- The chosen solution should be realistic when pertaining to a particular flow variable. For example, if the physics of the problem demands positive temperature, the obtained manufactured solution must be compatible.

The following are the manufactured solutions used in this thesis,

$$\begin{aligned} T(x,y,z,t) &= e^{4\pi^2 t} cos^2(2\pi x) cos^2(2\pi y) cos^2(2\pi z) \\ u(x,y,z,t) &= ((0.5 cos(\pi x) cos(\pi y) cos(\pi z) cos(\pi t)) + 0.5) \\ v(x,y,z,t) &= ((0.25 sin(\pi x) sin(\pi y) cos(\pi z) cos(\pi t)) + 0.5) \\ w(x,y,z,t) &= ((0.25 sin(\pi x) cos(\pi y) sin(\pi z) cos(\pi t)) + 0.5) \\ p(x,y,z,t) &= ((cos(\pi x) cos(\pi y) cos(\pi z) cos(\pi t)) + 0.5) \end{aligned} \tag{4.5}$$

The velocity solutions are obtained in a way as to satisfy the continuity equation.

4.3 Initial and boundary conditions

Initial solution is obtained using the manufactured solutions in (4.5). A solution for a differential equation can be obtained using different types of boundary conditions. Using this principle, the boundary values are obtained from the manufactured solutions. For example, consider a case where the code obtains a solution using Dirichlet boundary condition on one of the boundary faces. This implementation can be directly performed and compared using the manufactured solutions. If the same solution is required using a Neumann condition on the same face, the values can be directly obtained from the derivatives of the manufactured solutions.

4.4 Discretization error

The normalized global discretization error of the obtained numerical solution and manufactured solution is obtained by employing L_2 norm,

$$L_{2,i} = \Big(\frac{\sum_{n=1}^{N} |f_{i,n} - f_{exact,n}|^2}{N} \Big)^{1/2} \tag{4.6}$$

Here, i is for a particular mesh level.

4.5 Strengths and Limitations of MMS

Some of the strengths of MMS are,

- Most of the coding options can be verified using this method.
- It has the capability to handle nonlinear and coupled equations.
- It can be used to detect mistakes in the solution algorithm.
- It works equally well for finite difference, finite volume and finite element schemes.

Some of the limitations are,

- It is required to change the source code in order to accommodate the changes in the governing equations. The changes include the source term, initial, and boundary condition terms. Therefore, this process cannot be conducted as a black box analysis.
- To test other model options in the code, it's required to change the source term which can be time-consuming.
- The solutions are assumed to be smooth in the domain. Therefore, it is challenging to obtain manufactured solutions in cases involving discontinuities (shock waves, etc.).

5

Geometry and Mesh

5.1 Domain Geometry

For the geometry of the domain, a Square duct is considered throughout the thesis.

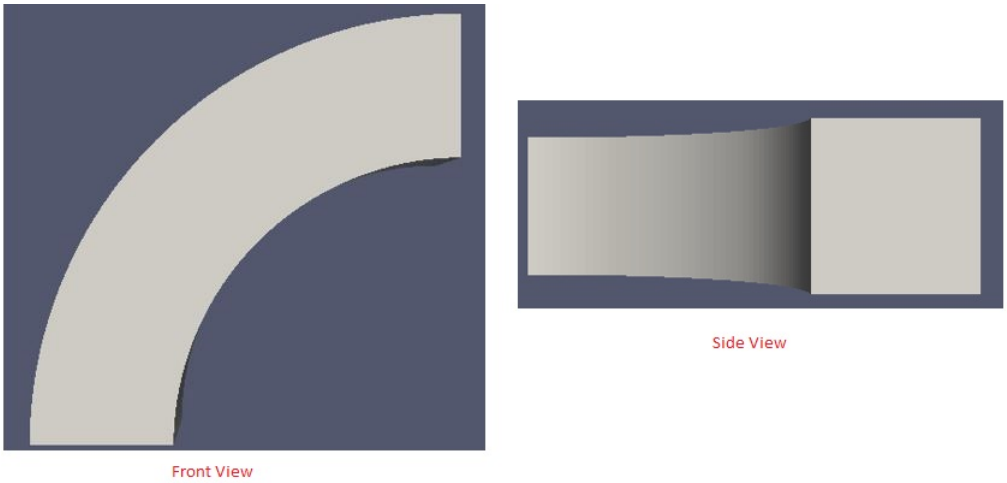


Figure 5.1: Domain Geometry

The dimension of the square is $0.5m \times 0.5m$ and the height of the geometry is $1.5m$.

5.2 Grid Generation

A 3D Structured, Curvilinear grid (figure 5.2) is initially generated using the BlockMesh utility on OpenFOAM and later imported into MATLAB to make certain changes which will explained in the sections below.

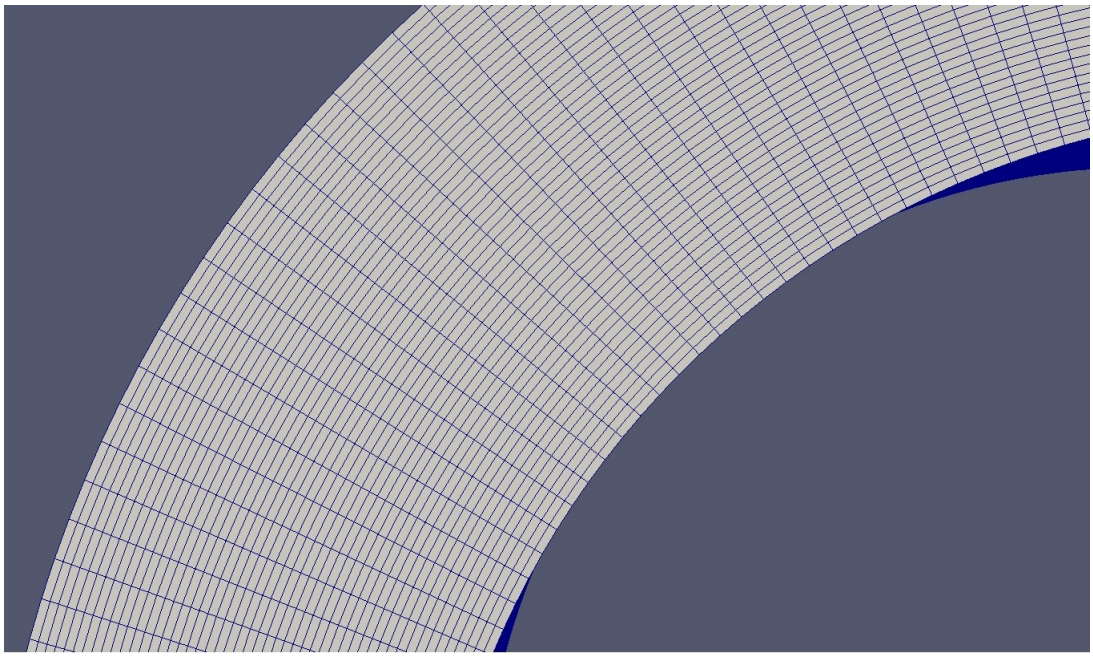


Figure 5.2: Uniform Grid (50x50x50)

5.2.1 Degree of Skewness

The evaluation of the numerical scheme is performed by testing it on several grids which includes some level of skewness.

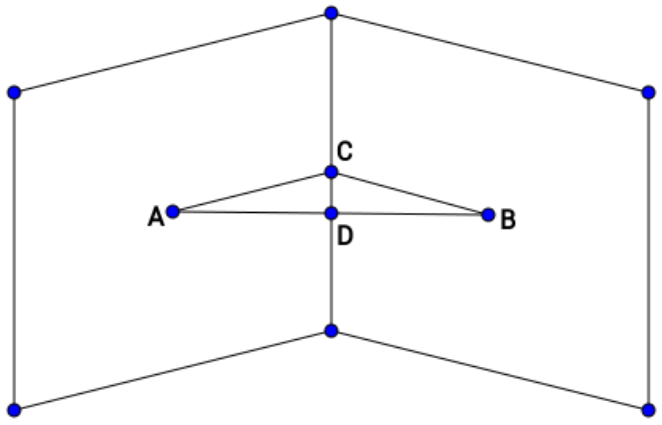


Figure 5.3: Degree of skewness

From figure (5.3), degree of skewness (DOS) can be defined as,

$$DOS = \frac{|CD|}{|AB|} \tag{5.1}$$

Using equation (5.1) the required non-orthogonality is obtained in the domain (see figure 5.4).

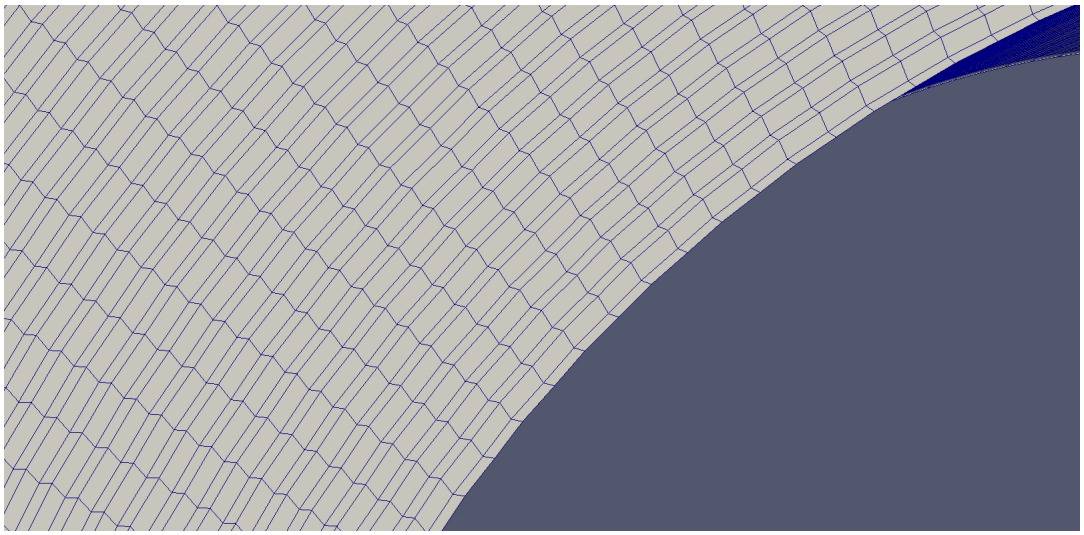


Figure 5.4: Example grid for DOS-0.05 (50x50x50 grid)

5.2.2 Grids

During phase one of the thesis, three grid configurations are used.

Grids for phase one			
Grid size	Uniform	DOS-0.02	DOS-0.05
25x25x25	YES	NO	YES
50x50x50	NO	YES	YES

During phase two of the thesis, five grids of two grid configurations are used.

Grids for phase two (uniform and DOS-0.009)		
Grids	Gird size	Grid spacing
Grid 1	129x129x129	1
Grid 2	65x65x65	2
Grid 3	33x33x33	4
Grid 4	17x17x17	8
Grid 5	9x9x9	16

6

Results and discussion

The Results section is divided into two parts. The first section relates to the results obtained from the first phase which is the MATLAB implementation of the new numerical scheme. The second section relates to the results obtained from the second phase of the thesis i.e, MMS implementation for the numerical scheme evaluation.

6.1 Phase one

Phase one results are obtained for the grids provided in (5.2.2) with initial and boundary conditions given in (2.4). A line plot analysis is performed to visualize the temperature distribution obtained from PDS and CD. Further, they are compared to the line plots obtained from the analytical results. The line plot is taken along the boundaries across the domain as shown in figure (6.1). It is taken in a way as to obtain the temperature variation within the boundaries through the internal domain. Even though the simulations are performed over time, the interest of this project lies in understanding the diffusion gradients captured by the numerical schemes and further comparing it to the analytical solution for any particular time step. Therefore, the results presented below are for a particular time step.

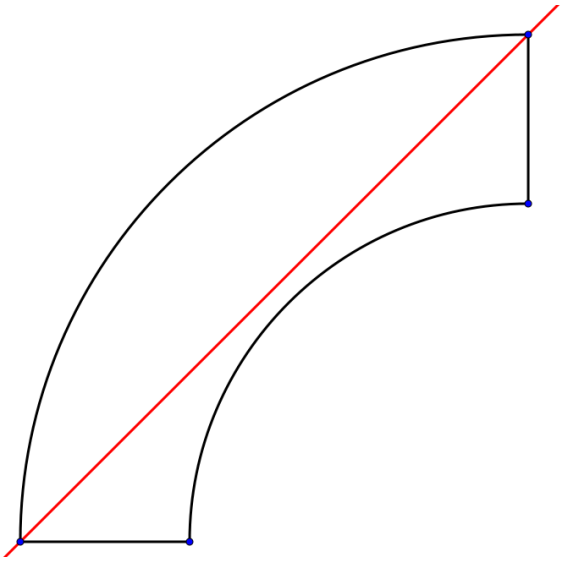


Figure 6.1: Reference Line for line plots

6.1.1 Uniform Orthogonal grid

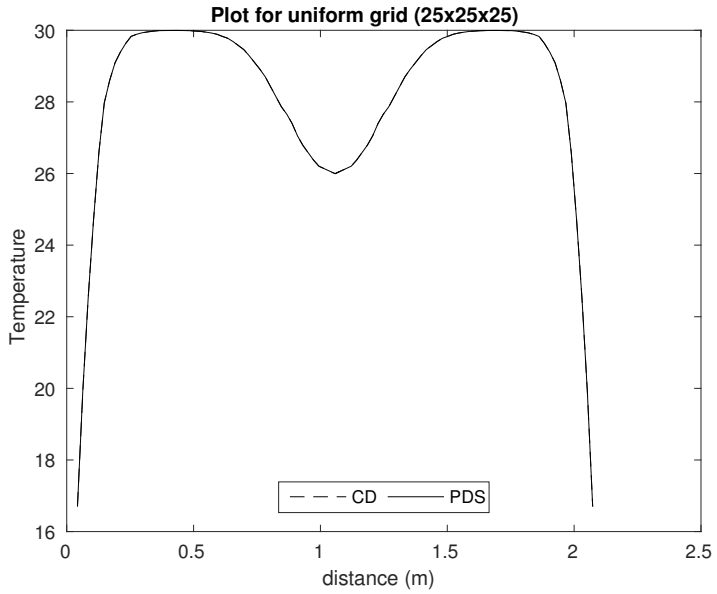


Figure 6.2: Temperature line Plot for PDS and CD with uniform orthogonal grid, t=100

The above line plot represents the temperature distribution obtained for PDS and CD on an uniform orthogonal grid. Both the schemes overlap which shows that PDS behaves like CD when used on an uniform orthogonal grid.

6.1.2 Test for Skewness

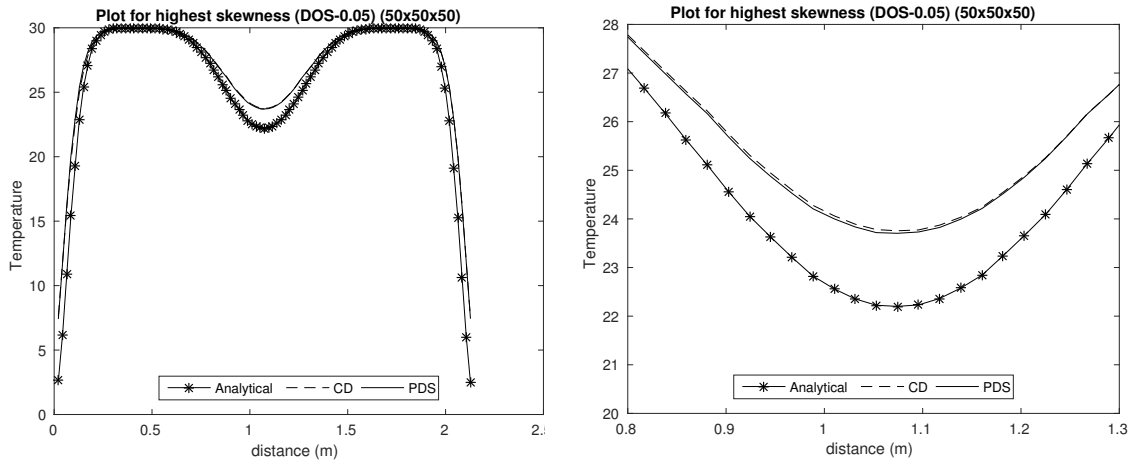


Figure 6.3: Temperature line Plot for PDS and CD with DOS-0.05, t=100

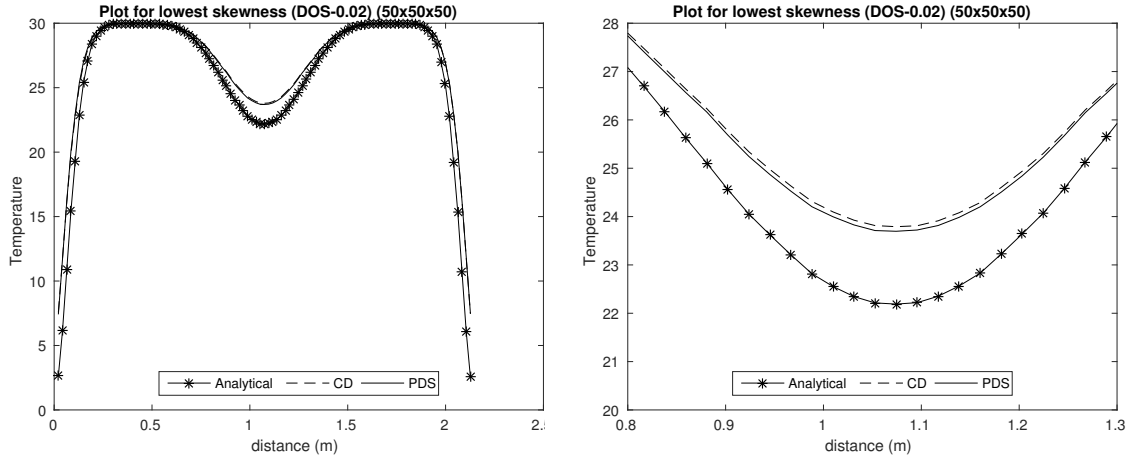


Figure 6.4: Temperature line Plot for PDS and CD with DOS-0.02, t=100

The grid (50x50x50) is subjected to two levels of skewness (0.02 and 0.05) and the plots above represent the temperature plot for PDS, CD and the analytical solution. Plots on the right represent a zoomed in portion of the plots on the left. In both the cases, PDS is closer to the analytical solution but it is not significant. Also, there is a high deviation between the numerical and the analytical solutions near the boundaries.

6.1.3 Test for mesh resolution

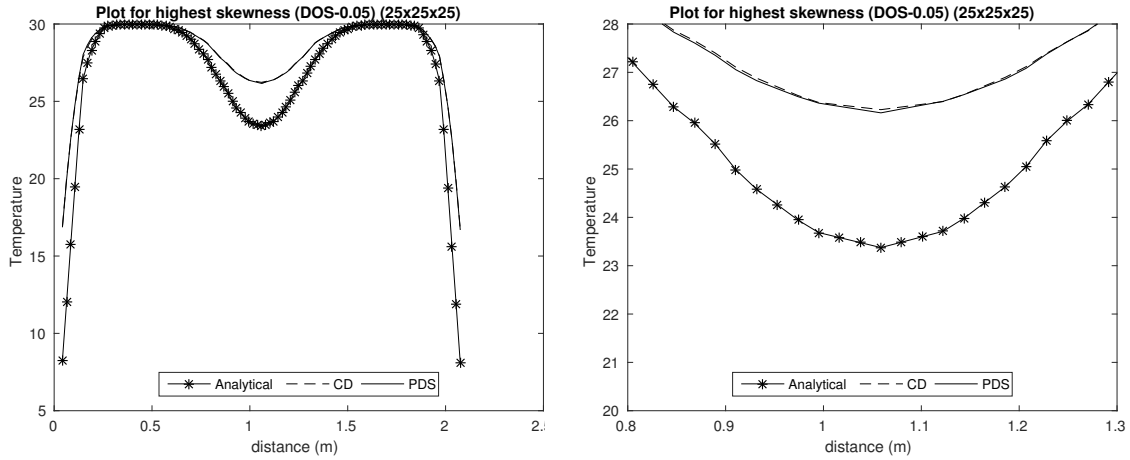


Figure 6.5: Temperature line Plot for PDS and CD with DOS-0.05, t=100

The above plot is obtained for a coarse grid. When compared to figure (6.3), PDS is still relatively closer to the analytical solution when tested for a coarse grid.

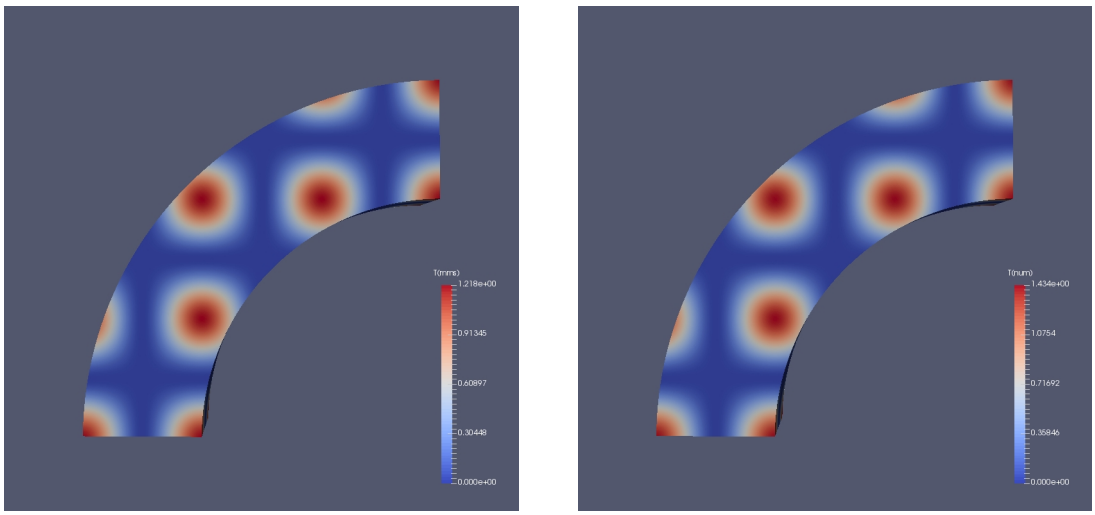
6.2 Phase two

As explained in section 4, the scheme is evaluated using method of manufactured of solutions. The discretization error required to obtain the observed order of accuracy is computed using L_2 norm (4.6) between the numerical and the manufactured solutions.

The results shown below basically demonstrates if the scheme behaves in a second order manner with increasing grid resolution. Here, diffusion gradient terms are mainly studied.

6.2.1 MMS implementation for heat equation

The results below represent the variation of the discretization error for the heat gradients. An example plot for the manufactured solution (6.6) is provided below. It can be observed that the numerical solution obtained from the MMS implementation in CALC++ is similar to the manufactured solution.



Manufactured solution Numerical solution

Figure 6.6: Comparison plots for the manufactured and numerical solutions, t=100

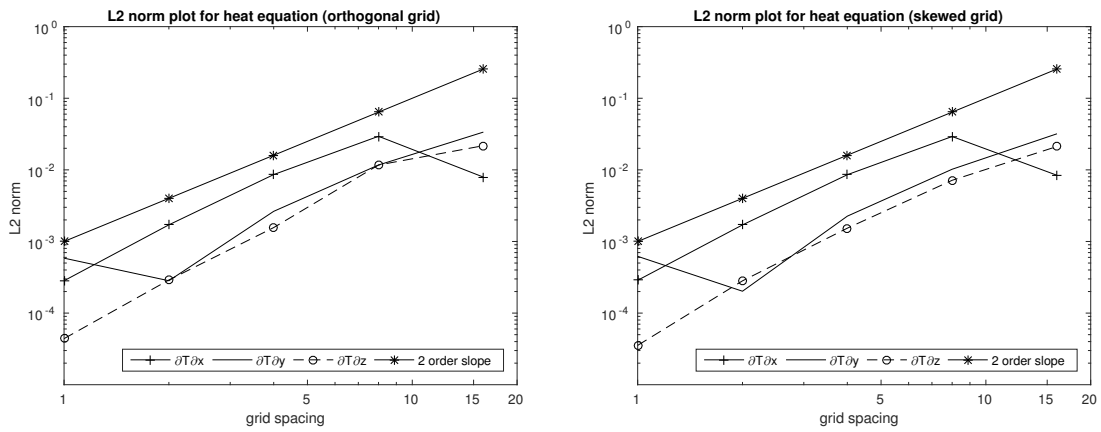


Figure 6.7: Discretization error plots for uniform orthogonal grid and grid with DOS=0.0009, t=100

From figure (6.7), it can be observed that the error reduction almost follows the second order slope but in some cases it deviates. The tables below represent the observed order of accuracies captured with increasing grid resolution. Negative orders of accuracies basically represent that the error is increasing with increasing grid resolution.

Discretization error and observed order of accuracy for uniform orthogonal grid								
Grids	Gird size	Grid spacing	$\frac{\partial T}{\partial x}$		$\frac{\partial T}{\partial y}$		$\frac{\partial T}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0002		0.0005		0.00004	
Grid 2	65x65x65	2	0.001	2.5	0.0002	-1.04	0.0002	2.7
Grid 3	33x33x33	4	0.008	2.3	0.002	3.2	0.001	2.4
Grid 4	17x17x17	8	0.02	1.7	0.01	2.1	0.01	2.9
Grid 5	9x9x9	16	0.007	-1.9	0.03	1.5	0.02	0.8

Discretization error and observed order of accuracy for skewed grid of DOS-0.0009								
Grids	Gird size	Grid spacing	$\frac{\partial T}{\partial x}$		$\frac{\partial T}{\partial y}$		$\frac{\partial T}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0002		0.0006		0.00003	
Grid 2	65x65x65	2	0.001	2.5	0.0002	-1.7	0.0002	2.9
Grid 3	33x33x33	4	0.008	2.3	0.002	3.48	0.001	2.4
Grid 4	17x17x17	8	0.02	1.7	0.01	2.1	0.007	2.2
Grid 5	9x9x9	16	0.008	-1.79	0.03	1.6	0.02	1.5

6.2.2 MMS implementation for incompressible Navier Stokes equation

A similar implementation is performed for the incompressible Navier stokes equation on CALC++. The following results represent the behaviour of the discretization error of the velocity gradients with increasing grid refinement.

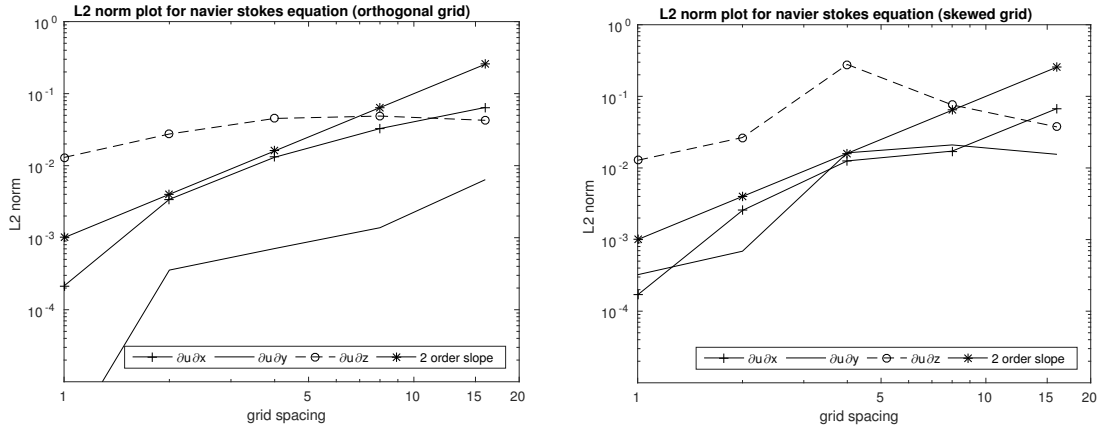


Figure 6.8: Discretization error plots for uniform orthogonal grid and grid with DOS=0.0009 for u velocity gradients, t=100

Discretization error and observed order of accuracy for uniform orthogonal grid								
Grids	Gird size	Grid spacing	$\frac{\partial u}{\partial x}$		$\frac{\partial u}{\partial y}$		$\frac{\partial u}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0002		0.000001		0.01	
Grid 2	65x65x65	2	0.003	3.99	0.0003	8.09	0.03	1.08
Grid 3	33x33x33	4	0.01	1.9	0.0007	0.99	0.04	0.7
Grid 4	17x17x17	8	0.03	1.3	0.001	0.96	0.04	0.1
Grid 5	9x9x9	16	0.06	0.97	0.006	2.21	0.04	-0.19

Discretization error and observed order of accuracy for skewed grid of DOS-0.0009								
Grids	Gird size	Grid spacing	$\frac{\partial u}{\partial x}$		$\frac{\partial u}{\partial y}$		$\frac{\partial u}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0001		0.0003		0.01	
Grid 2	65x65x65	2	0.002	3.9	0.0006	1.1	0.02	1.05
Grid 3	33x33x33	4	0.01	2.29	0.01	4.56	0.27	3.4
Grid 4	17x17x17	8	0.01	0.45	0.02	0.36	0.07	-1.8
Grid 5	9x9x9	16	0.06	1.97	0.01	-0.43	0.03	-1.02

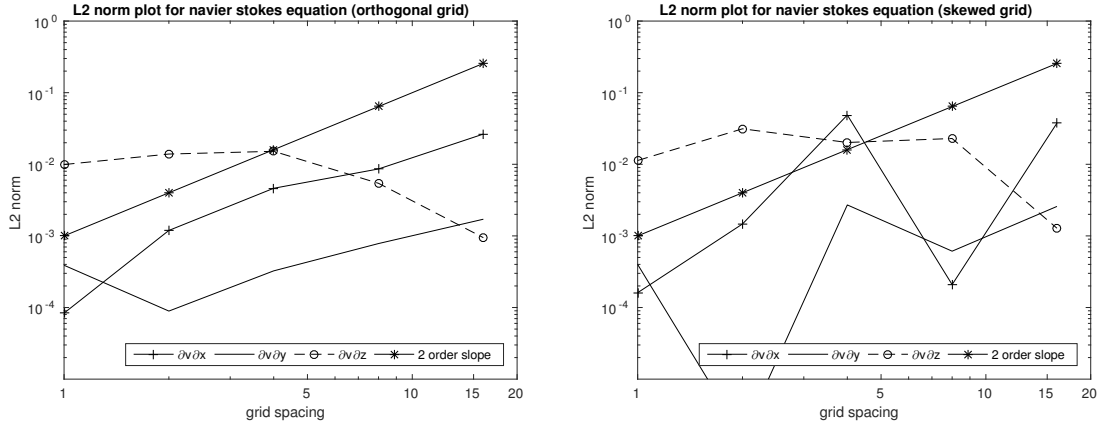


Figure 6.9: Discretization error plots for uniform orthogonal grid and grid with DOS=0.0009 for v velocity gradients, t=100

Discretization error and observed order of accuracy for uniform orthogonal grid								
Grids	Gird size	Grid spacing	$\frac{\partial v}{\partial x}$		$\frac{\partial v}{\partial y}$		$\frac{\partial v}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.00008		0.0003		0.009	
Grid 2	65x65x65	2	0.001	3.83	0.00008	0.48	0.01	0.22
Grid 3	33x33x33	4	0.004	1.9	0.0003	0.12	0.01	-0.18
Grid 4	17x17x17	8	0.008	0.89	0.0007	-1.48	0.005	1.22
Grid 5	9x9x9	16	0.02	1.6	0.001	-2.52	0.0009	2.2

Discretization error and observed order of accuracy for skewed grid of DOS-0.0009								
Grids	Gird size	Grid spacing	$\frac{\partial v}{\partial x}$		$\frac{\partial v}{\partial y}$		$\frac{\partial v}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0001		0.0003		0.01	
Grid 2	65x65x65	2	0.001	3.18	0.000001	-7.97	0.03	1.46
Grid 3	33x33x33	4	0.04	5.05	0.002	10.7	0.02	-0.63
Grid 4	17x17x17	8	0.0002	-7.8	0.0006	-2.1	0.02	0.18
Grid 5	9x9x9	16	0.03	7.4	0.002	2.07	0.001	-4.1

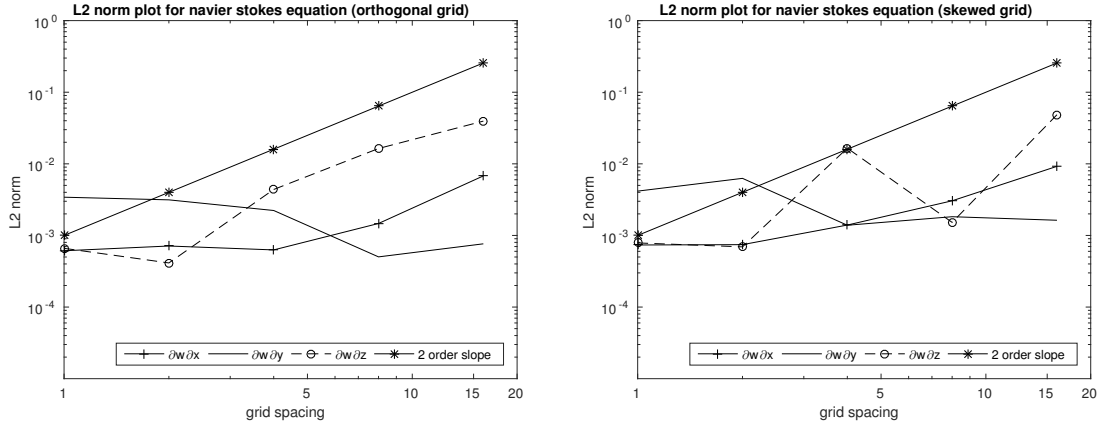


Figure 6.10: Discretization error plots for uniform orthogonal grid and grid with DOS=0.0009 for w velocity gradients, t=100

From the discretization error plots it can be observed that the rate of reduction of the error does not exactly follow the second order slope. Also, the values obtained for the observed order of accuracies are abnormal.

Discretization error and observed order of accuracy for uniform orthogonal grid								
Grids	Gird size	Grid spacing	$\frac{\partial w}{\partial x}$		$\frac{\partial w}{\partial y}$		$\frac{\partial w}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0006		0.003		0.0006	
Grid 2	65x65x65	2	0.0007	0.22	0.003	-0.12	0.0004	-0.6
Grid 3	33x33x33	4	0.0006	-0.18	0.002	-0.49	0.004	3.4
Grid 4	17x17x17	8	0.001	1.22	0.0005	-2.15	0.01	1.9
Grid 5	9x9x9	16	0.006	2.2	0.0007	0.6	0.03	1.2

Discretization error and observed order of accuracy for skewed grid of DOS-0.0009								
Grids	Gird size	Grid spacing	$\frac{\partial w}{\partial x}$		$\frac{\partial w}{\partial y}$		$\frac{\partial w}{\partial z}$	
			L2 norm	OOA	L2 norm	OOA	L2 norm	OOA
Grid 1	129x129x129	1	0.0007		0.004		0.0007	
Grid 2	65x65x65	2	0.0007	0.01	0.006	0.59	0.0006	-0.18
Grid 3	33x33x33	4	0.001	0.9	0.001	-2.17	0.01	4.5
Grid 4	17x17x17	8	0.003	1.1	0.001	0.3	0.001	-3.44
Grid 5	9x9x9	16	0.009	1.6	0.001	-016	0.04	5

6.3 Conclusion

Both CD and PDS perform similarly when subjected to a uniform orthogonal grid. This can be further proved mathematically by reducing the equations in Appendix A to central differencing equations. Based on the obtained results from phase one, it can be concluded that the new transformation technique has a slight edge over the conventional central differencing technique. Based on the evaluation results obtained in phase two, it can be observed that the new numerical scheme does not exhibit a second order behavior.

6.4 Future Work

A conclusion of whether the new numerical scheme is good or bad cannot be drawn with the available evaluation results. The scheme should be further subjected to several other tests to determine its performance. Considering the implementation of MMS is accurate in this work, the two major areas of future work could be in checking the implementation of the numerical scheme (algorithm) in CALC++ and/or the mathematics of the numerical scheme. Further, solution verification can be performed to estimate the formal order of accuracy.

Bibliography

[1] Andersson, N., (2005) "A Study of Subsonic Turbulent Jets and Their Radiated Sound Using Large-Eddy Simulation", Ph.D. thesis, Department of Applied Mechanics, Chalmers University of Technology, Göteborg, Sweden, ISBN 91-7291-679-6, pp.30-31.

[2] Necati Ozisik, M.,W. Hahn, D., (1993) "Heat Conduction",2nd edition, ISBN 04-7153-256-8, pp.99-133.

[3] Juretic, F., (2004) "Error Analysis in Finite Volume CFD", Ph.D. thesis, Department of Mechanical Engineering, Imperial College , London, England.

[4] Roache, P.J, "Code Verification by the Method of Manufactured Solutions", ASME Journal, Vol. 124, 2002.

[5] Roy, C.J, "Review of code and solution verification procedures for computational simulation", Journal of Computational Physics 205, 131-156, 2005.

[6] Veluri, S.P., Roy, C.J, and Choudhary, A., Edward, A.L., “Finite Volume Diffusion Operators for Compressible CFD on Unstructured Grids”, AIAA, 2009
2009-4141

A

Preferred Direction Diffusion Scheme

This chapter provides a detailed derivation of the integral calculations to obtain the coefficients in (3.9).

Cell Properties

Twenty cell properties are calculated for each cell using an uniquely defined reference node point $(x_{ref}, y_{ref}, z_{ref})$.

$$\mathcal{V}_{o_k} = \int\int\int_{\Omega_k} d\Omega$$
$$\mathcal{V}_{x_k} = \int\int\int_{\Omega_k} (x - x_{ref}) d\Omega$$
$$\mathcal{V}_{y_k} = \int\int\int_{\Omega_k} (y - y_{ref}) d\Omega$$
$$\mathcal{V}_{z_k} = \int\int\int_{\Omega_k} (z - z_{ref}) d\Omega$$
$$\mathcal{V}_{xx_k} = \int\int\int_{\Omega_k} (x - x_{ref})^2 d\Omega$$
$$\mathcal{V}_{xy_k} = \int\int\int_{\Omega_k} (x - x_{ref})(y - y_{ref}) d\Omega$$
$$\mathcal{V}_{xz_k} = \int\int\int_{\Omega_k} (x - x_{ref})(z - z_{ref}) d\Omega$$
$$\mathcal{V}_{yy_k} = \int\int\int_{\Omega_k} (y - y_{ref})^2 d\Omega$$
$$\mathcal{V}_{yz_k} = \int\int\int_{\Omega_k} (y - y_{ref})(z - z_{ref}) d\Omega$$
$$\mathcal{V}_{zz_k} = \int\int\int_{\Omega_k} (z - z_{ref})^2 d\Omega$$
$$\mathcal{V}_{xxx_k} = \int\int\int_{\Omega_k} (x - x_{ref})^3 d\Omega$$
$$\mathcal{V}_{xxy_k} = \int\int\int_{\Omega_k} (x - x_{ref})^2 (y - y_{ref}) d\Omega$$
$$\mathcal{V}_{xxz_k} = \int\int\int_{\Omega_k} (x - x_{ref})^2 (z - z_{ref}) d\Omega$$
$$\mathcal{V}_{xyy_k} = \int\int\int_{\Omega_k} (x - x_{ref})(y - y_{ref})^2 d\Omega$$

$$
\begin{aligned}
\mathcal{V}_{xyz_k} &= \int\int\int_{\Omega_k} (x - x_{ref})(y - y_{ref})(z - z_{ref}) d\Omega \\
\mathcal{V}_{xzz_k} &= \int\int\int_{\Omega_k} (x - x_{ref})(z - z_{ref})^2 d\Omega \\
\mathcal{V}_{yyy_k} &= \int\int\int_{\Omega_k} (y - y_{ref})^3 d\Omega \\
\mathcal{V}_{yyz_k} &= \int\int\int_{\Omega_k} (y - y_{ref})^2 (z - z_{ref}) d\Omega \\
\mathcal{V}_{yzz_k} &= \int\int\int_{\Omega_k} (y - y_{ref})(z - z_{ref})^2 d\Omega \\
\mathcal{V}_{zzz_k} &= \int\int\int_{\Omega_k} (z - z_{ref})^3 d\Omega
\end{aligned}
\tag{A.1}
$$

Tri-Linear Parametrization

The Cell properties in A are calculated for each of the ten cells per face using Gauss-point Quadrature. Initially a local coordinate system (x', y', z') is defined for each cell. This system with respect to the global co-ordinate system is obtained using tri-linear parametrization.

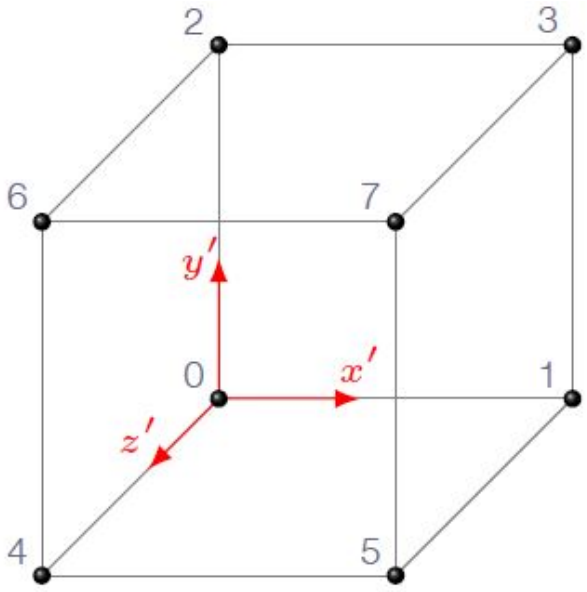


Figure A.1: Representation of local co-ordinate sytem of a cell in physical space

$$
\begin{gathered}
x = (1-x')(1-y')(1-z')x_0 + x'(1-y')(1-z')x_1 + \\
(1-x')y'(1-z')x_2 + x'y'(1-z')x_3 + (1-x')(1-y')z'x_4 + \\
x'(1-y')z'x_5 + (1-x')y'z'x_6 + x'y'z'x_7 \\
y = (1-x')(1-y')(1-z')y_0 + x'(1-y')(1-z')y_1 + \\
(1-x')y'(1-z')y_2 + x'y'(1-z')y_3 + (1-x')(1-y')z'y_4 + \\
x'(1-y')z'y_5 + (1-x')y'z'y_6 + x'y'z'y_7 \\
z = (1-x')(1-y')(1-z')z_0 + x'(1-y')(1-z')z_1 + \\
(1-x')y'(1-z')z_2 + x'y'(1-z')z_3 + (1-x')(1-y')z'z_4 + \\
x'(1-y')z'z_5 + (1-x')y'z'z_6 + x'y'z'z_7
\end{gathered}
\tag{A.2}
$$

Gauss-point Quadrature

Using the equations in (A.2), the volume integrals in (A.1) are calculated using Gauss-point Quadrature numerical integration method. The general representation of the integration is given by,

$$
\int\int\int_{\Omega_k} \phi(x,y,z)\,dxdydz = \int_0^1\int_0^1\int_0^1 \phi(x',y',z')J\,dx'dy'dz'
$$

where Jacobian J is defined as,

$$
\begin{vmatrix}
\frac{\partial x}{\partial x'} & \frac{\partial x}{\partial y'} & \frac{\partial x}{\partial z'} \\
\frac{\partial y}{\partial x'} & \frac{\partial y}{\partial y'} & \frac{\partial y}{\partial z'} \\
\frac{\partial z}{\partial x'} & \frac{\partial z}{\partial y'} & \frac{\partial z}{\partial z'}
\end{vmatrix}
\tag{A.3}
$$

In order to use the Gauss-point quadrature, the range of the local coordinate system must be redefined from -1 to 1.

$$
\int\int\int_{\Omega_k} \phi(x,y,z)\,dxdydz = \frac{1}{8}\int_{-1}^1\int_{-1}^1\int_{-1}^1 \phi(x'',y'',z'')J(x'',y'',z'')\,dx''dy''dz'' \tag{A.4}
$$

$$
\int\int\int_{\Omega_k} \phi(x,y,z)\,dxdydz = \frac{1}{8}\sum_{i=1}^n\sum_{j=1}^n\sum_{k=1}^n w_i w_j w_k \phi(p_i,p_j,p_k)J(p_i,p_j,p_k) \tag{A.5}
$$

Where p_i, p_j, p_k are the Gauss points and w_i, w_j, w_k are the corresponding weights.

Volume Integrals

The integrals in matrix A is calculated using the relationship between the (ξ,η,ζ) space and (x,y,z) space in (3.2).

$$
\begin{aligned}
\xi &= \Psi_{11}(x-x_o) + \Psi_{12}(y-y_o) + \Psi_{13}(z-z_o) \\
\eta &= \Psi_{21}(x-x_o) + \Psi_{22}(y-y_o) + \Psi_{23}(z-z_o) \\
\zeta &= \Psi_{31}(x-x_o) + \Psi_{32}(y-y_o) + \Psi_{33}(z-z_o)
\end{aligned}
\tag{A.6}
$$

The following relations are used to rewrite the integrals,

$$
\begin{aligned}
(x-x_o) &= ((x-x_{ref}) + (x_{ref}-x_o)) \\
(y-y_o) &= ((y-y_{ref}) + (y_{ref}-y_o)) \\
(z-z_o) &= ((z-z_{ref}) + (z_{ref}-z_o))
\end{aligned}
\tag{A.7}
$$

Introducing the constansts $\Delta x, \Delta y, \Delta z$,

$$\begin{aligned} \Delta x &= x_{ref} - x_o \\ \Delta y &= y_{ref} - y_o \\ \Delta z &= z_{ref} - z_o \end{aligned} \tag{A.8}$$

which results in,

$$\begin{aligned} (x - x_o) &= ((x - x_{ref}) + \Delta x) \\ (y - y_o) &= ((y - y_{ref}) + \Delta y) \\ (z - z_o) &= ((z - z_{ref}) + \Delta z) \end{aligned} \tag{A.9}$$

The terms in the first column are obtained using the volume integrals in equations (A.1). Further, the second, third and fourth columns which consists of first order terms are obtained using the direct relations from equations (A.6-A.9).

$$\begin{aligned} \int\int\int_{\Omega_k} \xi \, d\Omega &= \Psi_{11}(\mathcal{V}_{x_k} + \Delta x \mathcal{V}_{o_k}) + \Psi_{12}(\mathcal{V}_{y_k} + \Delta y \mathcal{V}_{o_k}) + \Psi_{13}(\mathcal{V}_{z_k} + \Delta z \mathcal{V}_{o_k}) \\ \int\int\int_{\Omega_k} \eta \, d\Omega &= \Psi_{21}(\mathcal{V}_{x_k} + \Delta x \mathcal{V}_{o_k}) + \Psi_{22}(\mathcal{V}_{y_k} + \Delta y \mathcal{V}_{o_k}) + \Psi_{23}(\mathcal{V}_{z_k} + \Delta z \mathcal{V}_{o_k}) \\ \int\int\int_{\Omega_k} \zeta \, d\Omega &= \Psi_{31}(\mathcal{V}_{x_k} + \Delta x \mathcal{V}_{o_k}) + \Psi_{32}(\mathcal{V}_{y_k} + \Delta y \mathcal{V}_{o_k}) + \Psi_{33}(\mathcal{V}_{z_k} + \Delta z \mathcal{V}_{o_k}) \end{aligned} \tag{A.10}$$

Further, the quadratic and cubic terms in matrix A are calculated by rewriting and expanding the products of ξ, η and ζ.

$$\begin{gathered}
\xi\eta = \Psi_{11}\Psi_{21}(x-x_o)^2 + (\Psi_{11}\Psi_{22}+\Psi_{12}\Psi_{21})(x-x_o)(y-y_o)+ \\
(\Psi_{11}\Psi_{23}+\Psi_{13}\Psi_{21})(x-x_o)(z-z_o) + \Psi_{12}\Psi_{22}(y-y_o)^2+ \\
(\Psi_{12}\Psi_{23}+\Psi_{13}\Psi_{22}(y-y_o)(z-z_o) + \Psi_{13}\Psi_{23}(z-z_o)^2 \\
\xi\zeta = \Psi_{11}\Psi_{31}(x-x_o)^2 + (\Psi_{11}\Psi_{32}+\Psi_{12}\Psi_{31})(x-x_o)(y-y_o)+ \\
(\Psi_{11}\Psi_{33}+\Psi_{13}\Psi_{31})(x-x_o)(z-z_o) + \Psi_{12}\Psi_{32}(y-y_o)^2+ \\
(\Psi_{12}\Psi_{33}+\Psi_{13}\Psi_{32}(y-y_o)(z-z_o) + \Psi_{13}\Psi_{33}(z-z_o)^2
\end{gathered}$$

$$\begin{gathered}
\zeta^2 = \Psi_{21}\Psi_{21}(x-x_o)^2 + 2\Psi_{21}\Psi_{22}(x-x_o)(y-y_o)+ \\
2\Psi_{21}\Psi_{23}(x-x_o)(z-z_o) + \Psi_{22}\Psi_{22}(y-y_o)^2+ \\
2\Psi_{22}\Psi_{23}(y-y_o)(z-z_o) + \Psi_{23}\Psi_{23}(z-z_o)^2 \\
\eta^2 = \Psi_{31}\Psi_{31}(x-x_o)^2 + 2\Psi_{31}\Psi_{32}(x-x_o)(y-y_o)+ \\
2\Psi_{31}\Psi_{33}(x-x_o)(z-z_o) + \Psi_{32}\Psi_{32}(y-y_o)^2+ \\
2\Psi_{32}\Psi_{33}(y-y_o)(z-z_o) + \Psi_{33}\Psi_{33}(z-z_o)^2
\end{gathered}$$

$$\begin{gathered}
\xi\eta^2 = \Psi_{11}\Psi_{21}\Psi_{21}(x-x_o)^3+ \\
(2\Psi_{11}\Psi_{21}\Psi_{22}+\Psi_{12}\Psi_{21}\Psi_{21})(x-x_o)^2(y-y_o)+ \\
(2\Psi_{11}\Psi_{21}\Psi_{23}+\Psi_{13}\Psi_{21}\Psi_{21})(x-x_o)^2(z-z_o)+ \\
(\Psi_{11}\Psi_{22}\Psi_{22}+2\Psi_{12}\Psi_{21}\Psi_{22})(x-x_o)(y-y_o)^2+ \\
(2\Psi_{11}\Psi_{22}\Psi_{23}+2\Psi_{12}\Psi_{21}\Psi_{23}+2\Psi_{13}\Psi_{21}\Psi_{22}) \\
(x-x_o)(y-y_o)(z-z_o)+ \\
(\Psi_{11}\Psi_{23}\Psi_{23}+2\Psi_{13}\Psi_{21}\Psi_{23})(x-x_o)(z-z_o)^2+ \\
\Psi_{12}\Psi_{22}\Psi_{22}(y-y_o)^3+ \\
(2\Psi_{12}\Psi_{22}\Psi_{23}+\Psi_{13}\Psi_{22}\Psi_{22})(y-y_o)^2(z-z_o)+ \\
(\Psi_{12}\Psi_{23}\Psi_{23}+2\Psi_{13}\Psi_{22}\Psi_{23})(y-y_o)(z-z_o)^2+ \\
\Psi_{13}\Psi_{23}\Psi_{23}(z-z_o)^3 \\
\xi\zeta^2 = \Psi_{11}\Psi_{31}\Psi_{31}(x-x_o)^3+ \\
(2\Psi_{11}\Psi_{31}\Psi_{32}+\Psi_{12}\Psi_{31}\Psi_{31})(x-x_o)^2(y-y_o)+ \\
(2\Psi_{11}\Psi_{31}\Psi_{33}+\Psi_{13}\Psi_{31}\Psi_{31})(x-x_o)^2(z-z_o)+ \\
(\Psi_{11}\Psi_{32}\Psi_{32}+2\Psi_{12}\Psi_{31}\Psi_{32})(x-x_o)(y-y_o)^2+ \\
(2\Psi_{11}\Psi_{32}\Psi_{33}+2\Psi_{12}\Psi_{31}\Psi_{33}+2\Psi_{13}\Psi_{31}\Psi_{32}) \\
(x-x_o)(y-y_o)(z-z_o)+ \\
(\Psi_{11}\Psi_{33}\Psi_{33}+2\Psi_{13}\Psi_{31}\Psi_{33})(x-x_o)(z-z_o)^2+ \\
\Psi_{12}\Psi_{32}\Psi_{32}(y-y_o)^3+ \\
(2\Psi_{12}\Psi_{32}\Psi_{33}+\Psi_{13}\Psi_{32}\Psi_{32})(y-y_o)^2(z-z_o)+ \\
(\Psi_{12}\Psi_{33}\Psi_{33}+2\Psi_{13}\Psi_{32}\Psi_{33})(y-y_o)(z-z_o)^2+ \\
\Psi_{13}\Psi_{33}\Psi_{33}(z-z_o)^3
\end{gathered} \tag{A.11}$$

Further, the unknown products of $(x - x_o)$, $(y - y_o)$ and $(z - z_o)$ in equations (A.11) are obtained by expanding and rewriting them using equation (A.9).

$$(x - x_o)^2 = (x - x_{ref})^2 + 2\Delta x(x - x_{ref}) + \Delta x^2$$

$$(x - x_o)(y - y_o) = (x - x_{ref})(y - y_{ref}) + \Delta y(x - x_{ref}) + \Delta x(y - y_{ref}) + \Delta x\Delta y$$

$$(x - x_o)(z - z_o) = (x - x_{ref})(z - z_{ref}) + \Delta z(x - x_{ref}) + \Delta x(z - z_{ref}) + \Delta x\Delta z$$

$$(y - y_o)^2 = (y - y_{ref})^2 + 2\Delta y(y - y_{ref}) + \Delta y^2$$

$$(y - y_o)(z - z_o) = (y - y_{ref})(z - z_{ref}) + \Delta z(y - y_{ref}) + \Delta y(z - z_{ref}) + \Delta y\Delta z$$

$$(z - z_o)^2 = (z - z_{ref})^2 + 2\Delta z(z - z_{ref}) + \Delta z^2$$

$$(x - x_o)^3 = (x - x_{ref})^3 + 3\Delta x(x - x_{ref})^2 + 3\Delta x^2(x - x_{ref}) + \Delta x^3$$

$$(x - x_o)^2(y - y_o) = (x - x_{ref})^2(y - y_{ref}) + 2\Delta x(x - x_{ref})(y - y_{ref}) + \Delta x^2(y - y_{ref}) + \Delta y(x - x_{ref})^2 + 2\Delta x\Delta y(x - x_{ref}) + \Delta x^2\Delta y^2$$

$$(x - x_o)^2(z - z_o) = (x - x_{ref})^2(z - z_{ref}) + 2\Delta x(x - x_{ref})(z - z_{ref}) + \Delta x^2(z - z_{ref}) + \Delta z(x - x_{ref})^2 + 2\Delta x\Delta z(x - x_{ref}) + \Delta x^2\Delta z^2$$

$$(x - x_o)(y - y_o)^2 = (x - x_{ref})(y - y_{ref})^2 + 2\Delta y(x - x_{ref})(y - y_{ref}) + \Delta y^2(x - x_{ref}) + \Delta x(y - y_{ref})^2 + 2\Delta x\Delta y(y - y_{ref}) + \Delta x\Delta y^2$$

$$(x - x_o)(z - z_o)^2 = (x - x_{ref})(z - z_{ref})^2 + 2\Delta z(x - x_{ref})(z - z_{ref}) + \Delta z^2(x - x_{ref}) + \Delta x(z - z_{ref})^2 + 2\Delta x\Delta z(z - z_{ref}) + \Delta x\Delta z^2$$

$$(y-y_o)^3 = (y-y_{ref})^3 + 3\Delta y(y-y_{ref})^2 + 3\Delta y^2(y-y_{ref}) + \Delta y^3$$

$$\begin{gathered}(y-y_o)^2(z-z_o) = (y-y_{ref})^2(z-z_{ref}) + \\ 2\Delta y(y-y_{ref})(z-z_{ref}) + \\ \Delta y^2(z-z_{ref}) + \Delta z(y-y_{ref})^2 + \\ 2\Delta y\Delta z(y-y_{ref}) + \Delta y^2\Delta z^2\end{gathered}$$

$$\begin{gathered}(y-y_o)(z-z_o)^2 = (y-y_{ref})(z-z_{ref})^2 + \\ 2\Delta z(y-y_{ref})(z-z_{ref}) + \\ \Delta z^2(y-y_{ref}) + \Delta y(z-z_{ref})^2 + \\ 2\Delta y\Delta z(z-z_{ref}) + \Delta y\Delta z^2\end{gathered} \tag{A.12}$$

$$\begin{gathered}(z-z_o)^3 = (z-z_{ref})^3 + 3\Delta z(z-z_{ref})^2 + \\ 3\Delta z^2(z-z_{ref}) + \Delta z^3\end{gathered}$$

$$\begin{gathered}(x-x_o)(y-y_o)(z-z_o) = (x-x_{ref})(y-y_{ref})(z-z_{ref}) + \\ \Delta y(x-x_{ref})(z-z_{ref}) + \\ \Delta x(y-y_{ref})(z-z_{ref}) + \\ \Delta x\Delta y(z-z_{ref}) + \\ \Delta z(x-x_{ref})(y-y_{ref}) + \\ \Delta y\Delta z(x-x_{ref}) + \\ \Delta x\Delta z(y-y_{ref}) + \Delta x\Delta y\Delta z\end{gathered}$$

Using equations in (A.11 and A.12) together with the cell properties, the quadratic and cubic terms of matrix A can be calculated.

Domain Boundaries

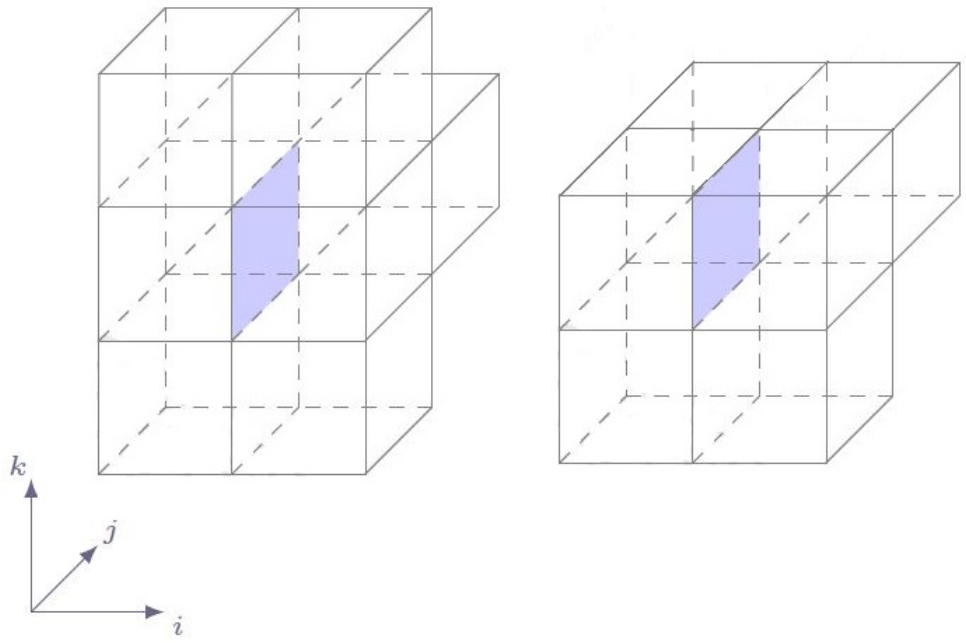


Figure A.2: Representation of Domain Boundaries

The cell faces present along the domain boundaries do not have ten cells surrounding them. Therefore a truncated flux molecule is considered for the analysis in these cases. A general description of the treatment which is analogous for flux molecules in other directions is discussed below.
The first case consists of eight cells with two missing in η direction. This reflects in the removal of the terms which includes η^2 from the equation (3.3). Analogous to this case, the terms containing ζ^2 are removed in the other direction.

$$\varphi(\xi, \eta, \zeta) = C_0 + C_1\xi + C_2\eta + C_3\zeta + C_4\xi\eta + C_5\xi\zeta + C_6\zeta^2 + C_7\xi\zeta^2 \tag{A.13}$$

The second case consists of six cells with two missing in η and ζ directions. This reflects in the removal of the terms which includes η^2 and ζ^2 from the equation (3.3).

$$\varphi(\xi, \eta, \zeta) = C_0 + C_1\xi + C_2\eta + C_3\zeta + C_4\xi\eta + C_5\xi\zeta \tag{A.14}$$

B

Analytical Solution

This section provides a detailed derivation of the analytical solution for the considered geometry in this project. The Governing Equation solved here is a 2D Unsteady heat Conduction equation in Cylindrical Coordinate system.

$$\frac{\partial T}{\partial t} = \alpha(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \Theta^2}) \tag{B.1}$$

The initial and Boundary conditions are,

$$\begin{gathered} T(r = a = 1m) = 0 \qquad T(r = b = 1.5m) = 0 \\ T(\Theta = 0) = 0 \qquad T(\Theta = \frac{\pi}{2}) = 0 \\ T(t = 0) = F(r, \Theta) = 30K \end{gathered} \tag{B.2}$$

Using Separation of Variables method of solving Partial Differential Equations,

$$\begin{gathered} \text{let} \qquad T(t, \Theta, t) = R(r).\chi(\Theta).\Gamma(t) \\ \text{substituting in (B.1) results in,} \\ \frac{1}{R}[R'' + \frac{1}{r}R'] + \frac{1}{r^2\chi}\chi'' = \frac{1}{\alpha\Gamma}\frac{\partial \Gamma}{\partial t} = -\lambda^2 \end{gathered} \tag{B.3}$$

The general solution of the unsteady term in (B.3) results in,

$$\Gamma(t) = C_1.e^{-\alpha\lambda^2 t} \tag{B.4}$$

Further, the general solution of the χ is obtained as,

$$\begin{gathered} \chi(\Theta) = C_2 cos(\nu\Theta) + C_3 sin(\nu\Theta) \\ \text{Applying Boundary conditions,} \\ \chi(\Theta = 0) = 0 = C_2 \\ \chi(\Theta = \frac{\pi}{2}) = 0 = C_3 sin(\nu\frac{\pi}{2}) \\ \nu_n = 2n \quad n = 1, 2, 3... \end{gathered} \tag{B.5}$$

Here ν_n are the Eigen values. Further, the general equation for the R term is obtained as,

$$R'' + \frac{1}{r}R' + (\lambda^2 - \frac{\nu^2}{r^2})R = 0$$
$$R(r) = C_4 J_\nu(\lambda r) + C_5 Y_\nu(\lambda r)$$

Here J_ν and Y_ν are Bessel functions of the first and second order respectively.

Further, applying the Boundary conditions,

$$R(a) = 0 = C_4 J_\nu(\lambda a) + C_5 Y_\nu(\lambda a)$$

$$C_5 = -C_4 \frac{J_\nu(\lambda a)}{Y_\nu(\lambda a)}$$

$$R(r) = \frac{C_4}{Y_\nu(\lambda a}(Y_\nu(\lambda a)J_\nu(\lambda r) - J_\nu(\lambda a)Y_\nu(\lambda r)) \tag{B.6}$$

$$R(r) = C_6(Y_\nu(\lambda a)J_\nu(\lambda r) - J_\nu(\lambda a)Y_\nu(\lambda r))$$

$$R(b) = 0 = Y_\nu(\lambda a)J_\nu(\lambda r) - J_\nu(\lambda a)Y_\nu(\lambda r)$$

$$Y_\nu(\lambda a)J_\nu(\lambda r) - J_\nu(\lambda a)Y_\nu(\lambda r) = 0 \qquad \lambda_m = 1, 2, 3...$$

Each value of ν_n yields λ_m Eigen Values
thereby resulting in λ_{nm} of Eigen values.

Using the Solutions from equations (B.4-B.6), the Temperature is described as,

$$T(r, \Theta, t) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} C_{nm} sin(\nu_n \Theta) e^{(-\alpha\lambda_{nm}^2 t)}[Y_\nu(\lambda_{nm}a)J_\nu(\lambda_{nm}r) - J_\nu(\lambda_{nm}a)Y_\nu(\lambda_{nm}r)]$$

Applying Initial Condition,

$$F(r, \Theta) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} C_{nm} sin(\nu_n \Theta) e^{(-\alpha\lambda_{nm}^2 t)}[Y_\nu(\lambda_{nm}a)J_\nu(\lambda_{nm}r) - J_\nu(\lambda_{nm}a)Y_\nu(\lambda_{nm}r)]$$

Using the orthogonal Properties of Bessel Functions,

$$C_{nm} = \frac{\int_{r=a}^{b}\int_{\Theta=0}^{\frac{\pi}{2}} rF(r,\Theta)sin(\nu_n\Theta)[Y_\nu(\lambda_{nm}a)J_\nu(\lambda_{nm}r) - J_\nu(\lambda_{nm}a)Y_\nu(\lambda_{nm}r)]\, d\Theta dr}{\int_{r=a}^{b} r[Y_\nu(\lambda_{nm}a)J_\nu(\lambda_{nm}r) - J_\nu(\lambda_{nm}a)Y_\nu(\lambda_{nm}r)]^2 dr. \int_{\Theta=0}^{\frac{\pi}{2}} sin^2(\nu\Theta)]d\Theta} \tag{B.7}$$

C

Procedure for MMS

This section provides a detailed procedure of the method of manufactured solutions using 1D unsteady heat equation as an example,

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0 \tag{C.1}$$

Consider a manufactured solution for T,

$$T(x,t) = e^{-t} sin(\pi x) \tag{C.2}$$

Applying the manufactured solution to the governing equation,

$$\begin{aligned} \frac{\partial T}{\partial t} &= -e^{-t} sin(\pi x) \\ \frac{\partial^2 T}{\partial x^2} &= -e^{-t} \pi^2 sin(\pi x) \end{aligned} \tag{C.3}$$

Using equations (C.1 and C.3) the obtained source term is,

$$Q = e^{-t} sin(\pi x)(\alpha \pi^2 - 1) \tag{C.4}$$

Using (C.4) the governing equation with (C.2) as the solution is written as,

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = Q \tag{C.5}$$

The initial and the corresponding boundary conditions are applied using the manufactured solution (C.2). For example, the initial and the dirchlet boundary condition is represented as,

$$\begin{aligned} T(x,0) &= sin(\pi x) \\ T(x_B,t) &= e^{-t} sin(\pi x_B) \end{aligned} \tag{C.6}$$

The equation (C.5) is numerically modelled and run using the corresponding numerical schemes. A set of results using systematic grid refinement is obtained and the order of accuracy is computed using equation (1.5). Further, this method can also be used to test different boundary conditions.