Acknowledgement

First I'd like to express my sincere gratitude to my project guide, **Dr. Atul Ramesh Bhagat**, Sc 'E' at High Temperature Composite Center (HTCC), Advanced System Laboratories, DRDO. I was very fortunate to work with an extremely involved and encouraging project guide, who not only guided me but also inspired me to venture into a territory unexplored before by me.

I'd like to thank **Advanced System Laboratories, DRDO, Hyderabad** for letting me use their premises and equipment for my project purpose.

A special thanks to **G. Rohini Devi** ma'am and **M. Rajagopal Reddy, Head, HRDG, ASL, DRDO** for giving me this great opportunity to pursue.

Abstract

A numerical method is used to solve an inverse heat conduction problem using finite difference method and one dimensional Newton-Raphson optimization technique. A thermocouple placed anywhere on the one dimensional rod will read the temperature at that point, this temperature when fed into the FORTRAN code can predict the heat flux subjected onto the rod. The code has been further modified to predict variable heat flux (with time) as well. Error contribution of distance of thermocouple from source and time is demonstrated. Accurate prediction of heat flux variable with time has also been validated. First, a FORTRAN code was written to simulate and solve a transient model of a rod subjected to constant temperatures on both sides using Finite Difference Method. Next, a FORTRAN code was written to solve a steady state, and consequently a transient, model of a rod subjected to heat flux from one side using FDM and is tested to measure temperature at any node on the rod. Further, this code was modified to predict heat flux based on temperature data provided using Newton-Raphson optimization technique.

Contents

Acknowledgement 1

Abstract 2

1. **List of Tables, Figures and Graphs** 4
2. **List of Symbols and Abbreviations** 5
3. **Introduction** 6
4. **Heat Transfer Equation** 7
5. **Thomas Algorithm for solving TDMA** 14
6. **Inverse Conduction** 16
7. **Numerical Example** 18
8. **Conclusion** 35
9. **References** 36

1. List of Tables, Figures and Graphs

1.1. Tables

i. Different materials used with properties

1.2. Figures

1. Dirichlet Condition
2. Neumann Condition
3. Inverse Conduction Flowchart
4. Spherical Sensor
5. Flat Face Sensor
6. Variable Input
7. Variable Node – Spherical
8. Variable Node – Flat face
9. Variable Node – Aluminum
10. Step Input – Spherical
11. Step Input – Flat face
12. Step Input – Aluminum
13. Linearly Varying Input – Spherical
14. Linearly Varying Input – Flat face
15. Linearly Varying Input – Aluminum
16. Sine Input – Spherical
17. Sine Input – Flat face
18. Sine Input – Aluminum

2. List of Symbols and Abbreviations

2.1. Symbols

i. T – Temperature

ii. x,y,z – Spatial coordinates in x, y and z direction

iii. α – Thermal Diffusivity

iv. k – Thermal Conductivity

v. C_p – Specific Heat Capacity

vi. ρ – Density

vii. $\frac{\partial}{\partial x}$ – Partial Differential

viii. Δx – Change in x

ix. Δt – Change in time

x. θ - Deciding Factor

xi. q – Heat Flux

xii. a_1, b_1, c_1 – Diagonal Elements of TDMA

xiii. γ_1, β_1 – Triangularization Terms

xiv. T_c – Calculated Temperature

xv. T_m – Measured Temperature

xvi. ε - Error

2.2. Abbreviations

i. 1D – One Dimensional

ii. 3D – Three Dimensional

iii. FDM - Finite Difference Method

3. Introduction

Inverse heat conduction techniques are used when output can be measured, consequently the input has to be calculated. It finds its application in various fields. To predict heat generated inside a furnace by measuring temperature from the outside wall, to tweak the parameters according to the needs. To predict heat flux acting on the head of a re-entry vehicle by measuring temperature from the inside, so that appropriate materials can be used to make the heat shield.

Literature Survey

Numerical methods for such problems have been developed before, most of them being based on discretization by finite difference scheme. Vogel et al. [1] take a 'Boundary inverse heat conduction problem' in which a one dimensional rod has 2 temperature boundary conditions and 2 thermocouples placed at 10% away from each end. The end temperatures are estimated by using inverse conduction technology considering the temperature history from the thermocouples. R.C.Mehta [2] considers a case with heat flux on one side of the rod and the other side open ended. In the inverse conduction technique used, he assumes an initial value for heat flux and reduces the difference between the calculated temperature and the temperature reading using Newton-Raphson method. R.C.Mehta[3], a solution is proposed for inverse heat conduction with a heat flux and a radiation boundary condition using Regula-Falsi iterative method.

The present reports on firstly, calculating temperature on nodes on the rod between 2 constant boundary condition. Next, calculating temperature on the rod with heat flux from one side and the other side being open ended. Lastly, estimating heat flux based on temperature on the rod after a certain time using iterative procedures. The calculated values of the first 2 steps are compared with theoretical solutions for validation of FORTRAN code.

4. Heat Transfer Equation

$$\alpha\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right]=\frac{\partial T}{\partial t}$$

The above equation is the 3-D heat transfer equation. In its present form it is the governing equation for heat conduction or diffusion in an isotropic medium. Its steady state form is

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}=0$$

Where change in temperature with time is zero ($\frac{\partial T}{\partial t}=0$).

A 1D form of the heat equation in unsteady state is a parabolic equation. Ignoring the partial derivatives in y and z directions, we have

$$\alpha\frac{\partial^2 T}{\partial x^2}=\frac{\partial T}{\partial t}$$

Methods of solving the heat equation

- Finite Volume Method
- Finite Element Method
- Finite Difference Method

In this report, we will be using **finite difference method**.

4.1. Finite Difference approximations

4.1.1. First Order Approximations

4.1.1.1. Forward Difference

$$\frac{\partial T}{\partial x}=\frac{T_{i+1}-T_i}{\Delta x}$$

4.1.1.2. Backward Difference

$$\frac{\partial T}{\partial x}=\frac{T_i-T_{i-1}}{\Delta x}$$

4.1.2. Second Order Approximations

4.1.2.1. Central Difference

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2}$$

4.2. Schemes for solving the equation

Here the subscript refers to the node number, whereas the superscript refers to the time step. So T_i^n refers to temperature at ith node at nth time step.

4.2.1. Explicit Method (Forward Time, Centered Space)

In this scheme, the time derivative is approximated with forward difference.

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

And the space derivative uses central difference approximation.

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}$$

Substituting these 2 equations in the governing equation, we get

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}$$

Re-arranging this equation, we get

$$T_i^{n+1} = T_i^n + \lambda(T_{i+1}^n - 2T_i^n + T_{i-1}^n)$$

$$\lambda T_{i+1}^n + (1 - 2\lambda)T_i^n + \lambda T_{i-1}^n = T_i^{n+1}$$

here, $\lambda = \alpha \frac{\Delta t}{(\Delta x)^2}$

4.2.2. Implicit Method (Backward Time, Centered Space)

In this scheme, the time derivative is approximated with forward difference.

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

The space derivative uses central difference approximation

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2}$$

Substituting these 2 equations in the governing equation, we get

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2}$$

4.2.3.

Rearranging the equation, we get

$$-\lambda T_{i+1}^{n+1} + (1+2\lambda)T_i^{n+1} - \lambda T_{i-1}^{n+1} = T_i^n$$

Here, $\lambda = \alpha \frac{\Delta t}{(\Delta x)^2}$

4.2.4. Crank-Nicholson Method

In this scheme, the time derivative is approximated with forward difference.

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

And the space derivative can be determined at midpoint by averaging the central difference approximation beginning at n and ending at n+1 time increment.

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{2}\left(\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}\right)$$

Substituting these equations in the governing equation, we get:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{1}{2}\left(\frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} + \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}\right)$$

Rearranging the equation, we get:

$$-\lambda T_{i+1}^{n+1} + 2(1+\lambda)T_i^{n+1} - \lambda T_{i-1}^{n+1} = \lambda T_{i+1}^n + 2(1-\lambda)T_i^n + \lambda T_{i-1}^n$$

Here, $\lambda = \alpha \frac{\Delta t}{(\Delta x)^2}$

4.2.5. Universal Form

To get the universal form of the above 3 schemes, we use the theta (θ) rule on the spatial derivative. The time derivative is the same as Crank-Nicholson and Implicit Method.

$$\frac{\partial T}{\partial t} = \frac{T_i^{n+1} - T_i^n}{\Delta t}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{2}\left(\theta \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} + (1-\theta)\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{(\Delta x)^2}\right)$$

Substituting these 2 equations in the governing equation, we get

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \alpha \frac{1}{2}\left(\theta \frac{T_{i+1}^{n+1} - 2T_i^{n+1} + T_{i-1}^{n+1}}{(\Delta x)^2} + (1 - \theta)\frac{T_{i+1}^{n} - 2T_i^{n} + T_{i-1}^{n}}{(\Delta x)^2}\right)$$

Rearranging the equation, we get

$$-\lambda\theta T_{i+1}^{n+1} + (1 + 2\theta\lambda)T_i^{n+1} - \lambda\theta T_{i-1}^{n+1} = \lambda(1 - \theta)T_{i+1}^{n} + \big(1 - 2\lambda(1 - \theta)\big)T_i^{n} + \lambda(1 - \theta)T_{i-1}^{n}$$

Here, $\theta = 0$, for Explicit Scheme

$\theta = 1$, for Implicit Scheme

$\theta = 0.5$, for Crank-Nicholson Scheme

4.3. Initial Condition

Let the length of the 1D rod be 'L'. The initial condition for the solution will be

$$T(x, 0) = f(x) \qquad \text{For, } 0 \le x \le L$$

Where, f(x) is a function that'll give the temperature at all the nodes at the zeroth time step. For a constant temperature, $(x, 0) = c$. This condition can also be written in this form,

$$T_i^0 = c \qquad \text{for, } i = 1, 2, \dots \dots m - 1$$

4.4. Boundary Conditions

4.4.1. Dirichlet Condition

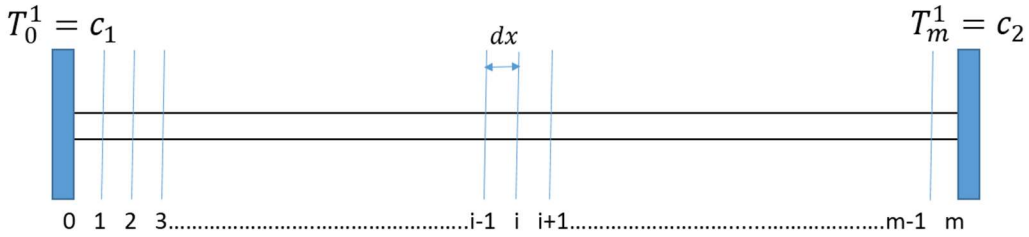

Figure 1 - Dirichlet Condition

$$T(0, t) = T_0^j = c_1 \qquad \text{For, } 0 \le t \le t_c \text{ or j = 1, 2 … n}$$

$$T(L, t) = T_m^j = c_2 \qquad \text{For, } 0 \le t \le t_c \text{ or j = 1, 2 … n}$$

The system of equations for nodes 0 to m, for the j+1[th] time step, will look like this.

$$[A][x] = [B]$$

$$\begin{bmatrix} 1+2\lambda\theta & -\lambda\theta & 0 & \dots & \dots & \dots & \dots & 0 \\ -\lambda\theta & 1+2\lambda\theta & -\lambda\theta & 0 & \dots & \dots & \dots & 0 \\ 0 & -\lambda\theta & 1+2\lambda\theta & -\lambda\theta & \dots & \dots & \dots & 0 \\ \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots & \dots \\ \dots & \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots \\ \dots & \dots & \dots & \dots & \ddots & 1+2\lambda\theta & -\lambda\theta & 0 \\ \dots & \dots & \dots & \dots & \dots & -\lambda\theta & 1+2\lambda\theta & -\lambda\theta \\ 0 & 0 & 0 & 0 & \dots & 0 & -\lambda\theta & 1+2\lambda\theta \end{bmatrix} \begin{bmatrix} T_1^{j+1} \\ T_2^{j+1} \\ T_3^{j+1} \\ \dots \\ \dots \\ \dots \\ \dots \\ T_{m-1}^{j+1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ d_{m-1} \end{bmatrix}$$

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ d_{m-1} \end{bmatrix} = \begin{bmatrix} \lambda(1-\theta)T_0^j + (1-2\lambda\theta)T_1^j + \lambda(1-\theta)T_2^j + \lambda\theta T_0^j \\ \lambda(1-\theta)T_1^j + (1-2\lambda\theta)T_2^j + \lambda(1-\theta)T_3^j \\ \lambda(1-\theta)T_2^j + (1-2\lambda\theta)T_3^j + \lambda(1-\theta)T_4^j \\ \dots \\ \dots \\ \dots \\ \dots \\ \lambda(1-\theta)T_{m-2}^j + (1-2\lambda\theta)T_{m-1}^j + \lambda(1-\theta)T_m^j + \lambda\theta T_m^j \end{bmatrix}$$

This set of equations can be solved using Thomas Algorithm, as it is in the form of a TDMA (Tri-Diagonal Matrix). However, to solve for j+2[th] time step, replace the solution matrix into [x] in [A][x] = [B] and solve the TDMA again.

4.4.2. Neumann Condition

$$-k\frac{\partial T}{\partial x} = q\Big|_{x=0}$$

$$-k\frac{\partial T}{\partial x} = 0\Big|_{x=L}$$

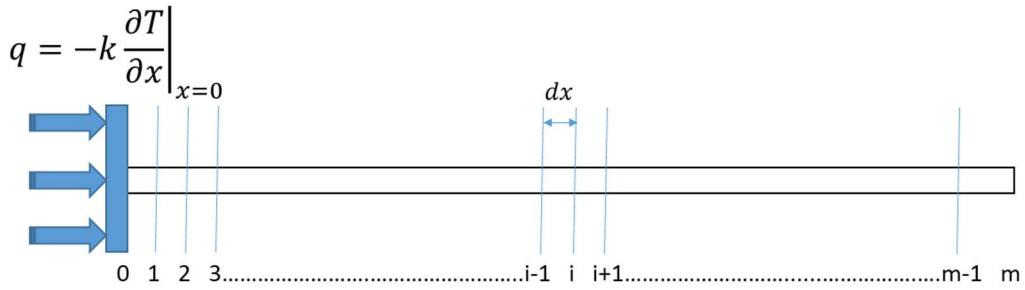

Figure 2 - Neumann Condition

The heat flux condition on one end and insulated on the other end can be formulated in the following way.

$$-k\frac{\partial T}{\partial x} = q - \rho C_p \frac{\partial T}{\partial t}\left(\frac{\Delta x}{2}\right)\Big|_{x=0}$$

$$-k\frac{\partial T}{\partial x} = -\rho C_p \frac{\partial T}{\partial t}\left(\frac{\Delta x}{2}\right)\Big|_{x=L}$$

Applying finite difference approximations and theta rule on the boundary conditions, we get

$$-k\left[\theta\frac{T_0^{j+1} - T_1^{j+1}}{\Delta x} + (1-\theta)\frac{T_0^{j} - T_1^{j}}{\Delta x}\right] = q - \rho C_p\left[\frac{T_0^{j+1} - T_0^{j}}{\Delta t}\right]\left(\frac{\Delta x}{2}\right)$$

$$-k\left[\theta\frac{T_m^{j+1} - T_{m-1}^{j+1}}{\Delta x} + (1-\theta)\frac{T_m^{j} - T_{m-1}^{j}}{\Delta x}\right] = -\rho C_p\left[\frac{T_m^{j+1} - T_m^{j}}{\Delta t}\right]\left(\frac{\Delta x}{2}\right)$$

We have used forward difference at x=0, and backward difference at x = L at j^{th} time step.

When we rearrange the terms, we have

$$(1+2\lambda\theta)T_0^{j+1} - 2\lambda\theta T_1^{j+1} = 2\lambda(1-\theta)T_1^{j} + \big(1-2\lambda(1-\theta)\big)T_0^{j} + \frac{2q\Delta t}{\rho C_p \Delta x}$$

$$-2\lambda\theta T_{m-1}^{j+1} + (1+2\lambda\theta)T_m^{j+1} = 2\lambda(1-\theta)T_{m-1}^{j} + \big(1-2\lambda(1-\theta)\big)T_m^{j}$$

For all the nodes between, from i = 1,2……m-1, we can use the equation from Dirichlet condition.

$$-\lambda\theta T_{i-1}^{j+1} + (1+2\lambda\theta)T_i^{j+1} - \lambda\theta T_{i+1}^{j+1} = \lambda(1-\theta)T_{i-1}^{j} + \big(1-2\lambda(1-\theta)\big)T_i^{j} + \lambda(1-\theta)T_{i+1}^{j}$$

This system of linear equations can be written in matrix form.

$$\begin{bmatrix} 1+2\lambda\theta & -2\lambda\theta & 0 & \dots & \dots & \dots & \dots & 0 \\ -\lambda\theta & 1+2\lambda\theta & -\lambda\theta & 0 & \dots & \dots & \dots & 0 \\ 0 & -\lambda\theta & 1+2\lambda\theta & -\lambda\theta & \dots & \dots & \dots & 0 \\ \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots & \dots \\ \dots & \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots \\ \dots & \dots & \dots & \dots & \ddots & 1+2\lambda\theta & -\lambda\theta & 0 \\ \dots & \dots & \dots & \dots & \dots & -\lambda\theta & 1+2\lambda\theta & -\lambda\theta \\ 0 & 0 & 0 & 0 & \dots & 0 & -2\lambda\theta & 1+2\lambda\theta \end{bmatrix} \begin{bmatrix} T_0^{j+1} \\ T_1^{j+1} \\ T_2^{j+1} \\ \dots \\ \dots \\ \dots \\ \dots \\ T_m^{j+1} \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \dots \\ \dots \\ \dots \\ \dots \\ d_m \end{bmatrix}$$

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \dots \\ \dots \\ \dots \\ \dots \\ d_m \end{bmatrix} = \begin{bmatrix} \big(1-2\lambda(1-\theta)\big)T_0^j + 2\lambda(1-\theta)T_1^j + \dfrac{2q\Delta t}{\rho C_p \Delta x} \\ \lambda(1-\theta)T_0^j + \big(1-2\lambda(1-\theta)\big)T_1^j + \lambda(1-\theta)T_2^j \\ \lambda(1-\theta)T_1^j + \big(1-2\lambda(1-\theta)\big)T_2^j + \lambda(1-\theta)T_3^j \\ \dots \\ \dots \\ \dots \\ \dots \\ 2\lambda(1-\theta)T_{m-1}^j + \big(1-2\lambda(1-\theta)\big)T_m^j \end{bmatrix}$$

This set of equations can be solved using Thomas Algorithm, as it is in the form of a TDMA (Tri-Diagonal Matrix).

However, to solve for j+2th time step, replace the solution matrix into [x] in [A][x] = [B] and solve the TDMA again.

5. Thomas Algorithm for solving TDMA

Thomas algorithm is an algorithm used to solve a **T**ri-**D**iagonal **MA**trix. Consider a system of equations given in the following form:

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & \dots & \dots & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & \dots & \dots & 0 \\ \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots & \dots \\ \dots & \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots \\ \dots & \dots & \dots & \dots & \ddots & b_{m-2} & c_{m-2} & 0 \\ \dots & \dots & \dots & \dots & \dots & a_{m-1} & b_{m-1} & c_{m-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_m & b_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ x_m \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ d_m \end{bmatrix}$$

5.1.1. Step 1: Triangularization

$$\gamma_1 = \frac{c_1}{b_1}$$

$$\gamma_k = \frac{c_k}{b_k - a_k \gamma_{k-1}} \qquad \text{for k = 2, 3, ……(m-1)}$$

$$\beta_1 = \frac{d_1}{b_1}$$

$$\gamma_k = \frac{d_k - a_k \beta_{k-1}}{b_k - a_k \gamma_{k-1}} \qquad \text{for k = 2, 3, ……m}$$

This operation will give rise to

$$\begin{bmatrix} 1 & \gamma_1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & \gamma_2 & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & 1 & \gamma_3 & \dots & \dots & \dots & 0 \\ \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots & \dots \\ \dots & \dots & \dots & \ddots & \ddots & \ddots & \dots & \dots \\ \dots & \dots & \dots & \dots & \ddots & 1 & \gamma_{m-2} & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 & 1 & \gamma_{m-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ x_m \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \\ \dots \\ \dots \\ \dots \\ \beta_m \end{bmatrix}$$

5.1.2. Step 2: Backward Sweep

Now we move towards the solution vector

$$x_m = \beta_m$$

$$x_k = \beta_k - \gamma_k x_k \qquad \text{for, k = (n-1), (n-2), ……, 1}$$

6. Inverse Conduction

Inverse conduction is a concept in which input heat flux has to be calculated when temperature distribution across the body has been given. In this report, we will be looking at a 1D rod subjected to an unknown heat flux. Temperature at the end node after a certain amount of time is measured using a thermocouple. Using FDM and Newton Raphson, with an iterative procedure, we calculate the input heat flux.

First we assume an initial amount of heat flux and get the temperature at a designated node after a certain amount of time using FDM. This is calculated temperature (T_c). If the difference between T_c and measured temperature (T_m) is more than a designated error (ε) value, the assumed heat flux is either increased or decreased by an amount. This increase or decrease in assumed heat flux is given by Newton Raphson.

$$q_{i+1} = q_i - \frac{f(x)}{f'(x)}$$

Here, $f(x) = (T_c - T_m)$, and $f'(x)$ is given by calculating slope of change in temperature for a linearly increasing heat flux(q) for a designated amount of time, in this case, it is for however long the flux is being subjected onto the 1D rod. Once $(T_c - T_m) \leq \varepsilon$, the iterative loop stops and q_{i+1} is the calculated heat flux.

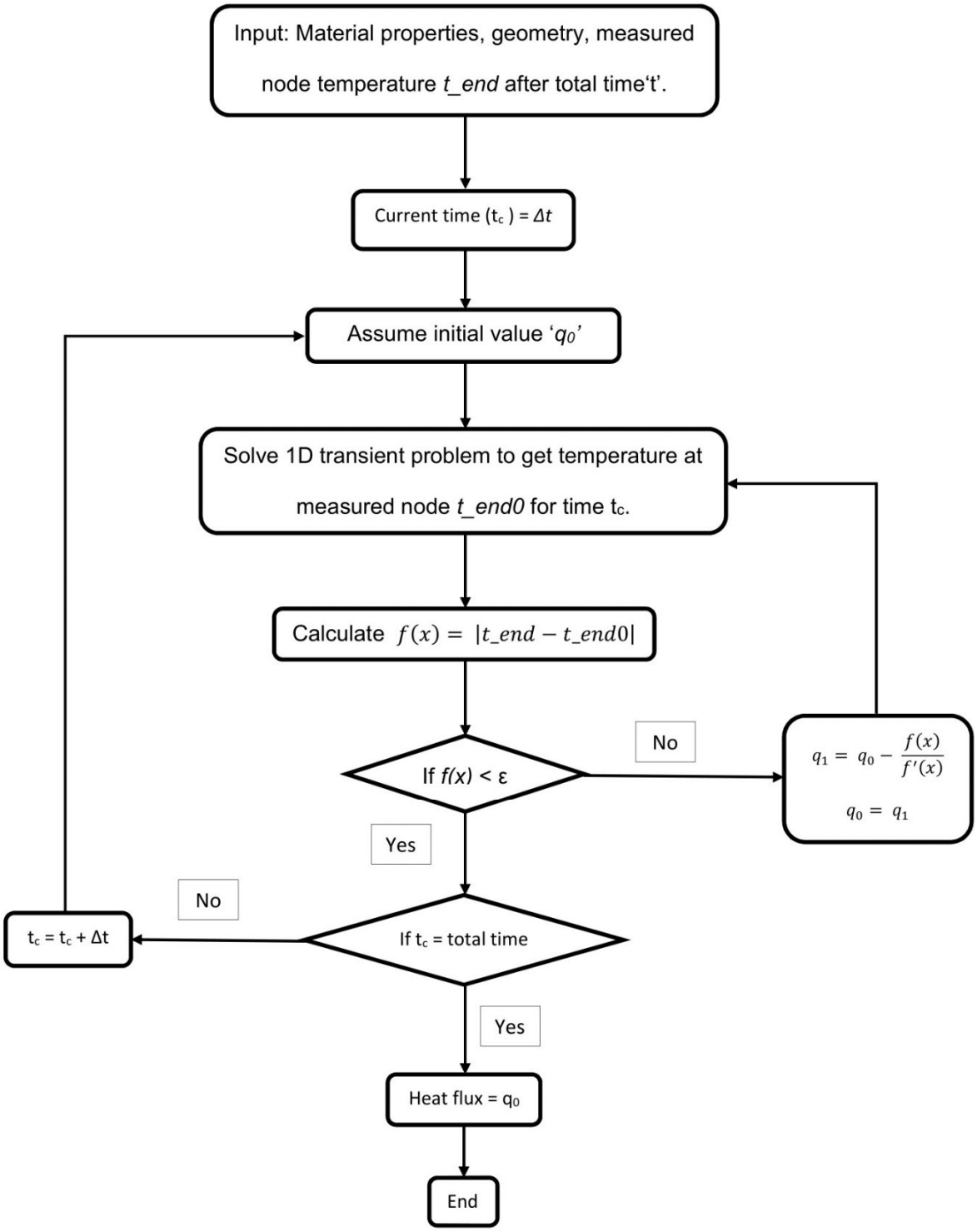

Figure 3 - Inverse Conduction Flowchart

7. Numerical Example

Three cases with difference property 1D rods have been chosen. Their Properties are as follows:

Properties	Copper(Spherical)	Copper(Flat face)	Aluminum
Length(m)	0.00218m	0.0025m	0.05m
Density(kg/m^3)	8960	8960	7800
Thermal Conductivity(W/m.K)	100	100	50
C_p	1000	1000	1000

Table No. 1 – Different Materials used with properties

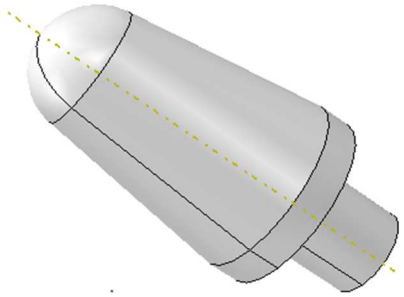

Figure 4 - Spherical Sensor

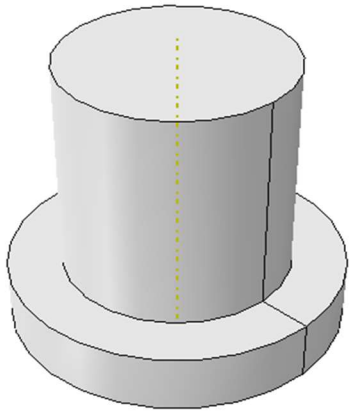

Figure 5 - Flat Face Sensor

These 3 rods have been subjected through various tests with the FORTRAN code. These tests are as follows:

a. Estimation of heat flux when thermocouple is placed at difference distances from the source.

Estimation of heat flux, when it varying at source with time. The following figure shoes the different cases chosen.

b. Heat Flux increasing in step form.
c. Heat Flux increasing linearly.
d. Heat Flux varying in Sine form.

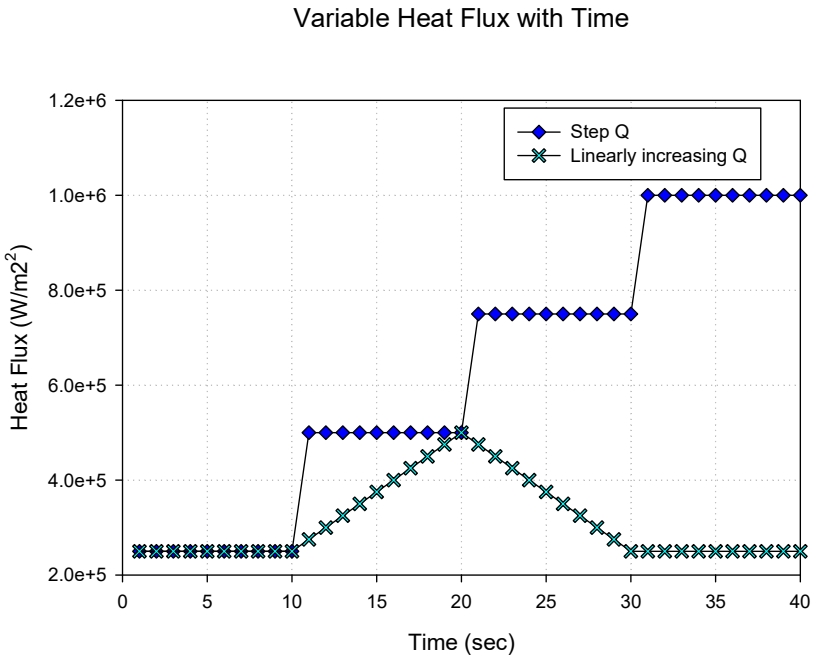

Figure 6 - Variable Input (1)

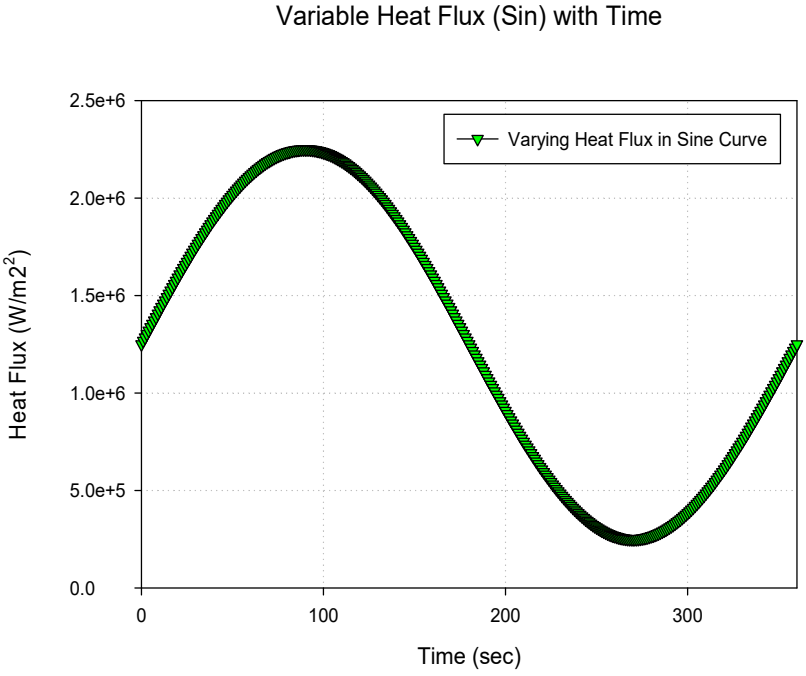

Figure 6 - Variable Input (2)

Results

a. For Variable node vs Percent of length

1. Copper(Spherical)

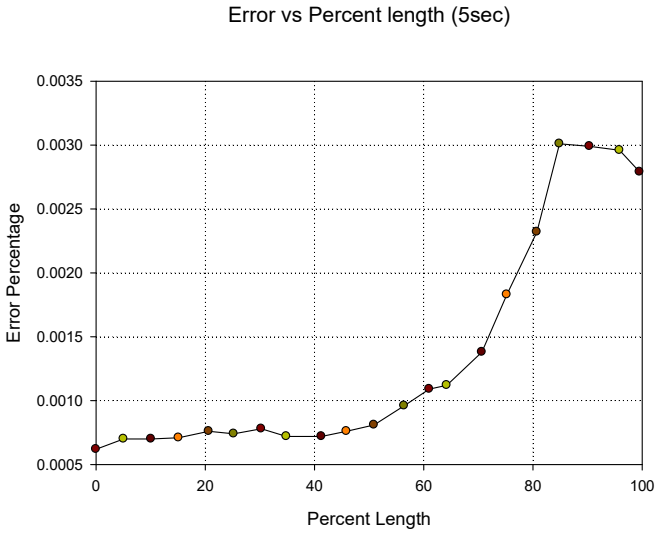

Figure 7 - Variable Node - Spherical (1)

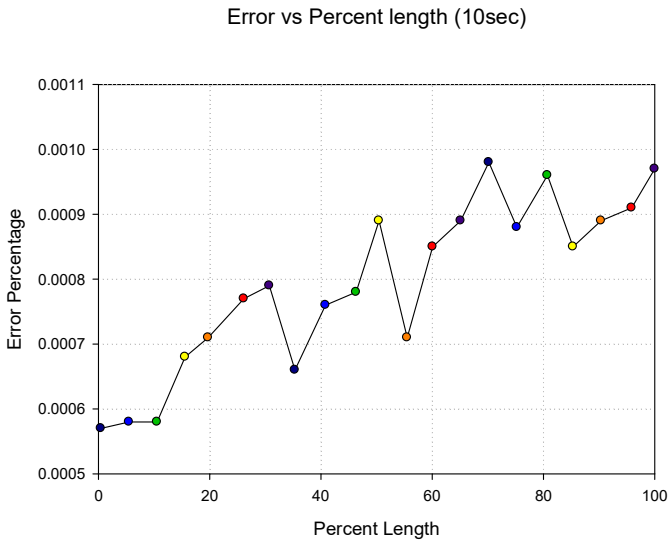

Figure 7 - Variable Node – Spherical (2)

From the above result we can see that error percent increases as we move away from the source. This may be due to the fact that the temperature towards the end might not have changed much in a short span of time, hence predicting the exact heat flux will be difficult. We can also see

that, the overall magnitude of error reduces as time progresses. We can observe that the oscillations in the graph increase as we go further away from the source in the 2nd graph.

2. Copper(Flat Face)

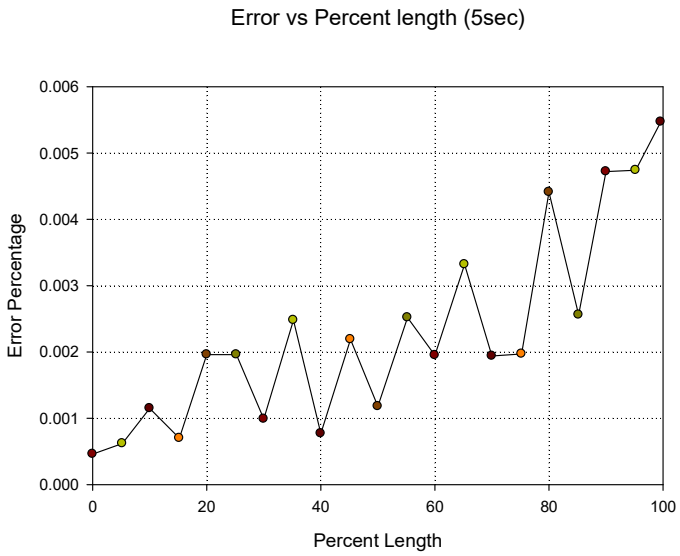

Figure 8 - Variable Node - Flat face (1)

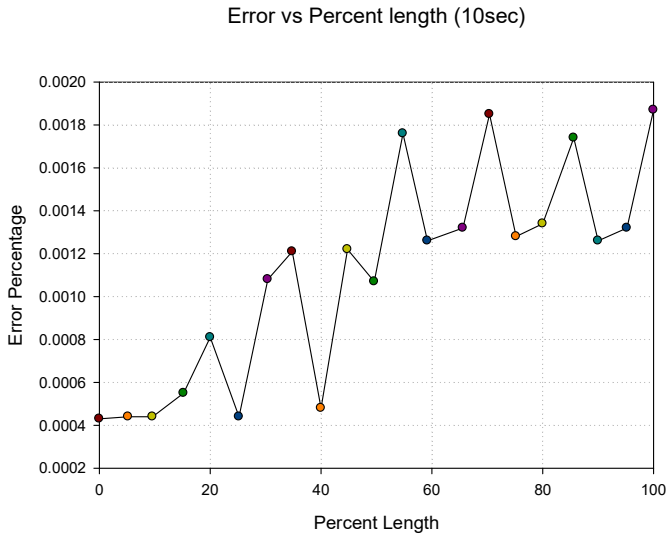

Figure 8 – Variable Node - Flat face (2)

A similar trend can be observed in this example as well, however there seems to be oscillations in error percent even in the 5sec graph. This could be due to increase in the length, as the effect of the heat flux wouldn't have reached the farther end.

3. Aluminum

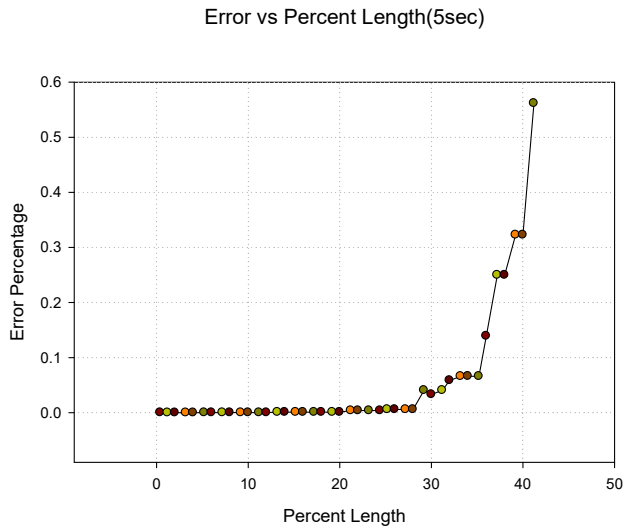

Figure 9 - Variable Node - Aluminum (1)

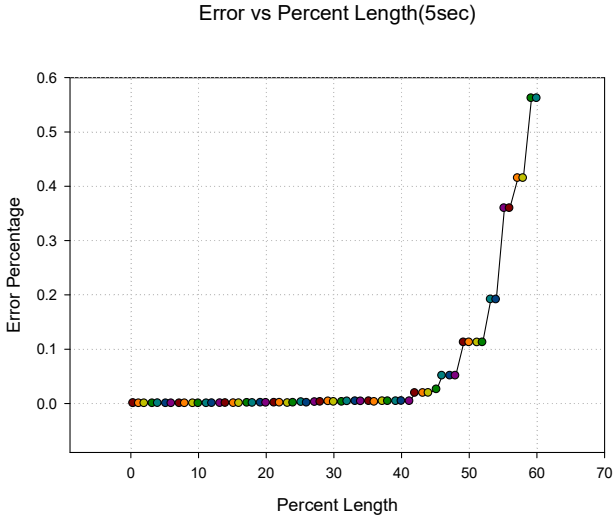

Figure 9 - Variable Node - Aluminum (2)

In the case aluminum a new observation can be made. When the FORTRAN code was run with these parameters calculating beyond 40% and 60%, for 5 and 10 sec respectively, was taking an extremely high number of iterations to converge. In the case of 5sec, calculating beyond 50% didn't yield any result as the effect of the source after such a short period of time didn't have any effect on the rod beyond 50%. We can also observe that the error increases after 28% and 42% respectively. Do note that the percentage at which error increases moves farther away as we progress in time.

For the upcoming calculations, we have considered the end nodes for the first 2 cases, whereas for the 3rd case we have a considered node at distance of 20% length from the source for better accuracy.

b. For varying heat flux input in step form

1. Copper(Spherical)

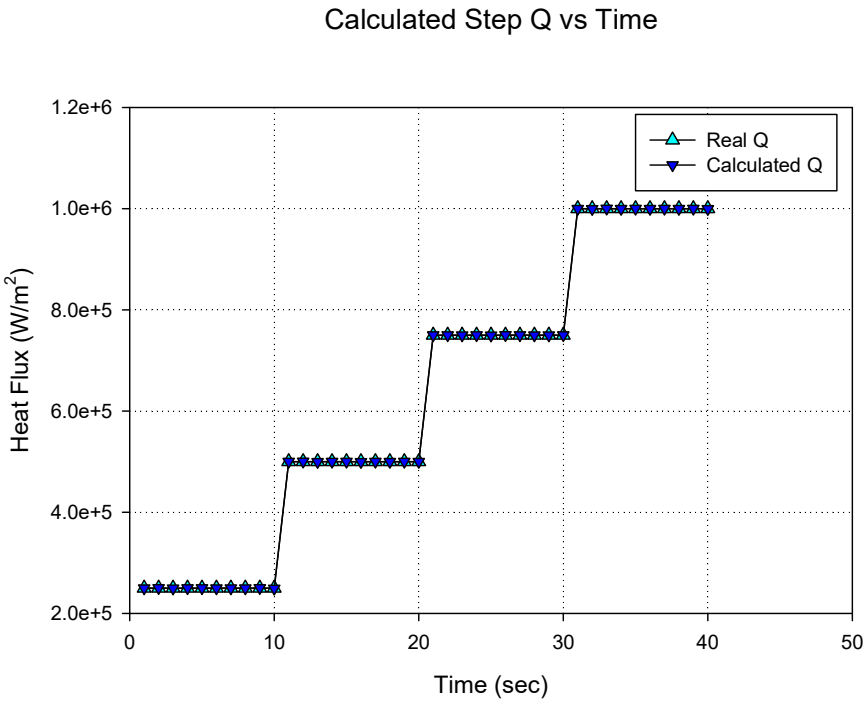

Figure 10 - Step Input - Spherical (1)

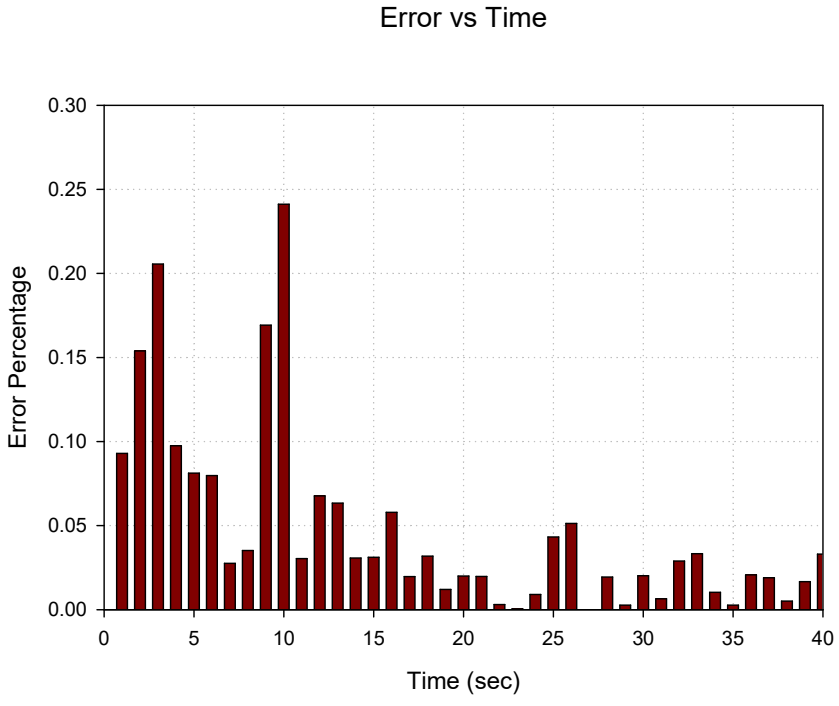

Figure 10 - Step Input - Spherical (2)

2. Copper(Flat Face)

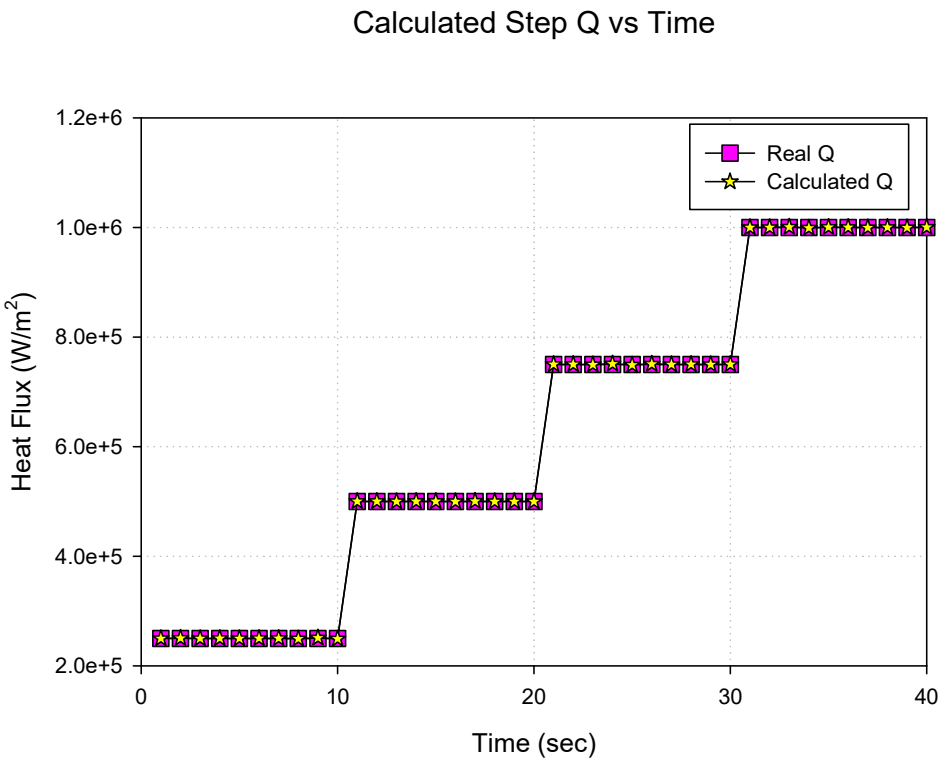

Figure 11 - Step Input - Flat face (1)

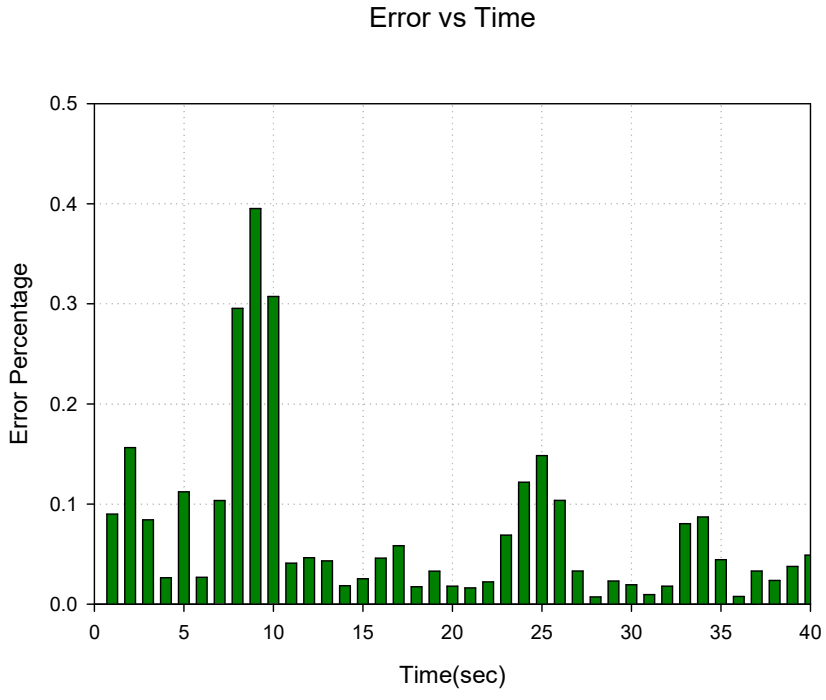

Figure 11 - Step Input - Flat face (2)

3. Aluminum

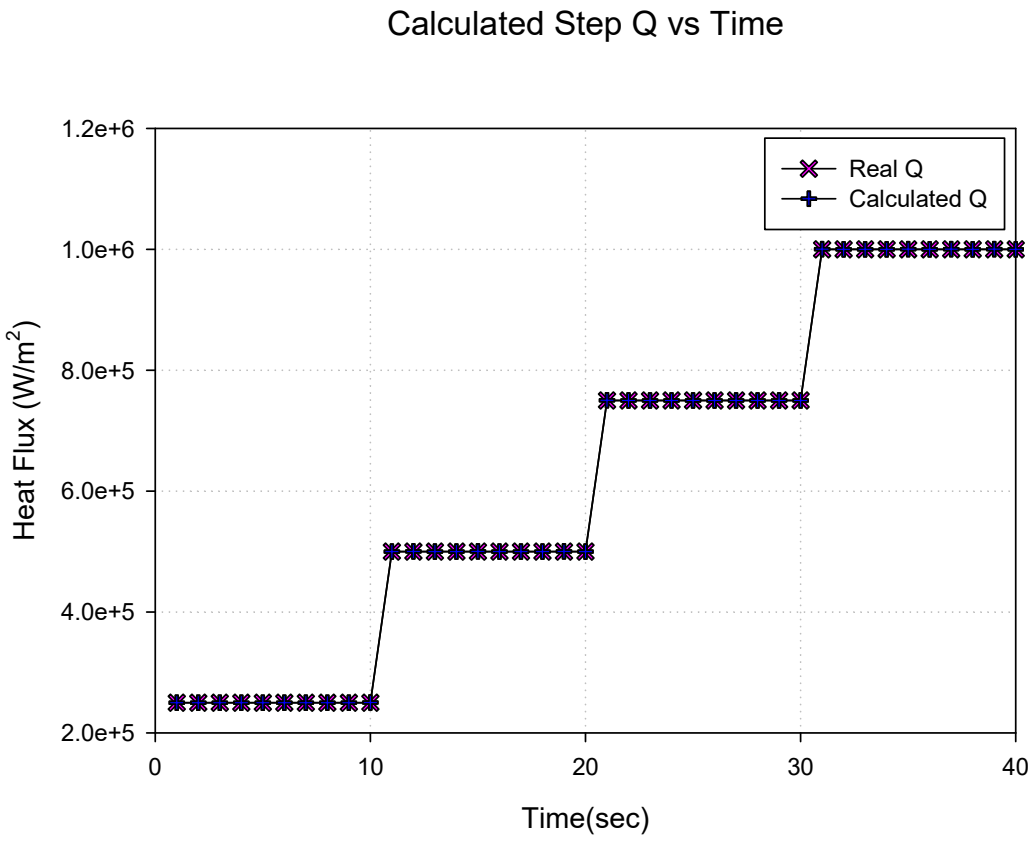

Figure 12 - Step Input - Aluminum (1)

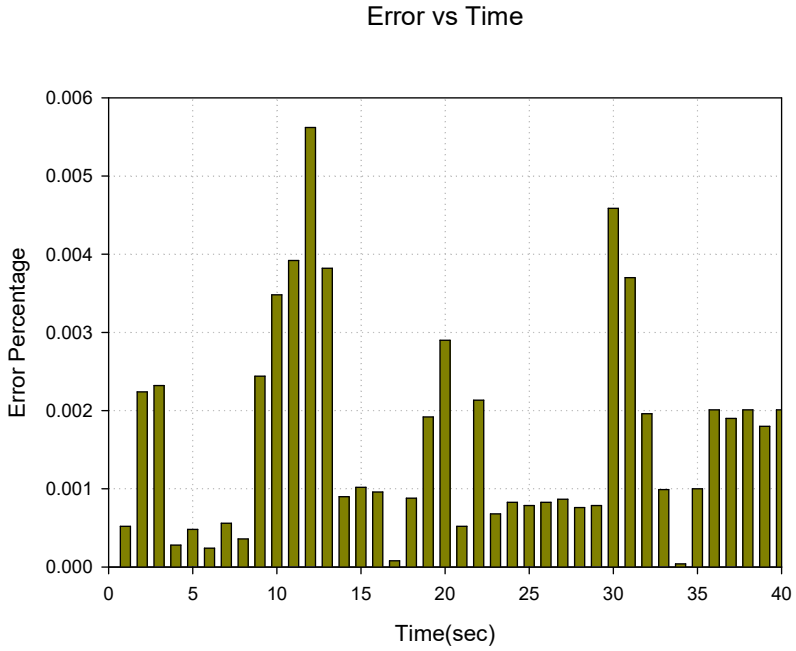

Figure 12 - Step Input - Aluminum (2)

As seen in the above graphs, the FORTRAN code is able to predict step based input for all the 3 cases. However, a small numerical error of the order 10^{-1}(or less) is present arising due to Newton-Raphson optimization. Error is higher for the first 2 cases as end node is considered whereas for the 3rd case node at 20% length is considered.

c. For heat flux input in linearly varying form

1. Copper(Spherical)

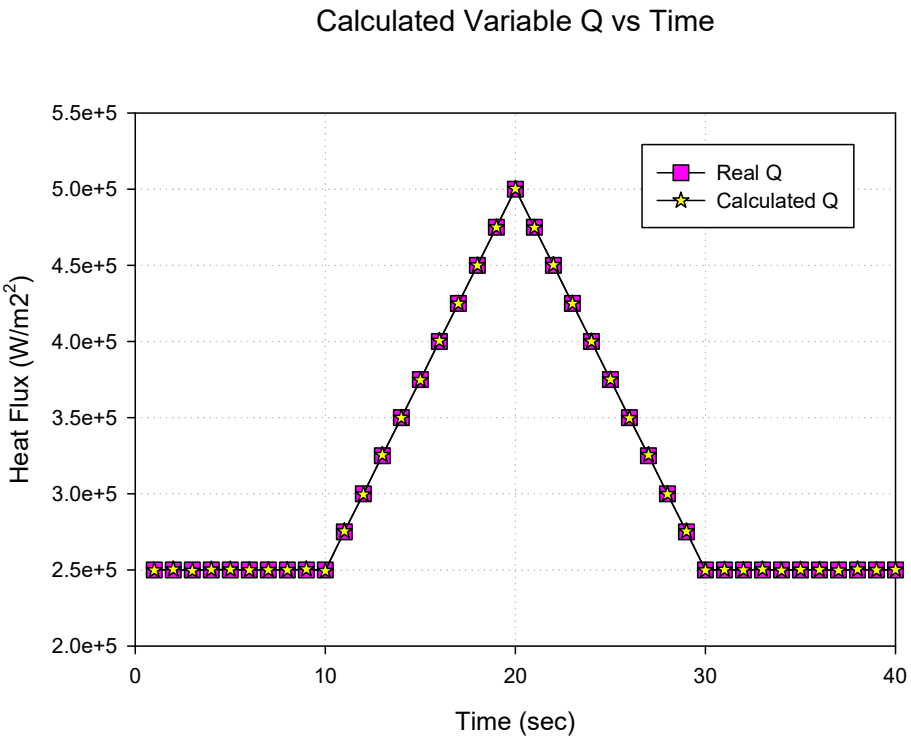

Figure 13 - Linearly Varying Input - Spherical (1)

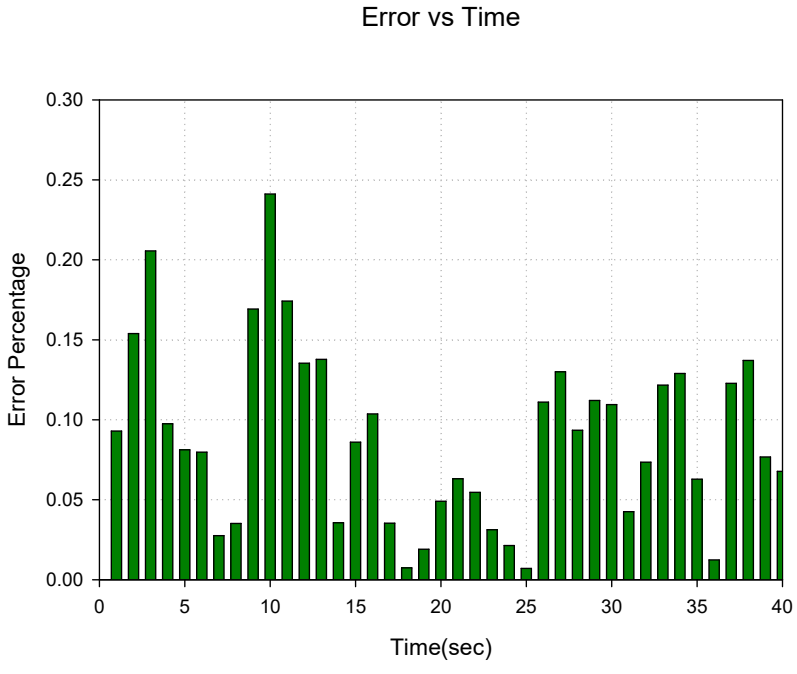

Figure 13 - Linearly Varying Input - Spherical (2)

2. Copper(Flat Face)

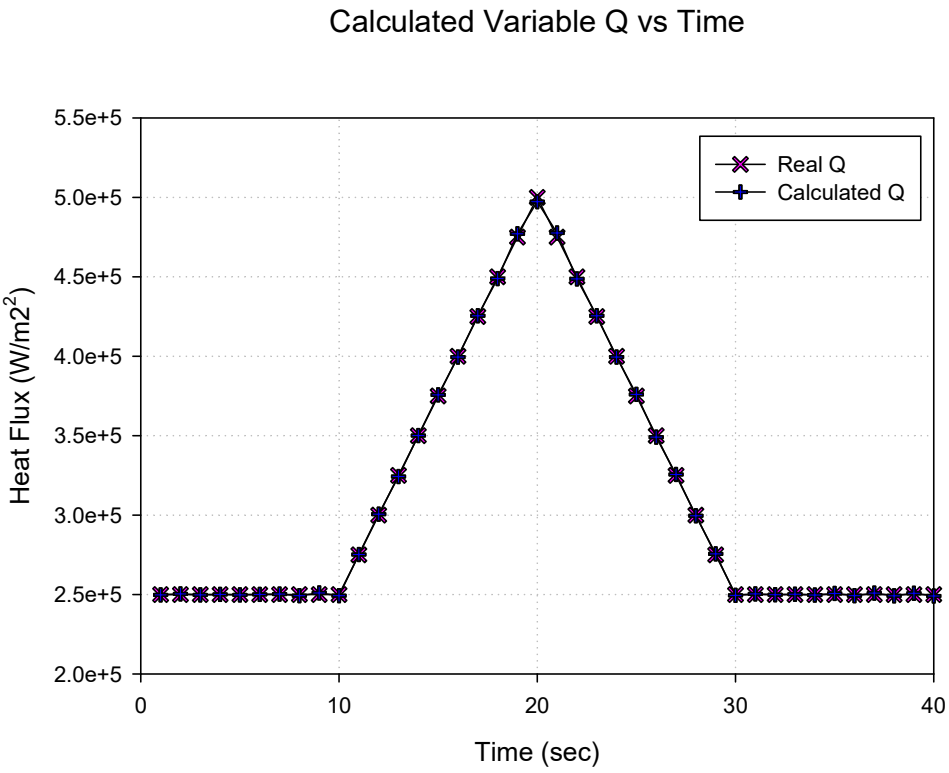

Figure 14 - Linearly Varying Input - Flat face (1)

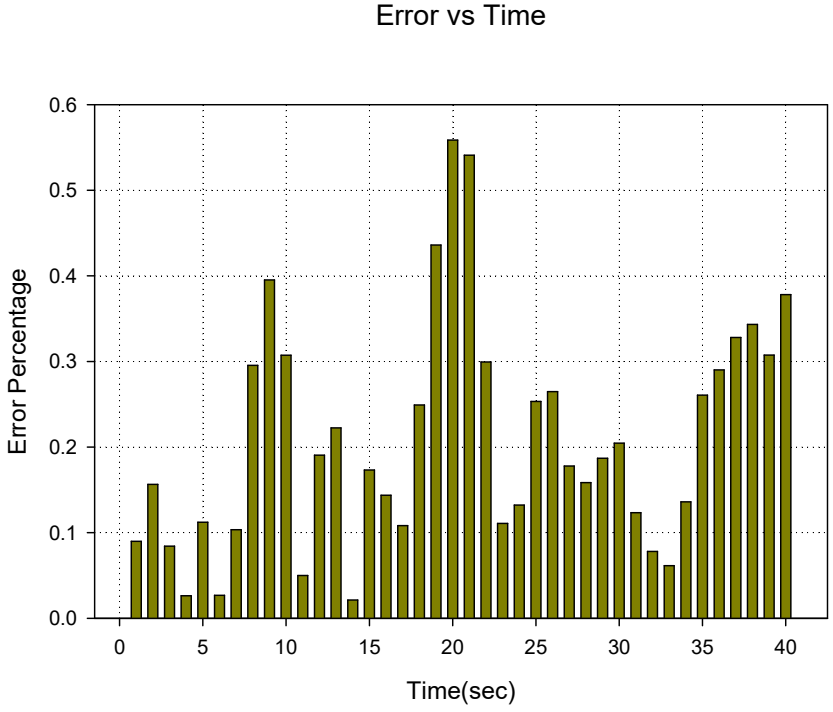

Figure 14 - Linearly Varying Input - Flat face (2)

3. Aluminum

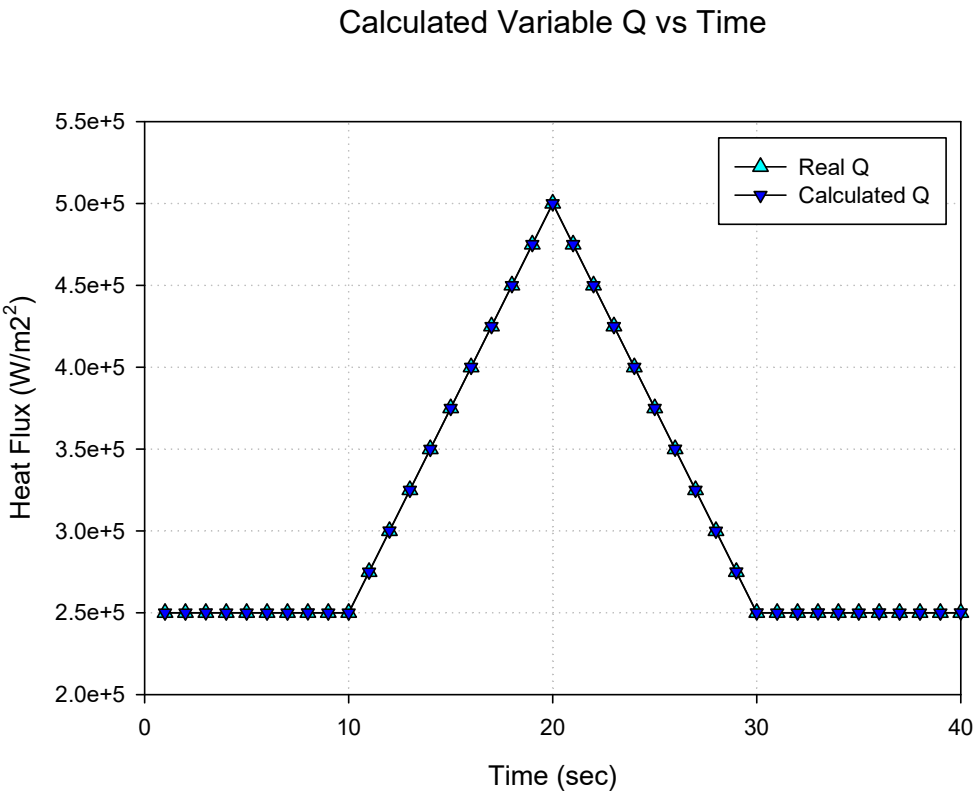

Figure 15 - Linearly Varying Input - Aluminum (1)

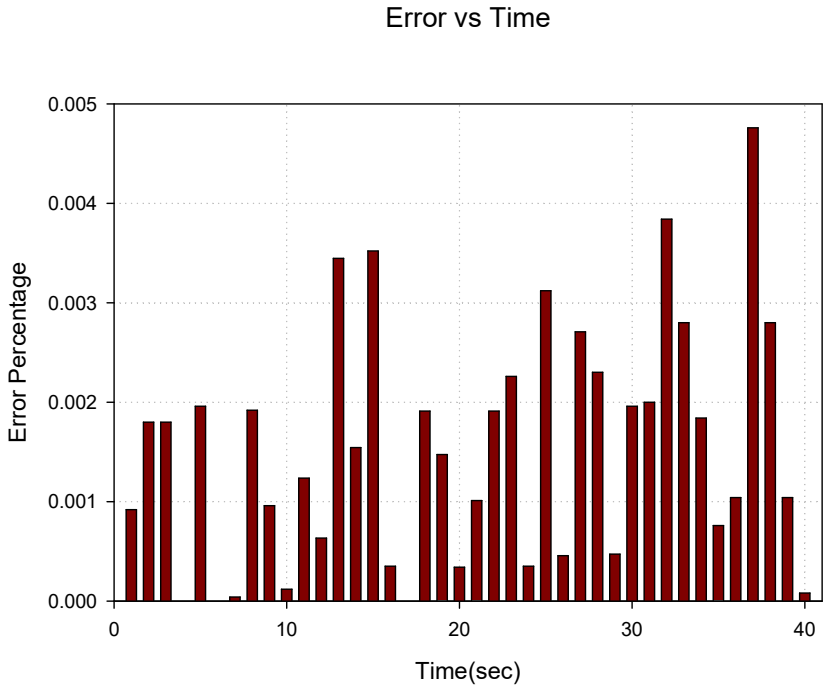

Figure 15 - Linearly Varying Input - Aluminum (2)

As seen in the above graphs, the FORTRAN code is able to predict linearly varying input for all the 3 cases. However, a small numerical error of the order 10^{-1}(or less) is present arising due to Newton-Raphson optimization. Error is higher for the first 2 cases as end node is considered whereas for the 3rd case node at 20% length is considered.

d. For heat flux input in sine curve form

1. Copper(Spherical)

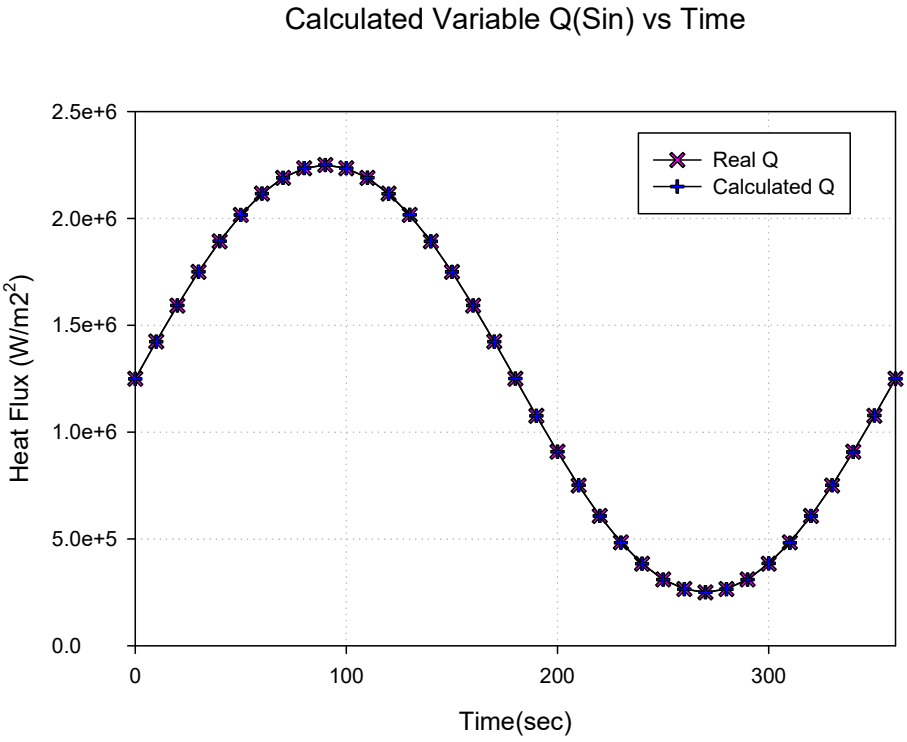

Figure 166 - Sine Input – Spherical (1)

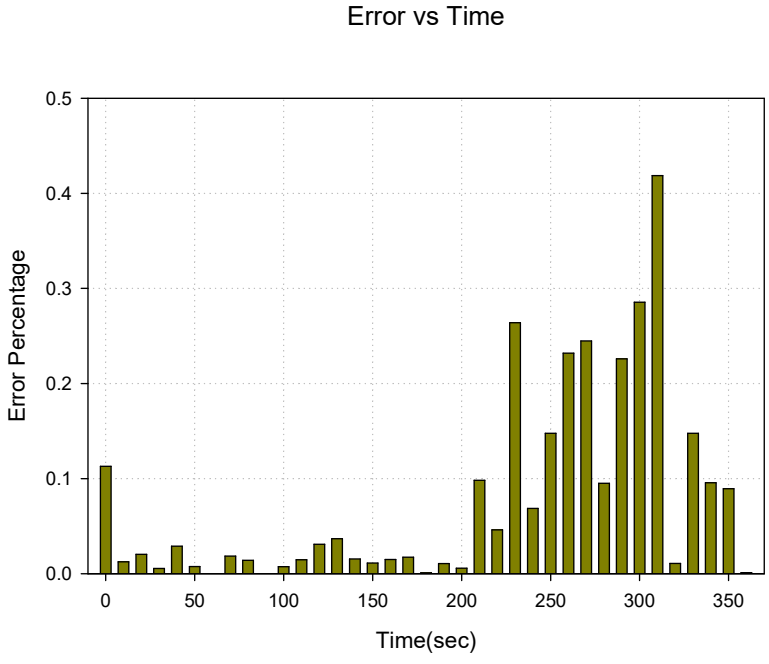

Figure 17 - Sine Input - Spherical (2)

2. Copper(Flat Face)

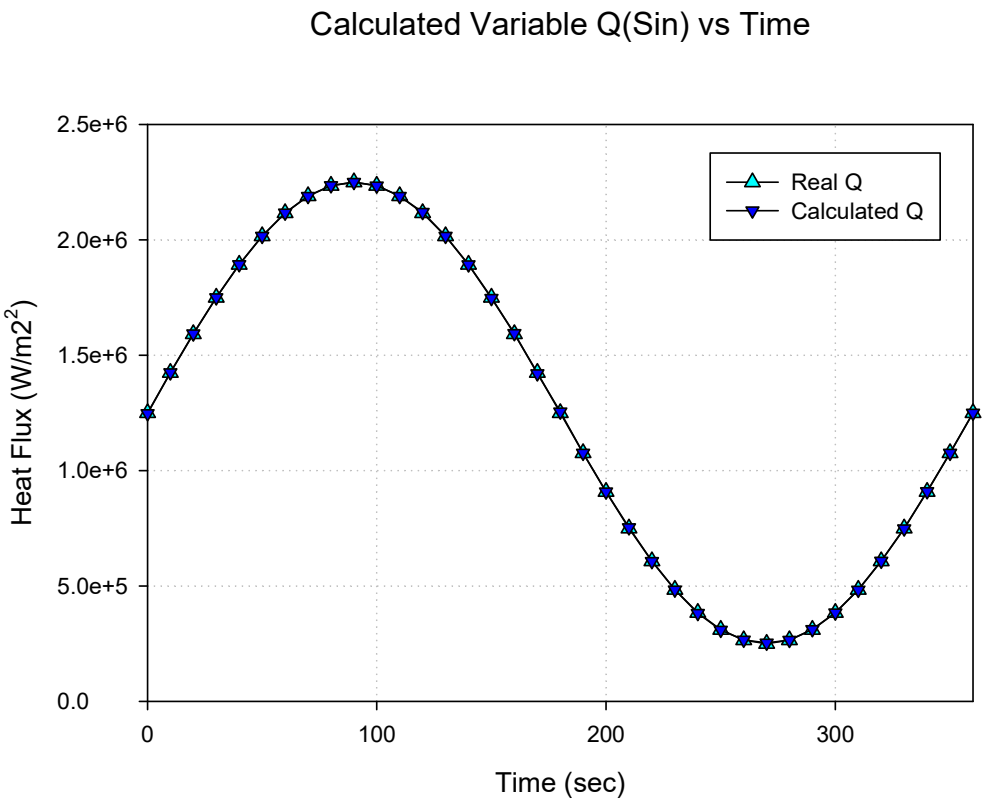

Figure 177- Sine Input - Flat face (1)

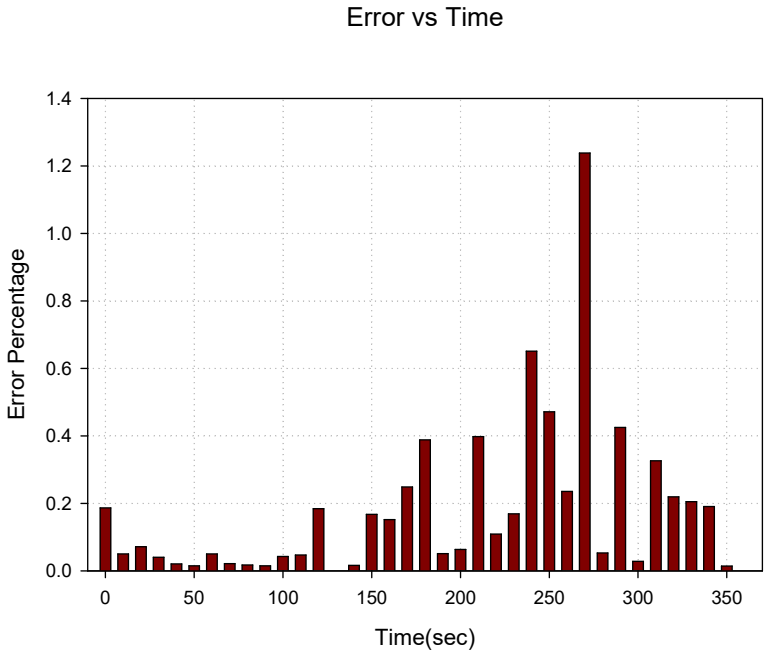

Figure 18- Sine Input - Flat face (2)

3. Aluminum

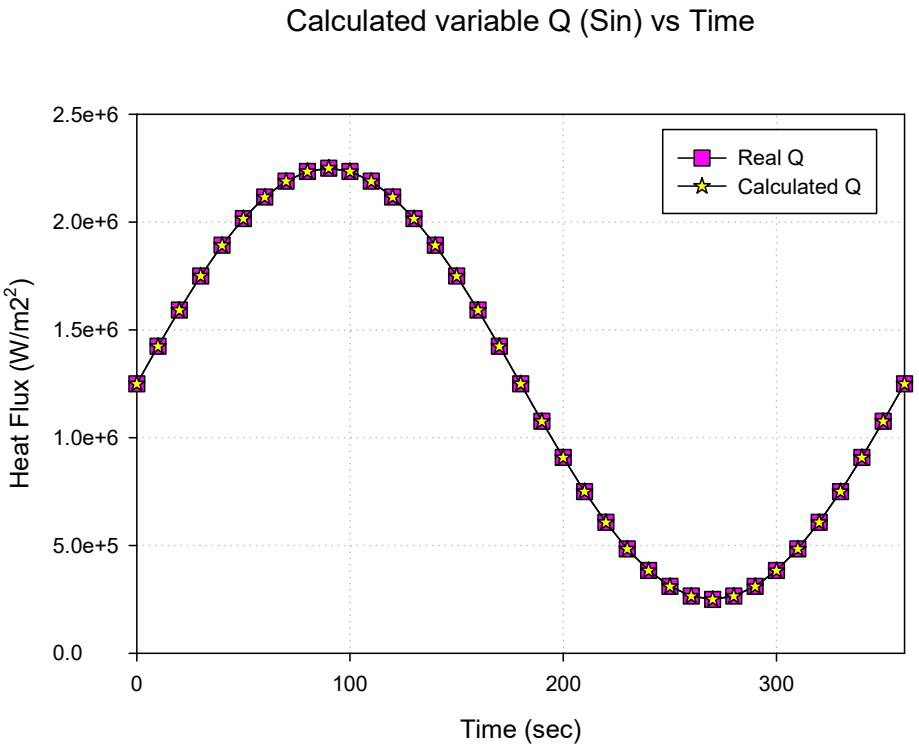

Figure 18 - Sine Input - Aluminum (1)

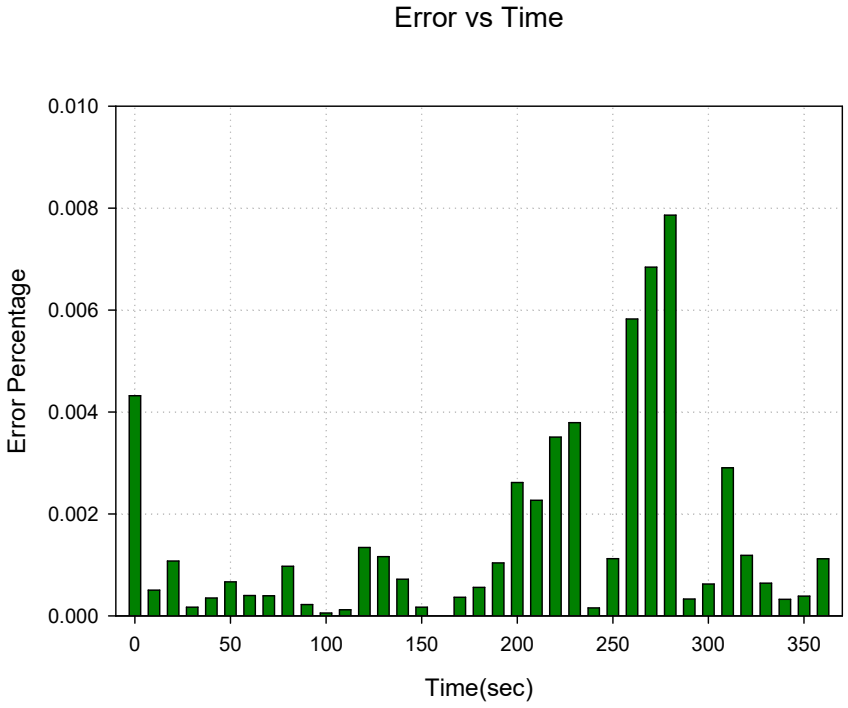

Figure 18 - Sine Input - Aluminum (2)

As seen in the above graphs, the FORTRAN code is able to predict input in the form of Sine curve for all the 3 cases. However, a small numerical error of the order 10^{-1}(or less) is present arising due to Newton-Raphson optimization. Error is higher for the first 2 cases as end node is considered whereas for the 3rd case node at 20% length is considered.

After keen observation, it can be said that the error curve seems to the mirror image of input. This is due to the fact that error is low for a high amount of heat flux as and vice versa.

8. Conclusion

From the results above, it can be observed that Newton-Raphson optimization is a great tool to use for inverse conduction. Although a few drawbacks have come up, the solution is stable and well within an acceptable range of error.

In case of constant heat flux, tradeoff has to be made between time and placement of measured node. If thermocouple is placed only at the end node, then a large amount of time has to pass by for better and more accurate results. If faster results are desired, then accurate result can be obtained by placing the thermocouple as close to the source as possible.

In case of varying heat flux, the temperature history at the measured node has to be obtained. If the time difference between 2 readings is less, then thermocouple has to be placed closer to the source, this will give flux readings at those time steps. The smaller the time difference, the more accurately we can predict the change in heat flux.

9. References

[1]J.Vogel, L.Sara and L.Krejc, "A simple inverse heat conduction method with optimization," International Journal for Heat Mass Transfer, Vol. 36, No. 17, pp.4215 – 4220, 1993.

[2]R.C.Mehta, "Estimation of Heat Transfer Coefficient in a Rocket Nozzle," AIAA, Vol. 19, pp. 1085-1086, August 1981.

[3]R.C.Mehta, "Numerical Solution of nonlinear inverse heat conduction problem with radiation boundary condition," International Journal for Numerical Methods in Engineering, Vol. 20, pp. 1057-1066, 1984.

[4]Steven C. Chapra, Raymond P. Canale, "Numerical Methods for Engineers," Sixth Edition.

[5]John C. Tannehill, Dale A. Anderson, Richard H. Pletcher, "Computational Fluid Mechanics and Heat Transfer," Second Edition.

[6]Suhas V. Patankar, "Numerical Heat Transfer and Fluid Flow".

[7]" http://www.fortrantutorial.com/documents/IntroductionToFTN95.pdf".