PARALLEL SOLVERS USING PETSC FOR NONLINEAR CONDUCTION PROBLEMS IN HIGHER DIMENSIONS

By:

MUHAMMAD HISYAMUDDIN BIN ROSLI

(Matric No.: 123116)

Supervisor:

Dr. –Ing. Muhammad Razi bin Abdul Rahman

May 2018

This dissertation is submitted to

Universiti Sains Malaysia

as partial fulfillment of the requirement to graduate with honors degree in

BACHELOR OF ENGINEERING (MECHANICAL ENGINEERING)

School of Mechanical Engineering

Engineering Campus

Universiti Sains Malaysia

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed…….. (Muhammad Hisyamuddin bin Rosli)
Date……..

STATEMENT 1
This thesis is the result of my own investigations, except where otherwise stated.
Other sources are acknowledged by giving explicit references.
Bibliography/references are appended.

Signed…….. (Muhammad Hisyamuddin bin Rosli)
Date……..

STATEMENT 2
I hereby give consent for my thesis, if accepted, to be available for photocopying and for interlibrary loan, and for the title and summary to be made available outside organizations.

Signed…….. (Muhammad Hisyamuddin bin Rosli)
Date……..

ACKNOWLEDGEMENT

The completion of this project is not possible without the help from many people. First of all, I would like to express my gratitude to Universiti Sains Malaysia for giving me the chance to complete this four-year course in Mechanical Engineering.

I would like to express my gratitude to my dedicated supervisor, Dr. –Ing. Muhammad Razi bin Abdul Rahman, who has given me the chance to carry out this interesting and yet challenging project. Beside this, thanks for giving constant guidance and advice throughout this whole project. It would not be possible for the projects to go this far without your patience and enthusiasm encouragement and guidance.

Next, I would like to extend my gratitude to the staffs of the School of Aerospace Engineering, especially Mrs. Rahayu binti Dorahim@Abdul Rahim for providing the access to the HPC server. I would also like to thank Mr. Mohd Najib bin Mohd Hussain for resolving the problem encountered when the server was offline for more than a week. Apart from that, I would like to thank all my fellow friends who provide me the encouragement and assistant whenever needed.

Last but not least, I would like to thank my dearest family and friends for their continuous support along the journey of completion of this final year project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Explanation
AMG	algebraic multigrid
ASM	Additive Schwarz Method
BJACOBI	Block Jacobi
BLAS	Basic Linear Algebra Subprograms
CG	conjugate gradient
DM	domain management
FEM	finite element method
GCC	GNU Compiler Collection
HPC	high-performance computing
ILU	incomplete LU (matrix factorisation method)
LAPACK	Linear Algebra Package
LU	lower upper (matrix factorisation method)
MG	multigrid method
MPI	Message Passing Interface
NUMA	non-uniform memory access
OpenMP	Open Multi-Processing
PC	preconditioner
PDE	partial differential equation
PETSc	Portable, Extensible Toolkit for Scientific Computation
SGS	subgrid scale
SMP	symmetric multiprocessing
SNES	Scalable Nonlinear Equations Solver
SOR	successive over relaxation
SPD	symmetric positive definite

ABSTRAK

Pakej Set Perisisan Mudah Alih, Boleh Perluas untuk Pengiraan Saintifik (PETSc) ialah alat pengkomputeran yang popular dalam kalangan para penyelidik kerana pakej ini menyediakan kemampuan untuk melaksanakan pengaturcaraan selari dalam menyelesaikan pelbagai jenis analisis unsur terhingga melalui penggunaan pelbagai rutin dan set perisian yang dapat dimanipulasi melalui pilihan waktu eksekusi. PETSc, seperti yang didakwa, membolehkan pertukaran kod saintifik sedia ada dari struktur bersiri ke dalam struktur selari dengan cara yang mudah.

Tumpuan utama projek ini adalah untuk menilai prestasi kod selari unsur terhingga melalui penggunaan PETSc. Perbandingan prestasi antara pelbagai jenis penyelesai selari juga dilakukan dalam kajian ini. Tahap pengoptimuman selari yang berbeza juga dikaji. Platform ujian terdiri daripada pemproses Intel Xeon X5650 yang mempunyai 6 teras fizikal (12 teras logik) dan 24GB RAM serta kod kaedah unsur terhingga (FEM) dibina dengan pengkompil versi GCC 4.4 dan OpenMPI 3.0.

Hasil dari projek ini menunjukkan prestasi pengiraan selari yang agak baik berbanding prestasi pengiraan bersiri. Sebagai contoh, kebanyakan simulasi pengiraan selari boleh mendapat kelajuan 3 kali lebih laju berbanding pengiraan bersiri. Selain itu, pelaksanaan pilihan pengoptimuman waktu eksekusi tidak semestinya meningkatkan prestasi kod selari tersebut. Dalam kajian ini, penggunaan penyelesai selari yang berbeza iaitu penyelesai LU tidak lengkap (ILU) dan multigrid (MG) juga menunjukkan perbezaan dalam prestasi pengiraan selari.

Ringkasnya, kerja ini menunjukkan bahawa prestasi pengiraan selari jauh lebih baik daripada prestasi pengiraan bersiri. Walau bagaimanapun, terdapat banyak ruang untuk meningkatkan kelajuan pengiraan selari kerana didapati kadar peningkatan dalam kelajuan pengiraan selari mula menyusut apabila bilangan teras pemproses yang agak banyak mula digunakan. Selain itu, kajian ini menunjukkan bahawa beberapa penyesuaian perlu dilakukan untuk meningkatkan prestasi penyelesai selari versi optimum.

ABSTRACT

The Portable, Extensible, Toolkit for Scientific Computation (PETSc) library package is a popular computational tool among researchers that provides the capability of implementing parallel schemes in solving various kinds of finite element analysis through the use of its many routines and libraries that can be manipulated through runtime options. PETSc, as is claimed, allows a migration of existing scientific code from a sequential structure into a scalable, parallel paradigm structure in a convenient manner.

The main focus of this project is to evaluate the parallel performance of finite element code through the use of PETSc. Performance comparisons between different types of parallel solver are also done in this study. Different levels of parallel optimisation are also studied. The test platform consists of an Intel Xeon X5650 processor which has 6 physical cores (12 logical cores) and 24GB RAM and the FEM code is built with GCC compiler version 4.4 and OpenMPI 3.0.

The results from this project show a considerably good parallel performance compared to the serial performance. For instance, most of the parallel computation simulations could gain at least 3 times speedup compared to the serial computation. Apart from this, the implementation of build-time optimisation option does not necessarily increase the parallel performance of the code. In this study, the use of different parallel solvers, in this case, the incomplete LU (ILU) and multigrid (MG) solvers, also shows difference in the parallel performance.

In brief, this work shows that the performance of the parallel computation is considerably better than serial computation performance. However, there is much room for improvement of the scalability of the parallel computation speedups since the most speedups start to scale poorly when more number of cores used. Moreover, this work hints that some tunings of the compiler should be taken to enhance the optimised versions of the parallel solver.

CHAPTER 1: INTRODUCTION

1.1 The Portable, Extensible Toolkit for Scientific Computation (PETSc)

Developing parallel, nontrivial partial differential equation (PDE) solvers that deliver high performance is still difficult and requires long period of concentrated effort. The Portable, Extensible Toolkit for Scientific Computation (PETSc) is a toolkit that can ease these difficulties and reduce the development time. PETSc is portable and can be installed on any operating systems. The software has a powerful set of tools for the numerical solution of partial differential equations and related problems on high-performance computers (Gropp, 2001).

PETSc includes linear system solvers (sparse/dense, iterative/direct), nonlinear system solvers, tools for distributed matrices and support for profiling, debugging and graphical output. PETSc provides abstract interface that eases user to manipulate the routines provided by PETSc through runtime options ('PETSc: Features', n.d.).

1.2 Advantages of Using PETSc

The operation performed on the objects has abstract interface which is simply a set of calling sequences, which makes the use of PETSc easy during the development of large-scale scientific application codes. The use of libraries provided in PETSc could deliver high performance computing.

PETSc includes libraries of numerical methods that can be applied directly to applications. With abstraction level interfaces, process of porting PETSc into the existing application codes becomes easier than developing. The parallel finite element computation of interested problems is done with PETSc subroutines.

Routines provided with PETSc enable scalable parallelism. PETSc library provides its users a platform to develop applications exploiting fully parallelism and the flexibility to experiment many different models, linear and nonlinear large system solving methods avoiding explicit calls to MPI library. PETSc could reduce computational time in solving linear system since all linear solvers in PETSc are iterative solvers (iterative solvers use successive approximations of the solution

instead of solving a linear system Ax = b with Gaussian elimination which can take lots of time and memory).

1.3 Parallelisation Using PETSc

To enable parallel computation using PETSc, a message passing interface library is needed. The Message Passing Interface (MPI) is a library for parallel communication. It is a system for launching parallel jobs. PETSc users can use shared memory programming through MPI library. MPI has basic tools that send elementary datatypes between processors while PETSc has intermediate tools that insert matrix element in arbitrary location and do parallel matrix-vector product.

Parallel computation is preferred over the serial computation in finite element modelling because the problem can be solved faster. Hence, it is possible to obtain a more accurate solution or solve a more complicated problem in the same amount of time.

1.4 Problem Statement

The serial computing takes much longer time compared to parallel computing. The big interest in creating finite element analysis (FEM) program is the ability to implement parallel computing into its solvers. Though practically it is almost impossible to achieve ideal scaling of parallel speedup, the implementation of parallel computing still can greatly increase the speed of the FEM simulation of the program.

Free and open-source FEM code implementations are widely available. However, parallel FEM code is limited. On the other hand, PETSc is an established toolkit and library for parallel computing. The parallel performance of PETSc for using in FEM needs to be understood as far as computing time and memory requirements are concerned. PETSc could provide scalability, which is an important concern of parallel processing (‘Measuring Parallel Scaling Performance - Documentation’, 2016), through established MPI libraries such OpenMPI library.

1.5 Objectives of Project

i) To acquire and set up case studies for parallel solution with PETSc.
ii) To evaluate and analyse the performance of the parallel solvers through strong scaling analysis.
iii) To characterise the algebraic multigrid solver.

1.6 Scope of Work

This project focuses to implement parallel scheme for solving finite element modelling problem through the use of open-source software, which is PETSc toolkit in this case. The parallel schemes are implemented through runtime options which allow users to manipulate certain routines for implementation of parallel solver. The measuring of the parallel performance is done by using only one node of the HPC server.

This study aims to analyse the efficiency and performance of parallel computing of the finite element analysis problem through the use of a parallel solver available in PETSc. There are so many types of algorithm can be tested and used to set up the solver through runtime options. Different type of solvers will influence the perfomance of parallel computation. In this project, a parallel solver will be created based on the algebraic multigrid method (linear solver) and compared to another typical solver.

The problems involved for the implementation of the parallel solver only include the steady-state cases, for instance, a 3D steady-state thermal conduction problem. Hence, the efficiency of the simulation of the typical 3D conduction problem will provide a platform to analyse and evaluate the performance and effectiveness of parallel solver implementation for other similar finite element modelling cases.

CHAPTER 2: LITERATURE REVIEW

2.1 Use of PETSc

In finite element analysis of large scale, most of the computation time is spent on the solution of the assembled matrix system which sometimes includes time integration. This project however, does not include transient problem. The computation time could be reduced significantly if parallel solver is implemented. PETSc is one of the softwares that provide toolkit for development and implementation of parallel solver. PETSc provides an abstract interface for users to PETSc enables users to easily implement the formation and solution of finite element analysis. PETSc has been used for parallel finite element modelling in many areas which include computational fluid dynamics, aerodynamics, material science and even earthquakes (Knepley, Katz, & Smith, 2006).

2.2 Parallel Finite Element Modelling Using PETSc

One particular research has been done to develop a parallel finite element model for elasticity problems (Zhang, 2015). In this study of, the conjugate gradient (CG) algorithm is used since the linear system derived from finite element discretization of the elasticity problems is sparse and symmetric positive definite (SPD). A comparative analysis of serial performance of various preconditioned CG solvers is conducted. The precondioners tested are Jacobi, successive over-relaxation (SOR), and algebraic multigrid (AMG) and none preconditioner. Besides that, parallel performance of the finite element linear system solution stage on coarse and fine meshes for Jacobi and AMG also has been measured.

Apart from that, PETSc routines also had been used to create a parallel FEM scheme for the simulation of large scale thermochemical energy storage (Wang, Kolditz, & Nagel, 2017). Here, PETSc routines embedding parallelisation approach are used to conduct an efficient simulation of the thermochemical heat storage model. The finite element method used in this specific study includes weighted residual method for the weak forms formulation of the problem and isoparametric Galerkin

mixed finite element approach for the discretisation of the weak forms. Picard method (fixed point iteration) is used for the linearisation. The solver used is basically KSP (Krylov subspace) linear solver. The evaluation of the parallel computation in this particular study is based on the comparison with the serial computation of the simulation of the thermochemical heat storage problem.

In another study, the principle of thermal recovery simulation is analyzed through a parallel scheme created using PETSc (Liu, Xue, Shu, & Ma, 2010). This study compared the solution time for two preconditioners which are Additive Schwarz Method (ASM) and block Jacobi (BJACOBI). This study also compared parallel computation with the serial computation of the thermal recovery simulation.

There are also studies that were conducted to analyze hybrid parallel programming techniques (Castro, Paz, Storti, & Sonzogni, 2009; Paz, Storti, & Castro, 2010). In these studies, parallel hybrid finite element code is developed and its performance evaluated, using Message Passing Interface (MPI) through PETSc for communication between cluster nodes and Open Multi-Processing (OpenMP) for parallelism within a symmetric multiprocessing (SMP) node.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Hardware and System Requirements

Computations were carried out on the high-performance computing (HPC) server that consists of three nodes with each node consists of an Intel Xeon X5650 processor, a 24 GB RAM system and 1 TB hard disk. Each of the Intel Xeon X5650 processor has 6 physical cores with total of 12 logical cores (threads) which runs at 2.67 GHz speed with 3.1 GHz turbo speed. This project used one node.

The HPC server uses Red Hat Enterprise Linux Server release 6.7 (Santiago) operating system. A personal laptop is with any Linux operating system is needed to connect to the HPC server through a Linux console. In this project, openSUSE of Tumbleweed version is installed on the personal laptop.

Figure 3.1 shows the graphical reperesentation for the HPC system.

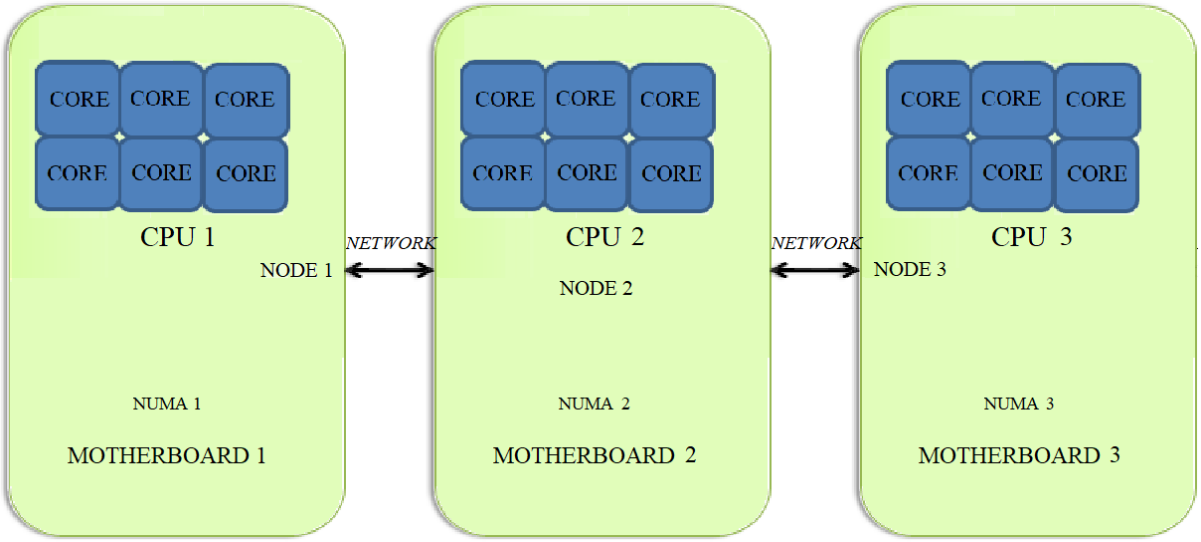

Figure 3.1: Graphical definition of processor, core and node.

Figure 3.1 above shows how a node is defined in this particular case. The HPC server has 3 nodes, however only one node is used in this project with total of 6 cores (12 threads) are usable.

3.2 Configuring PETSc

PETSc is installed on the HPC server. The version of the PETSc used is version 3.8.1 which was released on 4 November 2017.

In this work, the solution stage of the parallel finite element computation of interested problems is implemented using PETSc subroutines. Figure 3.2 below various numerical libraries and objects available through PETSc.

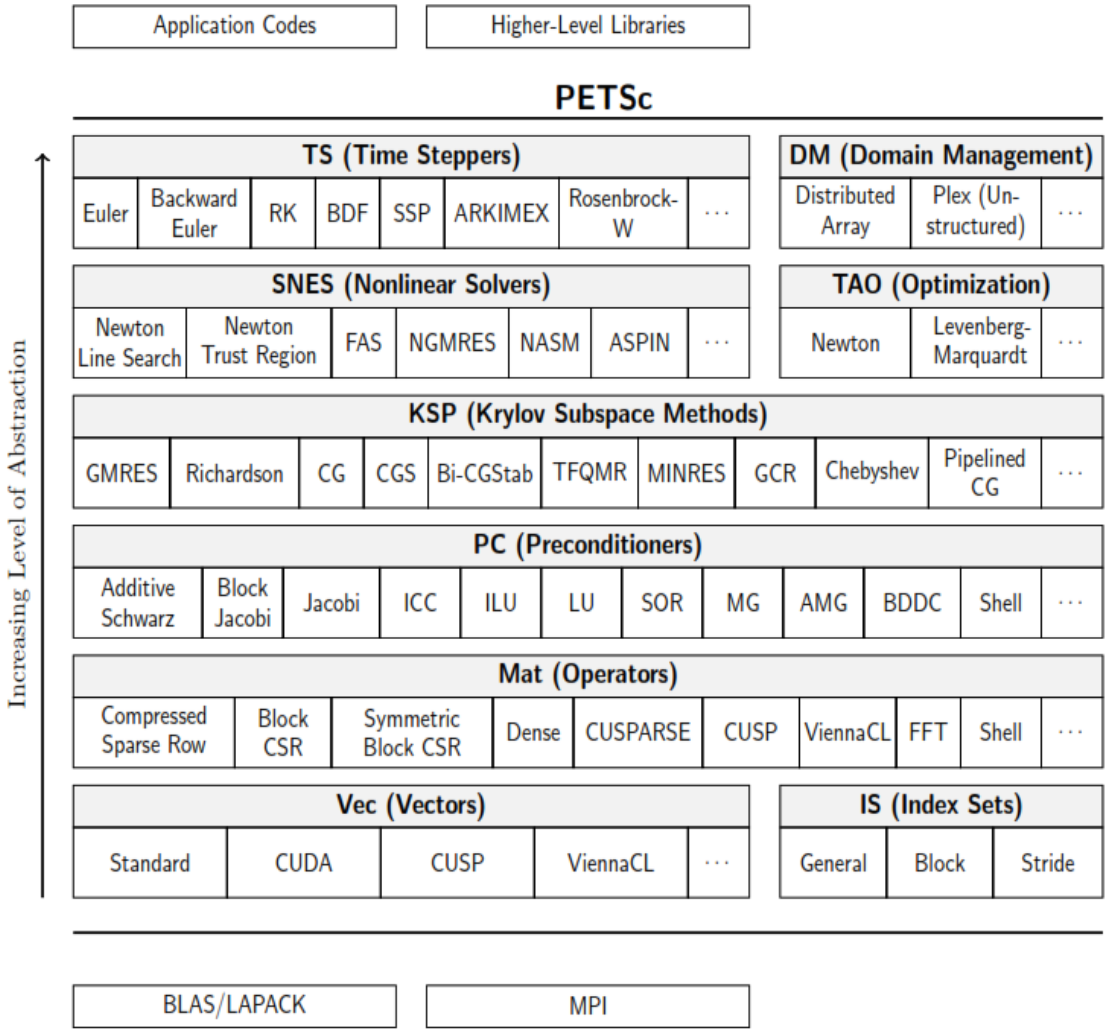

Figure 3.2: Numerical libraries of PETSc (Gropp, 2001).

From Figure 3.2, each library manipulates a particular family of objects such as vectors and also the operations performed on the objects. The operation performed on the objects has abstract interface which is simply a set of calling sequences. Thus, PETSc enables users to employ the level of abstraction that is most appropriate for a particular problem through runtime options.

In this project, the main objects that were manipulated through the runtime options were those under the Domain Management (DM) and Preconditioners (PC) libraries. The sample of the runtime options implemented is attached in Appendices section.

As shown in the Figure 3.2, Basic Linear Algebra Subprograms (BLAS), Linear Algebra Package (LAPACK) and Message Passing Interface (MPI) libraries are also needed to use PETSc. Next sections briefly provide some information on these libraries.

3.2.1 MPI

The Message Passing Interface (MPI) is a library specification that allows the passing of information between various nodes and clusters of computers. This enables the implementation of parallel programming in the HPC server. In this project, the specific library used for the MPI implementation was OpenMPI. The option to use OpenMPI library was done in the configuration sricpt of PETSc. OpenMPI is an open-source, portable implementation of the MPI standard. The version of the OpenMPI used was OpenMPI 3.0 together with GCC compiler version 4.4.

3.2.2 BLAS and LAPACK Library

The BLAS (Basic Linear Algebra Subprograms) provide routines for standard building blocks to enable the basic operations of vector and matrix. There are three main levels in BLAS. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS mainly perform matrix-matrix operations. BLAS are commonly used in the development of high quality linear algebra software, such as LAPACK.

LAPACK (Linear Algebra Package) provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. LAPACK also allows efficient computation on shared-memory vector and parallel processors (‘LAPACK — Linear Algebra PACKage’, 2017).

3.3 Case Studies

Two PETSc application codes were used as main case studies for parallel computation analysis in this project. The code solves a 2D driven cavity problem while the second code solves 3D thermal conductivity problems that are modelled in terms of Poisson's equation.

3.3.1 2D Driven Cavity Problem

The first case study is a about a 2D nonlinear driven cavity problem.

Figure 1 below shows the graphical representation of the problem. This problem is solving values for temperature, velocity and vorticity of 2D driven cavity problem inside a unit square domain.

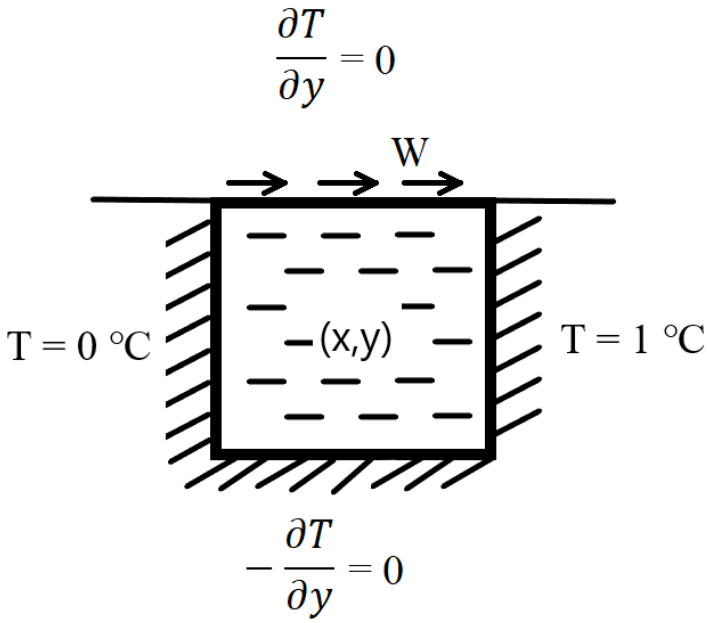

Figure 3.3: Boundary conditions for the nonlinear driven cavity problem.

Figure 3.3 shows the driven cavity problem represented in a unit square domain.

Boundary Conditions

No-slip, rigid-wall conditions are used for the walls of the cavity. As shown in Figure 1, Dirichlet conditions are used for temperature on the left and right walls, and insulation homogeneous Neumann conditions are used for temperature on the top and bottom walls. All the boundary conditions can be expressed by equations:

$$T = 0 \text{ on the left wall} \tag{3.1}$$

$$T = 1 \text{ on the right wall} \tag{3.2}$$

$$\frac{\partial T}{\partial y} = 0 \text{ on the top wall} \tag{3.3}$$

$$-\frac{\partial T}{\partial y} = 0 \text{ on the bottom wall} \tag{3.4}$$

The lid velocity, W, is set to different values for different grid refinement level. Table 3.1 below shows different values used for the lid velocity.

Table 3.1: Lid velocity for different grid refinement levels.

Grid Refinement Level	Lid Velocity (m/s)
6 (Coarse)	2.6846e-05
7 (Medium)	6.7465e-06
8 (Small)	1.6910e-06
9 (Fine)	4.2330e-07

Governing Equations

A number of partial differential equations are used to generate the mesh of structured grid for solving the nonlinear driven cavity problem.
The velocity of the fluid in the cavity is given by

$$\boldsymbol{\rho} = U_{x,y}\boldsymbol{i} + V_{x,y}\boldsymbol{j} \tag{3.5}$$

The vorticity is represented by equation

$$\Omega = -\nabla_y U + \nabla_x V \tag{3.6}$$

where along each constant coordinate boundary, the tangential derivative is zero which is given by equations

$$-\Delta U - \nabla_y \Omega = 0 \tag{3.7}$$

$$-\Delta V - \nabla_x \Omega = 0 \tag{3.8}$$

Two more partial differential equations used to model this problem are given by

$$-\Delta\Omega + \nabla.([U * \Omega, V * \Omega]) - GR * \nabla_x T = 0 \tag{3.9}$$

$$-\Delta T + PR * \nabla.([U * T, V * T]) = 0 \tag{3.10}$$

where *GR* is the Grashof number, the dimensionless temperature gradient and *PR* is the Prandtl, the dimensionless thermal or momentum diffusivity ratio. In this problem both *PR* and *GR* are set to 1.0.

The problem is uniformly discretised in each of x and y in the unit square domain.

3.3.2 3D Thermal Conduction Problem with Constant Coefficient

This problem aims to determine values of u(x,y,z) in one unit cube domain such that

$$-\nu.\nabla^2 u = f \tag{3.11}$$

where $\nu = 1$ and $f = 4$.

Hence, the equation can be rewritten as

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = 4 \tag{3.12}$$

The boundary conditions of the problem are defined as Dirichlet conditions based on the exact solution of the equation 3.12. The exact solution is:

$$u(x,y,z) = \frac{2}{3}(x^2 + y^2 + z^2) \quad (3.13)$$

Hence, the boundary conditions for all six faces of the cube domain are given by the following equations:

$$u = \frac{2}{3}(x^2 + y^2 + 1) \text{ on the top area } (z = 1) \quad (3.14)$$

$$u = \frac{2}{3}(x^2 + y^2) \text{ on the bottom area } (z = 0) \quad (3.15)$$

$$u = \frac{2}{3}(x^2 + z^2) \text{ on the front area } (y = 0) \quad (3.16)$$

$$u = \frac{2}{3}(x^2 + z^2 + 1) \text{ on the back area } (y = 1) \quad (3.17)$$

$$u = \frac{2}{3}(y^2 + z^2 + 1) \text{ on the right area } (x = 1) \quad (3.18)$$

$$u = \frac{2}{3}(y^2 + z^2) \text{ on the left area } (x = 0) \quad (3.19)$$

Figure 3.4 shows the orientation of axis and notation of surface orientation of 3D unit cube of the 3D thermal conduction problem.

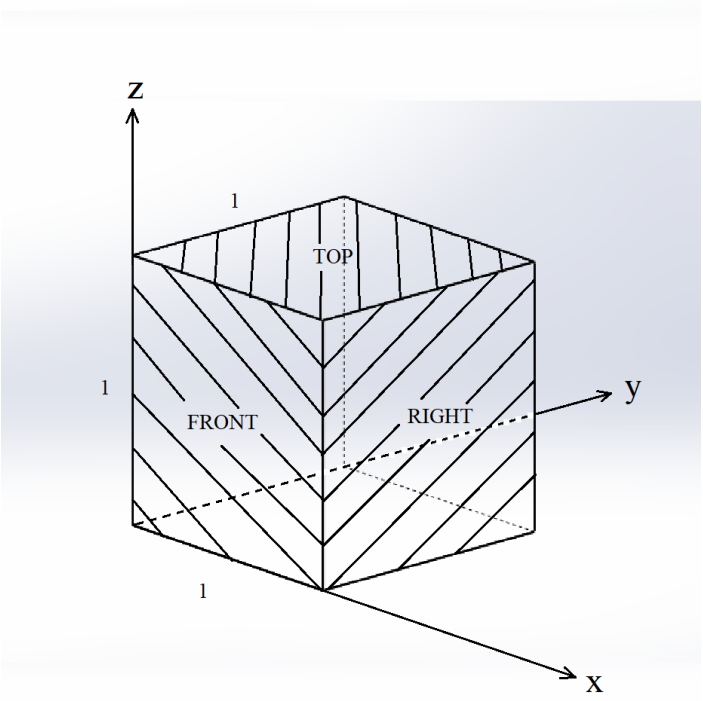

Figure 3.4: The cube domain and showing the surface orientation of the cube domain.

3.3.3 3D Thermal Conduction Problem with Nonlinear Coefficient

This problem aims to determine values of *u(x,y,z)* in one unit cube domain such that

$$-\nabla.\nu(x,y,z)\nabla u = f \tag{3.20}$$

where f is given by equation

$$f = 16(x^2 + y^2 + z^2) \tag{3.21}$$

The exact solution is used for the Dirichlet conditions with a nonlinear coefficient (p-Laplacian with p = 4). The exact solution is given by equation

$$u(x,y,z) = \frac{2}{3}(x^2 + y^2 + z^2) \tag{3.22}$$

Hence, for this specific case, the value of ν is

$$\nu = \frac{12}{5}(x^2 + y^2 + z^2) \tag{3.23}$$

With

$$r^2 = x^2 + y^2 + z^2,$$

then

$$-\left(\frac{\partial}{\partial x}r^2\frac{\partial u}{\partial x} + \frac{\partial}{\partial y}r^2\frac{\partial u}{\partial y} + \frac{\partial}{\partial z}r^2\frac{\partial u}{\partial z}\right) = \frac{20}{3}(x^2 + y^2 + z^2) \tag{3.24}$$

The Dirichlet boundary conditions are based on exact solution are given by same equations 3.14 to 3.19, the exact same conditions as in linear case in described in section 3.3.2.

3.4 Implementation of PETSc Routines

The subsections below provide brief explanation of the PETSc routines that are used in implementing the parallel solver through runtime options.

3.4.1 Control of Domain Management

The meshing for 2D driven cavity problem is structured. The example of main routine used to vary the grid size to refinement level 2 is

> –da_refine 2

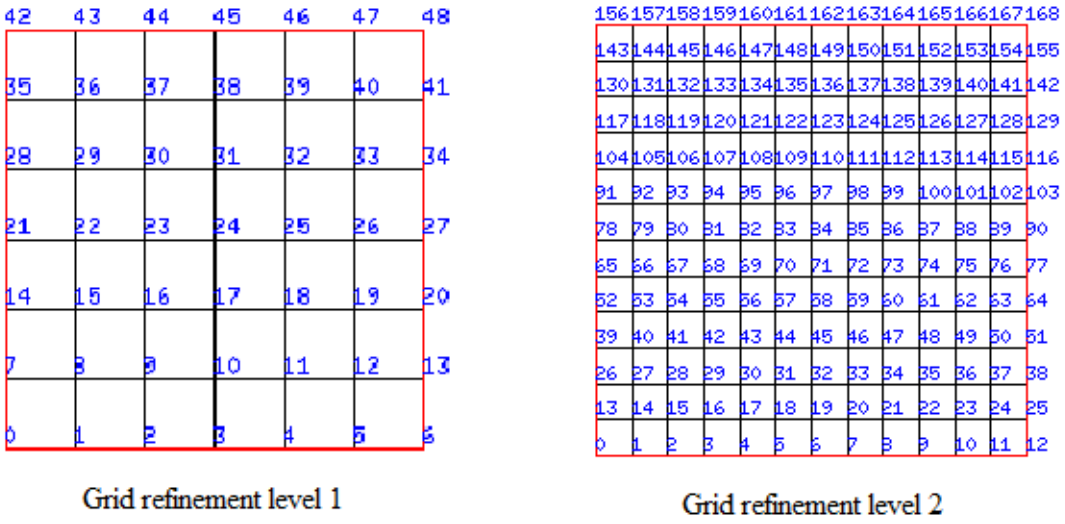

Figure 3.5: Images showing how the grid sizes vary from one level to one level.

Figure 3.5 shows how grid size varies when the grid refinement level was increased by one level. In this study 4 different grid refinement levels are used starting from level 6 to level 9. Table 3.2 below shows the number of unknowns associated with each level used.

Table 3.2: Number of unknowns for different grid sizes used.

Grid Refinement Level	No. of Unknowns
6 (coarse size)	148,996
7 (medium size)	592,900
8 (small size)	2,365,444
9 (fine size)	188,891,152

For 3D thermal problem, the meshing is unstructured. Table 3.3 shows the main differences between structured mesh and unstructured mesh.

Table 3.3: Main differences between structured and unstructured grids or meshes.

Structured Grid	Unstructured Grid
Domain is divided into a structured assembly of quadrilateral cells.	Computational domain is divided into an unstructured assembly of computational cells.
Each interior nodal point is surrounded by the same number of mesh cells (or elements).	The number of cells surrounding each interior node is not necessarily constant.
Directions within the mesh can be immediately identify by associating a curvilinear co-ordinates system (represented by i, j, k indices).	The nodes and the elements have to be numbered.

Figure 3.6 shows the unstructured grid of the 3D thermal problem.

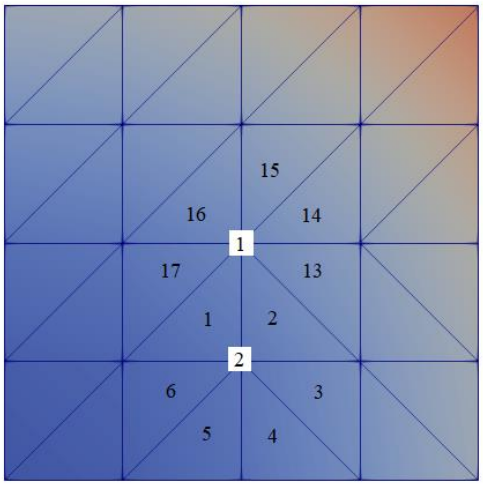

Figure 3.6: Unstructured mesh of the 3D thermal problem.

From Figure 3.6, it is clearly shown that node 1 and 2 are surrounded by different number of elements. Node 1 is surrounded by 7 elements while node 2 is surrounded by 6 elements.

In this project, meshing refinement level is fixed to level 7 both linear and nonlinear cases. 16,581,375 unknowns are solved for this particular mesh size. Runtime option shown below is the main command used to refine the mesh to level 7.

> –dm_refine 7

Figure 3.7 shows how mesh size refined when meshing refinement level is increased by one level.

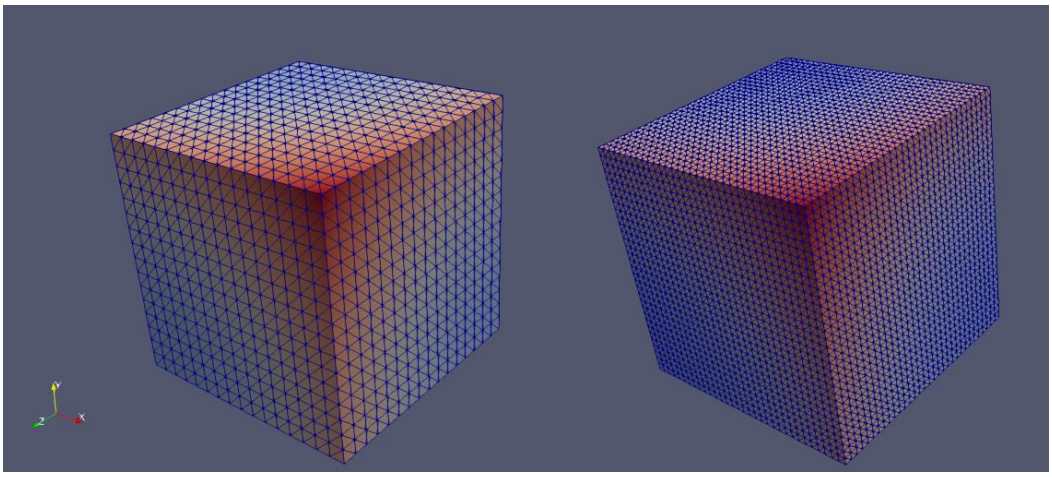

Figure 3.7: Pictures from the left to right indicate the meshing refinement level 4 and 5 respectively.

3.4.2 Using MPI

PETSc is configured together with OpenMPI library to allow the manipulation of number of cores used in running the simulation of the problem codes. This will enable the measuring of the scaling efficiency of PETSc. This could indicate how efficient PETSc parallel solvers are when using increasing number of cores. Example use of routine for this purpose is:

> –mpirun –n *no. of cores*

The number of cores ranges from 1 to 12.

3.4.3 Assigning Type of Solver

Figure 3.8 gives a brief definition of solver and preconditioner in solving a system of equations.

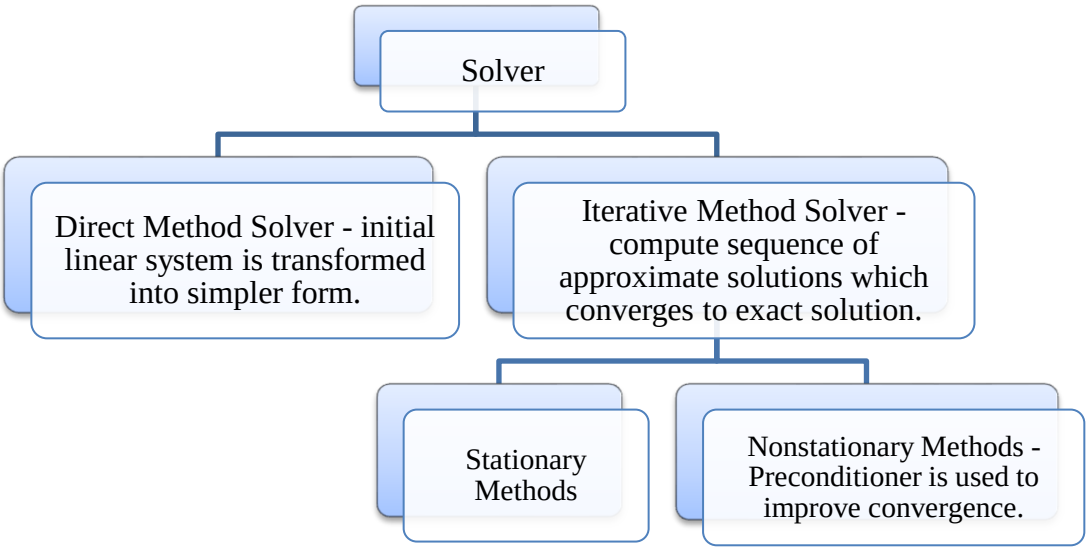

Figure 3.8: Definition of solver.

The main types of solver used include Newton's method, incomplete LU factorisation (ILU) method and multigrid method. The solver type could be specified using command:

> –pc_type
> –snes_type

The Scalable Nonlinear Equations Solver (SNES) used is Newton's line search method.
Newton's method is really useful in solving nonlinear equations (Briggs, Henson, & McCormick, 2000).

Suppose that, to solve the scalar equation *F(x) = 0*, the *F* term is expanded in a Taylor series about an initial guess *x*:

$$F(x+s) = F(x) + sF'(x) + \frac{s^2}{2}F''(\xi) \tag{3.25}$$

where ξ is between x and $x + s$. If $x + s$ is the solution, then (neglecting the higher-order terms) the series becomes $0 = F(x) + sF'(x)$, from which $s = -F(x)/F'(x)$. Thus, the initial guess of x can be updated using:

$$x \leftarrow x - \frac{F(x)}{F\prime(x)} \tag{3.26}$$

The ILU solver is a modification of LU factorisation method. Suppose that a matrix system Ax = b can be expressed by a mathematically equal preconditioned linear system expressed as follows:
$M^{-1}Ax = M^{-1}b$, where M is a preconditioner. One simple way to construct preconditioners is to split A into A = M − N. In theory, any splitting with nonsingular M which is close to A in some sense can be used.

The Jacobi preconditioner is a commonly used preconditioner with the form of M = diag(A). The successive over relaxation (SOR), Gauss-Seidel or subgrid scale (SGS) preconditioning matrix is of the form M = LU, where L and U are the lower triangular part and the upper triangular part of A, respectively. Another simple way of defining a preconditioner is incomplete factorisation of the matrix A. These incomplete LU factorisation (ILU) preconditioners perform decomposition of the form A = LU−R, where L and U are the lower and upper parts of A with the same nonzero structure and R is the residual of the factorisation (Zhang, 2015).

Multigrid method mainly can be divided into two types. Geometric multigrid is used for structured grid while algebraic multigrid is used for unstructured grid.
Suppose that for matrix equation Ax = b, the x can approximated as v which then gives the error, e = x – v. Then, the residual is r = b – Av. Since e = x - v, the system Ax = b can be written as A(v + e) = b which means that Ae = b - Av ≡ r. Residual equation is given by equation Ae = r while the residual correction is given by equation x = v + e. The nonlinear residual equation is given by A(v + e) - A(v) = r.

3.4.4 Optimisation of Code

The codes in PETSc could be run either without optimisation or with optimisation option. To run the code in optimisation option, the PETSc was configured with debugging option. Debugging option was used in both case studies.

On the other hand, the debugging option is turned off to optimise the code. The optimization level could be manipulated from level 0 to level 3. Despite of some errors during configuring with optimisation of levels 2 and 3, the code for 2D driven cavity problem could still be run with the options. The screenshot that shows errors while configuring for the implementation of optimisation level 2 is attached as Appendix D.

Hence, all optimisation levels were used for the 2D driven cavity problem while for the 3D thermal problems, only optimisation level 1 is used to be compared with the code performance without optimisation. The 3D thermal problem could not run with the optimisation of levels 2 and 3. This is due to the errors during configuration of optimisation of levels 2 and 3. The script samples for configuring PETSc without optimisation and with optimisation are attached as Appendix A and Appendix B respectively.

3.4.5 Obtaining Graphics for 2D Driven Cavity Problem

PETSc has integrated various routines that could be called during runtime options. Among them are the options to use X Window system, a system for managing a windowed graphical user interface in a distributed network. Hence it enables the display of graphical solution for 2D driven cavity problem. Below are some runtime options to enable the graphics display of this specific problem solution:

> –contour –draw_save –draw_pause –1

3.4.6 Obtaining Graphics for 3D Problem

For a 3D problem, the graphical result need to be saved as a file before opening the saved file using a supported software, which in this case is ParaView. To enable the view in ParaView, the file is usually saved in .vtu or .vts format. For this specific study, the file is saved in .vtu extension since the code involves unstructured mesh.

> –vec_view vtk:*filename*.vtu:vtk_vtu

3.4.7 Obtaining Data for Analysis of the Parallel Implementation

The output during each problem solution is saved into a log file that records all the data and information needed for the analysis of parallel performance of the codes. The information needed includes total running time of the code, total memory used and total number of unknowns. All these information are recorded and saved into a log file through options:

> –log_view –memory_view –ksp_view –malloc_info

A sample of log file is attached as Appendix C.

3.5 Analysis of Results

The main interests of the analysis are to study about the speedup of the codes, the memory used, number of unknowns associated and relative performance of parallel solvers used. Hence, based on the simulations' result, a number of graphs were plotted to analyse the interested parameters. The graphs plotted include information on speedup, solution time and also comparison between different optimisation levels.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 2D Driven Cavity Problem

The case study of this problem had been successfully set up by using incomplete LU decomposition solver. The main interest for this 2D driven cavity problem is to analyse the speedup of parallel computation for different sizes of meshing and to study the effect of optimisation of the code.

4.1.1 Graphical Solution

Figure 4.1 shows the velocity in x-direction of the fluid in the cavity.

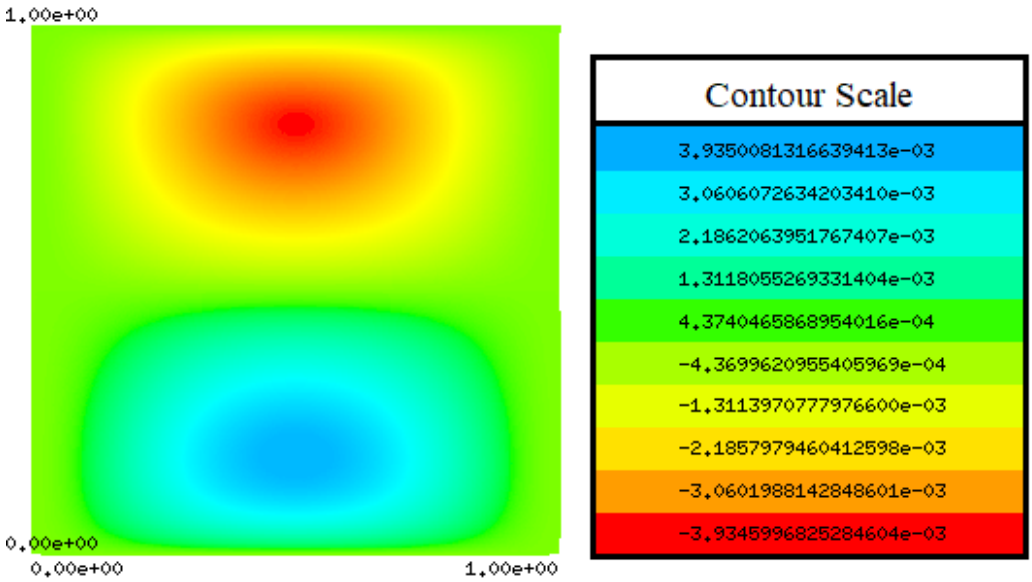

Figure 4.1: Velocity in x direction inside the cavity.

Based on Figure 4.1, the graphical solution shows that the velocity in x-direction range between -0.0039 m/s to 0.0039 m/s.

Figure 4.2 shows the velocity in y-direction of the fluid in the cavity which lies in the same range as that of velocity in x-direction.

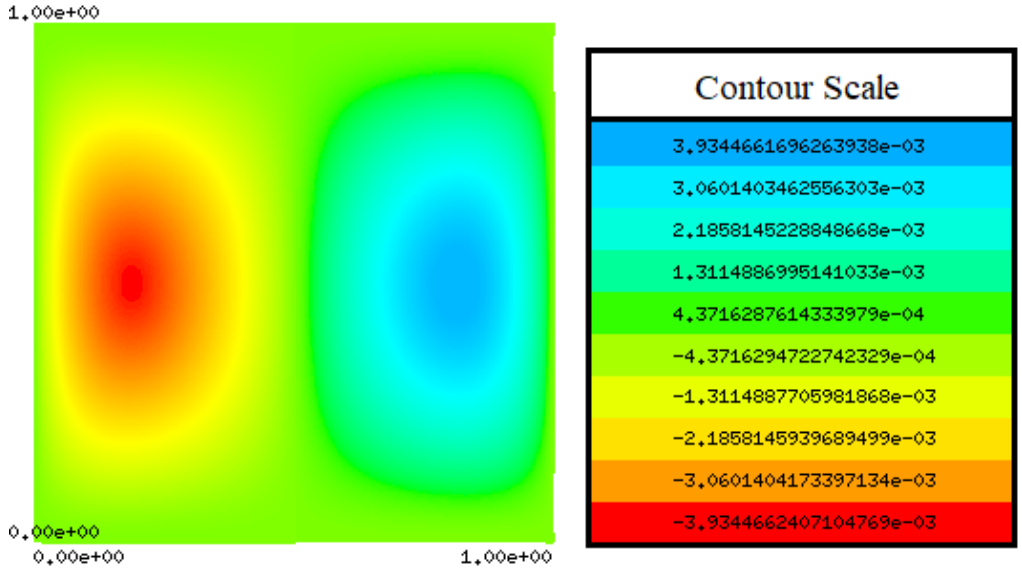

Figure 4.2: Velocity in y direction inside the cavity.

Figure 4.3 shows the PETSc solution for the vorticity of the fluid inside the cavity.

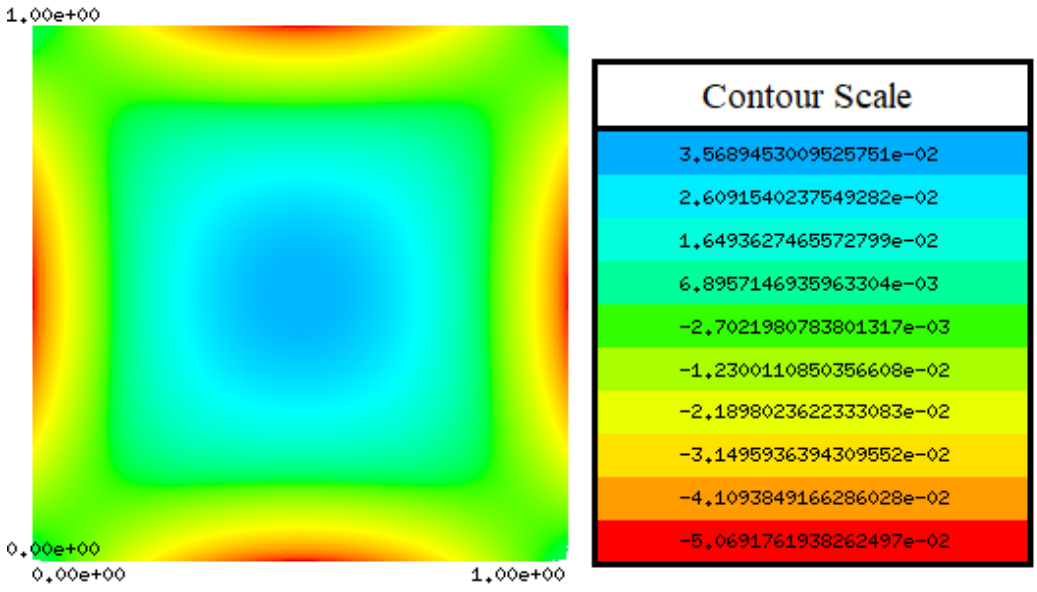

Figure 4.3: Vorticity of the fluid inside the cavity.

The vorticity ranges from -0.0507 rad/s to 0.0357 rad/s.

Figure 4.4 shows the temperature distribution inside the cavity.

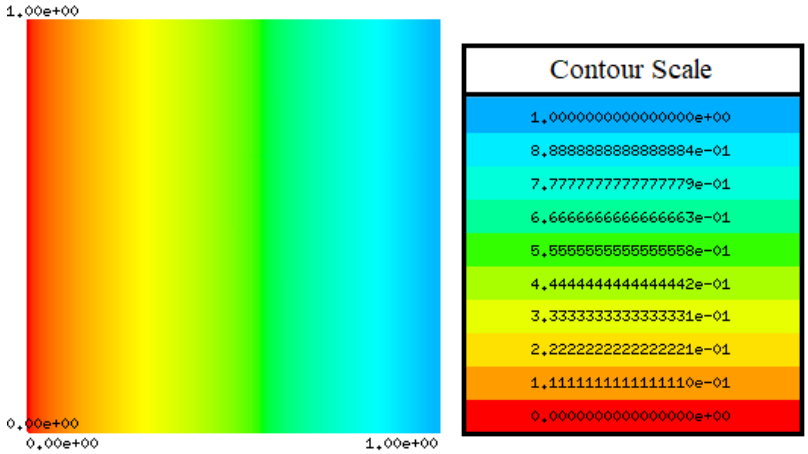

Figure 4.4: Temperature distribution of the fluid inside the cavity.

From the Figure 4.4, the temperature varies in x-direction from 0 ℃ at the left wall to 1℃ at the right wall as described by Dirichlet conditions.

4.1.2 Speedup of Code

Figure 4.5 shows the scalability analysis of the code without optmisation.

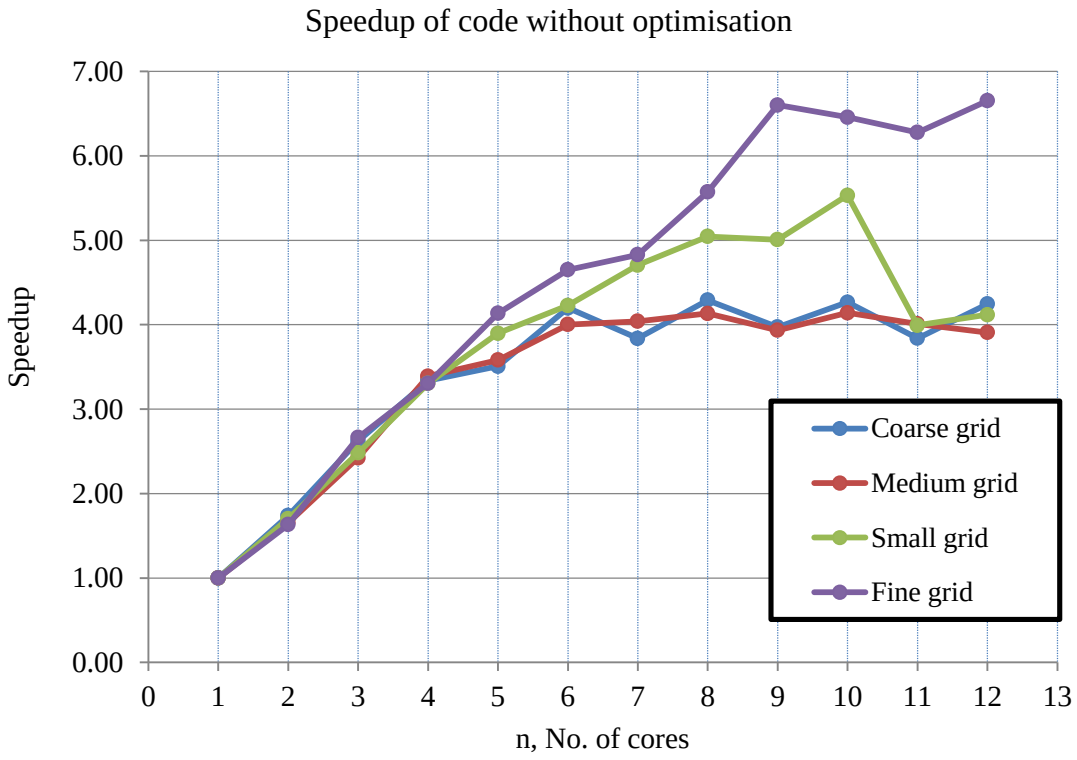

Figure 4.5: Speedup of 2D driven cavity problem code without optimisation.

Based on Figure 4.5, the speedup of the code starts to hit the peak with n = 6 for coarse and medium grid. The speedup then stays about the same until n = 12.

For small grid, the speedup increases until n = 10 but then significantly drops when the number of cores to 11 and remains constant with maximum cores used. For fine grid, the speedup gradually increases from n = 1 core to n = 12.

Figure 4.6 shows the speedup for optimisation level 0.

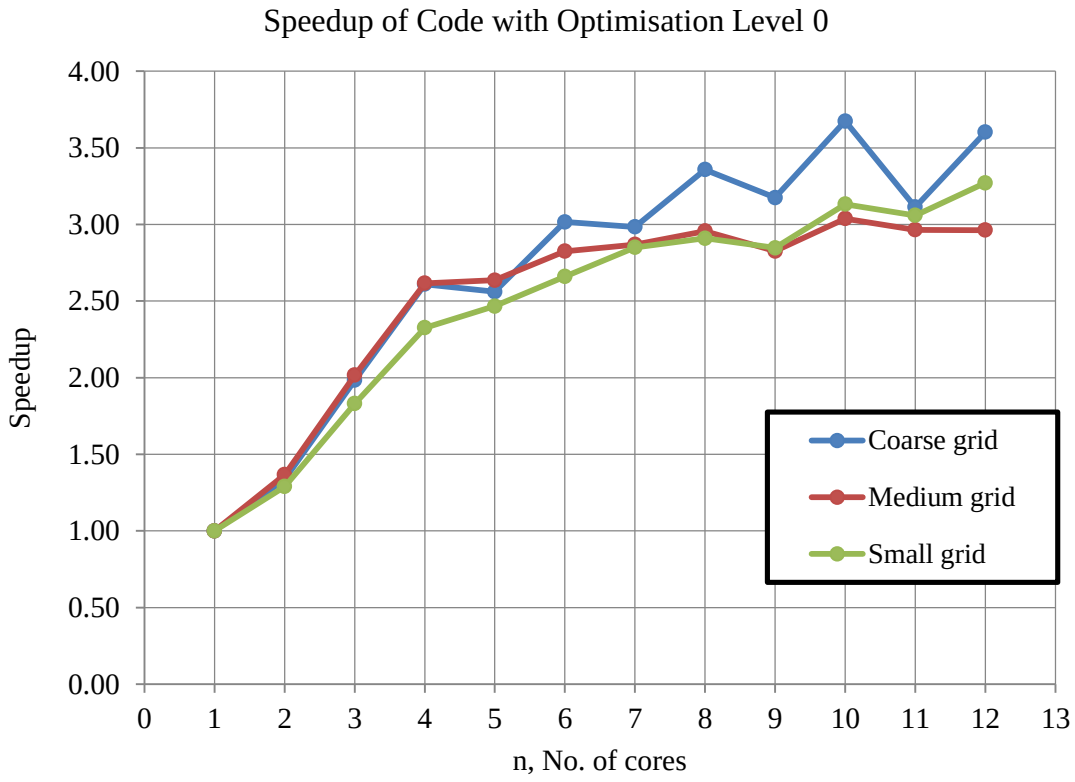

Figure 4.6: Speedup of code for three different grid sizes with optimisation level 0.

From Figure 4.6, the speedup for coarse gradually increases but then starts to fluctuate at n = 8 until n = 12. The speedup for the speedup trend for both medium seems about the same. The speedup seems starts to hit the peak at 6 cores. This speedup remains constant until 12 cores for medium grid but it shows a slight increase from n = 11 to n = 12 for small grid. Generally, the peak speedup of the parallel computation achieved lies in between 3 to 4 compared to the serial computation with n = 1.