

# **Implementation of Multigrid Method for Solving Coupled Problem**

By:

**Lim Kean Siang**

(Matric no.: 137820)

Supervisor:

**Dr. Muhammad Razi Bin Abdul Rahman**

July 2021


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 been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed  ..... (LIM KEAN SIANG)


Date.....

Statement 1: This journal is the result of my own investigation, except where otherwise stated. Other sources are acknowledged by giving explicit references. Bibliography/references are appended.

Signed  ..... (LIM KEAN SIANG)

Date.....

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

Signed  ..... (LIM KEAN SIANG)

Date.....

## **ACKNOWLEDGEMENT**

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Muhammad Razi Bin Abdul Rahman for giving me this opportunity to learn and study coupled problems in detail under his supervision. I am very thankful for his support, guidance, advice, and patience throughout the whole project. This project would not be complete without his guidance. He has provided me proper and patient guidance in the simulations of the project.

I would also like to thank my friends for providing morale support whenever I face difficulties in my work.

Lastly, I would like to thank my parent and my brother for being the pillar of support in my life. Their supported my studies and dealt with the problems at home, enabling me to focus on my work.

## ABSTRAK

*Coupled problem* adalah sistem yang melibatkan beberapa sistem fizikal yang saling berinteraksi, adalah perkara biasa di alam, dan ini menjadikan kajiannya berguna. Kaedah multigrid adalah algoritma yang berguna untuk menyelesaikan masalah multiresolusi. Namun, sastera mengenai penggunaan kaedah multigrid untuk menyelesaikan *coupled problem* kekurangan. Oleh itu, projek ini dilaksanakan dengan tujuan menggunakan kaedah multigrid untuk menyelesaikan *coupled problem* dan mengkaji ciri-ciri prestasinya. Contoh kod PETSc diubah dan digunakan untuk mengkaji pelaksanaan multigrid untuk menyelesaikan *coupled problem*. Kod berjaya dijalankan untuk menghasilkan solusi. Kod itu kemudian diubah dengan *solver* dan saiz grid yang berbeza untuk mengkaji ciri-ciri prestasinya. Data prestasi yang dihasilkan diplotkan untuk mengenal pasti hubungan antara prestasi dan saiz grid, dan perbandingan prestasi untuk *solver* yang berbeza. Saiz grid yang besar didapati memberi kesan negatif kepada prestasi kod. *Solver* yang paling cekap dijumpai dari *solver* yang dikaji dalam projek ini.

## **ABSTRACT**

Coupled problems were systems involving multiple interacting physical systems, were commonplace in nature, making study of it useful. Multigrid methods were algorithms useful for solving multiresolution problems. However, there was a lack of literature on the use of multigrid methods for solving coupled problems. Hence, this project was carried out with the aim of implementing multigrid on coupled problems and studying its performance characteristics. A PETSc example code was modified and used to study the implementation of multigrid for solving a coupled problem. The code was successfully executed to yield a solution. The code was then executed with different solver types and grid sizes to study its performance characteristics. The resulting performance data was plotted to identify the relationship between performance and grid size, and the performance comparison of different solver types. Large grid sizes were found to negatively impact performance of code. The most efficient solver was found from the solvers studied in this project.

## TABLE OF CONTENTS

<b>DECLARATION.....</b>	<b>I</b>
<b>ACKNOWLEDGEMENT.....</b>	<b>II</b>
<b>ABSTRAK.....</b>	<b>III</b>
<b>ABSTRACT.....</b>	<b>IV</b>
<b>TABLE OF CONTENTS .....</b>	<b>V</b>
<b>LIST OF FIGURES .....</b>	<b>VII</b>
<b>LIST OF TABLES .....</b>	<b>VIII</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>IX</b>
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>10</b>
1.1 Research Background.....	10
1.2 Problem Statement .....	12
1.3 Objective .....	12
1.4 Scope of Research .....	12
<b>CHAPTER 2 LITERATURE REVIEW .....</b>	<b>13</b>
2.1 Multigrid Method .....	13
2.2 Coupled Systems.....	14
2.3 PETSc.....	15
2.4 Solvers.....	16
2.4.1 ksponly solver.....	16
2.4.2 newtonls solver.....	16
2.4.3 newtontr solver .....	18
2.4.4 qn solver .....	20
<b>CHAPTER 3 RESEARCH METHODOLOGY .....</b>	<b>21</b>
3.1 Introduction .....	21
3.2 Hardware & Software.....	24

<b>CHAPTER 4</b>	<b>RESULTS AND DISCUSSION</b>	<b>25</b>
4.1	Introduction	25
4.2	Solution	26
4.2.1	Coupled Problem	26
4.2.2	U given fixed K	28
4.2.3	K given fixed U	30
4.3	Tabulated Performance Results	32
4.4	Plotted Performance Graphs	34
4.4.1	Performance against Grid Size	34
4.4.1(a)	ksponly solver	35
4.4.1(b)	newtonls solver	36
4.4.1(c)	newtontr solver	38
4.4.1(d)	qn solver	40
4.4.2	Performance Characteristics Comparison of different solvers	42
4.5	Coupled vs Non-Coupled Performance	45
<b>CHAPTER 5</b>	<b>CONCLUSION AND FUTURE WORK</b>	<b>47</b>
5.1	Conclusion	47
5.2	Future Work	47
<b>REFERENCES</b>		<b>48</b>
<b>APPENDICES</b>		<b>50</b>

## LIST OF FIGURES

		<b>Page</b>
Figure 3-1	Overall Methodology .....	21
Figure 4-1	Graph showing solution of $u(x)$ against $x$ for coupled problem.....	26
Figure 4-2	Graph showing solution of $k(u)$ against $x$ for coupled problem. ....	27
Figure 4-3	Graph showing solution of $u(x)$ against $x$ for fixed $k(u)$ . .....	28
Figure 4-4	Graph showing fixed value of $k(u)$ against $x$ . .....	29
Figure 4-5	Graph showing solution of $k(u)$ against $x$ for fixed $u(x)$ . .....	30
Figure 4-6	Graph showing fixed value of $u(x)$ against $x$ . .....	31
Figure 4-5	Line graph of Flops and Max Memory against Grid Size for ksponly solver.....	35
Figure 4-6	Line graph of Flops and Max Memory against Grid Size for newtonls solver.....	36
Figure 4-7	Line graph of Total Iteration against Grid Size for newtonls solver. 37	37
Figure 4-8	Line graph of Flops and Max Memory against Grid Size for newtontr solver.....	38
Figure 4-9	Line graph of Total Iteration against Grid Size for newtontr solver.. 39	39
Figure 4-10	Line graph of Flops and Max Memory against Grid Size for qn solver. ....	40
Figure 4-11	Line graph of Total Iteration against Grid Size for qn solver.....	41
Figure 4-12	Bar Graph of Flop against Solver Types.....	42
Figure 4-13	Bar Graph of Maximum Memory against Solver Types. ....	43
Figure 4-14	Bar Graph of Total Iteration against Solver Types. ....	44



## LIST OF TABLES

		<b>Page</b>
Table 2-1	Default PETSc options for newtonls solver.....	17
Table 2-2	Default PETSc options for newtontr solver. ....	19
Table 2-3	Default PETSc options for qn solver. ....	20
Table 4-1	Solution of $u(x)$ for coupled problem. ....	26
Table 4-2	Solution of $k(u)$ for coupled problem. ....	27
Table 4-3	Solution of $ux$ given fixed $ku$ . ....	28
Table 4-4	Fixed value of $k(u)$ . ....	29
Table 4-5	Solution of $ku$ given fixed $ux$ . ....	30
Table 4-6	Fixed value of $u(x)$ . ....	31
Table 4-7	Performance data for different grid sizes using ksponly solver.....	32
Table 4-8	Performance data for different grid sizes using newtonls solver. ....	32
Table 4-9	Performance data for different grid sizes using newtontr solver. ....	33
Table 4-10	Performance data for different grid sizes using qn solver. ....	33
Table 4-11	Performance characteristics of coupled and non-coupled problems for ksponly solver. ....	45
Table 4-12	Performance characteristics of coupled and non-coupled problems for newtonls solver.....	45
Table 4-13	Performance characteristics of coupled and non-coupled problems for newtontr solver.....	45
Table 4-14	Performance characteristics of coupled and non-coupled problems for qn solver. ....	46

## LIST OF ABBREVIATIONS

CPU	Central Processing Unit
ODE	Ordinary Differential Equation
PETSc	Portable, Extensible Toolkit for Scientific Computation
SNES	Scalable Nonlinear Equations Solvers
Tao	Toolkit for Advanced Optimization

# CHAPTER 1

## INTRODUCTION

### 1.1 Research Background

Multigrid methods are a class of algorithms used to solve computational problems (Borz`1, n.d.). It is primarily used for solving linear and non-linear boundary value problems (Borz`1, n.d.).

Multigrid methods are generally accepted as the fastest numerical methods for the solution of elliptic partial differential equations (Trottenberg et al., 2000). They are also considered to be among the fastest methods for many other problems (Trottenberg et al., 2000). If the multigrid idea is generalised to structures besides grids, one yields multiresolution, multilevel, or multiscale methods, which can be used to solve very different problems defined by different type of structures (Trottenberg et al., 2000).

Multigrid methods are most useful for solving multiresolution problems, i.e., problems with different scales. Multiresolution problems oscillate smoothly at some parts and oscillate rapidly at other parts (Borz`1, n.d.). Use of coarse discretization scale for smoother parts and finer discretization scale for the rapidly oscillating part would solve the multiresolution problem in a more optimal manner (Borz`1, n.d.). Hence, multiresolution methods such as multigrid which can use different scales for each component of a problem is important for solving multiresolution problems most optimally.

Coupled problems are systems where multiple physical systems interact with one another, requiring simultaneously solving all interacting systems to obtain a solution (Zienkiewicz & Taylor, 2000). For example, solving for the electric current of a wire requires simultaneously solving for the resistance of the wire which depends on temperature. In this example, there is coupling between the electrical and thermodynamic system. The electrical system causes heating of the wire which affects the thermodynamic system, while the thermodynamic system cause temperature changes that influences electrical properties of the wire. Due to the relationship of the 2 systems that affect each other, both must be solved simultaneously to obtain a

solution. Most systems in the real world are coupled problems that involve interaction of different fields of physics, making solving of coupled problems important.

Another example of coupled problems is nonlinear thermoelectric effect. Thermoelectric effects refer to phenomenon such as the Seebeck effect and Peltier effect (Tritt, 2002). In Seebeck effect, a temperature difference applied across 2 dissimilar materials would generate a potential difference (Tritt, 2002). Peltier effect is the opposite of Seebeck effect, whereby a current driven across the dissimilar material would cause a heat flow (Tritt, 2002). Thermoelectric effects therefore involve a coupling of thermodynamic and electric systems and is a coupled problem. Thermoelectric effects enable conversion of energy from heat to electricity and vice versa. This grants it a multitude of applications such as in heating, refrigeration, air-conditioning, electricity generation, and measuring instruments (Patidar, 2018). Its multitude of applications is due to its simplicity of construction and mechanism, and its portability (Patidar, 2018). It is simple in that it consists of just 2 different metals joint together and has no moving parts. The wide applications brought by a coupled problem such as the thermoelectric effect shows just how important coupled problems are in our world. Therefore, the study of coupled problems is important and have huge implications.

Solution of nonlinear coupled problems are complicated and can only be done through numerical methods. The complicated nature of coupled problems makes it more computationally expensive to solve compared to non-coupled problems which involves just 1 physical system. Nonlinear problems are also more complicated and more expensive to solve than linear problems. Nonlinear coupled problem in this project will be solved numerically using a computer.

The interest of this project was in using multigrid methods for solving coupled problems.

## **1.2 Problem Statement**

There was a lack of literature on the use of multigrid methods for solving nonlinear coupled problems. More studies could be done on the implementation of multigrid methods for solving coupled problems and the resulting performance characteristics of such implementations. Given the benefits of using multigrid methods and the importance of coupled problems in the world, there are reasons for embarking on studies in this area.

## **1.3 Objective**

The specific objective of this project was:

1. Implement a multigrid method for solving a coupled problem.
2. Set up a 1D nonlinear coupled problem for performance analysis
3. Identify performance characteristics of using different solvers and grid sizes.

## **1.4 Scope of Research**

This project involves simulation of a specific 1D coupled problem involving 2 equations using a multigrid method. Different solvers and grid sizes were used for the simulated coupled problem to obtain performance characteristics. Simulation was done using C++ language and PETSc library. The code was created by modifying an existing example code.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Multigrid Method

Numerical solution of a sparse system of linear equations can be solved by methods that largely fall into two large categories: direct methods and iterative (relaxation) methods (Briggs et al., 2000). Direct methods determine the solution exactly in a finite number of arithmetic steps but are restricted primarily to systems that arise from separable self-adjoint boundary value problems (Briggs et al., 2000). Iterative methods begin with an initial guess which acts as an approximate solution (Briggs et al., 2000). This approximate solution is improved by iterations and should ideally converge to the exact solution (Briggs et al., 2000). Classical relaxation methods are easy to implement and may be successfully applied to more general linear systems than most direct methods but suffers from limitations and multigrid methods have evolved from attempts to deal with these limitations (Briggs et al., 2000).

Relaxation methods with a multigrid setting are competitive with the fast direct methods when applied to model problems, and they have more generality and a wider range of application (Briggs et al., 2000). The original multigrid ideas have been extended to what are more appropriately called multilevel methods (Briggs et al., 2000). Purely algebraic problems (for example, network and structural problems) have led to the development of algebraic multigrid (AMG) (Briggs et al., 2000).

## 2.2 Coupled Systems

The coupling of a coupled systems may be strong or weak depending on the degree of interaction between the involved systems (Zienkiewicz & Taylor, 2000). In a current carrying conductor where its resistance is subject to its temperature, the temperature dependence of the conductor will determine how well coupled the system of electricity and thermodynamics is. A high temperature dependence will result in a high degree of interaction between temperature and resistivity causing the electrical system to be heavily influenced by the change of temperature in the thermodynamic system, leading to a strong coupling.

Coupled systems can be classified into 2 classes (Zienkiewicz & Taylor, 2000). Class I is where coupling occur at domain interfaces through imposed boundary conditions (Zienkiewicz & Taylor, 2000). An example of this is an aerofoil, where 2 physically different domains, fluid and the structure interact (Zienkiewicz & Taylor, 2000). Class II is where various domains overlap either partially or totally (Zienkiewicz & Taylor, 2000). Here, coupling occurs through the governing differential equations describing different physical phenomenon (Zienkiewicz & Taylor, 2000). An example of this, is the previously mentioned current carrying conductor whose resistivity is temperature dependant. In that example, the temperature domain overlaps the electrical conducting domain of the conductor completely and the interaction between the systems occur within the body of the conductor rather than the boundary. The coupled problem studied in this project is of class II.

### **2.3 PETSc**

Provide software for the scalable (parallel) solution of algebraic systems arising from partial differential equation simulations (PDEs) (Smith, n.d.). It is a supported research code, is portable to virtually any system, and is largely funded by the US Department of Energy (Smith, n.d.). PETSc places an emphasis on algorithmic and discrete mathematics interface, whereby programmers manipulate solvers, sparse matrices, nonlinear equations (Smith, n.d.). PETSc makes uses of object-oriented programming where you deal with performing operations on a data rather than the data itself (Smith, n.d.).



## 2.4 Solvers

Several non-linear solvers from PETSc were explored and used for solving the coupled problems. The section below introduces the various solvers used in this project.

### 2.4.1 ksponly solver

ksponly was a nonlinear solver that performs one Newton step without a line search and does not compute norms (*SNESKSPONLY*, n.d.). The solver is used for solving linear problems using the SNES interface, without additional overhead in the form of vector operations (*SNESKSPONLY*, n.d.).

### 2.4.2 newtonls solver

newtonls was a Newton based nonlinear solver which uses line search, and it was the default solver in SNES (*SNESNEWTONLS*, n.d.).

When finding the roots of a function, an efficient method of approximating the solution is needed (Burton, 2009). Newton-Raphson method or more commonly known as Newton's method is an efficient algorithm for approximating the solution to a function or a system of equations (Burton, 2009). It involves an initial guess of the solution, followed by successive iterations that convergences the guess towards the solution (Burton, 2009).

A line search is an algorithm for finding the minimum of a function (Gockenbach, n.d.). There are many types of algorithms for line search, with backtracking being the most used algorithm (Gockenbach, n.d.). When minimising the function, a descend direction and step length needs to be identified (Gockenbach, n.d.). Given a descend direction, backtracking line search first tries a step length of one, and gradually reduces it until an acceptable step length is found (Gockenbach, n.d.).

Table 2-1 Default PETSc options for newtonls solver.

Options	Default Value
Type of line search.	Backtracking line search over the L2 norm of the function.
Order of line search.	Cubic order.
Computation of final norms in the line search.	True
Alpha used in determining if reduction in function norm is sufficient.	$10^{-4}$
Maximum stepsize the line search will use.	$10^8$
Minimum lambda the line search will tolerate.	$10^{-12}$

### 2.4.3 newtontr solver

newtontr was a “Newton based nonlinear solver that uses a trust region.” (*SNESNEWTONTR*, n.d.).

Trust region is another method aside from line search which is used for minimising a function. For trust region method, a model such as a quadratic is used to approximate the objective function and this model is minimised in a neighbourhood of the defined trust region (Dutta, 2016). Forward steps in the model are taken according to the model, with the step size determined before improving the step direction or both at the same time (Ye, 2014).

If a notable decrease is seen due to the forward step, then the model is believed to represent the objective function in a good way (Ye, 2014). However, if the improvement is too subtle or negative, the model is believed to badly represent the objective function within the region (Ye, 2014). The size of the trust region can be adjusted and is reduced if the model is bad, while it is increased if the model is reliable (“Trust-Region Methods,” 2006).

Trust region algorithms are less mature compared to line search and are more limited in applications, as research on trust region mostly started in the 80s (Dutta, 2016). However, trust region methods are more reliable, robust and have very strong convergence properties (Dutta, 2016). Trust regions methods are effective in nonlinear optimizations and are also useful for non-convex optimizations and non-smooth optimizations (Dutta, 2016). Line search methods can be considered as a special case of trust region methods (Dutta, 2016).

Table 2-2 Default PETSc options for newtontr solver.

<b>Trust Region Parameters</b>	<b>Default Value</b>
mu	0.25
eta	0.75
sigma	0.0001
delta0	0.2
delta1	0.3
delta2	0.75
delta3	2

#### 2.4.4 qn solver

qn refers to Limited-Memory Quasi-Newton methods for solving nonlinear systems (*SNESQN*, n.d.).

They are a class of optimising methods used when full Newton's method is too time consuming or difficult to use (Cericola, 2015). Specifically, qn is used to find the global minimum of a twice-differentiable function (Cericola, 2015). There are advantages to using qn over full Newton's method for expansive and complex non-linear problems (Cericola, 2015). The procedure of qn is the same as a regular Newton's method but with modifications to the Hessian Matrix step which depends on the type of qn used (Cericola, 2015).

qn is faster and has no need to solve for second derivatives and linear system of equations (Cericola, 2015). However, qn require more convergence steps and has less precise convergence path compared to full Newton's method (Cericola, 2015).

Table 2-3 Default PETSc options for qn solver.

Options	Default Value
QN type.	LBFGS
Type of scaling performed on inner Jacobian.	scalar
Type of line search.	Critical point secant line search assuming $F(x) = \text{grad } G(x)$ for some unknown $G(x)$ .
Maximum stepsize the line search will use.	$10^8$
Minimum lambda the line search will tolerate.	$10^{-12}$

## CHAPTER 3

### RESEARCH METHODOLOGY

#### 3.1 Introduction

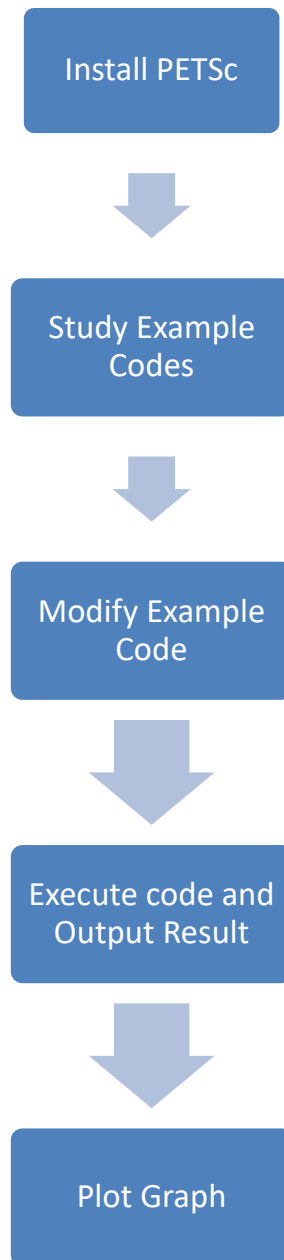


Figure 3-1 Overall Methodology

“PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well

as hybrid MPI-GPU parallelism. PETSc (sometimes called PETSc/Tao) also contains the Tao optimization software library.”(*PETSc/Tao: Home Page*, n.d.). PETSc was used to run the simulation of the coupled problem.

Initially, installation of PETSc was done on a laptop which uses Windows 10. A Linux emulator, Cygwin was installed on the Windows 10 laptop to enable the laptop to run PETSc commands. This was necessary as PETSc uses Linux commands. Cygwin was then used for installing PETSc. After installation, relevant PETSc example codes were studied to gain familiarity with basic usage of PETSc functions which was used to do simulations.

Example file ex28 from the PETSc SNES tutorials was used for doing the coupled problem simulation. This example uses multigrid for solving a 1D coupled problem which involves 2 nonlinear ODEs. The coupled problem used in ex28 was as follows:

$$-\frac{d}{dx}\left(k(u)\frac{du}{dx}\right) = 1, \quad 0 < x < 1, \quad u(0) = 0, \quad u(1) = 1$$

$$e^{k-1} + k = ((1 + u)^{-1} + (1 + \left(\frac{du}{dx}\right)^2)^{-1})^{-1}$$

The first equation expresses  $u(x)$  in terms of  $k(u)$  which makes it a coupled problem since  $k(u)$  was not fixed. The second equation expresses  $k(u)$  in terms of  $u$ . This makes it necessary to solve both equations simultaneously to obtain a solution. Also specified was the Dirichlet boundary condition of  $u(x)$ , and  $x$  was limited to a range from 0 to 1.

This example file was modified to use a chosen grid size and solver, and to output the solution on a text file and print the performance data on the console. The example file was a C source file and was modified using a C++ code editor. After modification, compilation of the modified file was done using Linux ‘make’ command in Cygwin.

An ‘option’ file was also created and written in plain text file format using a notepad. The list of PETSc ‘options’ that were going to be used during runtime were written in the ‘option’ file. PETSc ‘options’ were user specified commands used to modify the PETSc executable during runtime. The use of PETSc ‘options’ to issue

commands offer greater flexibility compared to writing the desired commands in the C source file. This was because PETSc 'options' enable changing of the user specified commands without requiring recompilation of the PETSc C source file. On the other hand, the commands written in the C source file requires recompilation whenever it was modified. Commands such as changing of solver type was issued through the PETSc 'option' file to enable more efficient changing of codes during simulation.

After compilation of the C source file was done, it was executed in Cygwin together with the 'option' file and results in outputs of the solution and performance data. The C source file was placed in the same directory as the option file, and the code in the C source file was written to enable it to read data in the option file when ran. The resulting output was then plotted in Matlab to yield a graph.



### 3.2 Hardware & Software

This project was a simulation/programming type.

Hardware:

1. A Windows 10 laptop. To run the software.

Software:

1. PETSc. Code library for scientific computations. Provides functions used in C source files of PETSc code.
2. Cygwin. To emulate Linux environment on Windows, which was necessary for running PETSc. Used to install PETSc, compile PETSc C source files, and execute PETSc executables.
3. Any C++ editor. To edit the C source file of PETSc code.
4. Matlab. To plot graphs for output data.