A NEW DIVERGENCE METHOD FOR HEAT TRANSFER WITH NEUMANN BOUNDARY CONDITION

FATIN NABILAH BINTI SABRI

CIVIL ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

SEPTEMBER 2017

A New Divergence Method for Heat Transfer with Neumann Boundary Condition

by

Fatin Nabilah Binti Sabri
19318

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Engineering (Hons)
(Civil Engineering)

SEPTEMBER 2017

Universiti Teknologi PETRONAS
32610, Bandar Seri Iskandar,
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

A New Divergence Method for Heat Transfer with Neumann Boundary Condition

By

Fatin Nabilah Binti Sabri
19318

A dissertation submitted to the
Civil Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(CIVIL ENGINEERING)

Approved by,

_____

(DR AIRIL YASREEN BIN MOHD YASSIN)

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR, PERAK
September 2017

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

(FATIN NABILAH BINTI SABRI)

ABSTRACT

New Divergence Theorem is a new formulation for the simplest case of heat transfer problem involving Neumann Boundary Condition which combines two different numerical techniques which are Finite Element Method (FEM) and Finite Volume Method (FVM). The use of numerical techniques to solve such problems is therefore considered essentials since the powerful Finite Element Method (FEM) is capable in solving heat transfer analysis by giving a piecewise approximation of the domain. This study aims to evaluate the hypothesis of combining FEM with the FVM and to develop a new formulation of heat transfer problem involving Neumann Boundary Condition. Finite Volume Method (FVM), which uses the concept of Green Divergence Theorem, where a surface integral can be transformed to line integral has less accuracy since it develops the assumption of constant flux. Meanwhile, for FEM, the accuracy is higher since FEM is based on polynomial interpolation using shape function. Accuracy or convergence rate increases as the order of polynomials increases. Comparing between linear interpolation and polynomials interpolation, the former will converge faster. By combining these two methods, the assumption of constant flux in FVM can be eliminate with the general polynomial interpolation of FEM. Another disadvantage of Finite Element Method (FEM) is its difficulty in accurately representing a geometrically complex domain. Taking the advantage of exact geometry representation of Finite Volume Method (FVM), the New Divergence Theorem is able to evaluate at the boundary of the domain, which may yield more accurate solution. In this study, the first step involves the establishment of partial differential equation (PDE) for heat transfer problem. By solving the equation based on a series of discretization work, the equation can be arranged in a matrix form and the temperature can be obtained by using MATLAB. Verification and parametric study was done and the New Divergence Method was verified to provide converged results. Despite that, the rate of convergence is lower than the established Finite Element Method. As the New Divergence Method is still in the developing phase, further study and enhancement are needed to improve the performance and compare it with the existing method.

ACKNOWLEDGEMENT

All praise to The Almighty, The Most Merciful for granting me His never-ending blessing and strength throughout the completion of my Final Year Project. I would like to express my deepest gratitude towards my supervisor, Dr Airil Yasreen Mohd Yassin for his endless support and constructive comments despite being busy with his overwhelming commitment and responsibilities. Continuous guidance by my supervisor is what steered me to the completion of this project. A big thank you also to those who was directly or indirectly involved to complete this project. To my dearest parents, there is nothing more I want than to make them proud after all the sacrifices they have made for me. I am beyond thankful for their encouragement and understanding in times when I have been occupied with the project. To my dearest parents, there is nothing more I want than to make them proud after all the sacrifices they have made for me. I am beyond thankful for their encouragement and understanding in times when I have been occupied with the project.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background Study

Numerical method is a method used to solve complicated mathematical equations such as Ordinary Differential Equation (ODE) and Partial Differential Equation (PDE). This method uses an integral form of equations governing mass, energy and momentum balance for an arbitrary control volume (DemirdiiC and Muzaferija,1994). It gives approximations to solve the problem by guessing the values and functions involved. Regardless of how precise the approximation is, in most cases, the exact answer cannot be obtained. However, numerical method is still the most used method because working out the exact answer using different approach seems quite impossible due to accuracy as compared to solving using numerical method and might be time consuming.

There are different types of numerical techniques available which are;

i. Finite Element Method (FEM)
ii. Finite Difference Method (FDM)
iii. Boundary Element Method (BEM)
iv. Meshless or Meshfree Method (MESHFREE)

For this study, the main interest is on Finite Element Method (FEM) and Finite Volume Method (FVM). FEM is one of the most commonly used method in solving problems related to structural mechanics as well as heat transfer and fluid mechanics. This method provides a systematic procedure in discretizing a domain where the domain will be subdivide into smaller and simpler parts called finite element. It is an

approach to solve complex partial differential equations that oversee the behavior of the structure in terms of the stresses, strains and other variables.

Meanwhile, Finite Volume Method (FVM) is a discretization technique for partial differential equations (Zhiqiang Cai, 1990) especially those that arise from physical conservation laws. FVM is frequently used methods in fluid mechanics as well as in heat and mass transfer problems. W.Q. Tao (2002) explained that they have the advantages of easily discretizing the governing equations and treating discontinuous physical phenomena, such as capturing shock waves and implementing upwind scheme.

Finite Volume Method uses the concept of Green Divergence Theorem. Green Divergence Theorem is a mathematical statement where in three-dimensional problem, volume integral can be converted to area integral.

$$\int_V \nabla . \overline{\mathrm{T}}\, dV = \int_A \overline{\mathrm{T}} . n\ dA \tag{1.1}$$

Meanwhile, for two-dimensional condition, surface integral can be transformed to line integral. Applying Divergence Theorem in discretizing heat transfer equation yields the following:

$$\int_A \frac{\delta^2 T}{\delta x^2} + \int_A \frac{\delta^2 T}{\delta y^2}\, dA = \int_A \frac{\delta^2 T}{\delta x^2}\, dA + \int_A \frac{\delta^2 T}{\delta y^2}\, dA$$

$$= \int_A \nabla . \frac{\delta T}{\delta x}\, dA + \int_A \nabla . \frac{\delta T}{\delta y}\, dA$$

$$= \int_A \frac{\delta T}{\delta x} . n_x\, dx + \int_A \frac{\delta T}{\delta y} . n_y\, dy \tag{1.2}$$

Eqn. (1.2) shows the transformed heat transfer equation when Divergence Theorem was applied.

1.2 Problem Statement

Finite Element Method (FEM) has been known as vortex-centric method whilst Finite Volume Method (FVM) is known for its volume centric nature. Whilst the former has been shown as being very rigor in various continuum analysis (e.g. solid and fluid), the latter has emerged as an effective numerical technique especially those that involve moving boundary. However, despite its practicality, FVM is still based on linear interpolation and hence the assumption of constant flux. It is the hypothesized in this study that such a limitation would not be necessary if the general polynomial interpolation of FEM is combined with the divergence theorem which is the basis of FVM. It is the interest of this study to evaluate such hypothesis and idea by developing new formulation for the simplest case of heat transfer problem involving Neumann Boundary Condition.

1.3 Objectives of the Study

This study is to evaluate the hypothesis of combining FEM with the divergence theorem and to develop a new formulation of heat transfer problem involving Neumann Boundary Condition. Thus, the objectives of this study are listed as follows:

1) To formulate, to write in Matlab software and to verify the new method for heat transfer problem
2) To conduct performance study by comparing against result obtained by FEM in terms of rate of convergence (in example: number of degree of freedoms)

1.4 Scope of Study

To fulfil the objectives of this study, below are the scopes that will be covered in this study:

1) Derivation of mathematical formulations and computer programming for 2D heat transfer problem which is in linear and isotropic condition using MATLAB.
2) Solving heat transfer problem with Neumann boundary condition will be allowed throughout the study

3) Verification of the formulation and codes applied by comparing the results against existing results.

CHAPTER 2

LITERATURE REVIEW

2.1 Numerical Techniques

As a general rule, analytical and experimental solutions are inapplicable in handling some heat transfer problem with significant complexity. For this reason, many numerical methods such as finite element (FE) methods (FEM), Finite Difference Methods (FDM), Finite Volume Methods (FVM), etc, have been developed. According to DemirdiiC and Muzaferija (1994), numerical method uses an integral form of equations governing mass, energy and momentum balance for an arbitrary control volume.

2.1.1 Finite Element Method

The Finite-Element Method, in its by and by acknowledged structures can be credited to no lesser a man than Richard L. Courant. According to Barkanov (2001), today, the finite element method (FEM) is considered as one of the well-established and convenient technique for the computer solution of complex problems in different fields of engineering: civil engineering, mechanical engineering, nuclear engineering, biomedical engineering, hydrodynamics, heat conduction, geo-mechanics, etc.

FEM is a formulation for both the continuous and the discrete problems. The formulation is obtained by multiplying the original equation by a “test function”. The continuous unknown is then approximated by a linear combination of “shape” functions. These shape functions are the test functions for the discrete formulation. The resulting equation is then integrated over the domain by the method of integration by parts.

The FVM is sometimes called a “discontinuous finite element method” since the original equation is multiplied by the characteristic function of each grid cell and the discrete unknown may be considered as a linear combination of shape functions.

Indeed, the FEM can be much more precise than FVM when using higher order polynomials, but it requires an adequate functional framework which is not always available in industrial problems. Other more precise methods are, for instance, particle methods or spectral methods but these methods can be more expensive and less robust than the finite volume method.

The common methods available for the solution of general field problems, like elasticity, fluid flow, heat transfer problems, etc., can be classified as presented in Fig. 2.1.

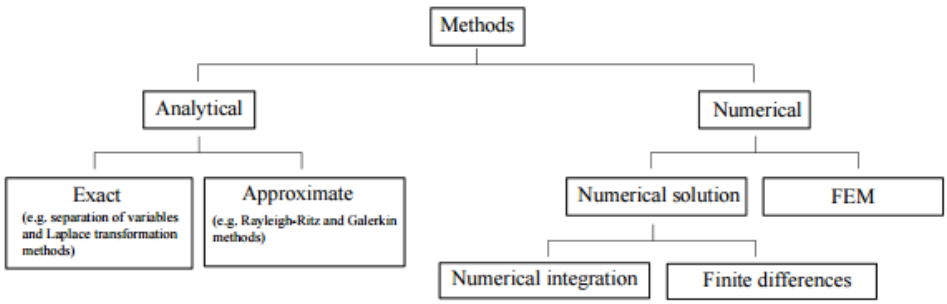


Figure 2.1: Different method in solving engineering problem

Nowadays, stabilized finite element methods are very popular and prevalently used in the numerical solutions of partial differential equations. There are many advantages from the stabilization methods such as enhancing stability, using simple node-based continuous Lagrange element and does not require special meshes are example of advantages from the stabilization methods.

2.1.2 Finite Volume Method

The finite volume method is a discretization strategy which is appropriate for the numerical simulation of different sorts (elliptic, parabolic or hyperbolic, for instance) of conservation laws. It has been broadly utilized as a part of a few building fields, for example, fluid mechanics, heat and mass transfer or petroleum engineering.

According to Oden (1991), some of the important features of the finite volume method are similar to those of the finite element method. The finite volume method is locally conservative because it is based on a " balance" approach: a local balance is written on each discretization cell which is often called "control volume"

The finite volume method is very unique from the finite difference method or the finite element method. The principle of the finite difference method is, given a number of discretization points which may be defined by a mesh, to assign one discrete unknown per discretization point, and to write one equation per discretization point. At each discretization point, the derivatives of the unknown are replaced by finite differences through the use of Taylor expansions. However, FDM is ends up noticeably hard to utilize when the coefficients associated with the condition are discontinuous. With the finite volume method, discontinuities of the coefficients will not be any problem if the mesh is chosen such that the discontinuities of the coefficients occur on the boundaries of the control volumes.

2.2 Coupling Different Numerical Techniques

According to J. Krok and J. Orkisz (2016), the coupling of FEM with other methods for computational analysis of boundary value problems (especially with meshless methods) is not new as it reaches early seventies of the previous century. From that point forward, this issue has been later explored by numerous different scientists and it is still being worked on these days.

In most cases, the principle thought behind the coupling strategies is either to wipe out or decrease downsides of one strategy (e.g. time-consuming mesh generation, low rate of derivatives convergence) or to utilize the upsides of other strategy (super-convergence, least squares smoothing, independent integration mesh and so on) in more powerful way.

One of the examples on the coupling method is the study on the combination of Finite Element Method (FEM) with Meshless Finite Difference Method (MFDM) in thermomechanical problems by J. Jaśkowie and S. Milewski in 2016. The MFDM is a representative approach among a wide class of meshless techniques. FEM utilizes structural of components (meshes) and unknown function approximation by means of the appropriate polynomial shape functions, built upon finite elements and associated with nodal degrees of freedom. MFDM, as one of the oldest meshless methods, may use both regular meshes or totally arbitrarily irregular clouds of nodes, without any imposed structure, like finite element, regular mesh or mapping restrictions.

The kind of issues where the coupling is legitimized is for instance the issues in which selected parts of the domain are under rapid change of subjected load, boundary conditions and/or material parameters (e.g. due to temperature dependency).

In such issues, applying a meshless approach (particularly the MFDM) to those areas is by all accounts sensible due to its higher flexibility in nodes generation, super-convergence of solution derivatives, as well as element-free determination of the approximation base (in general). In addition, non-linear problems, in which the basic linear problem has to be solved many times or situation in which the approximation mesh/cloud of nodes requires frequent refinements (e.g. due to the adaptation technique or shrinking of the material), are other examples.

In the coupling method, the domain is divided into two sub domains for FEM and MFDM, respectively. One scalar parameter, translated as a width of thin material layer between those two sub domains, has to be set. The scalar parameter depends neither on the type of considered

2.3 Use of Numerical Method in Structural Engineering

Utilization of finite element tools in building industry has not just permitted the effective building items, but also the development of accurate design method. Generally, engineers have utilized research laboratory testing to examine the structural behavior of steel building items and frameworks subject to the normal wind and earthquake before coming out with appropriate design of the structural members.

Be that as it may, such dependence on tedious and costly research laboratory testing has frustrated advance around there. In any case, progresses in the field of computer helped building amid the most recent two decades have changed this circumstance altogether in many designing work.

One of the applications of numerical method in structural engineering is the study on steel woof and wall cladding system in Australia. These claddings often suffer from local pull-through failures at their screw connections under wind uplift/suction loading caused by storms. [1,2]

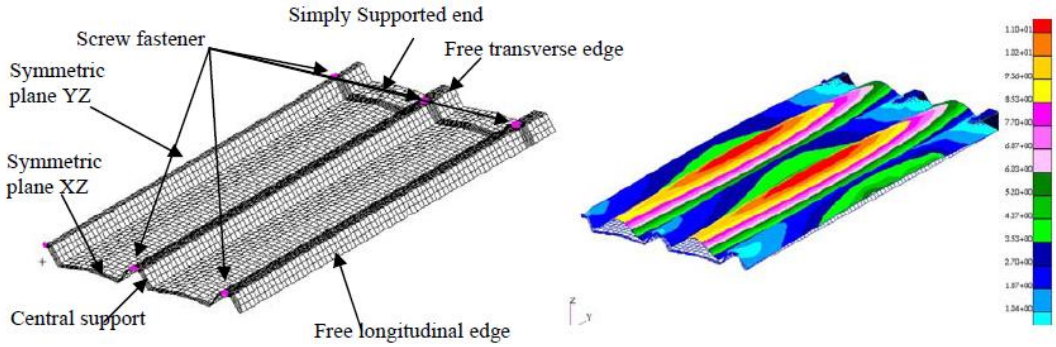


Figure 2.2 : Application of numerical method in structural engineering

CHAPTER 3

METHODOLOGY

Chapter 3 will discuss and outline the approach of methodological to this study. It provides the correct stage to the researcher to snoozing out the improvement work with the end goal to resolve the work using the correct technique within a given specifications.

3.1 Discretization

Discretization is the process of converting the continuous nature of PDE/ODE to 'equivalent' simultaneous algebraic equation. This process is usually carried out as a first step towards making them suitable for numerical evaluation and implementation on digital computers. In this study, a series of discretization work will be done in several steps. Figure 3.1 shows the brief flow of discretization work.

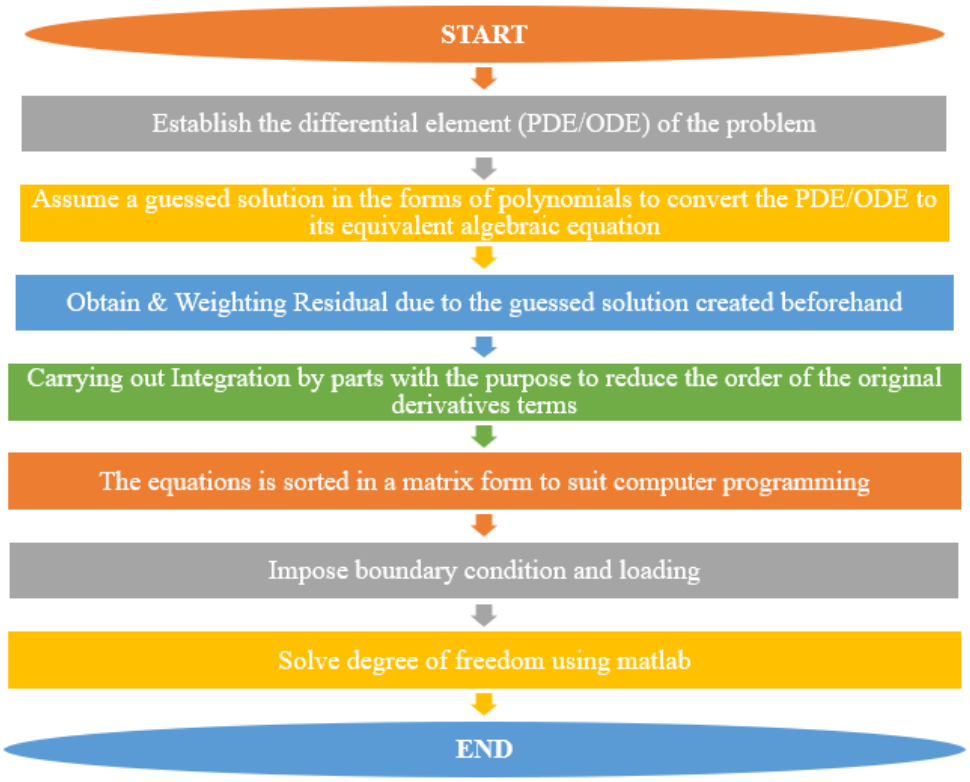


Figure 3.1 : Flowchart of discretization work

3.1.1 Discretization of Finite Element Method

In Finite Element Method, for heat transfer problem, the first step involves the establishment of partial differential equation as follows:

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) - \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) = -q_H \tag{3.1}$$

In order to solve the following partial differential equation, a guess equation for the function T in the form of polynomials is needed to convert the PDE into algebraic equation. Example of the trial function is:

$$T = a_1 + a_2 x + a_3 y + a_4 xy \tag{3.2}$$

Eqn. (3.2) will then be evaluated at the location of the nodes and the calculated values is equals to the corresponding degree of freedom, T_i

At $x = 0, y = 0$;

$$T = a_1 + a_2(0) + a_3(0) + a_4(0)(0) = \hat{T}1$$

At $x = a, y = 0$;

$$T = a_1 + a_2(a) + a_3(0) + a_4(a)(0) = \hat{T}2$$

At $x = a, y = b$; (3.3)

$$T = a_1 + a_2(a) + a_3(b) + a_4(a)(b) = \hat{T}3$$

At $x = 0, y = b$;

$$T = a_1 + a_2(0) + a_3(b) + a_4(0)(b) = \hat{T}4$$

Solving the above equations will result in obtaining the value for the polynomials coefficient and shape function of the equation. The following trial function are used corresponding to the Galerkin WRM.

$$T = N_i T_i = \{N\}\{T\}^T \quad (3.4)$$

Where N_i is the shape function and T_i is the degree of freedoms (dofs). The next step involve integration by part (IBP) with the intentions to relax the statement and to induce natural boundary conditions.

Eqn. (3.1) which are the Partial Differential Equation (PDE) for heat can be express in matrix forms as follows:

$$\{\partial\}[E]\{\partial\}^T T = -q_H \quad (3.5)$$

Where $[E] = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$ and $\{\partial\} = \{\frac{\partial}{\partial x} \frac{\partial}{\partial y}\}$

By replacing Eqn. (3.5) with Eqn. (3.4), the following equation obtained:

$$\{\partial\}[E]\{\partial\}^T\{N\}\{T\}^T = -q_H \quad (3.6)$$

In order to obtain a sufficient number of equations, Eqn (3.6) needs to be multiplied with shape function, $\{N\}$ and integrate it to get the independent equations.

$$\int_x \int_y \{N\}^T \left(\{\partial\}[E]\{\partial\}^T\{N\}\{T\}^T + q_H\right) dydx = 0 \quad (3.7)$$

Conducting IBP to Eqn (3.7) gives;

$$\int_x \int_y \{N\}^T\{\partial\}[E]\{\partial\}^T\{N\}\{T\}^T dydx = \int_x \int_y \{N\}^T q_H \, dydx + \int_s [N^T]\{\phi\}^T ds \quad (3.8)$$

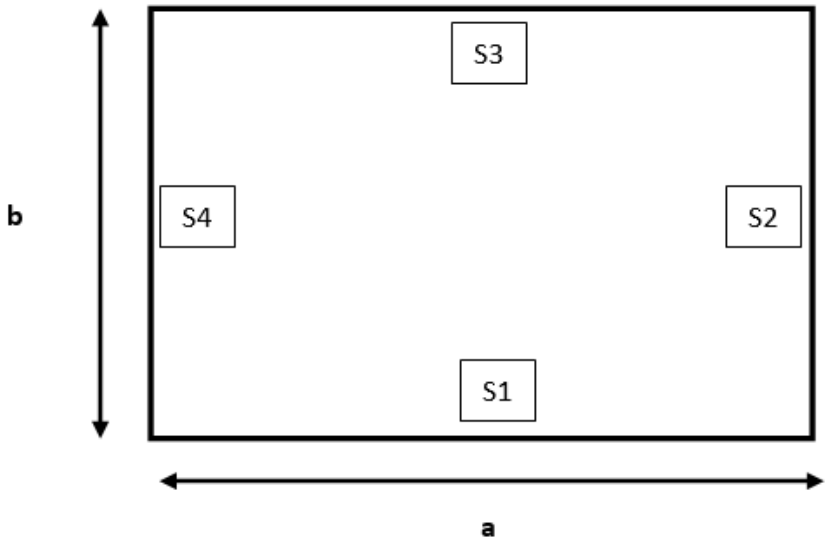
Where $\{\phi\}$ is the Neumann boundary condition terms from IBP.

By solving the IBP, the equation can be arranged in a matrix form and the degree of freedom can be solve by using MATLAB.

$$[K]\{T\}^{T} = \{q_H\} + \{\phi\} = \{r\}$$

3.1.2 Discretization of Finite Volume Method

In Finite Volume Method (FVM), Green's Divergence Theorem are used. Taking one simple closed region as an example, the equations are discretized as following:



$$S1 + S2 + S3 + S4 = 0 \tag{3.5}$$

Eqn. (3.5) shows the equation in evaluating the boundary of the domain.

$$\int_A \frac{\partial^2 T}{\partial x^2} + \int_A \frac{\partial^2 T}{\partial y^2}\, dA = \int_A \frac{\partial^2 T}{\partial x^2}\, dA + \int_A \frac{\partial^2 T}{\partial y^2}\, dA \tag{3.6}$$

By applying the divergence theorem, Eqn. (3.7) yields. Sign notation need to be synchronize to the figure shown.

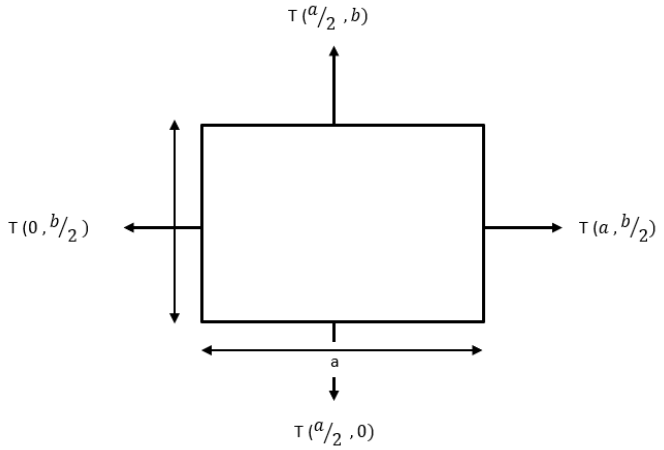
$$-\int_0^b \frac{\partial T}{\partial y}\,|_{y=0}\,dx + \int_0^a \frac{\partial T}{\partial x}\,|_{x=b}\,dy + \int_0^b \frac{\partial T}{\partial y}\,|_{y=a}\,dx - \int_0^a \frac{\partial T}{\partial x}\,|_{x=0}\,dy = 0 \qquad (3.7)$$

Eqn. (3.7) can be simplified in matrix form as follows:

$$[K_{ij}]\{T_i\} = 0$$

In FVM, the temperature gradient, T at the surface is assumed as constant, which eventually simplifies the equation to;

$$\left(\frac{\delta T}{\delta x}|_{x=b} - \frac{\delta T}{\delta x}\,|_{x=0}\right) Ly + \left(\frac{\delta T}{\delta y}|_{y=a} - \frac{\delta T}{\delta y}|_{y=0}\right) Lx = 0 \qquad (3.8)$$



Referring to the above figure, the equation can be simplified as below:

$$\left[\frac{T(a, b/2) - T(0, b/2)}{L_x}\right] L_y + \left[\frac{T(a/2, b) - T(a/2, 0)}{L_y}\right] L_x \qquad (3.9)$$

3.1.3 Discretization of New Divergence Method

$$-\int_0^b \frac{\delta}{\delta y}\,|_{y=0}\,dx + \int_0^a \frac{\delta T}{\delta x}\,|_{x=b}\,dy + \int_0^b \frac{\delta T}{\delta y}\,|_{y=a}\,dx - \int_0^a \frac{\delta T}{\delta x}\,|_{x=0}\,dy = 0 \qquad (3.10)$$

In New Divergence Theorem, instead of directly integrate Eqn. (3.10), Eqn. (3.4) which contains shape function will be included in the equation.

$T = N_i T_i$

$$\int \delta \frac{T_j N_j}{\delta x} . n_x \, dx + \int \delta \frac{T_j N_j}{\delta y} . n_y \, dy = 0 \tag{3.11}$$

$$-\int_0^b N_i \frac{\delta N_j}{\delta y} \, |_{y=0} \, dx + \int_0^a N_i \frac{\delta N_j}{\delta x} \, |_{x=b} \, dy + \int_0^b N_i \frac{\delta N_j}{\delta y} \, |_{y=a} \, dx - \int_0^a N_i \frac{\delta N_j}{\delta x} \, |_{x=0} \, dy = 0 \tag{3.12}$$

The purpose of using the concept of Green Divergence Theorem in Finite Volume Method is to make use of the shape function since Finite Volume Method carry the assumption of constant flux.

Eqn. (3.12) is the discretized equation for the proposed New Divergence Theorem, which can be presented in matrix form as follows:

$$[K_{ij}]\{T_i\} = 0$$

Solving the line integrals can be finalized as:

$$[K_{ij}] \begin{Bmatrix} T1 \\ T2 \\ T3 \\ T4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

3.2 Derivation of the New Divergence Method with Neumann Boundary Condition

In this study, the aim is to derive a New Divergence Formulation with Neumann Boundary Condition. The general steps in deriving the formulation are as follows:

i. Establishment of partial differential equation for heat

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) = 0 \tag{3.14}$$

ii. Discretization of the equation based on Finite Element Method. The discretized equation is shown as below. ϕ_x and ϕ_y is the Neumann Boundary Condition.

$$\int_A \left(k_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + k_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\right) \partial A \ \{T_i\} = \int_A \left(k_x N_i \phi_x + k_y N_i \phi_y\right) \partial A \tag{3.15}$$

$$where\ \phi_x = \frac{\partial T}{\partial x} \quad , \quad \phi_y = \frac{\partial T}{\partial y}$$

iii. The formulation is then discretized using the New Divergence Method and the discretized equation is as below.

$$\begin{aligned} -\int_0^a k_y N_i \frac{\partial N_i}{\partial y}\,|_{y=0}\,\partial x\,\{T_i\} + \int_0^b k_x N_i \frac{\partial N_i}{\partial x}\,|_{x=a}\,\partial y\,\{T_i\} \\ + \int_0^a k_y N_i \frac{\partial N_i}{\partial y}\,|_{y=b}\,\partial x\,\{T_i\} - \int_0^b k_x N_i \frac{\partial N_i}{\partial x}\,|_{x=0}\,\partial y\,\{T_i\} \\ = k_x \phi_x + k_y \phi_y \end{aligned} \tag{3.16}$$

iv. The equations can be arranged in matrix form of as follow

$$[K_{ij}]\{T_i\} = 0$$

3.2.1 Implementation of Neumann Boundary Condition in New Divergence Method

The implementation of Neumann Boundary Condition was based on linear Taylor Difference which can be illustrated as below.

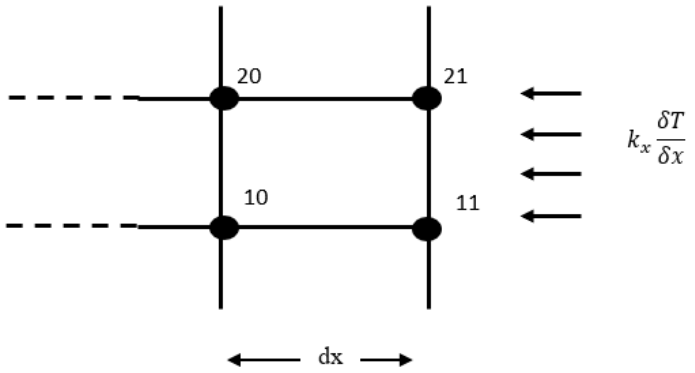


Figure 3.2 : Neumann B.C. at domain

$$\frac{\delta T}{\delta x} = \phi$$

$$where; \quad \text{Neumann}, \phi = \frac{T_{21} - T_{20}}{dx}$$

Taking figure above as the example, in coming out with the Neumann equation, the temperature at node 21, T_{21} is minus with temperature at node 20, T_{20} and divided with the distance between the two nodes, dx. The purpose of conducting Taylor Difference is to get the temperature gradient of the domain.

The implementation of Neumann Boundary Condition in source code for Finite Element Method and for the New Divergence Method is different. In Finite Element Method, the Neumann equation is included inside the stiffness matrix, [K] while for New Divergence Method, the Neumann equation is replacing the existing equation, which eventually takes a longer time for Matlab to work on it.

3.3 Computer source coding using Matlab

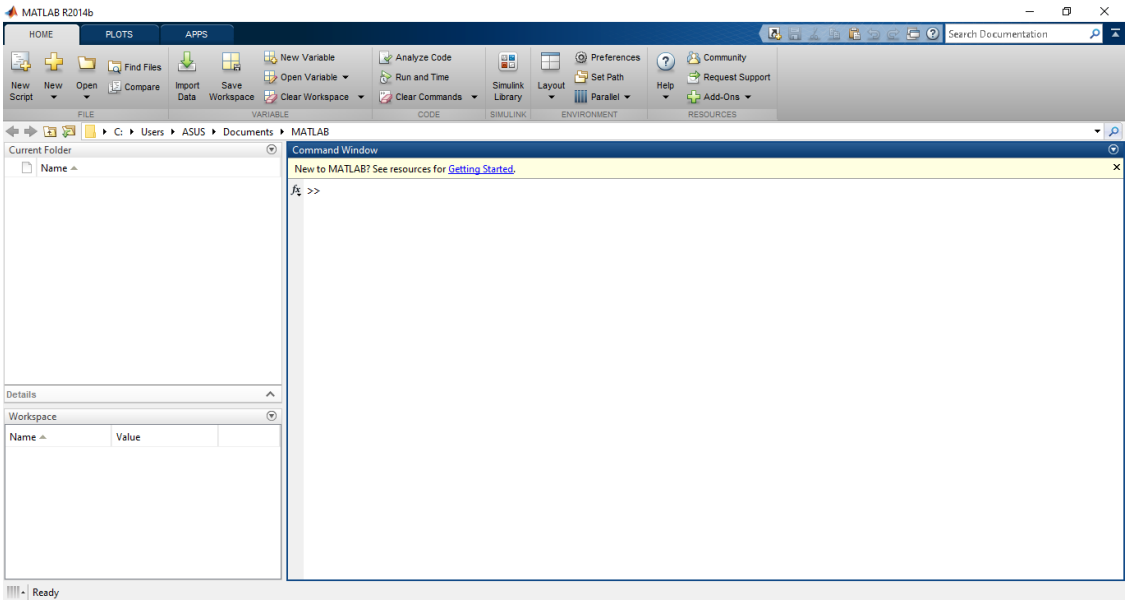


Figure 3.3 : MATLAB Command window

The name MATLAB stands for MATrix LABoratory. MATLAB is a software package for high performance numerical computation and visualization. It provides an interactive environment with hundreds of built-in functions for technical computation, graphics and animations. It also provides easy extensibility with its own high-level programming language.

MATLAB can be used to create arrays, vectors (vector or matrices) as well as performing operations on them. An array is a list of numbers or expressions arranged in horizontal and vertical columns. One dimensional arrays are called ***vector*** and two-dimensional arrays are called ***matrices.***

Below are the example of commands and the results that can be obtained from MATLAB:

Table 3.1 : MATLAB commands involving vectors

Commands	Results
>> x=[1 2 3] or >> x=[1,2,3]	Creates a row vector x = (1 2 3)

>> y=[3;7;9]	Creates a column vector y = 3 7 9
>> a= x+y	This will produce an error
>> a= x+z	a= (3 3 3)
>> c=x./y	This will produce an error
>> x'	Transpose of x. Ans = 1 2 3
>> y'	Transpose of y. Ans = (3 7 9)
>> t=linspace (0,10,6)	Creates a vector t = (0 2 4 6 8 10)
>> t=[0 : 0.2 : pi]	Creates the vector t = (0 0.2 0.4 ……pi)
>> t(5:length (t)) = []	Deletes all elements of vector t except 1 through 4. Ans t=(0 0.2 0.4 0.6)

Table 3.1 : MATLAB commands involving matrices

Commands	Results
>>A = [2,4,6; 1,7,3; 8,4,5]	Creates a matrix A=2 4 6 1 7 3 8 4 5
>>B = (1:2, 1:3)	Creates a matrix B from A. Thus B = 2 4 6 1 7 0 . Here '1:2' means from row 1 to row 2, and '1:3' means from column 1 to column 3
>>C = A (1:2, :)	Creates a matric C from A. Ans C = 2 4 6 1 7 0 . Here ':' means all columns, same as 1:3

>>A = [2,4,10,13; 16,3,7,18; 8,4,9,25; 3,12,15,17]	Creates a matrix A = $\begin{bmatrix} 2 & 4 & 10 & 13 \\ 16 & 3 & 7 & 18 \\ 8 & 4 & 9 & 25 \\ 3 & 12 & 15 & 17 \end{bmatrix}$
>>A (3,:) = []	Deletes the third row in A Ans A=$\begin{bmatrix} 2 & 4 & 10 & 13 \\ 16 & 3 & 7 & 18 \\ 3 & 12 & 15 & 17 \end{bmatrix}$
>>A = eye (3)	A is a 3x3 identity matrix
>>B = zeros (3)	B is a 3x3 matrix of zeros

In MATLAB there are two operators used for dividing operations which are:

1) The right division represented by the symbol '/' or known as 'slash'
2) The left division represented by the symbol '\' or also known as 'backslash'

These two operators differs from one another. The right division is as of what we usually make use of. Taking this example;

$$A = 8 \text{ , B} = 2$$

$$A/B = 8/2 = 4$$

Contrary to the right division, the left division works as follows (using the same example);

$$A\backslash B = B/A = 2/8 = 0.25$$

Therefore, if the matrix were to solve using backslash operators, the operation will be written as:

```
>> A=[1 2  ;  2 2];
   B=[3 2  ;  1 1];
   A\B
```

ans =

$$\begin{matrix} -2.0000 & -1.0000 \\ 2.5000 & 1.5000 \end{matrix}$$

3.4 Modelling Using COMSOL Software

3.4.1 Creating a model guided by the Model Wizard

1. Setting up a new model can be either guided by the **Model Wizard** or start from a **Blank Model**. In this project, **Model Wizard** was used as it guides in setting up the dimension, physics and study type.

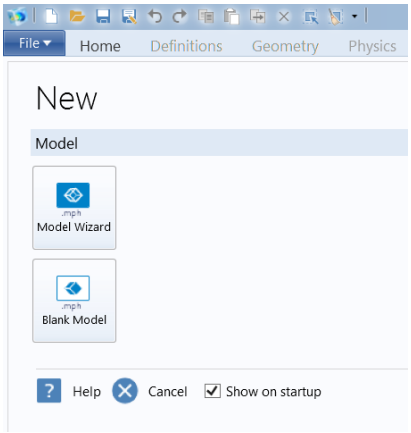


Figure 3.4 : Selection of model mode

2. **2D** space dimension was selected for the model component.

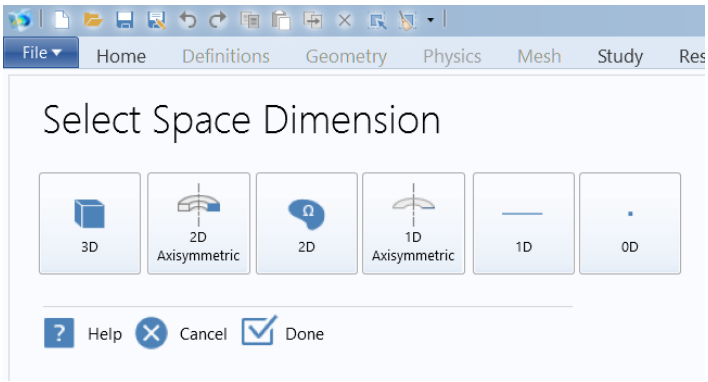


Figure 3.5 : Selection of space dimension

3. For the Physics interface, **Heat Transfer in Solids** was added. The Heat Transfer in Solids interface is used to model heat transfer by conduction, convection, and radiation. The temperature equation defined in solid domains corresponds to the differential form of the Fourier's law that may contain additional contributions like heat sources.

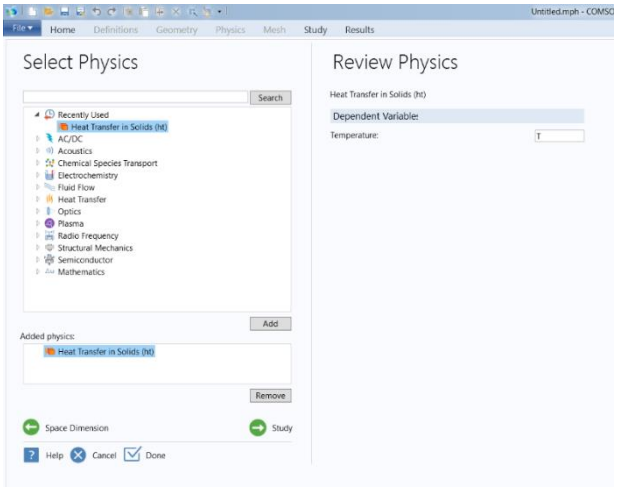


Figure 3.6 : Selection of physics interface

4. Study type selected was **Stationary**. The Stationary study is used when field variables do not change over time where in this heat transfer project, the study is to compute the temperature field at thermal equilibrium.

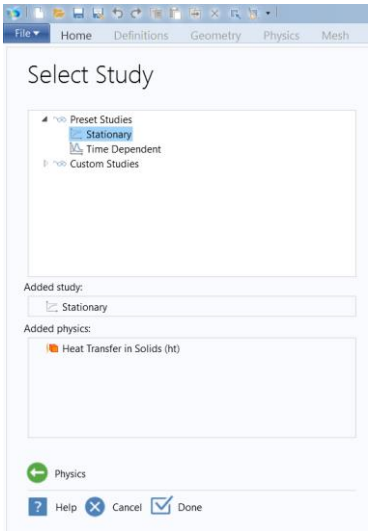


Figure 3.7 : Selection of study type

5. Next, click **Done** and the desktop will display the model tree configured based on the choices made.

3.4.2 Creating Component 1

1. In the **Model Builder** window, under **Component 1**, expand **Definitions** > expand **View 1** > **Axis**. The maximum and minimum length for both x-axis and y-axis are set as following:

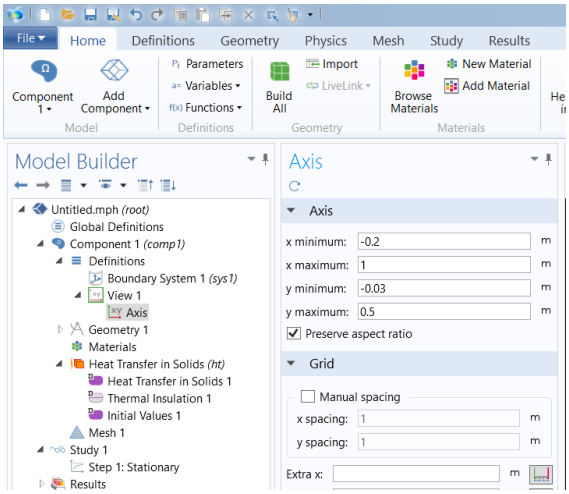


Figure 3.8 : Setting up axis

2. Right click **Geometry** and select **Rectangle**.

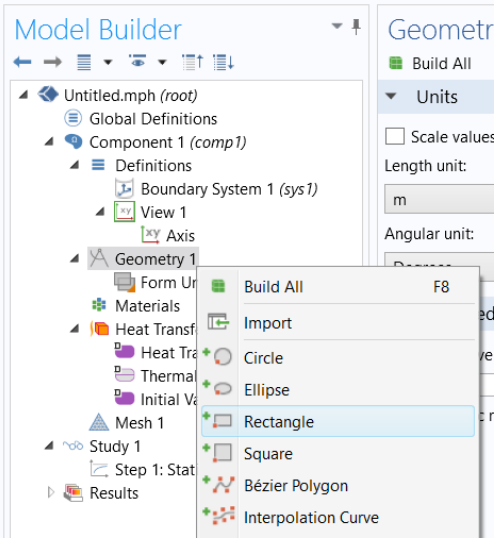


Figure 3.9 : Creating the model domain

3. Click at **Rectangle 1**. In the Rectangle window, specify the size of the model as 0.8(W) x 0.5(H) and then click **Build Selected**.

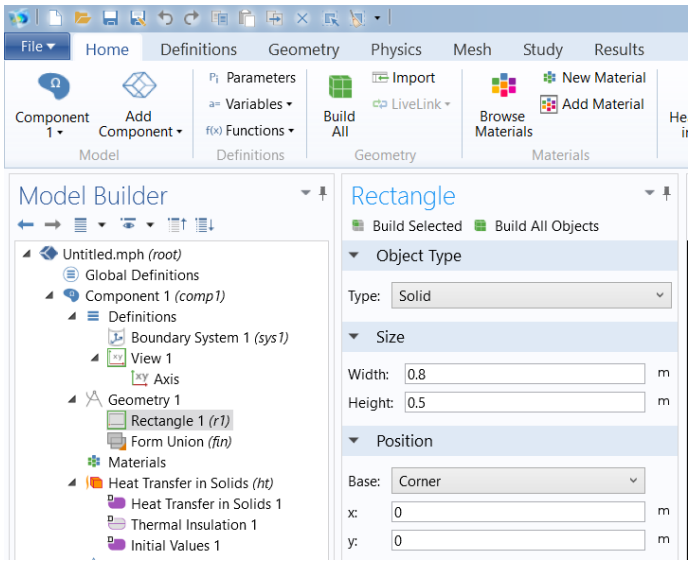


Figure 3.10 : Specify domain size

4. Click **Heat Transfer in Solids 1**. In the Heat Transfer in Solids window, change the following:

Thermal conductivity = 600 $W/(m.K)$

Density = 8700 kg/m^3

Heat capacity at constant pressure = 385 $J/(kg.K)$

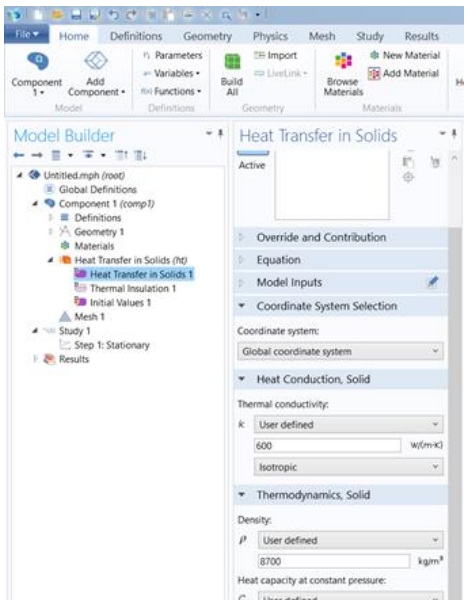


Figure 3.11 : Specify heat transfer properties

5. Click on **Initial Values 1** and specify the temperature, $\boldsymbol{T} = 273.15\ K$

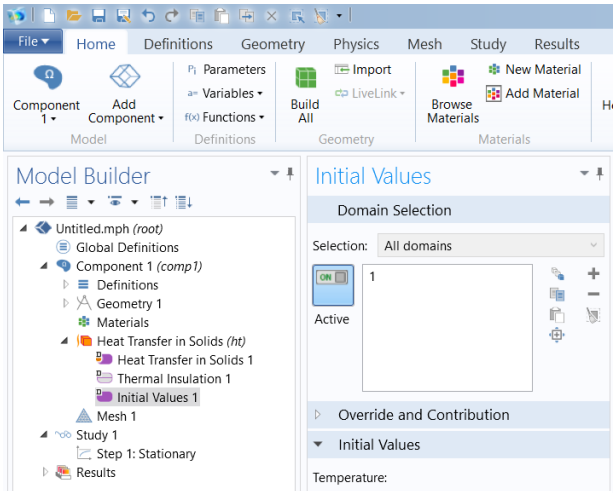


Figure 3.12 : Specify temperature value

6. Right click on **Heat Transfer in Solids (*ht*)** and click **Heat Source**

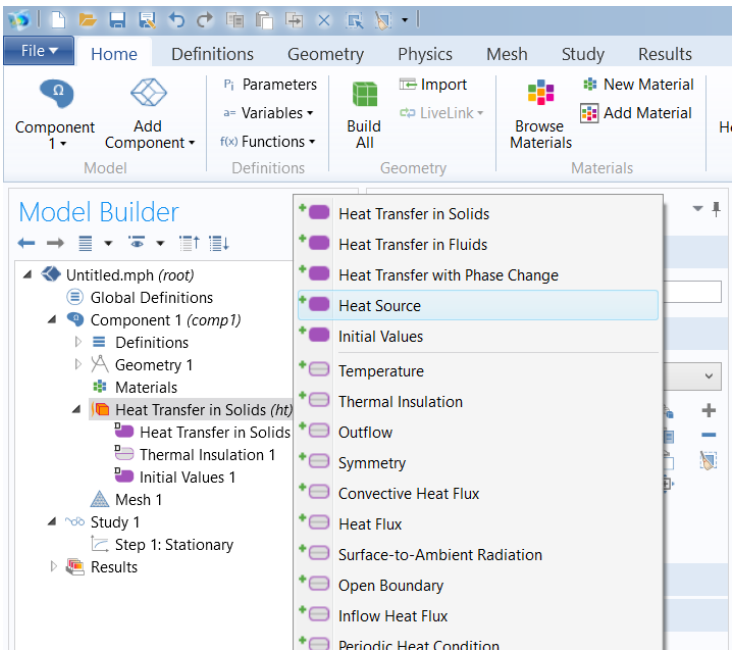


Figure 3.13 : Adding heat source to domain

7. Click on Heat Source 1. Under **Heat Source** tab, click on the **General source** and specify the

$$\mathbf{Q} = 0\ W/m^3$$

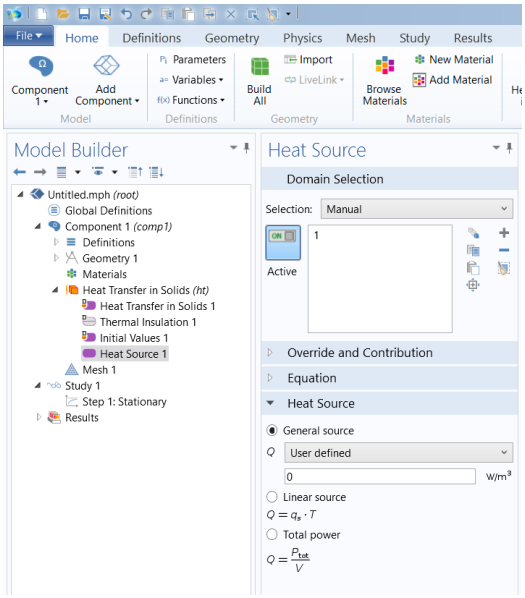


Figure 3.14 : Specify value for heat source

8. Right click on **Heat Transfer in Solids (*ht*)** and click **Temperature**

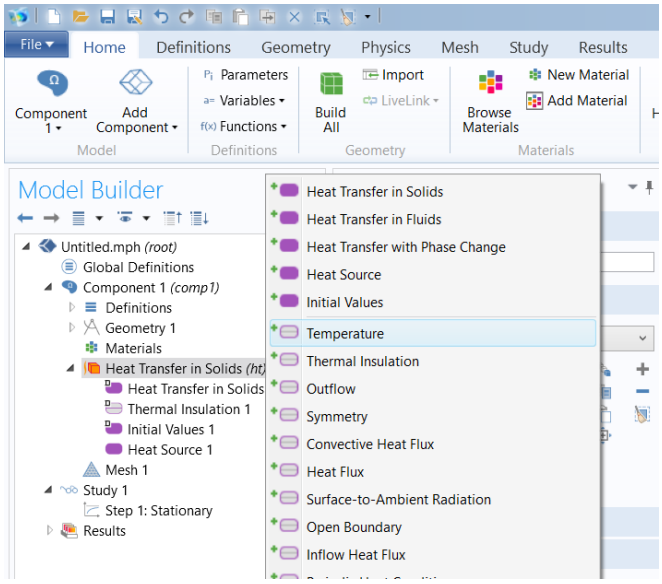


Figure 3.15 : Adding temperature to domain

9. Click on **Temperature 1** and at the Graphics window, click at the left boundary. Specify the temperature, **T** = 0 *K*

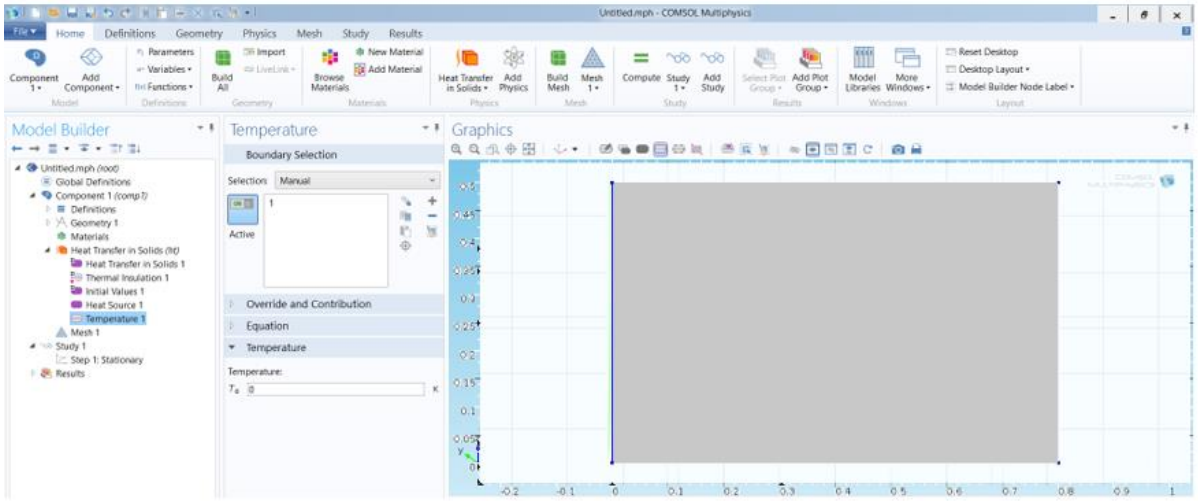


Figure 3.16 : Specify temperature at the domain

10. Repeat step 8 & 9 for temperature at the bottom boundary. Specify the temperature as T = 2 *K*

11. Right click on **Heat Transfer in Solids (*ht*)** and click **Heat Flux.**

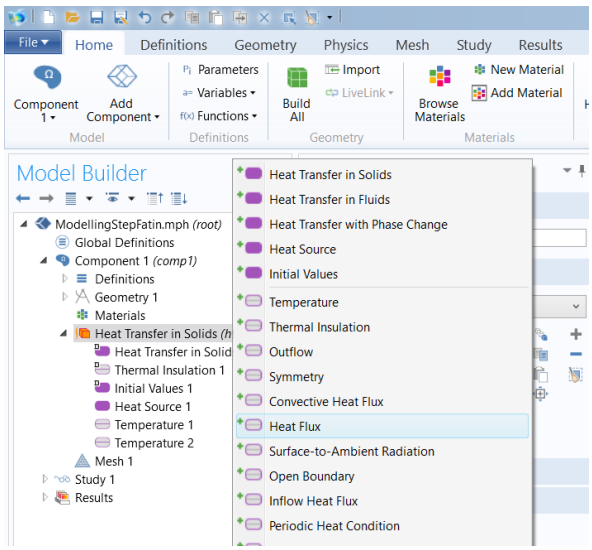


Figure 3.17 : Adding heat flux to the domain

12. Click on **Heat Flux 1** and at the Graphics window, click on the right boundary. Under **Heat Flux** tab, click on the **General inward heat flux** and specify the $q_0 = 3700\ W/m^2$

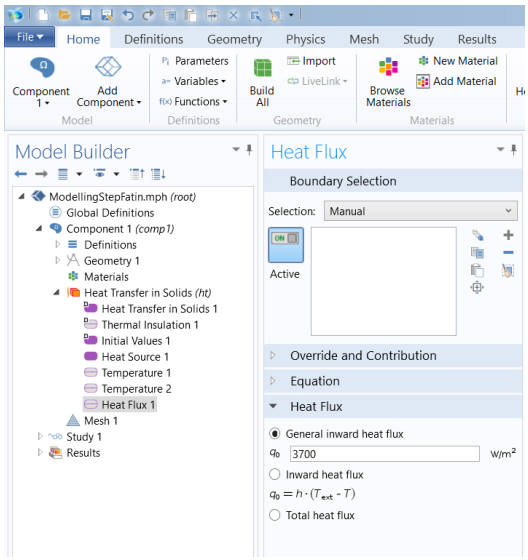


Figure 3.18 : Specify value for heat flux

13. Click on **Study 1** in the Model Builder and click **Compute** to get the contour of temperature distribution.

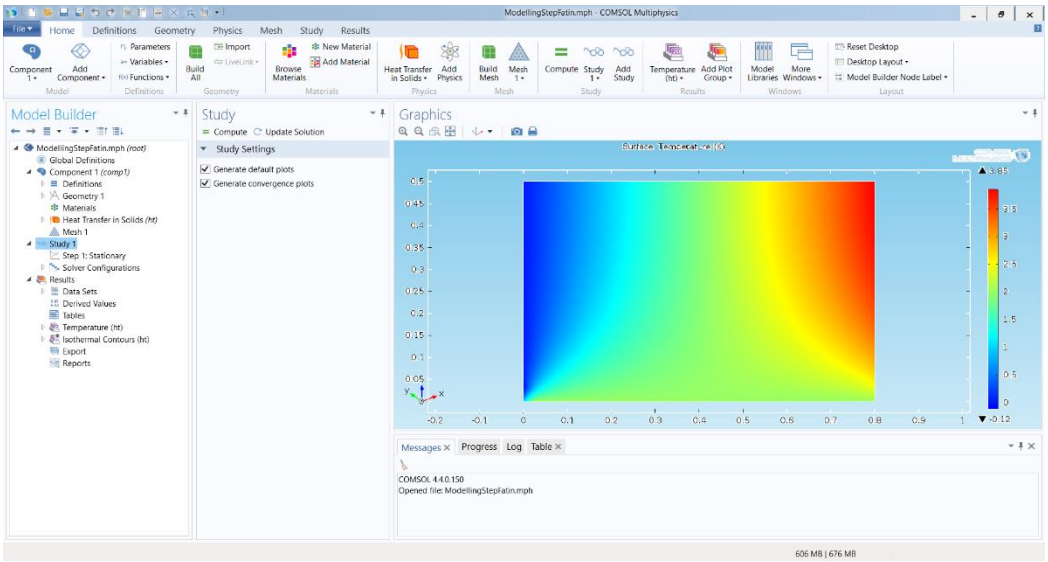


Figure 3.19 : Compute the domain to get the temperature contour

3.5 Verification of formula and codes

At the end of this study a new formulation is expected to be derived. Therefore, since it involves deriving a new formulation, the verification of the formula and codes need to be done by comparing the results against the existing result which used finite element method for heat transfer problem.

3.6 Parametric study

Choosing parameters for evaluation, characterize the parameter extend, determine the outline limitations, and examine the aftereffects of every parameter variety are the definition of parametric study. Therefore, it is crucial to access the performance of the new formulation based on the convergence rate

3.7 Discussion

To successfully achieve the objectives of this study, thorough discussion is needed throughout the study. In this study, detailed discussion will be on the variables involved in heat transfer problem which is temperature, the distribution of the heat and also the highest gradient of distribution. It is crucial to keep up with all the variables that may contribute to the end result. The main concern of new formulation is its accuracy as compared to the existing formulation. Apart from that, the computer consumption such as how long the time taken to process and how big storage needed to be used.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Preliminary Work

4.1.1 Heat Transfer Problem Using Finite Element Method

In this subtopic, a simple heat transfer problem will be solved using Finite Element Method formulation. Four assembled 4-nodes elements are used as the domain for this problem. Taking the size of domain as $0.4m(W)\ X\ 0.2m\ (H)$, heat source, $Q = 50W/m^3$ and other parameters as shown in the figure below, step by step manual calculations to evaluate the temperature at each node are shown below.

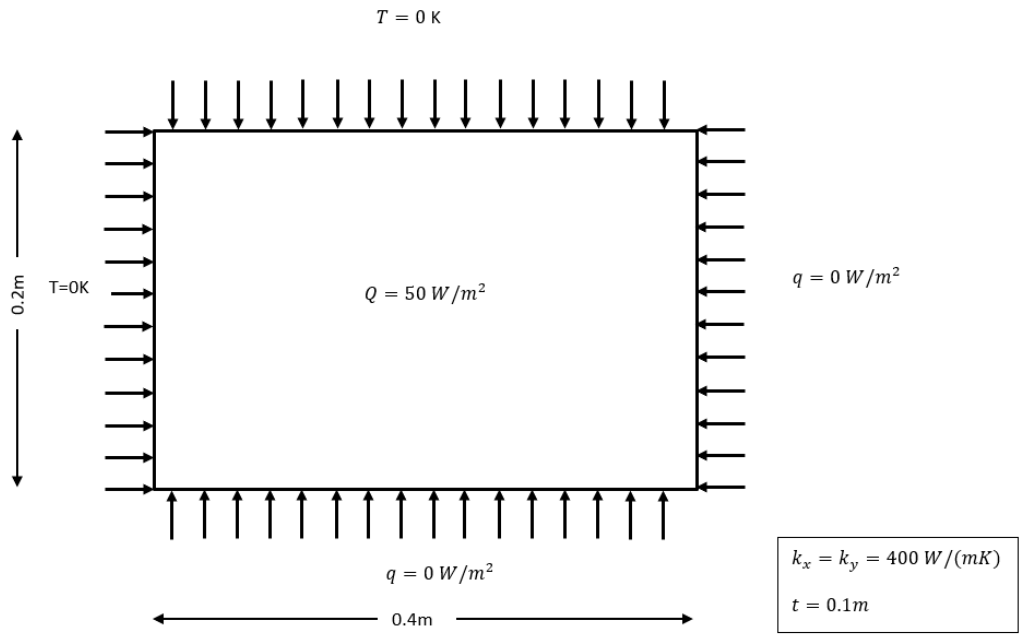


Figure 4.1 : Heat transfer domain

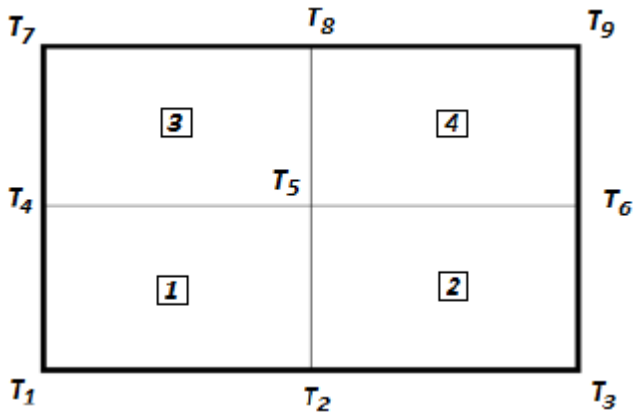


Figure 4.2 : Arrangement of elements and nodes

Figure above shows the arrangement of element and global degree of freedom numbering for 4-nodes element.

Each element would have similar conductance matrix and internal heat generation forces.

$$[k^1] = [k^2] = [k^3] = [k^4] \tag{4.2}$$

By solving the equation for this heat problem using Matlab command "\", the values of the temperature are obtained. The figure below shows the contour of the temperature distribution obtained once the equation is plotted in Matlab.

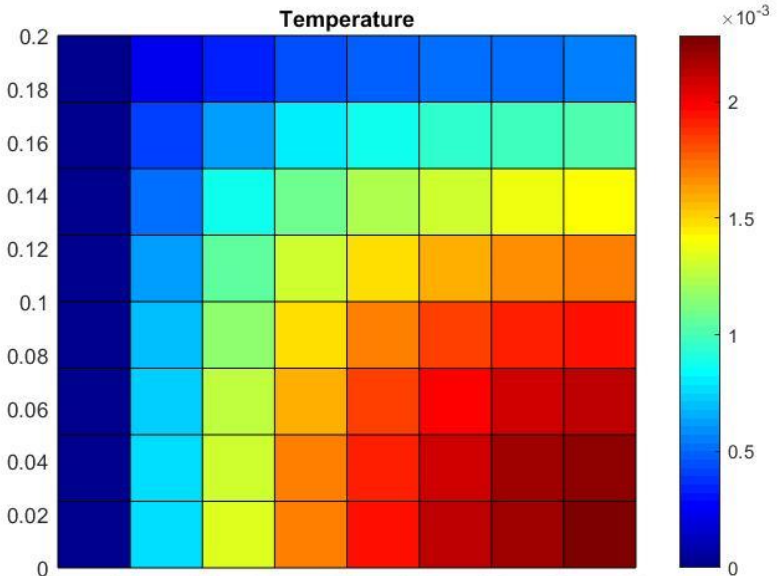


Figure 4.3 : Contour temperature distribution from MATLAB

Taking another different condition of heat transfer problem, two different methods, Finite Element Method (FEM) and the New Divergence Method (NDM) are used to solve this problem. Both methods are discretized following the steps as mentioned in the previous chapter.

4.1.2 Heat Transfer Problem (Comparison using Finite Element Method and New Divergence Method)

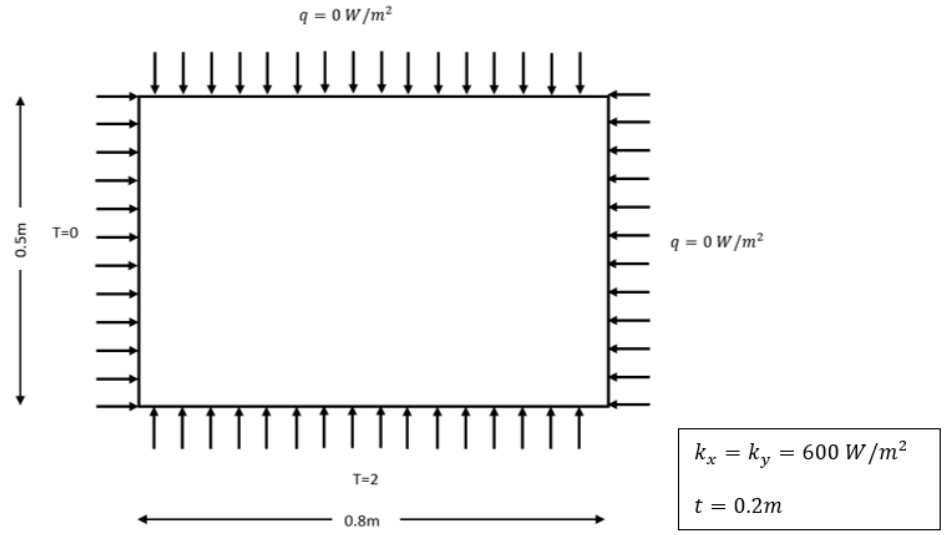


Figure 4.4 : Heat transfer domain

In this heat transfer problem, the rectangular domain has heat source of Q=$50W/m^3$ Therefore, in the Partial Differential Equation (PDE), the heat source must be included as well. Solving the problem using Matlab software yields the following results.

Considering the same condition and total degree of freedom of 64 elements, the problem was solved using MATLAB software and yield the following results. Figure below shows the contour of temperature distribution in Kelvin (K) for both FEM and NDM. The colour varies depending on the intensity of the temperature on the domain. The intensity scale can be seen on the right side of the figure.

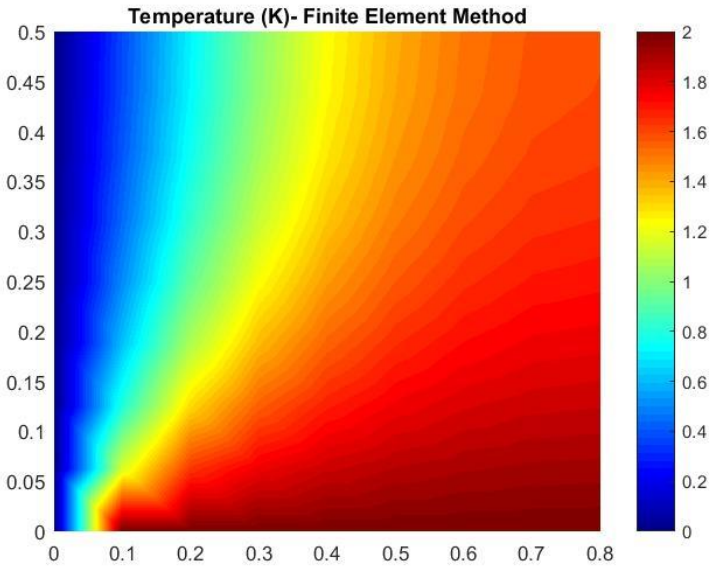


Figure 4.5 : Contour of temperature distribution from FEM

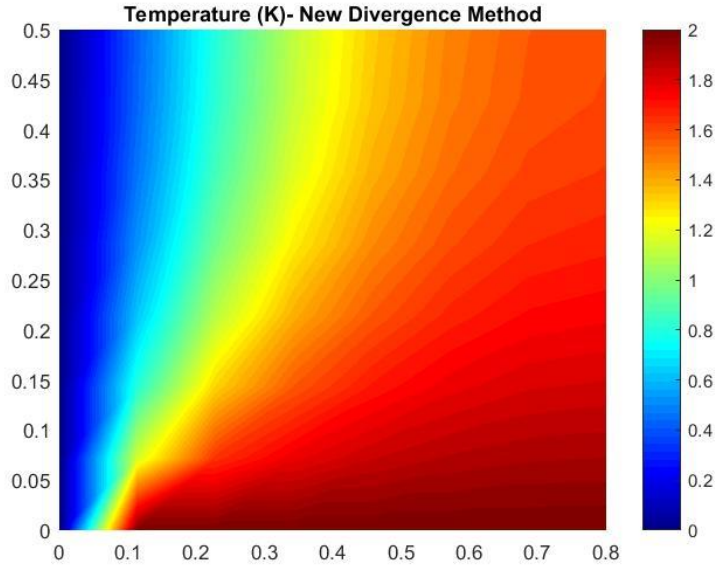


Figure 4.6 : Contour of temperature distribution from NDM

As can be seen above, at the top left of the figure. the contour of temperature distribution has a slight different between one another. This difference may be due to the number of decimal points taken at selected coordinate.

Since the work is about deriving a new formulation, verification was done by comparing the results between FEM and NDM. Temperature at selected coordinates was recorded and tabulated as Table 3 below.

Table 4.1 : Temperature for FEM and NDM at selected coordinate

x-coordinate	y-coordinate	Finite Element Method	New Divergence Method
		Temperature (K)	
0	0	0	0
0.2	0.25	0.927568371	0.918972275
0.4	0.25	1.429989029	1.41425864
0.6	0.25	1.646574765	1.642246696
0.8	0.25	1.70809284	1.709503999

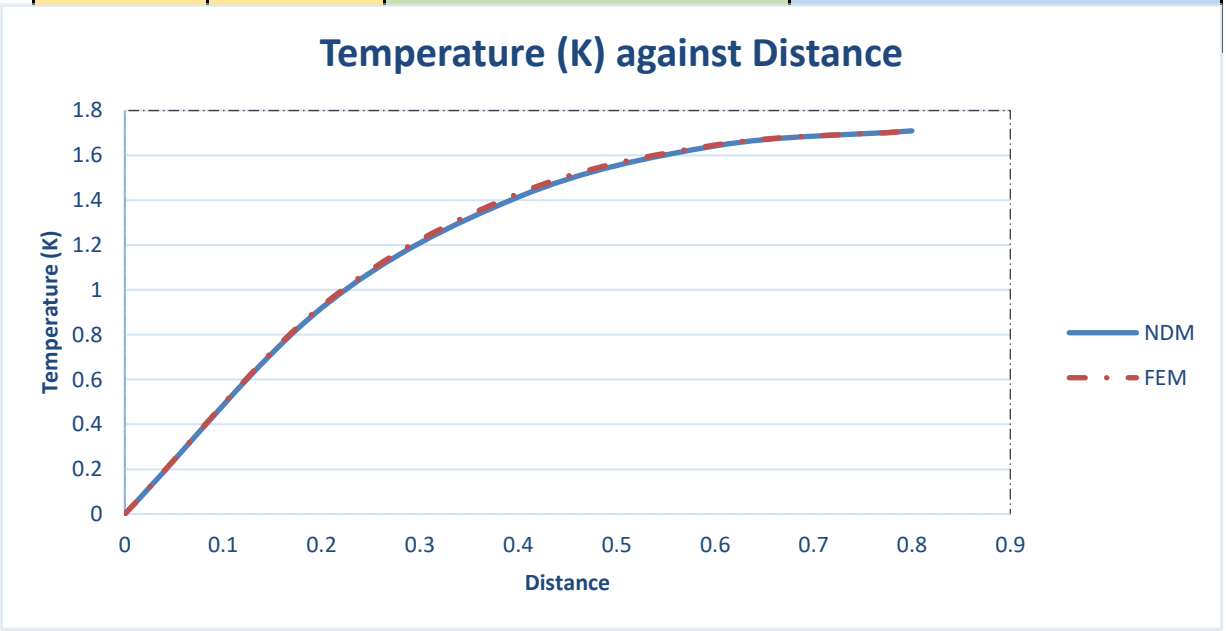


Figure 4.7 : Graph of temperature against distance for FEM and NDM

From the data table, there is no much difference on the value of temperature for both method. Therefore, it can be concluded that the New Divergence Method worked perfectly as the Finite Element Method.

Another checking that need to be done to New Divergence Theorem is the Convergence Test. One coordinate of (0.75,0.25) was chosen as point of interest. As the degree of freedom was increased, the temperature at the point of interest was recorded. The result was tabulated and plot in a graph as below.

Table 2.2 : Temperature for FEM and NDM at selected degree of freedom

Degree of Freedom	Temperature (K)	
	Finite Element Method	New Divergence Theorem
36	1.698981797	1.698981797
121	1.702696723	1.702696723
441	1.705211763	1.705211763
961	1.705678014	1.705678014
1681	1.705691773	1.705691773
2601	1.705691773	1.705819988
3721	1.705824073	1.705824073
5041	1.705835139	1.705835139
6561	1.70588507	1.70588507
8281	1.705860858	1.705860858
10201	1.705872742	1.705872742

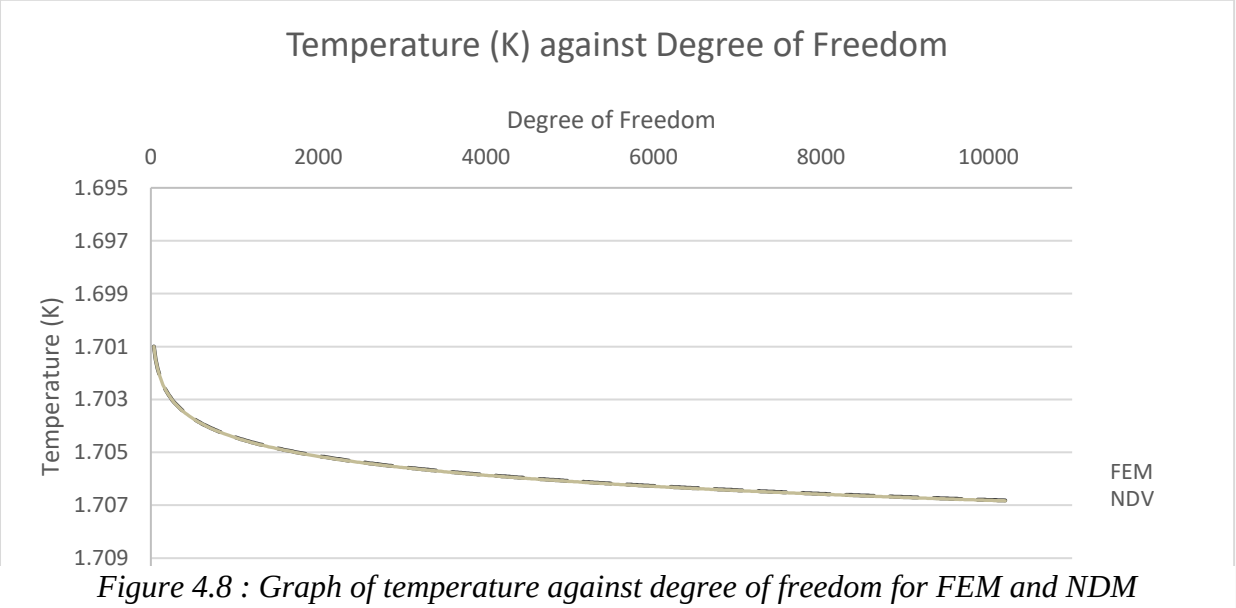


Figure 4.8 : Graph of temperature against degree of freedom for FEM and NDM

From the graph, it can be clearly seen that there is not much difference between the temperature at one point as the degree of freedom increases. This verify that the New Divergence Method worked.

4.2 Heat Transfer with Neumann Boundary Condition

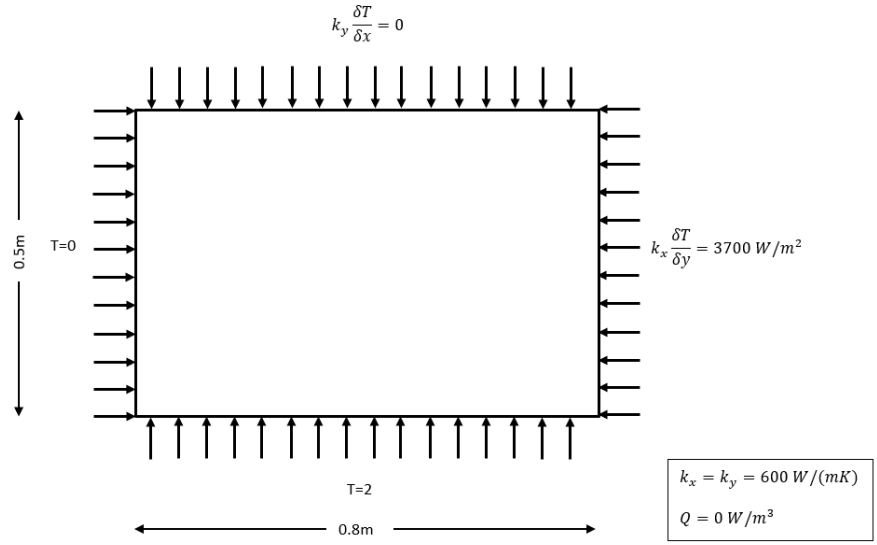


Figure 4.9 : Heat transfer problem with Neumann Boundary Condition

In this heat transfer problem, the problem is slightly different from the previous discussed chapter. This problem has a Neumann Boundary Condition on the right boundary of the domain. The derivation of formulation for heat transfer problem with Neumann Boundary Condition has been discussed in the previous chapter.

The figure below shows the contour of the temperature distribution obtained from Finite Element Method and Finite Volume Method once the equation is plotted in Matlab. Both method yields the same contour temperature distribution which proves that both method works the same. From the graph, it can be seen that on the right hand side of the domain, the intensity of the temperature is the highest. This is due to the presence of Neumann Boundary Condition which in terms of heat flux.

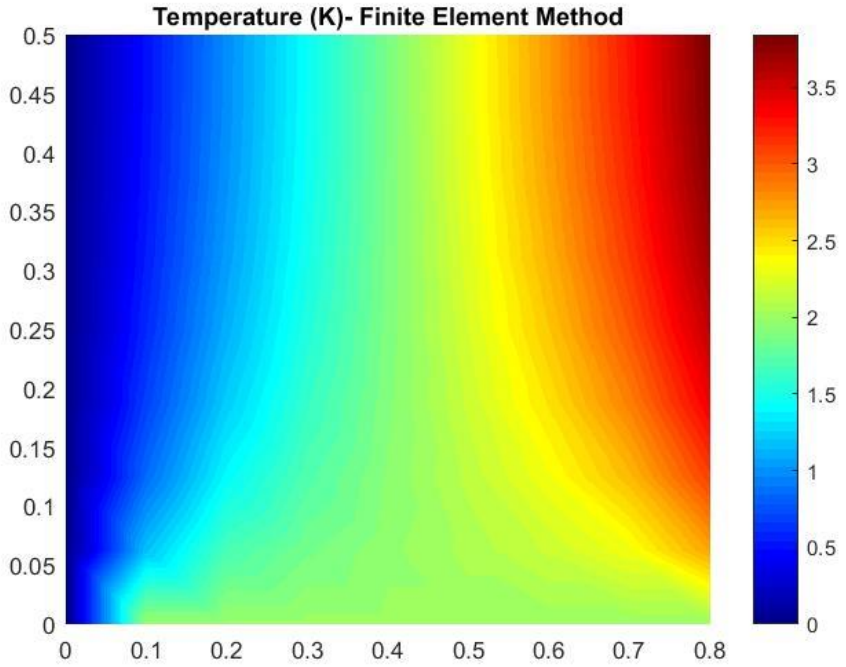


Figure 4.10 : Contour of temperature distribution with Neumann for FEM

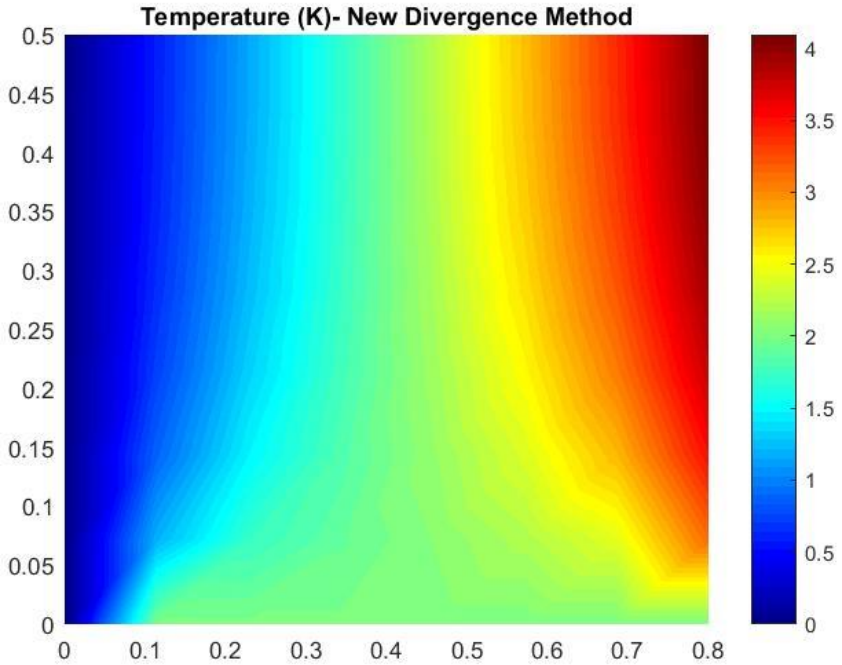


Figure 4.11 : Contour of temperature distribution with Neumann for NDM

Figure below shows the contour temperature distribution from COMSOL Multiphysics software. COMSOL Multiphysics is a platform for finite element analysis. Therefore, the result obtained from COMSOL should not have too much difference with the result obtained from Matlab using Finite Element Method. The slight difference in the contour temperature distribution may be due to source code restriction in Matlab.

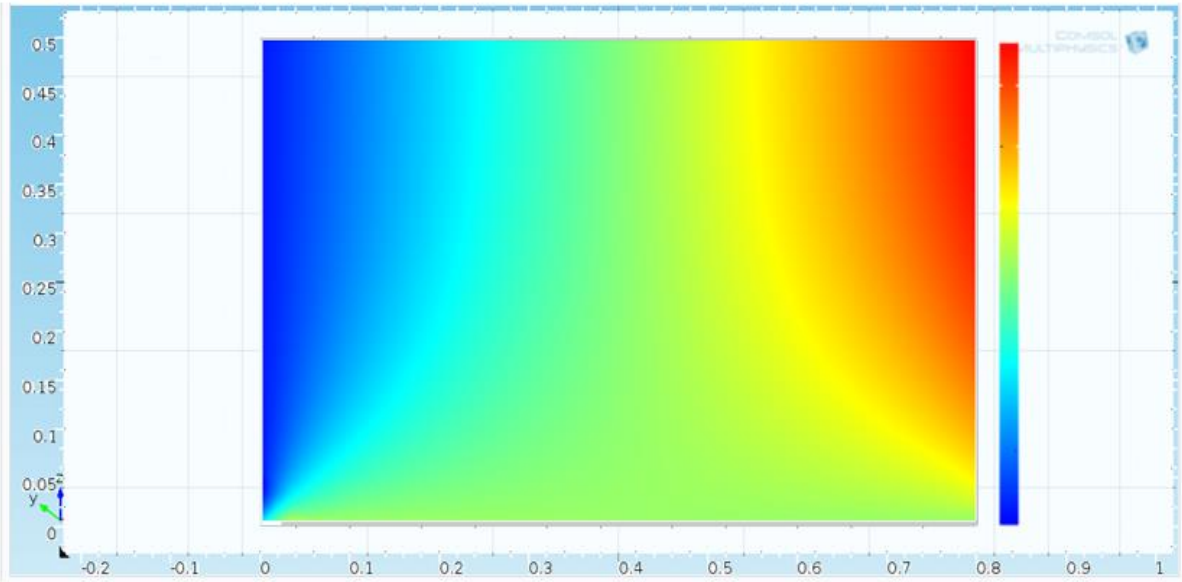


Figure 4.12 : Contour of temperature distribution from COMSOL software

4.2.1 Verification Study

Verification test was done by comparing the results between FEM and NDM. Temperature at selected distance was recorded and tabulated as Table 4.3.

Table 4.3 : Temperature for FEM and NDM at selected distance

Distance (x-direction)	Finite Element Method	New Divergence Method
	Temperature (K)	
0	0	0
0.2	1.11679	1.13192
0.4	1.888926	1.932565
0.6	2.581402	2.721189
0.8	3.562975	3.836548

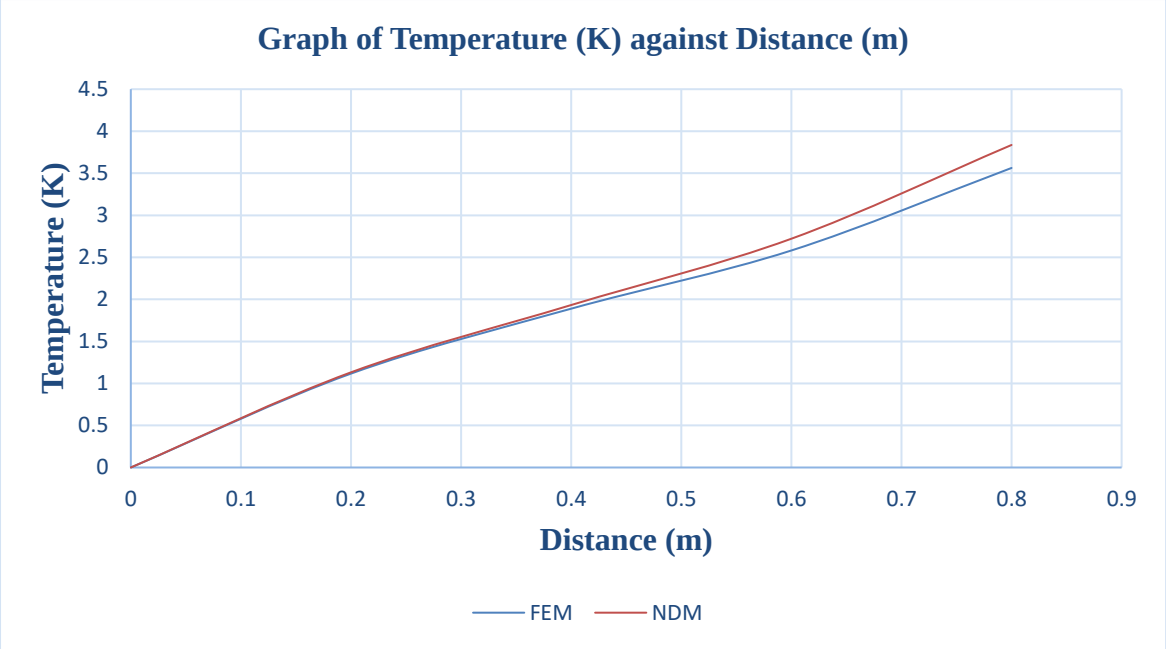


Figure 4.13 : Graph of temperature against distance for FEM and NDM

The figure above shows the graph of temperature against the distance in the domain. From the graph, it shows that as the distance increases, the temperature increases as well. However, there is a slight difference in temperature recorded between Finite Element Method and New Divergence Method. Since the work is about deriving a new formulation, the result from new method is expected to be a little bit lower or higher than the result from existing method. This is when tolerance error need to be taken in consideration. In this case, the result of New Divergence Method is slightly higher than the Finite Element Method. Despite the difference, the results are still acceptable.

4.2.2 Parametric Study

Parametric study was done to access the performance of the New Divergence Method to compare with the existing method. The parameters used in this parametric study are the degree of freedom and the temperature. Convergence rate between these two methods are compared by the plotting of residual error against the degree of freedom. The formula used for the calculation of residual error are as follows:

$$Percentage\ Error\ (\%) = \frac{Current\ Temperature - Converge\ Temperature}{Converge\ Temperature}\ X\ 100$$

Where;

Current temperature = Temperature recorded at certain degree of freedom in Kelvin

Converge temperature = Maximum temperature of the domain in Kelvin = 3.8477 K

Table 4.4 : Percentage error for FEM and NDM at selected degree of freedom

Degree of Freedom	Finite Element Method		New Divergence Method	
	Tmax	Percentage Error , %	Tmax	Percentage Error , %
2601	3.8476	0.00	3.8849	0.97
3721	3.8476	0.00	3.8788	0.81
5041	3.8476	0.00	3.8744	0.69
8281	3.8477	0.00	3.8685	0.54
10201	3.8477	0.00	3.8664	0.49
14641	3.8477	0.00	3.8633	0.41
29241	3.8477	0.00	3.8587	0.29

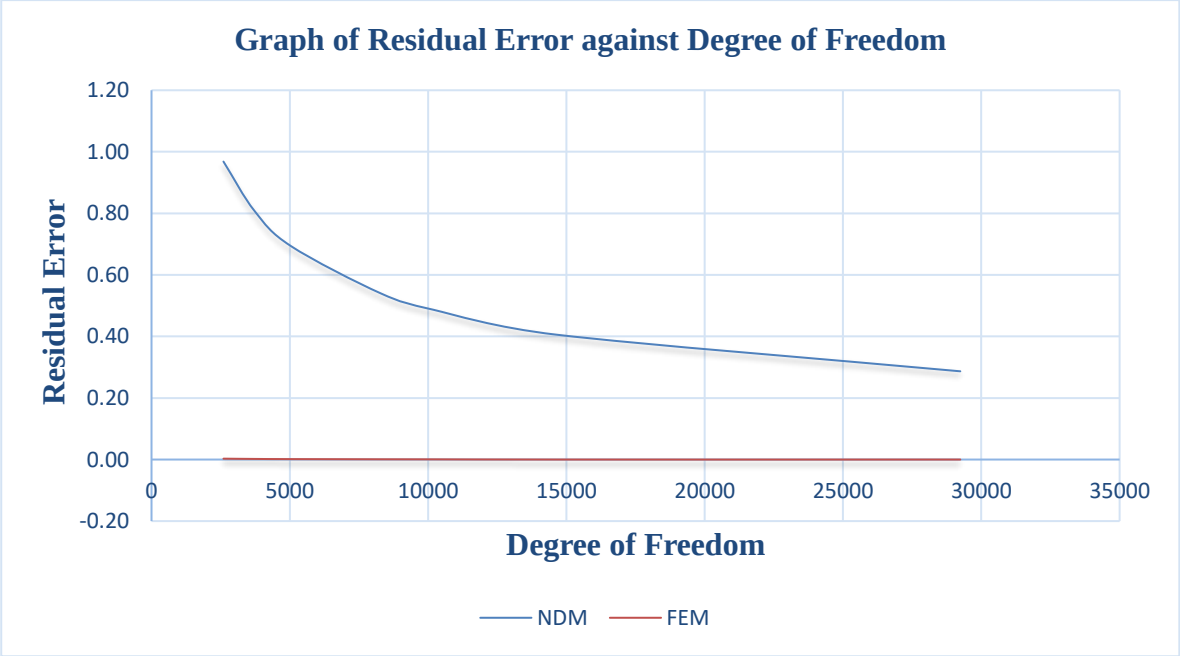


Figure 4.14 : Graph of residual error against degree of freedom for FEM and NDM

The graph above shows the comparison curve between Finite Element Method and New Divergence Method based on the plotting of residual error against the degree of freedom. As the degree of freedom increases, the temperature starts to converge. Thus, the residual error decreases. From the graph, the residual error of the New Divergence Method is quite significant to compare with the error from the Finite Element Method. This error was expected due to the way the Neumann Boundary Condition was implemented in the matrix.

The implementation of Neumann Boundary Condition which was based on linear Taylor Difference may result in less accuracy. Instead of calculating the temperature gradient between two nodes, taking several number of nodes and calculate the gradient would give a significant and more accurate value. However, a better convergence thus expected if a more appropriate interpolation of the boundary condition is formulated.

Another parametric study that was done is to access the time taken for each method to compute the temperature in computer software Matlab with the increment of degree of freedom. The elapsed time in seconds with the degree of freedom for each method are tabulated as below

Table 4.5 : Elapsed time for FEM and NDM at selected degree of freedom

Degree of Freedom	**Finite Element Method**	**New Divergence Method**
	Elapsed Time (s)	
25	0.008914	2.478023
81	0.017949	2.60808
169	0.055117	2.818525
289	0.021947	3.001262
441	0.039904	3.367504
961	0.050144	4.45111
1681	0.229369	6.19935
2601	0.448826	8.291786
3721	1.063554	11.275603
5041	2.247754	15.061531
8281	8.927647	28.565342
10201	19.004603	55.616099

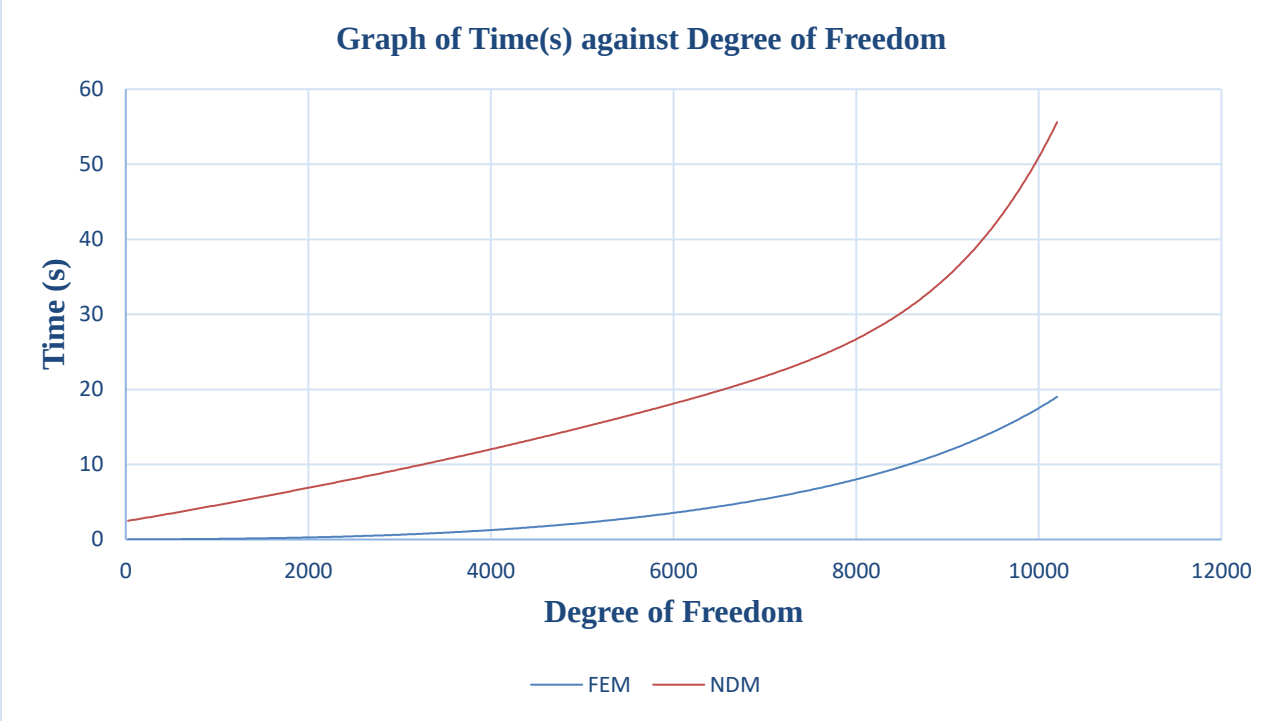


Figure 4.15 : Graph of time against degree of freedom for FEM and NDM

. From the table, the time taken between FEM and NDM shows a quite huge difference as the degree of freedom increases. This difference can be explained due to the different way the Neumann Boundary Condition was imposed. For Finite Element Method, the Neumann Boundary Condition was imposed by was including the Neumann equation inside the stiffness matrix. Meanwhile, for New Divergence Method, the existing equation are replaced with the Neumann equation which eventually takes a longer time for the software to compute the temperature.

Another factor that contributes to the difference in the elapsed time is due to the source code optimization. In writing the codes for the New Divergence Method, direct integration is used. In direct integration, the integration was done in symbolic, where it integrates from 0 to an indefinite number. Matlab software is not suitable to carry out a symbolic integration as it slows down the process.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

The idea of combining two numerical methods, Finite Volume Method and Finite Element Method works perfectly as expected. This can be proven from the parametric study that has been carried out based on the result obtained. The result obtained from FEM is the same as the New Divergence Method. Therefore, it can be strongly concluded that this New Divergence Method can be used for this study. The next step for this study is to include the Neumann Boundary Condition and verify the result.

The aim of this study is not only to create a new formulation. The formulation should work perfectly with the Neumann Boundary Condition. Based on the verification test and parametric study, the New Divergence Method with Neumann Boundary Condition was found as able to provide converged results. However, the rate of convergence is lower than the established Finite Element Method. This is identified as being caused by the satisfaction of the Neumann boundary condition using a linear Taylor difference. A better convergence thus expected if a more appropriate interpolation of the boundary condition is formulated.

However, the New Divergence Method is still in the developing stage. Further study and enhancement need to be done before the method can become useful tools for complex engineering problems.

REFERENCES

Airil, A.Akhbar, Erwan And Zhafri (2013). *Linear and Nonlinear Finite Element Analysis for Solids and Fluids with Matlab (Unedited First Draft).[*Manuscript in preparation].

C.Serraa, A.Tadeub, N.Simõesb (2016). "*3D heat diffusion simulation using 3D and 1D heat sources –Temperature and phase contrast results for defect detection using IRT*." Applied Mathematical Modelling: 1576–1587

Dong-yeon Seo a, C. K. b., Taehoon Hong b,⇑ (2015). "*A Lagrangian finite element model for estimating the heating and cooling demand of a residential building with a different envelope design.*" Applied Energy: 66–79.

Eric Li a, Zhongpu Zhang a, Z.C. He b, Xu Xuc, G.R. Liu d, Q. Li a (2014). *"Smoothed finite element method with exact solutions in heat transfer problems.*" International Journal of Heat and Mass Transfer **78**: 1219–1231.

Kanjanakijkasem, W. (2015). "*A finite element method for prediction of unknown boundary conditions in two-dimensional steady-state heat conduction problem*s." International Journal of Heat and Mass Transfer: 891–901.

Mahendran, M. (2007). *Applications of Finite Element Analysis in Structural Engineering*. Q. U. o. T. Faculty of Built Environment and Engineering, Australia**:** 38-46.

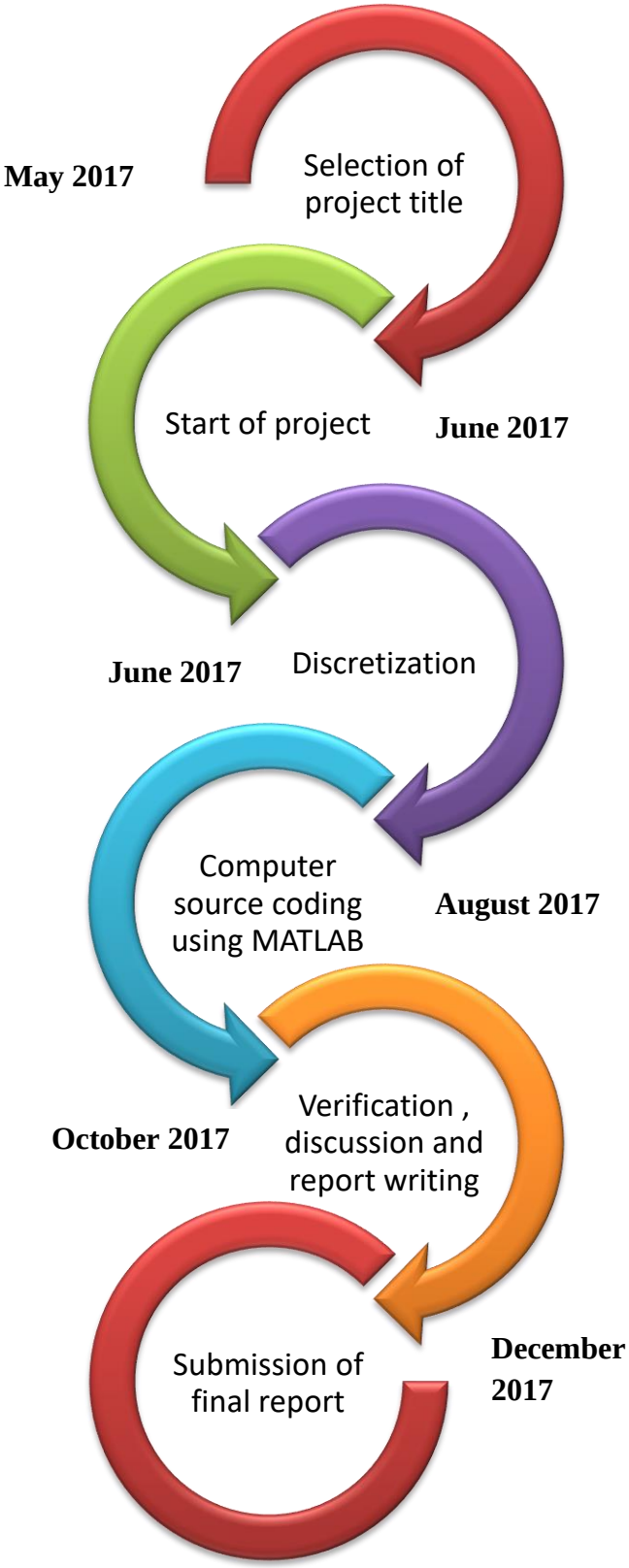
Pawe_l J. Matuszyk, M. S., and Maciej Paszy´nski (2015). "*Fully automatic 2D hp-adaptive Finite Element Method for Non-Stationary Heat Transfer*." Procedia Computer Science **51**: 2883–2887.

V.D. Thi a, b., M. Khelifa a,⇑, M. Oudjene a, M. El Ganaoui b, Y. Rogaume a *(2017). "Finite element analysis of heat transfer through timber elements exposed to fire*." Engineering Structures: 11-21.

Y.C. Ching a, Hakan F. Öztop b, M.M. Rahman c, M.R. Islam a, A. Ahsan d (2012). *"Finite element simulation of mixed convection heat and mass transfer in a right triangular enclosure*." International Communications in Heat and Mass Transfer **39**: 689–696.

APPENDICES

APPENDIX A


May 2017
Selection of project title
Start of project
June 2017
June 2017
Discretization
Computer source coding using MATLAB
August 2017
October 2017
Verification , discussion and report writing
Submission of final report
December 2017

APPENDIX B

```
clc;clear all;

kx1=400;
ky1=400;
Qa=50;

h1=0;
t1=0.1;

dx=8;
dy=8;

Lx=0.4;
Ly=0.2;

deltax=Lx/dx;
deltay=Ly/dy;

dofNo=[];
NodeNo=[];
index=0;
for i=1:dy
    for j=1:dx;
        index=index+1;
        val1=(dx+1)*(i-1)+j;
        val2=(dx+1)*(i)+j;
        Nd1=val1;
        Nd2=val1+1;
        Nd3=val2+1;
        Nd4=val2;
        NodeNo (index,:)=[Nd1 Nd2 Nd3 Nd4];
        dofNo(index,:)=NodeNo(index,:);

    end
end

TTDOF=max(max(dofNo));

% syms a b t kx ky h
[klocal]=klocalmatrix(t1,ky1,deltax,deltay,kx1);

% klocal=subs(k1,{a,b,t,kx,ky,h},{deltax,deltay,t1,kx1,ky1,h1})

kglobal=zeros(TTDOF,TTDOF);
flocal=zeros(TTDOF,1);

for i=1:size(dofNo,1)
    for j=1:4
        dof1=dofNo(i,j);
        for k=1:4
            dof2=dofNo(i,k);
            kglobal(dof1,dof2)=kglobal(dof1,dof2)+klocal(j,k);
        end
    end
end

a=deltax;
b=deltay;
for i=1:size(NodeNo,1)
    f1=[(Qa*a*b*t1)/4;(Qa*a*b*t1)/4;(Qa*a*b*t1)/4;(Qa*a*b*t1)/4];
    for j=1:4
        dof1=dofNo(i,j);
        flocal(dof1,1)=flocal(dof1,1)+f1(j,1);
    end
end
```

```
dofe=[81;80;79;78;77;76;75;74;73;64;55;46;37;28;19;10;1];
for i=1:size(dofe,1);
    val1=dofe(i,1);
    kglobal(val1,:)=[];
    kglobal(:,val1)=[];
    flocal(val1,:)=[];
end

T=kglobal\flocal

Tori=zeros(TTDOF,1);
Tori(2:9,1)=T(1:8,1);
Tori(11:18,1)=T(9:16,1);
Tori(20:27,1)=T(17:24,1);
Tori(29:36,1)=T(25:32,1);
Tori(38:45,1)=T(33:40,1);
Tori(47:54,1)=T(41:48,1);
Tori(56:63,1)=T(49:56,1);
Tori(65:72,1)=T(57:64,1);

Zx=[];
index=0;
for i=1:dy+1
    for j=1:dx+1
        index=index+1;
        x_val=Tori(index,1);
        Zx(i,j)=x_val;
    end
end

[X,Y] = meshgrid(0:deltax:Lx, 0:deltay:Ly);

% figure (1)
% surf(X,Y,Zx)
% title('Temperature (K)')
% colorbar

figure(2)
surf(X,Y,Zx)
colormap(jet)
title('Temperature')
% xlabel('string')
% ylabel('string')
colorbar
```

APPENDIX C

```
clc;clear all;

kx1=600;
ky1=600;
Qa=0;

h1=0;
t1=0.2;

dx=100;
dy=100;

Lx=0.8;
Ly=0.5;

deltax=Lx/dx;
deltay=Ly/dy;

Node_xy = [];
index = 0;
for iy = 1:dy+1
    ycoor = (iy-1)*deltay;
    for ix = 1:dx+1
        xcoor = (ix-1)*deltax;
        index = index + 1;
        Node_xy(index,:) = [xcoor ycoor];
    end
end

dofNo=[];
NodeNo=[];
index=0;
for i=1:dy
    for j=1:dx
        index=index+1;
        val1=(dx+1)*(i-1)+j;
        val2=(dx+1)*(i)+j;
        Nd1=val1;
        Nd2=val1+1;
        Nd3=val2+1;
        Nd4=val2;
        NodeNo(index,:)=[Nd1 Nd2 Nd3 Nd4];
        dofNo(index,:)=NodeNo(index,:);
    end
end



TTDOF= max(max(dofNo));

% syms a b t kx ky h
[klocal]=klocalmatrix (t1,ky1,deltax,deltay,kx1);

kglobal=zeros(TTDOF,TTDOF);
flocal=zeros(TTDOF,1);



for i=1:size(dofNo,1)
    for j=1:4
        dof1=dofNo(i,j);
        for k=1:4
            dof2=dofNo(i,k);
            kglobal(dof1,dof2)=kglobal(dof1,dof2)+klocal(j,k);
        end
    end
end
```

```
for iNode = 1:TTDOF
    xx = Node_xy(iNode,1);
    yy = Node_xy(iNode,2);

    if xx == 0
       kglobal(iNode, :)        = 0;
        kglobal(iNode, iNode)    = 1;
        flocal(iNode, :)        = 0;

    elseif yy == 0
        kglobal(iNode, :)        = 0;
        kglobal(iNode, iNode)    = 1;
        flocal(iNode, :)        = 2;
    end


end
T=kglobal\flocal

Zx=[];
index=0;
for i=1:dy+1
    for j=1:dx+1
        index=index+1;
        x_val=T(index,1);
        Zx(i,j)=x_val;
    end
end

[X,Y] = meshgrid(0:deltax:Lx, 0:deltay:Ly);

figure(1)

surf(X,Y,Zx,'edgecolor','none')
shading interp
title('Temperature (K)- Finite Element Method')
colorbar
colormap(jet)

%  Result interpolation
FT  = TriScatteredInterp(Node_xy(:,1),Node_xy(:,2),T);

% Display Result
Node_disp = [0.75 0.25;];

Data_Result = [];
for iNode = 1:size(Node_disp,1)
    xcoor = Node_disp(iNode,1);
    ycoor = Node_disp(iNode,2);
    TempV = FT(xcoor,ycoor);
    Data_Result(iNode,:) = [xcoor ycoor TempV];
end
```

APPENDIX D

```
function [klocal]=klocalmatrix(t,ky,a,b,kx)
klocal = [t * ky * a / b / 0.3e1 + t * kx / a * b / 0.3e1 t * ky * a / b / ...
    0.6e1 - t * kx / a * b / 0.3e1 -t * kx / a * b / 0.6e1 - t * ky * a / b...
    / 0.6e1 t * kx / a * b / 0.6e1 - t * ky * a / b / 0.3e1; t * ky * a / b...
    / 0.6e1 - t * kx / a * b / 0.3e1 t * ky * a / b / 0.3e1 + t * kx / a * ...
    b / 0.3e1 t * kx / a * b / 0.6e1 - t * ky * a / b / 0.3e1 -t * kx / a *...
    b / 0.6e1 - t * ky * a / b / 0.6e1; -t * kx / a * b / 0.6e1 - t * ky * ...
    a / b / 0.6e1 t * kx / a * b / 0.6e1 - t * ky * a / b / 0.3e1 t * ky * ...
    a / b / 0.3e1 + t * kx / a * b / 0.3e1 t * ky * a / b / 0.6e1 - t * kx ...
    / a * b / 0.3e1; t * kx / a * b / 0.6e1 - t * ky * a / b / 0.3e1 -t * ...
    kx / a * b / 0.6e1 - t * ky * a / b / 0.6e1 t * ky * a / b / 0.6e1 - t...
    * kx / a * b / 0.3e1 t * ky * a / b / 0.3e1 + t * kx / a * b / 0.3e1;];
end
```

APPENDIX E

```
clc;clear all;
tic
%--------------------------------------------------------------------------
% Input
%--------------------------------------------------------------------------
%
%     (0,b)   Side4   (a,b)
%       +---------------+
%       +               +
% Side1 +               + Side3
%       +               +
%       +---------------+
%     (0,0)   Side2   (a,0)

Lx = 0.8;
Ly = 0.5;
Q=0;
maxelement=50;
azizam=1;

for fatin=7%:2:maxelement;

DivX = fatin; %element in x-direction
DivY = fatin; %element in y-direction

a = Lx/DivX; % width of element
b = Ly/DivY; % height of element

% Generate Node
Node_xy = [];
index = 0;
for iy = 1:DivY+1
    ycoor = (iy-1)*b;
    for ix = 1:DivX+1
        xcoor = (ix-1)*a;
        index = index + 1;
        Node_xy(index,:) = [xcoor ycoor];
    end
end

% Generate connectivity matrix
TEle = DivX*DivY; % Total Element;
Ele_dof = zeros(DivX*DivY,4);
index = 0;
for iy = 1:DivY
    for ix = 1:DivX
        nd1 = (iy-1)*(DivX+1) + ix;
        nd2 = nd1 + 1;
        nd4 = iy*(DivX+1) + ix;
        nd3 = nd4 +1;

        index = index + 1;
        Ele_dof(index,:)=[nd1 nd2 nd3 nd4];
    end
end

TN = max(max(Ele_dof)) % Total node


%--------------------------------------------------------------------------
% Shape function
%--------------------------------------------------------------------------
% [Ni,dNidx,dNidy] = Q4_Phy_ShapeFunc(a,b);
% Ni = matlabFunction(Ni);
% dNidx = matlabFunction(dNidx);
% dNidy = matlabFunction(dNidy);
```

```
syms x y

N1 = @(x,y)1 - 1 / a * x - 1 / b * y + 1 / a / b * x * y;
N2 = @(x,y)0.1e1 / a * x - 0.1e1 / a / b * x * y;
N3 = @(x,y)0.1e1 / a / b * x * y;
N4 = @(x,y)0.1e1 / b * y - 0.1e1 / a / b * x * y;

dN1dx = @(y)-0.1e1 / a + 0.1e1 / a / b * y;
dN2dx = @(y)0.1e1 / a - 0.1e1 / a / b * y;
dN3dx = @(y)0.1e1 / a / b * y;
dN4dx = @(y)-0.1e1 / a / b * y;

dN1dy = @(x)-0.1e1 / b + 0.1e1 / a / b * x;
dN2dy = @(x)-0.1e1 / a / b * x;
dN3dy = @(x)0.1e1 / a / b * x;
dN4dy = @(x)0.1e1 / b - 0.1e1 / a / b * x;

%-------------------------------------------------------------------------
% R1
%-------------------------------------------------------------------------
% R1L1
Term11 = int(eval((subs(N1,x,0))*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N1,x,0))*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N1,x,0))*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N1,x,0))*(subs(dN4dx,x,0))),y,0,b);
TermR1L1 = [Term11 Term12 Term13 Term14];

% R1L2
Term11 = int(eval((subs(N1,y,0))*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N1,y,0))*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N1,y,0))*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N1,y,0))*(subs(dN4dy,y,0))),x,0,a);
TermR1L2 = [Term11 Term12 Term13 Term14];

% R1L3
Term11 = int(eval((subs(N1,x,a))*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N1,x,a))*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N1,x,a))*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N1,x,a))*(subs(dN4dx,x,a))),y,0,b);
TermR1L3 = [Term11 Term12 Term13 Term14];

% R1L4
Term11 = int(eval((subs(N1,y,b))*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N1,y,b))*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N1,y,b))*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N1,y,b))*(subs(dN4dy,y,b))),x,0,a);
TermR1L4 = [Term11 Term12 Term13 Term14];

R1 = - TermR1L1 - TermR1L2 + TermR1L3 + TermR1L4;



%-------------------------------------------------------------------------
% R2
%-------------------------------------------------------------------------
% R2L1
Term11 = int(eval((subs(N2,x,0))*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N2,x,0))*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N2,x,0))*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N2,x,0))*(subs(dN4dx,x,0))),y,0,b);
TermR2L1 = [Term11 Term12 Term13 Term14];

% R2L2
Term11 = int(eval((subs(N2,y,0))*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N2,y,0))*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N2,y,0))*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N2,y,0))*(subs(dN4dy,y,0))),x,0,a);
TermR2L2 = [Term11 Term12 Term13 Term14];

% R2L3
Term11 = int(eval((subs(N2,x,a))*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N2,x,a))*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N2,x,a))*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N2,x,a))*(subs(dN4dx,x,a))),y,0,b);
TermR2L3 = [Term11 Term12 Term13 Term14];
```

```
% R2L4
Term11 = int(eval((subs(N2,y,b))*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N2,y,b))*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N2,y,b))*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N2,y,b))*(subs(dN4dy,y,b))),x,0,a);
TermR2L4 = [Term11 Term12 Term13 Term14];

R2 = - TermR2L1 - TermR2L2 + TermR2L3 + TermR2L4;
%--------------------------------------------------------------------------
% R3
%--------------------------------------------------------------------------
% R3L1
Term11 = int(eval((subs(N3,x,0))*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N3,x,0))*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N3,x,0))*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N3,x,0))*(subs(dN4dx,x,0))),y,0,b);
TermR3L1 = [Term11 Term12 Term13 Term14];

% R3L2
Term11 = int(eval((subs(N3,y,0))*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N3,y,0))*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N3,y,0))*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N3,y,0))*(subs(dN4dy,y,0))),x,0,a);
TermR3L2 = [Term11 Term12 Term13 Term14];

% R3L3
Term11 = int(eval((subs(N3,x,a))*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N3,x,a))*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N3,x,a))*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N3,x,a))*(subs(dN4dx,x,a))),y,0,b);
TermR3L3 = [Term11 Term12 Term13 Term14];

% R3L4
Term11 = int(eval((subs(N3,y,b))*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N3,y,b))*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N3,y,b))*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N3,y,b))*(subs(dN4dy,y,b))),x,0,a);
TermR3L4 = [Term11 Term12 Term13 Term14];

R3 = - TermR3L1 - TermR3L2 + TermR3L3 + TermR3L4;
%--------------------------------------------------------------------------
% R4
%--------------------------------------------------------------------------
% R4L1
Term11 = int(eval((subs(N4,x,0))*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N4,x,0))*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N4,x,0))*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N4,x,0))*(subs(dN4dx,x,0))),y,0,b);
TermR4L1 = [Term11 Term12 Term13 Term14];

% R4L2
Term11 = int(eval((subs(N4,y,0))*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N4,y,0))*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N4,y,0))*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N4,y,0))*(subs(dN4dy,y,0))),x,0,a);
TermR4L2 = [Term11 Term12 Term13 Term14];

% R4L3
Term11 = int(eval((subs(N4,x,a))*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N4,x,a))*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N4,x,a))*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N4,x,a))*(subs(dN4dx,x,a))),y,0,b);
TermR4L3 = [Term11 Term12 Term13 Term14];

% R4L4
Term11 = int(eval((subs(N4,y,b))*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N4,y,b))*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N4,y,b))*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N4,y,b))*(subs(dN4dy,y,b))),x,0,a);
TermR4L4 = [Term11 Term12 Term13 Term14];

R4 = - TermR4L1 - TermR4L2 + TermR4L3 + TermR4L4;
```

```
%-------------------------------------------------------------------------
% klocal
%-------------------------------------------------------------------------
klocal1 = [R1;R2;R3;R4];

% klocal2 = [   a/(3*b) + b/(3*a),   a/(6*b) - b/(3*a), - a/(6*b) - b/(6*a),   b/(6*a) - a/(3*b);
%            a/(6*b) - b/(3*a),   a/(3*b) + b/(3*a),   b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a);
%          - a/(6*b) - b/(6*a),   b/(6*a) - a/(3*b),   a/(3*b) + b/(3*a),   a/(6*b) - b/(3*a);
%           b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a),   a/(6*b) - b/(3*a),   a/(3*b) + b/(3*a)]
%
%
% res = klocal2-klocal1;

klocal = klocal1;


%-------------------------------------------------------------------------
% flocal
%-------------------------------------------------------------------------
flocal=[Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;];

%-------------------------------------------------------------------------
% procedure
%-------------------------------------------------------------------------

KG = zeros(TN,TN);
FG = zeros(TN,1);

% Assemble
for iEle = 1:size(Ele_dof,1)
    uvw = Ele_dof(iEle,:);
    KG(uvw,uvw) = KG(uvw,uvw) + klocal;
    FG(uvw,1) = FG(uvw,1) + flocal;
end
% [ KG,FG ] = NEUMANN( KG,FG,Node_xy, a, TN)

% Specify DOF
for iNode = 1:TN
    xx = Node_xy(iNode,1);
    yy = Node_xy(iNode,2);

    if xx == 0
        KG(iNode, :)       = 0;
        KG(iNode, iNode)   = 1;
        FG(iNode, :)       = 0;
%     elseif xx == DivX*a
%         KG(iNode, :)       = 0;
%         KG(iNode, iNode)   = 1;
%         FG(iNode, :)       = 6;
    elseif yy == 0
        KG(iNode, :)       = 0;
        KG(iNode, iNode)   = 1;
        FG(iNode, :)       = 2;
    end


end

% solve simultaneous equation
dof = KG\FG;
% dof'
maxT(azizam,1)=TN;
maxT(azizam,2)=max(dof)
tt=toc
maxT(azizam,3)=tt;
azizam=azizam+1;
clc
end
maxT
```

```
%-------------------------------------------------------------------------
% plot result
%-------------------------------------------------------------------------

%  Result interpolation
FT  = TriScatteredInterp(Node_xy(:,1),Node_xy(:,2),dof);
%
% % Evaluate Result
coor_x = [];
coor_y = [];
coor_z = [];
disp_T = [];

for iele = 1:size(Ele_dof,1)
    Nd1 = Ele_dof(iele,1);
    Nd2 = Ele_dof(iele,2);
    Nd3 = Ele_dof(iele,3);
    Nd4 = Ele_dof(iele,4);

    x1 = Node_xy(Nd1,1);
    x2 = Node_xy(Nd2,1);
    x3 = Node_xy(Nd3,1);
    x4 = Node_xy(Nd4,1);

    y1 = Node_xy(Nd1,2);
    y2 = Node_xy(Nd2,2);
    y3 = Node_xy(Nd3,2);
    y4 = Node_xy(Nd4,2);

    T1 = FT(x1,y1);
    T2 = FT(x2,y2);
    T3 = FT(x3,y3);
    T4 = FT(x4,y4);

    coor_x(:,iele) = [x1;x2;x3;x4];
    coor_y(:,iele) = [y1;y2;y3;y4];
    coor_z(:,iele) = [1;1;1;1];
    disp_T(:,iele) = [T1;T2;T3;T4];
end

figure(2)
fill3(coor_x,coor_y,coor_z,disp_T,'EdgeColor','none')
colormap(jet)
title('Temperature (K)- New Divergence Method')%(Total Node = ' num2str(TN) ' and Total Ele = '
num2str(TEle) ' )'])
% xlabel('string')
% ylabel('string')
colorbar
% Display Result
Node_disp = [0 0;
             0.2 0.25;
             0.4 0.25;
             0.6 0.25;
             0.8 0.25];
Data_Result = [];
for iNode = 1:size(Node_disp,1)
    xcoor = Node_disp(iNode,1);
    ycoor = Node_disp(iNode,2);
    TempV = FT(xcoor,ycoor);
    Data_Result(iNode,:) = [xcoor ycoor TempV];
end
Data_Input = [DivX DivY TN]
Data_Result

% if centre 6,4 have dof
Val = [0 0];
for iNode = 1:TN
    xcoor = Node_xy(iNode,1);
    ycoor = Node_xy(iNode,2);
    if xcoor == 6 && ycoor == 4
        TempV = dof(iNode,1);
        Val = [iNode TempV];
    end
end
Val
```

APPENDIX F

```
clc;clear;
%--------------------------------------------------------------------------
% Input
%--------------------------------------------------------------------------
Lx  = 0.8;  % Total length in x-direction (m)
Ly  = 0.5;  % Total length in y-direction (m)
kxy = 600;  % Thermal conductivity (W/m.K)
Q   = 0;    % Heat source (W/m^3)
qx  = 3700; % Heat flux (W/m2)

tic

divX = 100;   % No of element in x-direction
divY = 100;   % No of element in y-direction

%--------------------------------------------------------------------------
% Prepare data for FEM procedure
%--------------------------------------------------------------------------
deltaX = Lx/divX;   % Elemental length in x-direction (m)
deltaY = Ly/divY;   % Elemental length in y-direction (m)

% Generate node coordinate, Node_xy
Node_xy = [];
iNode = 0;
for iy = 1:divY+1
    ycoor = (iy-1)*deltaY;
    for ix = 1:divX+1
        xcoor = (ix-1)*deltaX;
        iNode = iNode + 1;
        Node_xy(iNode,:) = [xcoor ycoor];
    end
end

% Generate Connectivity Matrix, ConvMat
ConvMat = [];
iNode = 0;
for ix = 1:divY
    for iy = 1:divX
        iNode   = iNode+1;
        val1    = (divX+1)*(ix-1)+iy;
        val2    = (divX+1)*(ix)+iy;
        Nd1     = val1;
        Nd2     = val1+1;
        Nd3     = val2+1;
        Nd4     = val2;
        ConvMat(iNode,:) = [Nd1 Nd2 Nd3 Nd4];
    end
end

TNodes = size(Node_xy,1);   % Total number of node
NumEle = size(ConvMat,1);   % Total number of element

%--------------------------------------------------------------------------
% Local stiffness,body force vector and flux vector
%--------------------------------------------------------------------------
a = deltaX;
b = deltaY;
klocal_xx = [kxy / a * b / 0.3e1 -kxy / a * b / 0.3e1 -kxy / a * b / 0.6e1 kxy / a * b / 0.6e1;
            -kxy / a * b / 0.3e1 kxy / a * b / 0.3e1 kxy / a * b / 0.6e1 -kxy / a * b / 0.6e1;
            -kxy / a * b / 0.6e1 kxy / a * b / 0.6e1 kxy / a * b / 0.3e1 -kxy / a * b / 0.3e1;
             kxy / a * b / 0.6e1 -kxy / a * b / 0.6e1 -kxy / a * b / 0.3e1 kxy / a * b / 0.3e1;];
klocal_yy = [kxy / b * a / 0.3e1 kxy / b * a / 0.6e1 -kxy / b * a / 0.6e1 -kxy / b * a / 0.3e1;
             kxy / b * a / 0.6e1 kxy / b * a / 0.3e1 -kxy / b * a / 0.3e1 -kxy / b * a / 0.6e1;
            -kxy / b * a / 0.6e1 -kxy / b * a / 0.3e1 kxy / b * a / 0.3e1 kxy / b * a / 0.6e1;
            -kxy / b * a / 0.3e1 -kxy / b * a / 0.6e1 kxy / b * a / 0.6e1 kxy / b * a / 0.3e1;];
```

```
klocal = klocal_xx + klocal_yy;

fheat = [Q * a * b / 0.4e1;
         Q * a * b / 0.4e1;
         Q * a * b / 0.4e1;
         Q * a * b / 0.4e1;];

fflux = [0;
         qx * b / 0.2e1;
         qx * b / 0.2e1;
         0;];
%--------------------------------------------------------------------------
% Assemble into global stiffness and vector
%--------------------------------------------------------------------------
Fglobal = zeros(TNodes,1);
Kglobal = zeros(TNodes,TNodes);

for iEle = 1:NumEle

    uv = ConvMat(iEle,:);   % Element connectivity

    Fglobal(uv,1)  = Fglobal(uv,1) + fheat;
    Kglobal(uv,uv) = Kglobal(uv,uv) + klocal;

    % Check element at flux boundary
    Node2 = uv(2);
    Node3 = uv(3);

    x2 = Node_xy(Node2,1);
    x3 = Node_xy(Node3,1);

    if x2 == Lx || x3 == Lx % If element node at flux boundary
        Fglobal(uv,1)  = Fglobal(uv,1) + fflux;
    end

end

%--------------------------------------------------------------------------
% Specify dirichlet boundary condition (Temperature,T)
%--------------------------------------------------------------------------

for iNode = 1:TNodes

    x_coor = Node_xy(iNode,1);
    y_coor = Node_xy(iNode,2);

    if x_coor == 0

        Kglobal(iNode, :)       = 0;
        Kglobal(iNode, iNode)   = 1;
        Fglobal(iNode, :)       = 0;

    elseif y_coor == 0

        Kglobal(iNode, :)       = 0;
        Kglobal(iNode, iNode)   = 1;
        Fglobal(iNode, :)       = 2;

    end

end

%--------------------------------------------------------------------------
% Solve simultaneous eqn for Temperature,T
%--------------------------------------------------------------------------
T = Kglobal \ Fglobal;

toc
```

```
%-------------------------------------------------------------------------
% Display result
%-------------------------------------------------------------------------
%  Result interpolation
FT  = scatteredInterpolant(Node_xy(:,1),Node_xy(:,2),T);

Node_disp = [0.75 0.4];

Data_Result = [];
for iNode = 1:size(Node_disp,1)
    xcoor = Node_disp(iNode,1);
    ycoor = Node_disp(iNode,2);
    TempV = FT(xcoor,ycoor);
    Data_Result(iNode,:) = [xcoor ycoor TempV];
end
FEM_Result = Data_Result



%-------------------------------------------------------------------------
% Plot result of Temperature,T
%-------------------------------------------------------------------------
Zx=[];
index=0;
for ix = 1:divY+1
    for iy = 1:divX+1
        index   = index+1;
        x_val   = T(index,1);
        Zx(ix,iy)   = x_val;
    end
end

[X,Y] = meshgrid(0:deltaX:Lx, 0:deltaY:Ly);

figure(1)

surf(X,Y,Zx,'edgecolor','none')
shading interp
title('Temperature (K)- Finite Element Method')
colorbar
colormap(jet)

%-------------------------------------------------------------------------
% End of coding
%-------------------------------------------------------------------------
```

APPENDIX G

```
clc;clear all;

%-------------------------------------------------------------------------
% Input
%-------------------------------------------------------------------------
%
%      (0,b)    Side4    (a,b)
%        +---------------+
%        +               +
% Side1  +               + Side3
%        +               +
%        +---------------+
%      (0,0)    Side2    (a,0)

Lx = 0.8;
Ly = 0.5;
kxy = 600;
Q = 0;
maxelement = 20;
azizam = 1;

tic

for fatin=100%:2:maxelement;

DivX = fatin; %element in x-direction
DivY = fatin; %element in y-direction

a = Lx/DivX; % width of element
b = Ly/DivY; % height of element

% Generate Node
Node_xy = [];
index = 0;
for iy = 1:DivY+1
    ycoor = (iy-1)*b;
    for ix = 1:DivX+1
        xcoor = (ix-1)*a;
        index = index + 1;
        Node_xy(index,:) = [xcoor ycoor];
    end
end

% Generate connectivity matrix
TEle = DivX*DivY; % Total Element;
Ele_dof = zeros(DivX*DivY,4);
index = 0;
for iy = 1:DivY
    for ix = 1:DivX
        nd1 = (iy-1)*(DivX+1) + ix;
        nd2 = nd1 + 1;
        nd4 = iy*(DivX+1) + ix;
        nd3 = nd4 +1;

        index = index + 1;
        Ele_dof(index,:)=[nd1 nd2 nd3 nd4];
    end
end

TN = max(max(Ele_dof)) % Total node


%-------------------------------------------------------------------------
% Shape function
%-------------------------------------------------------------------------
% [Ni,dNidx,dNidy] = Q4_Phy_ShapeFunc(a,b);
% Ni = matlabFunction(Ni);
% dNidx = matlabFunction(dNidx);
% dNidy = matlabFunction(dNidy);

syms x y
```

```
N1 = @(x,y)1 - 1 / a * x - 1 / b * y + 1 / a / b * x * y;
N2 = @(x,y)0.1e1 / a * x - 0.1e1 / a / b * x * y;
N3 = @(x,y)0.1e1 / a / b * x * y;
N4 = @(x,y)0.1e1 / b * y - 0.1e1 / a / b * x * y;

dN1dx = @(y)-0.1e1 / a + 0.1e1 / a / b * y;
dN2dx = @(y)0.1e1 / a - 0.1e1 / a / b * y;
dN3dx = @(y)0.1e1 / a / b * y;
dN4dx = @(y)-0.1e1 / a / b * y;


dN1dy = @(x)-0.1e1 / b + 0.1e1 / a / b * x;
dN2dy = @(x)-0.1e1 / a / b * x;
dN3dy = @(x)0.1e1 / a / b * x;
dN4dy = @(x)0.1e1 / b - 0.1e1 / a / b * x;

%-------------------------------------------------------------------------
% R1
%-------------------------------------------------------------------------
% R1L1
Term11 = int(eval((subs(N1,x,0))*kxy*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N1,x,0))*kxy*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N1,x,0))*kxy*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N1,x,0))*kxy*(subs(dN4dx,x,0))),y,0,b);
TermR1L1 = [Term11 Term12 Term13 Term14];

% R1L2
Term11 = int(eval((subs(N1,y,0))*kxy*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N1,y,0))*kxy*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N1,y,0))*kxy*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N1,y,0))*kxy*(subs(dN4dy,y,0))),x,0,a);
TermR1L2 = [Term11 Term12 Term13 Term14];

% R1L3
Term11 = int(eval((subs(N1,x,a))*kxy*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N1,x,a))*kxy*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N1,x,a))*kxy*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N1,x,a))*kxy*(subs(dN4dx,x,a))),y,0,b);
TermR1L3 = [Term11 Term12 Term13 Term14];

% R1L4
Term11 = int(eval((subs(N1,y,b))*kxy*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N1,y,b))*kxy*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N1,y,b))*kxy*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N1,y,b))*kxy*(subs(dN4dy,y,b))),x,0,a);
TermR1L4 = [Term11 Term12 Term13 Term14];

R1 = - TermR1L1 - TermR1L2 + TermR1L3 + TermR1L4;
%-------------------------------------------------------------------------
% R2
%-------------------------------------------------------------------------
% R2L1
Term11 = int(eval((subs(N2,x,0))*kxy*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N2,x,0))*kxy*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N2,x,0))*kxy*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N2,x,0))*kxy*(subs(dN4dx,x,0))),y,0,b);
TermR2L1 = [Term11 Term12 Term13 Term14];

% R2L2
Term11 = int(eval((subs(N2,y,0))*kxy*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N2,y,0))*kxy*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N2,y,0))*kxy*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N2,y,0))*kxy*(subs(dN4dy,y,0))),x,0,a);
TermR2L2 = [Term11 Term12 Term13 Term14];

% R2L3
Term11 = int(eval((subs(N2,x,a))*kxy*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N2,x,a))*kxy*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N2,x,a))*kxy*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N2,x,a))*kxy*(subs(dN4dx,x,a))),y,0,b);
TermR2L3 = [Term11 Term12 Term13 Term14];
```

```
% R2L4
Term11 = int(eval((subs(N2,y,b))*kxy*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N2,y,b))*kxy*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N2,y,b))*kxy*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N2,y,b))*kxy*(subs(dN4dy,y,b))),x,0,a);
TermR2L4 = [Term11 Term12 Term13 Term14];

R2 = - TermR2L1 - TermR2L2 + TermR2L3 + TermR2L4;


%-------------------------------------------------------------------------
% R3
%-------------------------------------------------------------------------
% R3L1
Term11 = int(eval((subs(N3,x,0))*kxy*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N3,x,0))*kxy*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N3,x,0))*kxy*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N3,x,0))*kxy*(subs(dN4dx,x,0))),y,0,b);
TermR3L1 = [Term11 Term12 Term13 Term14];

% R3L2
Term11 = int(eval((subs(N3,y,0))*kxy*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N3,y,0))*kxy*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N3,y,0))*kxy*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N3,y,0))*kxy*(subs(dN4dy,y,0))),x,0,a);
TermR3L2 = [Term11 Term12 Term13 Term14];

% R3L3
Term11 = int(eval((subs(N3,x,a))*kxy*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N3,x,a))*kxy*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N3,x,a))*kxy*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N3,x,a))*kxy*(subs(dN4dx,x,a))),y,0,b);
TermR3L3 = [Term11 Term12 Term13 Term14];

% R3L4
Term11 = int(eval((subs(N3,y,b))*kxy*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N3,y,b))*kxy*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N3,y,b))*kxy*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N3,y,b))*kxy*(subs(dN4dy,y,b))),x,0,a);
TermR3L4 = [Term11 Term12 Term13 Term14];

R3 = - TermR3L1 - TermR3L2 + TermR3L3 + TermR3L4;
%-------------------------------------------------------------------------
% R4
%-------------------------------------------------------------------------
% R4L1
Term11 = int(eval((subs(N4,x,0))*kxy*(subs(dN1dx,x,0))),y,0,b);
Term12 = int(eval((subs(N4,x,0))*kxy*(subs(dN2dx,x,0))),y,0,b);
Term13 = int(eval((subs(N4,x,0))*kxy*(subs(dN3dx,x,0))),y,0,b);
Term14 = int(eval((subs(N4,x,0))*kxy*(subs(dN4dx,x,0))),y,0,b);
TermR4L1 = [Term11 Term12 Term13 Term14];

% R4L2
Term11 = int(eval((subs(N4,y,0))*kxy*(subs(dN1dy,y,0))),x,0,a);
Term12 = int(eval((subs(N4,y,0))*kxy*(subs(dN2dy,y,0))),x,0,a);
Term13 = int(eval((subs(N4,y,0))*kxy*(subs(dN3dy,y,0))),x,0,a);
Term14 = int(eval((subs(N4,y,0))*kxy*(subs(dN4dy,y,0))),x,0,a);
TermR4L2 = [Term11 Term12 Term13 Term14];

% R4L3
Term11 = int(eval((subs(N4,x,a))*kxy*(subs(dN1dx,x,a))),y,0,b);
Term12 = int(eval((subs(N4,x,a))*kxy*(subs(dN2dx,x,a))),y,0,b);
Term13 = int(eval((subs(N4,x,a))*kxy*(subs(dN3dx,x,a))),y,0,b);
Term14 = int(eval((subs(N4,x,a))*kxy*(subs(dN4dx,x,a))),y,0,b);
TermR4L3 = [Term11 Term12 Term13 Term14];

% R4L4
Term11 = int(eval((subs(N4,y,b))*kxy*(subs(dN1dy,y,b))),x,0,a);
Term12 = int(eval((subs(N4,y,b))*kxy*(subs(dN2dy,y,b))),x,0,a);
Term13 = int(eval((subs(N4,y,b))*kxy*(subs(dN3dy,y,b))),x,0,a);
Term14 = int(eval((subs(N4,y,b))*kxy*(subs(dN4dy,y,b))),x,0,a);
TermR4L4 = [Term11 Term12 Term13 Term14];

R4 = - TermR4L1 - TermR4L2 + TermR4L3 + TermR4L4;
```

```
%--------------------------------------------------------------------------
% klocal
%--------------------------------------------------------------------------
klocal1 = [R1;R2;R3;R4];

% klocal2 = [   a/(3*b) + b/(3*a),   a/(6*b) - b/(3*a), - a/(6*b) - b/(6*a),   b/(6*a) - a/(3*b);
%            a/(6*b) - b/(3*a),   a/(3*b) + b/(3*a),   b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a);
%          - a/(6*b) - b/(6*a),   b/(6*a) - a/(3*b),   a/(3*b) + b/(3*a),   a/(6*b) - b/(3*a);
%           b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a),   a/(6*b) - b/(3*a),   a/(3*b) + b/(3*a)];


% res = klocal2-klocal1

klocal = klocal1;
%--------------------------------------------------------------------------
% flocal
%--------------------------------------------------------------------------
flocal=[Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;
        Q * a * b / 0.4e1;];

%--------------------------------------------------------------------------
% procedure
%--------------------------------------------------------------------------

KG = zeros(TN,TN);
FG = zeros(TN,1);

% Assemble
for iEle = 1:size(Ele_dof,1)
    uvw = Ele_dof(iEle,:);
    KG(uvw,uvw) = KG(uvw,uvw) + klocal;
    FG(uvw,1) = FG(uvw,1) + flocal;
end
[ KG,FG ] = NEUMANN( KG,FG,Node_xy,a,b,TN,Lx,Ly,kxy);

% Specify DOF
for iNode = 1:TN
    xx = Node_xy(iNode,1);
    yy = Node_xy(iNode,2);

    if xx == 0
        KG(iNode, :)       = 0;
        KG(iNode, iNode)   = 1;
        FG(iNode, :)       = 0;
    elseif yy == 0
        KG(iNode, :)       = 0;
        KG(iNode, iNode)   = 1;
        FG(iNode, :)       = 2;
    end


end

% solve simultaneous equation
dof = KG\FG;
% dof'
maxT(azizam,1)=TN;
maxT(azizam,2)=max(dof)
tt=toc
maxT(azizam,3)=tt;
azizam=azizam+1;
clc
end
maxT

toc
```

```
%-------------------------------------------------------------------------
% plot result
%-------------------------------------------------------------------------

%  Result interpolation
FT  = TriScatteredInterp(Node_xy(:,1),Node_xy(:,2),dof);
%
% % Evaluate Result
coor_x = [];
coor_y = [];
coor_z = [];
disp_T = [];
for iele = 1:size(Ele_dof,1)
    Nd1 = Ele_dof(iele,1);
    Nd2 = Ele_dof(iele,2);
    Nd3 = Ele_dof(iele,3);
    Nd4 = Ele_dof(iele,4);

    x1 = Node_xy(Nd1,1);
    x2 = Node_xy(Nd2,1);
    x3 = Node_xy(Nd3,1);
    x4 = Node_xy(Nd4,1);

    y1 = Node_xy(Nd1,2);
    y2 = Node_xy(Nd2,2);
    y3 = Node_xy(Nd3,2);
    y4 = Node_xy(Nd4,2);

    T1 = FT(x1,y1);
    T2 = FT(x2,y2);
    T3 = FT(x3,y3);
    T4 = FT(x4,y4);

    coor_x(:,iele) = [x1;x2;x3;x4];
    coor_y(:,iele) = [y1;y2;y3;y4];
    coor_z(:,iele) = [1;1;1;1];
    disp_T(:,iele) = [T1;T2;T3;T4];
end

figure(2)
fill3(coor_x,coor_y,coor_z,disp_T,'EdgeColor','none')
colormap(jet)
title('Temperature (K)- New Divergence Method')%(Total Node = ' num2str(TN) ' and Total Ele = '
num2str(TEle) ' )'])
% xlabel('string')
% ylabel('string')
colorbar


% Display Result
Node_disp = [0.75 0.25];

Data_Result = [];
for iNode = 1:size(Node_disp,1)
    xcoor = Node_disp(iNode,1)
    ycoor = Node_disp(iNode,2);
    TempV = FT(xcoor,ycoor);
    Data_Result(iNode,:) = [xcoor ycoor TempV];
end

Data_Input = [DivX DivY TN]
Data_Result


% if centre 6,4 have dof
Val = [0 0];
for iNode = 1:TN
    xcoor = Node_xy(iNode,1);
    ycoor = Node_xy(iNode,2);
    if xcoor == 6 && ycoor == 4
        TempV = dof(iNode,1);
        Val = [iNode TempV];
    end
end
Val
```

APPENDIX H

```
function [ KG,FG ] = NEUMANN( KG,FG,Node_xy,a,b,TN,Lx,Ly,kxy) % a=dx
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

for iNode = 1:TN
    xx = Node_xy(iNode,1);
    yy = Node_xy(iNode,2);

    if xx == Lx
        KG(iNode, :)         = 0;
        KG(iNode, iNode-1)   = -kxy/a;
        KG(iNode, iNode)     = kxy/a;
        FG(iNode,:)          = 3700;

%     elseif yy == Ly
%         KG(iNode, :)         = 0;
%         KG(iNode, iNode-1)     = -kxy/b;
%         KG(iNode, iNode)       = kxy/b;
%         FG(iNode,:)          = 0;
    end

end
```