Mathematical Modeling of Heat Transfer phenomenon in Selective Laser Sintering Process

Ghanshyam Chandra

June 27, 2019

Contents

Abstract

The study has been carried out for Mathematical modeling of selective laser sintering process. Since each time it's not possible to solve Governing equations exactly hence we need to depend on Numerical/Mathematical modeling but initially we need to verify accuracy of Numerical results with Exact results for that we need to highly depend on computational resources for convergence of both the results. Mathematical Modeling is just approximation of Field variable it may or may not converge with your exact solution but with proper meshing and correct aprroximation method will lead to converge results with slight error. Flexibility of Mathematical Modeling is that for same geometry and same parameter if you have verified results with exact results then you may go for different materials and different states for same governing equations. In present study we ave first verified results of Numerical modeling with Exact solution and then we have proceed for approximation of Transient Solution. Since Transient solution is also a steady state solution for a particuler time step.

1 Introduction

Mathematical Modeling of Governing equation has been done for heat interaction between heat source and aluminium bed. The study takes look at how much heat is required to penetrate desired tempearture upto desired length or vice-versa. The study is further extended to model Transient solution and Transient solution with moving heat source. FEniCS package is used to solve variational form for different states and parameters and convergence test has been test as been carried out to match exact result.

2 Mathematical Formulation of Governing Equation

2.1 Variational form Governing Equation in Cartesian Co-ordinates

From Energy Balance for steady state condition

$$(Q_{in})_X + (Q_{in})_Y + (Q_{in})_Z = (Q_{out})_X + (Q_{out})_Y + (Q_{out})_Z$$

So, equation of heat flow is derived as follows: -

Let

$$T - T_0 = \theta, dT = d\theta$$

$$\frac{d^2\theta}{dx^2} + \frac{d^2\theta}{dy^2} + \frac{d^2\theta}{dz^2} = \frac{1}{\alpha}\Big(\frac{d\theta}{dt} + v\frac{d\theta}{dx}\Big) \tag{1}$$

$$\frac{1}{\alpha}\Big(\frac{d\theta}{dt} + v\frac{d\theta}{dx}\Big) = \nabla^2\theta \tag{2}$$

Let T = Trial Function w = Test Function and V = FunctionSpace.
Where θ,$w \in V$.

here

$$\nabla^2\theta = \nabla.\nabla\theta \tag{3}$$

Hence

$$\int_\Omega \nabla.(\nabla\theta w)dx = \int_\Omega (\nabla.\nabla\theta)wdx + \int_\Omega (\nabla w).(\nabla\theta)dx \tag{4}$$

or

$$\int_\Omega (\nabla.\nabla\theta)wdx = \int_\Omega \nabla.(\nabla\theta w)dx - \int_\Omega (\nabla w).(\nabla\theta)dx \tag{5}$$

Volume Integral to Surface Integral conversion

$$\int_\Omega \nabla.(\nabla\theta w)dx = \int_{d\Omega} \frac{d\theta}{d\eta}wds \tag{6}$$

Hence

$$\int_\Omega \frac{1}{\alpha}\Big(\frac{d\theta}{dt} + v\frac{d\theta}{dx}\Big)wdx = \int_{d\Omega} \frac{d\theta}{d\eta}wds - \int_\Omega (\nabla w).(\nabla\theta)dx \tag{7}$$

$$\int_\Omega \frac{1}{\alpha}\Big(\frac{d\theta}{dt} + v\frac{d\theta}{dx}\Big)wdx + \int_\Omega (\nabla w).(\nabla\theta)dx = \int_{d\Omega} \frac{d\theta}{d\eta}wds \tag{8}$$

Using Euler Simple forward Interpolation (Here T_i is initial interpolated temperature, $T_i \in V$)

$$\int_\Omega \frac{1}{\alpha}(\frac{(\theta-\theta)}{\delta t} + v\frac{d\theta}{dx})wdx + \int_\Omega (\nabla w).(\nabla\theta)dx = \int_{d\Omega} \frac{d\theta}{d\eta}wds \tag{9}$$

$$\int_\Omega \frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_\Omega \frac{1}{\alpha}\Big(v\frac{d\theta}{dx}\Big)wdx + \int_\Omega (\nabla w).(\nabla\theta)dx = \int_{d\Omega} \frac{d\theta}{d\eta}wds + \int_\Omega \frac{\theta_i}{\delta t}wdx \tag{10}$$

2.2 Abstract Formulation of Governing Equation

From equation (10)

$$a(T,w) = \int_\Omega \frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_\Omega \frac{1}{\alpha}\Big(v\frac{d\theta}{dx}\Big)wdx + \int_\Omega (\nabla w).(\nabla\theta)dx \tag{11}$$

$$L(w) = \int_{d\Omega} \frac{d\theta}{d\eta}wds + \int_\Omega \frac{\theta_i}{\delta t}wdx \tag{12}$$

Where $a(\theta, w)$ is Bilinear form and $L(w)$ is linear form.

2.3 Energy Function (Π)

$$\Pi = \frac{1}{2}a(T,w) - L(w) \tag{13}$$

from equation (11) and (12)

$$\Pi = \frac{1}{2}\Big(\int_\Omega \frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_\Omega \frac{1}{\alpha}(v\frac{d\theta}{dx})wdx + \int_\Omega (\nabla w).(\nabla\theta)dx\Big) - \Big(\int_{d\Omega} \frac{d\theta}{d\eta}wds + \int_\Omega \frac{\theta_i}{\delta t}wdx\Big) \tag{14}$$

2.4 Variational Formulation applied to specific case

2.4.1 Steady State with Non-Moving Source

From equation(1)

$$\frac{d^2\theta}{dx^2} + \frac{d^2\theta}{dy^2} + \frac{d^2\theta}{dz^2} = 0 \tag{15}$$

From equation(10)

$$\int_{\Omega}(\nabla w).(\nabla\theta)dx = \int_{d\Omega}\frac{d\theta}{d\eta}wds \tag{16}$$

From equation (11) (12) and (13)

$$a(\theta, w) = \int_{\Omega}(\nabla w).(\nabla\theta)dx \tag{17}$$

$$L(w) = \int_{d\Omega}\frac{d\theta}{d\eta}wds \tag{18}$$

$$\Pi = \frac{1}{2}\Big(\int_{\Omega}(\nabla w).(\nabla\theta)dx\Big) - \int_{d\Omega}\frac{d\theta}{d\eta}wds \tag{19}$$

2.4.2 Transient with Non-Moving Source

From equation(1)

$$\frac{d^2\theta}{dx^2} + \frac{d^2T\theta}{dy^2} + \frac{d^2\theta}{dz^2} = \frac{1}{\alpha}\Big(\frac{d\theta}{dt}\Big) \tag{20}$$

From equation(10)

$$\int_{\Omega}\frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_{\Omega}(\nabla w).(\nabla\theta)dx = \int_{d\Omega}\frac{d\theta}{d\eta}wds + \int_{\Omega}\frac{\theta_i}{\delta t}wdx \tag{21}$$

From equation (11) (12) and (13)

$$a(\theta, w) = \int_{\Omega}\frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_{\Omega}(\nabla w).(\nabla\theta)dx \tag{22}$$

$$L(w) = \int_{d\Omega}\frac{d\theta}{d\eta}wds + \int_{\Omega}\frac{\theta_i}{\delta t}wdx \tag{23}$$

$$\Pi = \frac{1}{2}\Big(\int_{\Omega}\frac{1}{\alpha}\frac{\theta}{\delta t}wdx + \int_{\Omega}(\nabla w).(\nabla\theta)dx\Big) - \Big(\int_{d\Omega}\frac{d\theta}{d\eta}wds + \int_{\Omega}\frac{\theta_i}{\delta t}wdx\Big) \tag{24}$$

2.4.3 Transient with Moving Source

Equation (10) (11) (12) and (14) represents Variational Formulation for the required case.

2.4.4 Steady State subjected to Semi-infinite Boundary Condition

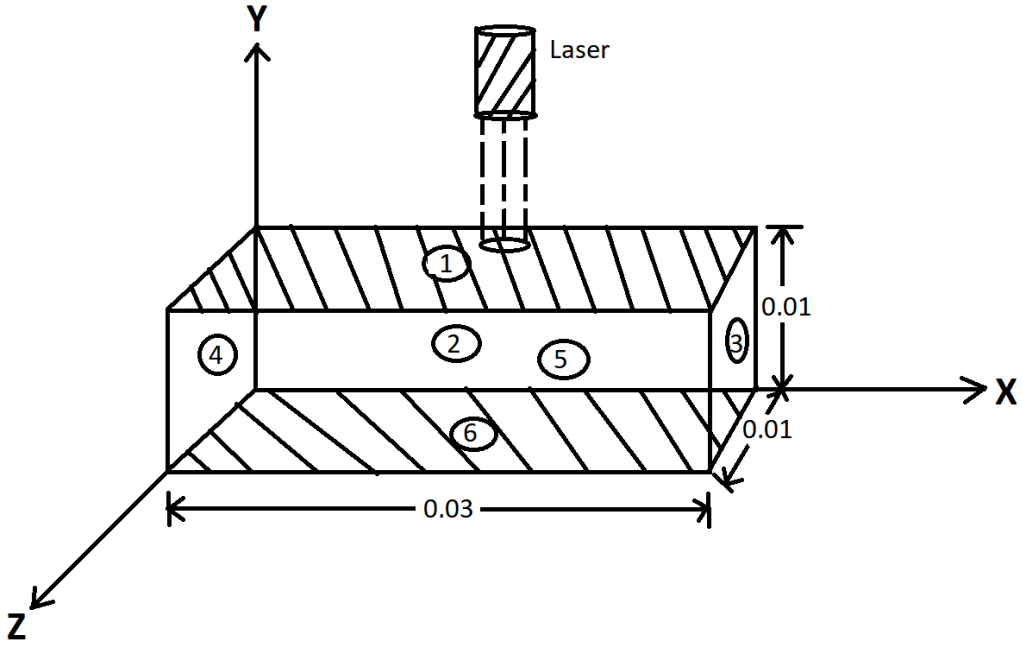


Figure 1: Physical Model of Aluminium Bed with Laser Source.

Boundary Conditions:-

1)Y = 0.00 plane(Insulated): -

$$\theta = 0$$

2)Z = 0.00 plane

$$\theta = 0$$

3)X = 0.03 plane: -

$$\theta = 0$$

4)X = 0.00 plane

$$\theta = 0$$

5)Z = 0.010 plane

$$\theta = 0$$

6)Y = 0.02 plane (Insulated): -

$$\frac{d\theta}{dy} = 0$$

7)Y = 0.02 plane(Constant Laser Heat Source): -

$$\frac{d\theta}{dy} = -\frac{Q}{KA}$$

Hence

$$\int_{d\Omega} \frac{d\theta}{d\eta} w ds(i) = \int_{d\Omega} \Big(\frac{-Q}{KA}\Big) w ds(7)$$

From equation (10)

$$\int_{\Omega} \frac{1}{\alpha} \Big(v \frac{d\theta}{dx}\Big) w dx + \int_{\Omega} (\nabla w).(\nabla \theta) dx = \int_{d\Omega} \Big(\frac{-Q}{KA}\Big) w ds(7) \tag{25}$$

2.4.5 Transient with Moving Source subjected to Natural Boundary Condition

Boundary Conditions:-

1)Y = 0.01 plane(Insulated): -

$$\frac{d\theta}{dy} = 0$$

2)Z = 0.00 plane

$$\frac{d\theta}{dz} = -\frac{h}{K}(\theta)$$

3)X = 0.03 plane: -

$$\frac{d\theta}{dx} = -\frac{h}{K}(\theta)$$

4)X = 0.00 plane

$$\frac{d\theta}{dx} = -\frac{h}{K}(\theta)$$

5)Z = 0 plane

$$\frac{d\theta}{dz} = -\frac{h}{K}(\theta)$$

6)Y = 0.02 plane (Insulated): -

$$\frac{d\theta}{dy} = 0$$

7)Y = 0.02 plane(Constant Laser Heat Source): -

$$\frac{d\theta}{dy} = -\frac{Q}{KA}$$

Hence

$$\int_{d\Omega} \frac{d\theta}{d\eta} w ds(i) = \int_{d\Omega} \Big(\frac{-Q}{KA}\Big) w ds(7)$$

From equation(10)

$$\int_{\Omega} \frac{1}{\alpha}\Big(v\frac{d\theta}{dx}\Big)wdx + \int_{\Omega}(\nabla w).(\nabla\theta)dx + \int_{d\Omega}\frac{-h}{K}(\theta)wds(2) + \int_{d\Omega}\frac{-h}{K}(\theta)wds(3) + \int_{d\Omega}\frac{-h}{K}(\theta)wds(4)+ \int_{d\Omega}\frac{-h}{K}(\theta)wds(5) + \int_{d\Omega}\frac{-h}{K}(\theta)wds(6) = \int_{d\Omega}\Big(\frac{-Q}{KA}\Big)wds(7) \tag{26}$$

Exact Solution:-

(1) For Square Heat Source(Cartesian Co-ordinate) :-

$$\int d\theta = \int_{-l/2}^{l/2}\int_{-l/2}^{l/2}\frac{q_0}{2\pi Ks}exp^{-\Big(\frac{v}{2K}(s-(x-x'))\Big)}dx'dy' \tag{27}$$

Where

$$s = \sqrt{\Big((x-x')^2 + (y-y')^2 + z^2\Big)}$$

(2) For Circular Heat Source(Polar Co-ordinate) :-

$$\int d\theta = \int_{0}^{2\pi}\int_{0}^{r_0}\frac{q_0 r}{2\pi Ks}exp^{-\Big(\frac{v}{2K}(s-(x-r\cos(\theta)))\Big)}drd\theta \tag{28}$$

Where

$$s = \sqrt{\Big((x-r\cos(\theta))^2 + (y-r\sin(\theta))^2 + z^2\Big)}$$

2.5 Normalisation of Governing Equations

$$\xi = \frac{x}{R_0}, \eta = \frac{y}{R_0}, \zeta = \frac{z}{R_0}, \xi' = \frac{x'}{r_0}, \eta' = \frac{y'}{r_0}, \zeta' = \frac{z'}{r_0}, \beta = \frac{R_0}{r_0}, Pe = \frac{vr_0}{2\alpha}$$

2.5.1 Exact Solution

(1) For Square Heat Source(Cartesian Co-ordinate) :-

$$\int d\theta = \int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\frac{q_0 r_0}{2\pi\beta Ks}exp^{-\Big(Pe\beta(\frac{s}{R_0}-(\xi-\frac{\xi'}{\beta}))\Big)}d\xi' d\eta' \tag{29}$$

Where

$$s = \sqrt{\Big((\xi-\frac{\xi'}{\beta})^2 + (\eta-\frac{\eta'}{\beta})^2 + \zeta^2\Big)}$$

(2) For Circular Heat Source(Polar Co-ordinate) :-

$$\int d\theta = \int_{0}^{2\pi}\int_{0}^{1}\frac{q_0 r r_0}{2\pi\beta Ks}exp^{-\Big(Pe\beta(\frac{s}{R_0}-(\xi-\frac{r\cos(\theta)}{\beta}))\Big)}drd\theta \tag{30}$$

Where

$$s = \sqrt{\Big((\xi - \frac{rcos(\theta)}{\beta})^2 + (\eta - \frac{r\sin(\theta)}{\beta})^2 + \zeta^2\Big)}$$

2.5.2 Governing Differential Equation (Steady State)

From equation(1)

$$\frac{d^2\theta}{d\xi^2} + \frac{d^2\theta}{d\eta^2} + \frac{d^2\theta}{d\zeta^2} = 2Pe\beta\Big(\frac{d\theta}{d\xi}\Big) \tag{31}$$

2.5.3 Variational Formulation

Equation(31) can be written as:-

$$\nabla.\nabla\theta = 2Pe\beta(\frac{d\theta}{d\xi})$$

Variational Form: -

$$\int_\Omega 2Pe\beta\Big(\frac{d\theta}{d\xi(0)}\Big)d\xi() + \int_\Omega \nabla(w).\nabla(\theta)d\xi() = \int_{d\Omega}\frac{d\theta}{d\eta}ds(i) \tag{32}$$

Subjected to Semi-infinite Boundary condition: -

$$\int_\Omega 2Pe\beta\Big(\frac{d\theta}{d\xi(0)}\Big)d\xi() + \int_\Omega \nabla(w).\nabla(\theta)d\xi() = \int_{d\Omega}\Big(R_0\frac{-Q}{KA}\Big)wds(7) \tag{33}$$

Subjected to Natural Boundary condition: -

$$\int_\Omega 2Pe\beta\Big(\frac{d\theta}{d\xi(0)}\Big)d\xi() + \int_\Omega \nabla(w).\nabla(\theta)d\xi() + \int_{d\Omega}\frac{-h}{K}(R_0\theta)wds(2) + \int_{d\Omega}\frac{-h}{K}(R_0\theta)wds(3) + \int_{d\Omega}\frac{-h}{K}(R_0\theta)wds(4) + \int_{d\Omega}\frac{-h}{K}(R_0\theta)wds(5) + \int_{d\Omega}\frac{-h}{K}(R_0\theta)wds(6) = \int_{d\Omega}\Big(\frac{-QR_0}{KA}\Big)wds(7) \tag{34}$$

Normalised length & radius of source: - Length = $\frac{1}{\beta}$, Radius = $\frac{1}{2\beta}$

3 Numerical Modeling

3.1 Generating Geometry

Generating file: -

```
gedit file_name.geo
```

Generating Points: -

```
Point(i) = {x,y,z,s};
```

Where i = point remark. s = element size.

Generating Line: -

```
Line(j) = {i,(i+1)};
```

Generating Suraface: -

```
Line Loop(j) = {a,b,c,d};
```

Where a,b,c,d are line remarks.
Generating Surface: -

```
Plane Surface(i) = {i};
Surface Loop(i) = {a,b,c,d,e,f};
```

Geneating volume from Surface: -

```
Volume(1) = {1};
```

3.2 Meshing

3.2.1 Gmsh

```
gmsh -d file_name.geo
%Parallel processing using "core" logical processors.
mpirun -np core gmsh -d file_name.geo
```

3.2.2 Netgen

Prepare $3D$ CAD Geometry using freeCAD.
Import *.step* files in Netgen using GUI and mesh.
Export mesh in *.msh* format.

3.3 Converting *.msh* to *.xml* format

3.3.1 mesio-convert

```
meshio-convert fine_name.msh file_name.xml
```

3.3.2 dolfin-convert

```
dolfin-convert fine_name.msh file_name.xml
```

No Mixed mesh conversion is supported(only Triangular and Quadrilateral are supported).

3.4 FEniCS Implementation

3.4.1 Getting Started

Importing FEniCS Packages in Python3

```
from fenics import *
```

Importing numpy and matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
```

3.4.2 Defining Geometry and Mesh import

```
mesh = BoxMesh(Point(X0,Y0,Z0),Point(L,H,W),Nx,Ny,Nz)
```

Here $X0, Y0, Z0$ are the co-ordinates of origin, L, H, W are length width and height corresponding to X,Y,Z axis and Nx,Ny and Nz are No. of Elements in X,Y,Z axis respectively.

```
mesh = Mesh('file_name.xml') # Mesh Import from .xml file.
```

3.4.3 Defining Function Space

```
V = FunctionSpace(mesh,'Lagrange',1)
```

Here we have choosen Lagrange Trial Polynomial with order = 1 in Function Space.We can also use "*CG*" (Continuous Galerkian) or "*DG*" (Discontinuous Galerkian) for Scaler Function Space. Ex :-

```
V = FunctionSpace(mesh,'CG',1)
V = FunctionSpace(mesh,'DG',1)
V = FunctionSpace(mesh,'CG',4)
```

Here in above example "4" represents order of Trial Function.
The Function Space can also be Vector Function Space or Tensor Function Space (Since here Temperature is scaler we have only focused on Scaler Function Space with Lagrange approach). Ex :-

```
V = FunctionSpace(mesh,'CG',4)
V = VectorFunctionSpace(mesh,'DG',5)
V = TensorFunctionSpace(mesh,'CG',6)
```

3.4.4 Defining Boundary Parts

```
dim = mesh.topology().dim()-1
boundary_parts = MeshFunction("size_t",mesh,dim)
boundary_parts.set_all(0) #Mark whole cuboid as domain "0"
```

Here "*dim*" represents topological dimension- (1) of mesh Function, "boundary_parts" represents domain of geometry and will be assigned later on boundaries.

3.4.5 Assigning Boundary Condition

There are Three Types of Boundary Condition
1)Dirichlet Boundary Condition :-

$$T_\Omega = 300.00$$

2)Neumann Boundary Condition :-

$$\left(\frac{dT}{dx}\right)_{d\Omega_N} = \left(\frac{-Q}{KA}\right)$$

3)Robbins Boundary Condition :-

$$\left(\frac{dT}{dx}\right)_{d\Omega_R} = \frac{-h}{K}(T - T_0)$$

Substitution of Boundary Condition:-

```
bc = DirichletBC(FunctionSpace,Constant/Function,boundary_domain)
bc = DirichletBC(FunctionSpace,Constant/Function,boundary_domain)
```

Defining Boundary Domain :-

```
def Left(x):
        return x[0] == 0.01 and x[1] == 0.02 and x[2] == 0.03
        #Define Point Source

def Left(x,on_boundary):
        return on_boundary #true for all Boundary and return bool (1 or 0)
```

For Neumann and Robbins Boundary Condition we need we to Include the Boundaries in Variational Formulation and Define Surface Integral Domain ($ds(i)$) for each Domain "i". Ex :-

```
class Left(SubDomain):
```

```
        def inside(self,x,on_boundary):
                return x[0] == 0.01
                #Here in 3D Space this will represent a plane

cx, cz, radius = 0.015, 0.005, 0.0007
class Seven(SubDomain):
    def inside(self, x,on_boundary):
        return pow(x[0] - cx, 2) + pow(x[2] - cz, 2) <= pow(radius, 2)
and x[1] == 0.01
        #Representing Circle on a plane in 3D Space

class Eight(SubDomain):
    def inside(self, x,on_boundary):
        return between(x[0],(0.02,0.03)) and near(x[1],0.02,1e-4) \
         and between(x[2],(0.01,0.02))
        #Representing a Rectangular Region in plane in 3D Space
```

3.4.6 Defining SubDomain Region and Calculating ds(i) and dx(i)

Marking Domain

```
seven = Seven()
seven.mark(boundary_parts, 7) #Mark Domain "seven" as "7" in boundary_parts
```

Measure ds(i) and dx(i)

```
dx = Measure('dx', domain = mesh, subdomain_data = boundary_parts)
ds = Measure('ds', domain = mesh, subdomain_data = boundary_parts)
```

3.4.7 Initializing Functions

```
T = TrialFunction(V) #Trial Function
w = TestFunction(V) #Test Function
```

Here "T" is a Trial Function (Lagrange) (i.e. Polynomial of defined order(3) in Function Space) and an be expressed as :-

$$T = C_1 + C_2\zeta + C_3\eta + C_4\zeta\eta + C_5\zeta^2 + C_6\eta^2 + C_7\zeta\eta^2 + C_8\zeta^2\eta + C_9\zeta^2\eta^2 + C_{10}\zeta^3 + C_{11}\eta^3 + C_{12}\zeta^3\eta + C_{13}\zeta\eta^3 + C_{14}\zeta^3\eta^2 + C_{15}\zeta^2\eta^3 + C_{16}\zeta^3\eta^3$$

(Consider (ζ, η) as Normalised co-ordinates of x[0] and x[1])
"w" is Test Function and can be expressed as :-

$$w = \frac{dT}{dc_i}$$

3.4.8 Interpolation of Initial Field Variable(Temperature)

```
T_1 = interpolate(T0,V)
T = Function(V)
```

Here we have Interpolated or just Substituted Initial Value of Temperature in Trial Function and hence T_1 is also a Function with assigned initial value of Temperature.

3.4.9 Defining Variational Formulation

There are two ways to Define Variational Formulation.
1) Use Equation (14) and (15) i.e. PreFormulated Linear and Bilinear Equations.
Ex:-

```
a = inner(nabla_grad(T) , nabla_grad(w))*dx() + vel*(1/alpha)*(Dx(T,0)*w)*dx()
#Function_1
L = (-Q/(K*A))*w*ds(7, subdomain_data = boundary_parts) #Function_2
```

2) Let FEniCS Decide and separate Linear and Bilinear Terms.

```
F = inner(nabla_grad(T) , nabla_grad(w))*dx() + vel*(1/alpha)*(Dx(T,0)*w)*dx()
-(-Q/(K*A))*w*ds(7, subdomain_data = boundary_parts)
a = lhs(F)
L = rhs(F)
```

Here a(w,T) is Biliner function and L(w) is a Linear Function.

3.4.10 Calculating Stiffness Matrix and VectorMatrix

3.4.11 Solver & Solver configuration

```
Problem = LinearVariationalProblem(a,L,T,bc)
Solver = LinearVariationalSolver(Problem)
Solver.solve()
```

or we can define Solver type and PreConditioners

```
T = Function(V)
Solver.parameters.linear_solver = 'gmres'
```

```
Solver.parameters.preconditioners = 'hypre_amg'
prm = Solver.parameters.preconditioners = solver.parameters.krylov_solver
#Short Form
prm.absolute_tolerence = 1E-07
prm.relative_tolerence = 1E-04
prm.maximum_iterations = 1000
```

3.4.12 Explicit Assembly and Solve

$$a(T, w) = L(w)$$

$$T' = \sum_{j=1}^{N} T_j \phi_j \tag{a}$$

T' Discrete solution can be computed by inserting equation(a) in a(T',w) for N Test Function $(\phi_1, \phi_2.....\phi_N)$. This implies that

$$\sum_{j=1}^{N} a(\phi_j \phi_i^{\cap}) T_j = L(\phi_i^{\cap}), i = 1, 2, 3, 4.....N. \tag{b}$$

Which is nothing but linear system.

$$AT' = b$$

Where entries of A and b are given by:-

$$A_{i,j} = a(\phi_j, \phi_i^{\cap}) b_i = L(\phi_i^{\cap})$$

on *FEniCS*

```
A = assemble(a)
b = assemble(L)
T'=Function(V)
T=T'.vector()
bc.apply(A,b) #Apply DirichletBC on A and b since we haven't applied initially
Solver.solve(A,T,b)
```

Specifying the solvers

```
solve(A,T,b,'gmres','hypre_amg')
```

or

```
Solver = KrylovSolver('gmres','ilu')
Prm = Solver.parameters
```

```
Prm.absolute_tolerence = 1E-07
Prm.relative_tolerance = 1E-04
Prm.maximum_iteration = 1000
T' = Function(V)
T = T'.vector()
Solver.solve(A,T,b)
```

3.4.13 Examining Matrix and Vector values

```
A = assemble(a)
b = assemble(L)
if mesh.num_cells() < 16 # print for small meshes only
Print(A.array())
Print(b.array())
bc.apply(A,b)
if mesh.num_cells() < 16
Print(A.array())
Print(b.array())
#Saving Matrix to file for Octave or MATLAB processing
import scipy.io as sc
sc.savemat('File_Name.mat':A.array(),'b':b.array())
```

3.4.14 Solvers & Preconditioner

Solvers

(1) **Linear Solver**

```
solve(a==L,T,bcs,solver_parameters = \
{'linear_solver':'mumps','preconditioner':'ilu'})
#or
solve(a==L,T,bcs,solver_parameters = dict(linear_solver = \
'gmres',preconditioner = 'amg'))
```

Look into Dolfin Docs for more solvers and preconditioners.

(1) **Non-linear Solver**

Discrete Non-linear problem can be written as:-

$$F(T, w) = 0, \forall T, w \in V$$

```
F = inner(nabla_grad(T) , nabla_grad(w))*dx() + vel*(1/alpha)*(Dx(T,0)*w)*dx()
-(-Q/(K*A))*w*ds(7, subdomain_data = boundary_parts)
T = TrialFunction(V)
T'=Function(V)
F=action(F,T)
J=derivative(F,T',T)
problem = NonlinearVariationalProblem(F, T',bcs,J)
solver=NonlinearVariationalSolver(problem)
solver.solve()
```

3.4.15 File Saving

Here *.pvd* is a file that contains information about other solution files that are in *.vtu* and *.pvtu* for serial and parallel processing respectively.

```
file = File('file_name.pvd')
file << T'
```

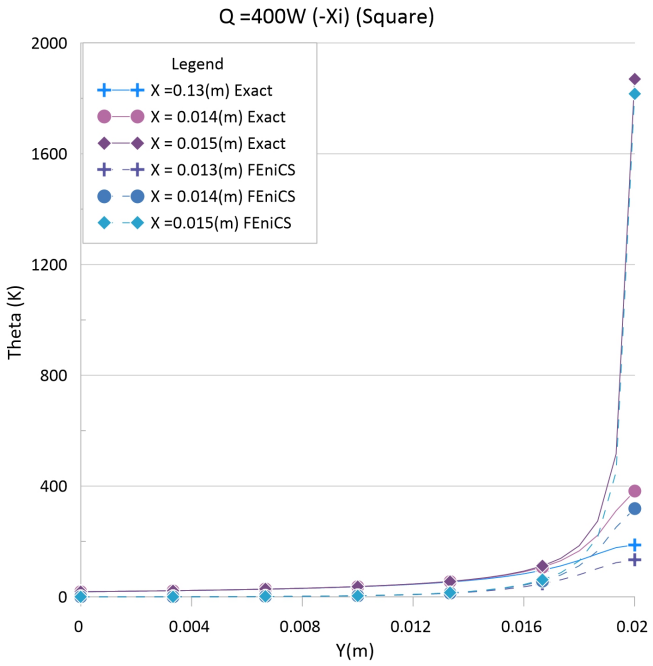
4 Results

For h = 10.45 $W/m^2 - K$, K = 167 $W/m - K$ and $\alpha = 6.9E - 05m^2/s$, T_0 = 300K

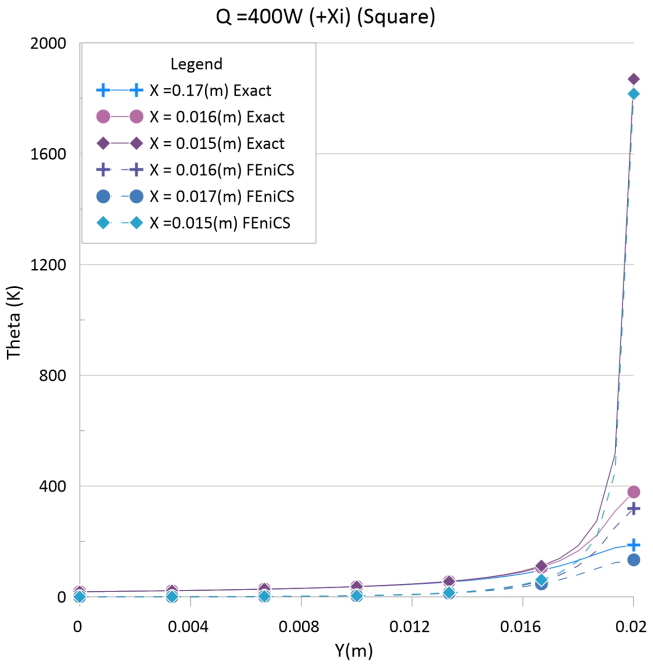
4.1 Verification with Exact Result and Contour Plots

Figure (2) - Figure (7) shows comparison of Numerical results with exact results for different parameters and as seen from the figure, the results are in a good agreement as well as the profiles. It is seen from the results that as we are increasing the velocity of powder bed the temperature in direction of motion increases, with increasing peclet no. the heat transfer by advection increases since the heat transfer by diffusion is fixed and also the source heat is same hence the peak temperature decreases.

Figure (8) - Figure (11) depicts the iso-contours of Temperature distribution in X-Y plane or plane perpendicular to the direction of incident heat. From figure it is clearly observed that with increase in velocity of bed the temperature contour shifts in direction of motion of bed and also with increase in the peclet no. velocity increases which further increase the advection rate hence the local peak temperature shift towards the direction of moving source.

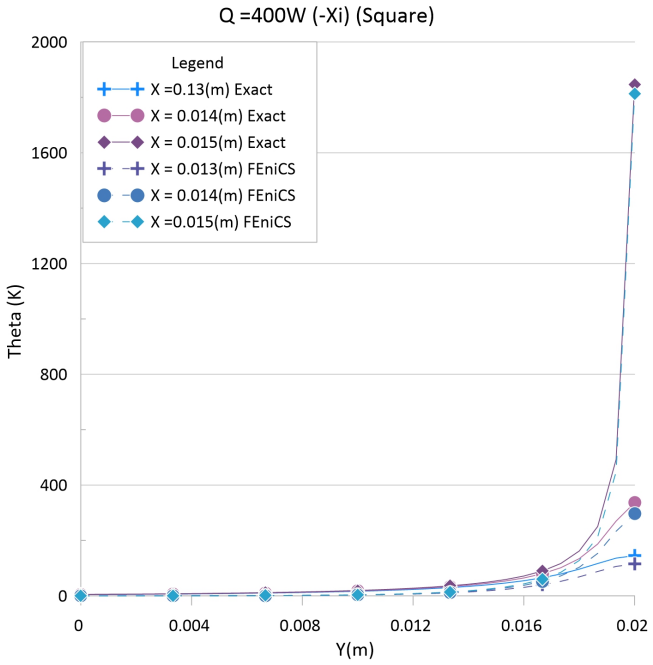


(a) In direction opposite to moving bed

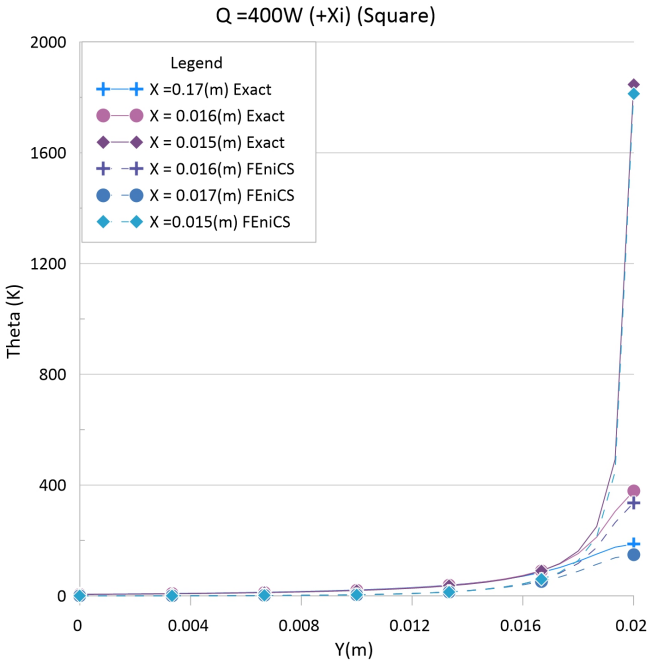


(b) In direction of moving bed

Figure 2: v = 0mm/s Square Heat Source

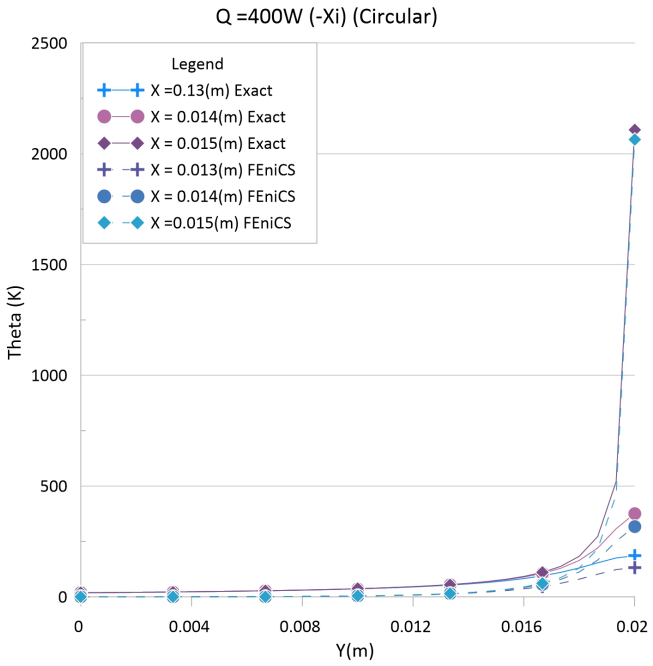


(a) In direction opposite to moving bed

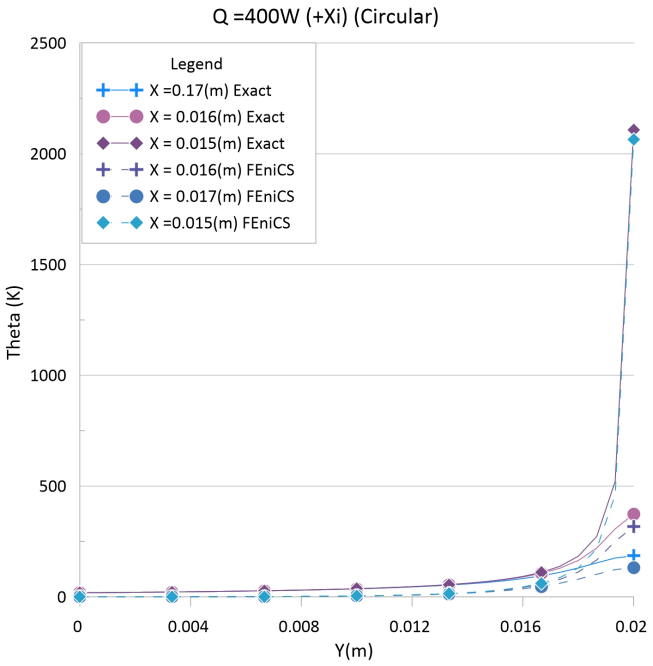


(b) In direction of moving bed

Figure 3: v = 9mm/s Square Heat Source

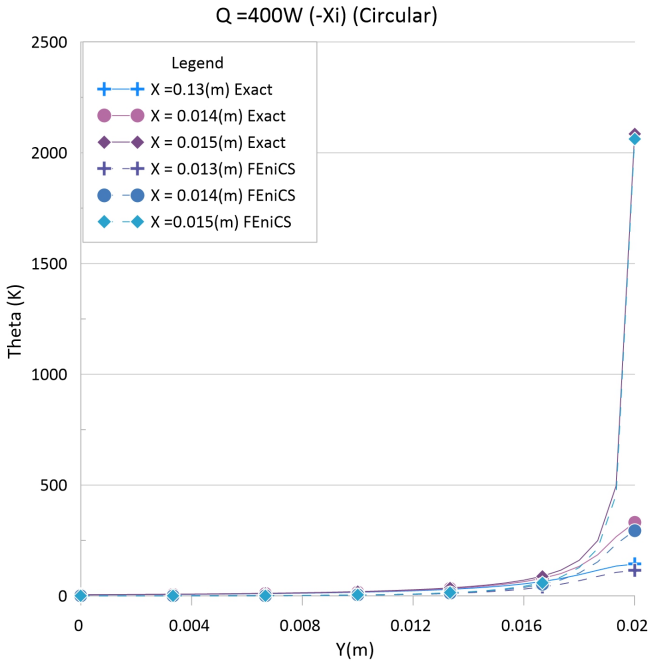


(a) In direction opposite to moving bed

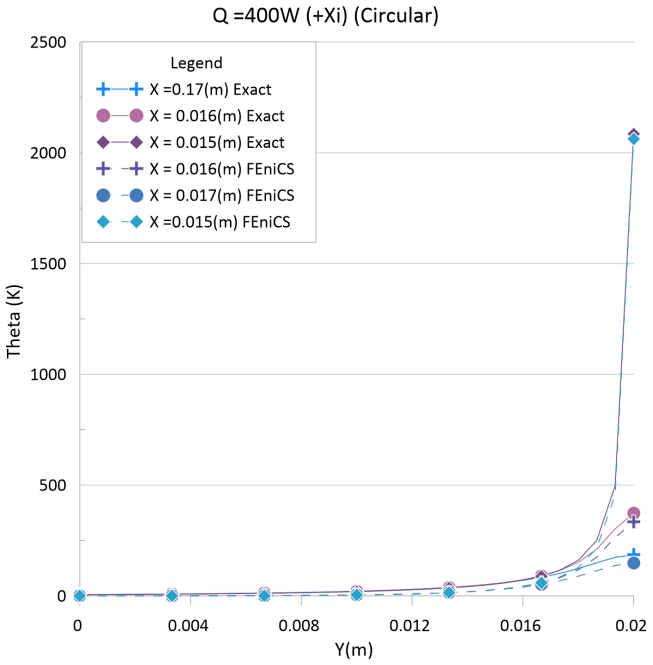


(b) In direction of moving bed

Figure 4: v = 0mm/s Circular Heat Source

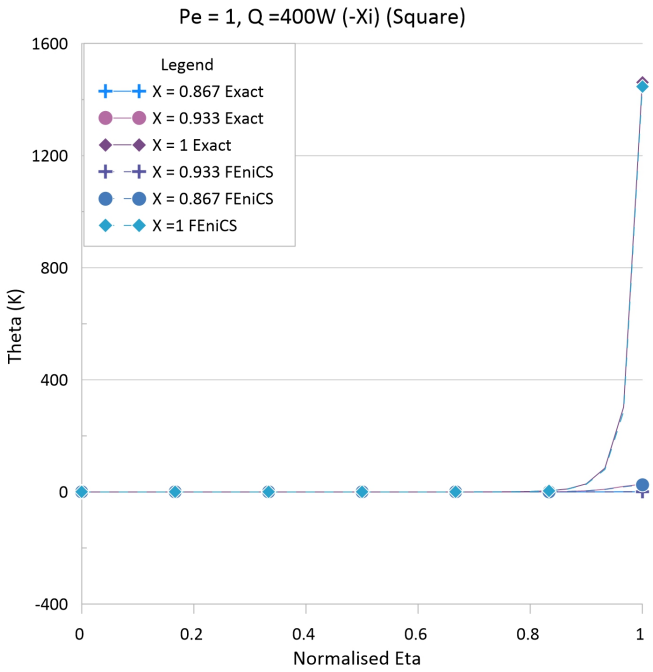


(a) In direction opposite to moving bed

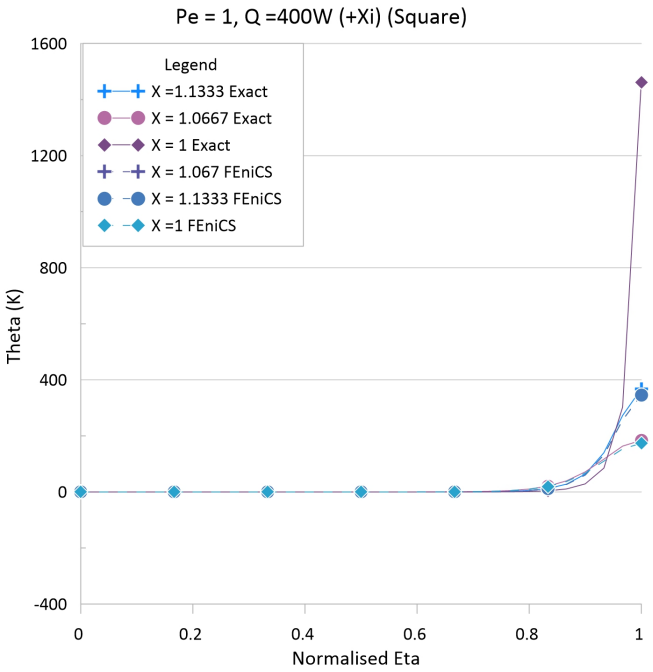


(b) In direction of moving bed

Figure 5: v = 9mm/s Circular Heat Source

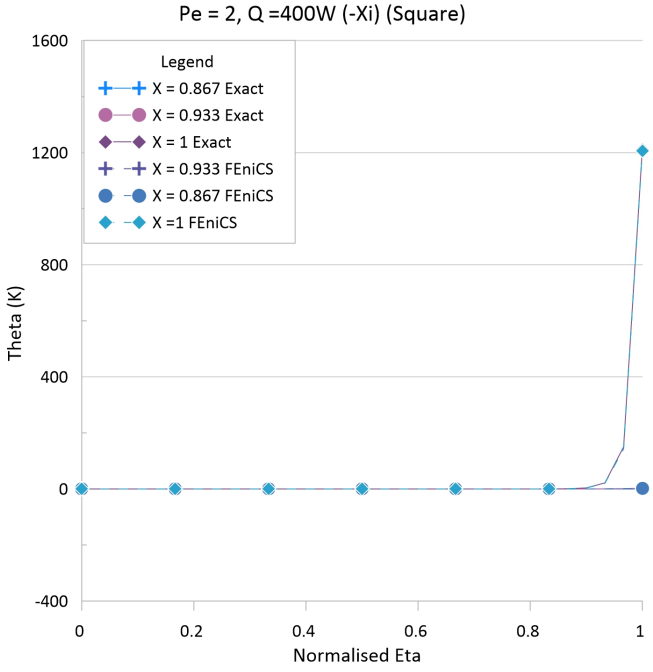


(a) In direction opposite to moving bed

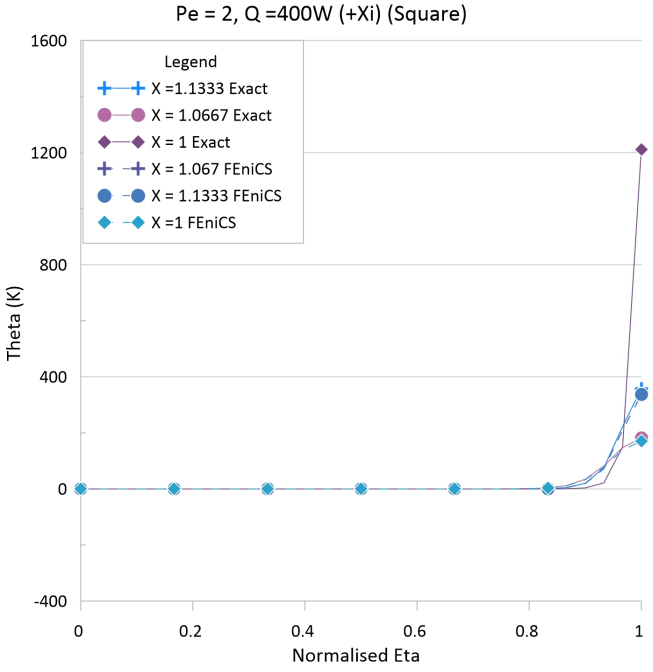


(b) In direction of moving bed

Figure 6: Peclet No. = 1 Square Heat Source (Normalised co-ordinte)

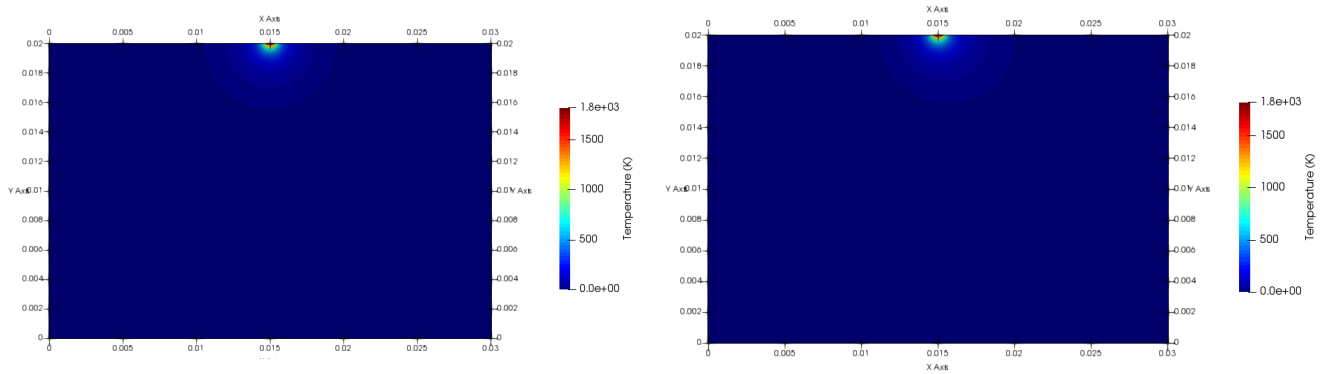


(a) In direction opposite to moving bed



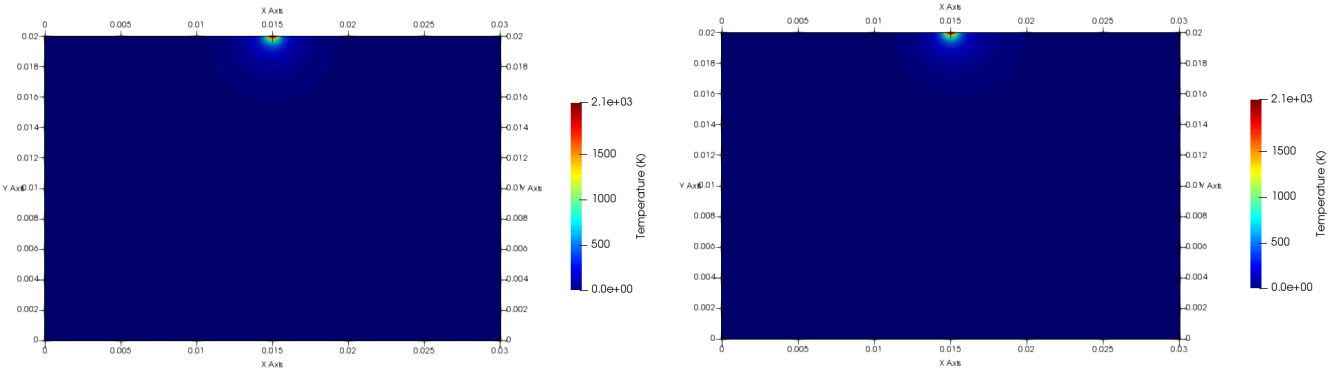
(b) In direction of moving bed

Figure 7: Peclet No. = 2 Square Heat Source (Normalised co-ordinate)



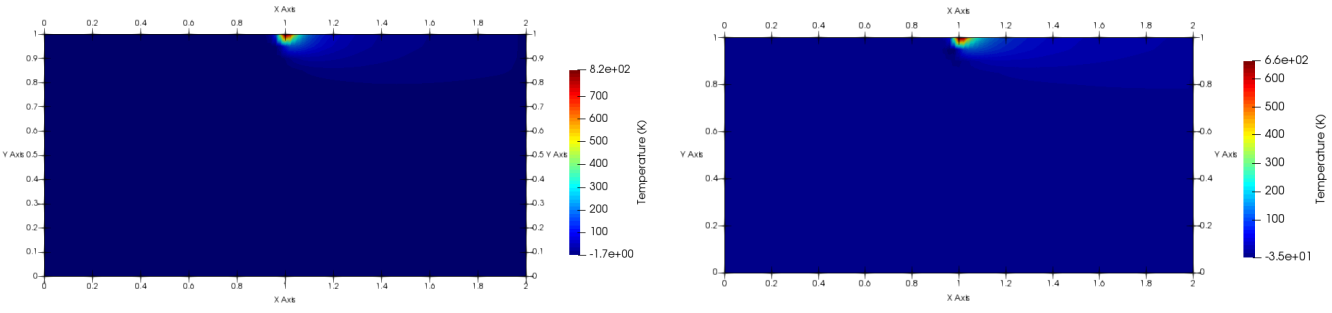
(a) v = 0mm/s (b) v = 9mm/s

Figure 8: Square Heat Source



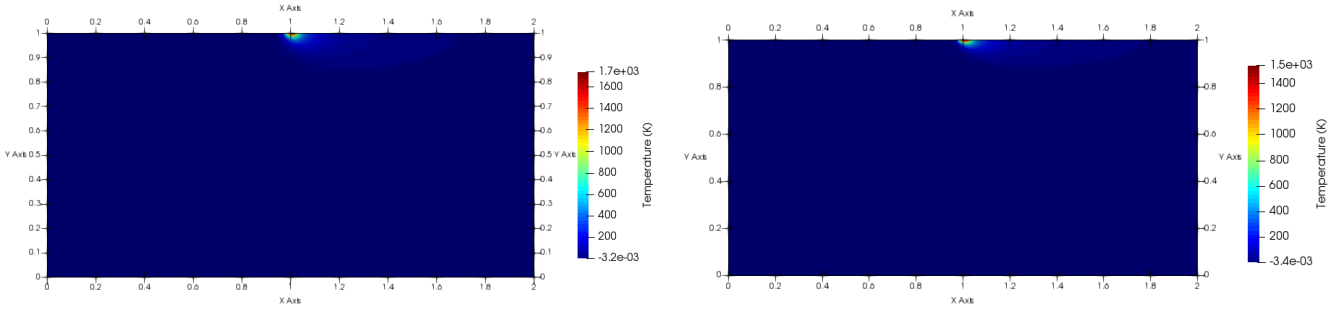
(a) v = 0mm/s (b) v = 9mm/s

Figure 9: Circular Heat Source



(a) Peclet No. = 1 (b) Peclet No. = 2

Figure 10: Square Heat Source



(a) Peclet No. = 1 (b) Peclet No. = 2

Figure 11: Circular Heat Source

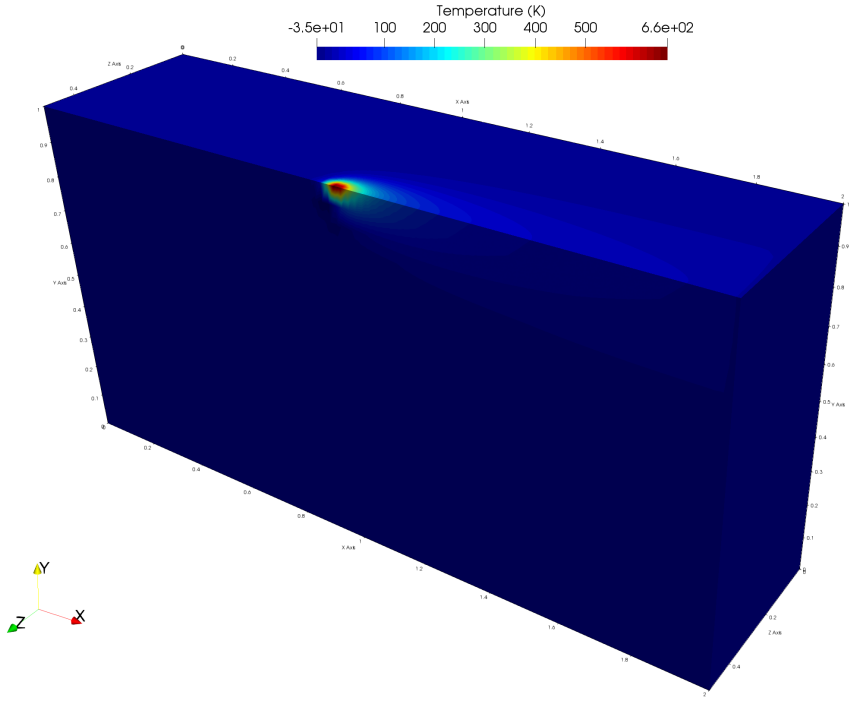


Figure 12: Sliced 3D plot for Square Heat Source with Peclet No. = 2

5 Conclusion

Since FEM is just approximation to get results which may or may not converge to exact result. There will always be an error, to avoid this error we may increase no of elements to an extent which may produce a converging result. Here we have exact result and we need to mesh geometry to an extent which may produce converging result. To reduce the use of intensive computational resources we have used refined mesh at the surface of Heat source. The exact results are converging with approximate results with maximum error of 2.887 % within all obtained results.
The results that we have got with Numerical Solution also has a same temperature profile as that of Exact solution. So this work can be extended with physical/Natural boundaries to analyze real life application of of Selective Laser Sintering process, apart from that it can also be applicable in welding process and we can also accommodate different welding process by changing the different boundary conditions. And here flexibility is that we can also change different materials such as isotropic & orthotropic materials.
Differential equations are modelled using [1] and the parameters are taken from [2] .

References

[1] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[2] Elham Mirkoohi, Jinqiang Ning, Peter Bocchini, Omar Fergani, Kuo-Ning Chiang, and Steven Liang. Thermal Modeling of Temperature Distribution in Metal Additive Manufacturing Considering Effects of Build Layers, Latent Heat, and Temperature-Sensitivity of Material Properties. *Journal of Manufacturing and Materials Processing*, 2(3):63, sep 2018.