Project Assignment #1

26 September 2018

MAE 610 Numerical Methods for Engineers

Dr. Loren Sumner

Jordan Hughes and Hamilton Wise

Abstract

This project models the heat transfer experienced by a semi-infinite solid medium being heated by a laser. The system also experiences convection at the surface at ambient temperature. First, the problem will be set up mathematically with a differential equation with boundary conditions. Then, the mathematical solution will be translated into a numerical solution for use in Matlab. A simple explicit forward difference approximation was used to step forward in time, and a second-order central difference approximation was used for spatial derivatives. The model is displayed using a contour plot showing temperature changing with time.

Mathematical Problem

Consider the following problem: A laser incident on a semi infinite solid medium (with thermal diffusivity *α*), introduces a heat flux over a circular area of diameter *d* at a constant rate of *q"*. Convection to the ambient air at temperature *Ta* occurs at the surface with a heat transfer coefficient of *h*. Assume that initially the solid medium is in equilibrium with the ambient air and that the problem is symmetric. Solve the unsteady diffusion equation for the temperature distribution in the medium *T(r,z,t)*. An illustration of the system can be seen in figure 1.

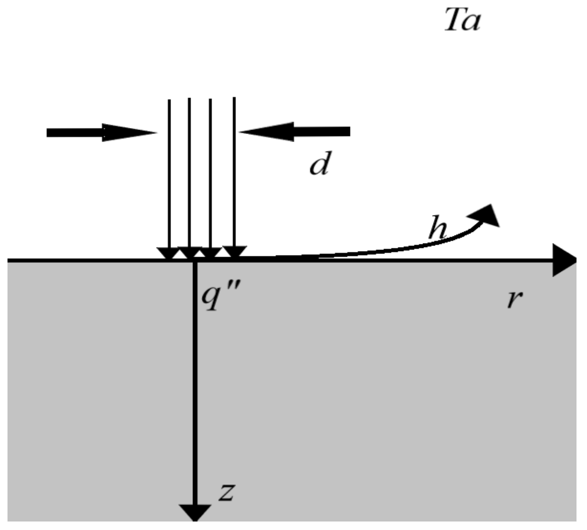


Figure 1: Problem Definition

In order to properly mathematically model the temperature distribution, the unsteady diffusion equation with several boundary conditions must be solved. It is assumed that the system follows the laws of heat transfer (i.e. Fourier's Law of Heat Conduction). The medium is considered semi-infinite, which gives the system insulated boundary conditions on all sides except for the top, where the medium experiences heating and convection. It is also assumed the material is homogeneous with constant material properties.

The governing differential equation that models the temperature distribution in this cylindrical medium is shown in equation 1.

$$T_t = \alpha \nabla^2 T \tag{1}$$

In this equation, T_t denotes the change in temperature with respect to time, $\alpha$is the thermal diffusivity, and $\nabla^2 T$ is the Laplacian operator. The Laplacian operator is shown in equation 2 for cylindrical coordinates.

$$\nabla^2 T = \frac{1}{r}\frac{\delta}{\delta r}\left(r\frac{\delta T}{\delta r}\right) + \frac{1}{r^2}\frac{\delta^2 T}{\delta \theta^2} + \frac{\delta^2 T}{\delta z^2} \tag{2}$$

Before equation 2 can be numerically approximated, the partial derivative with respect to the r-axis must be expanded into first and second derivatives resulting in equation 3.

$$\nabla^2 T = \frac{\delta^2 T}{\delta r^2} + \frac{1}{r}\frac{\delta\ \ T}{\delta r} + \frac{1}{r^2}\frac{\delta^2 a}{\delta \theta^2} + \frac{\delta^2 T}{\delta z^2} \tag{3}$$

Temperature is assumed to be constant at every angle θ, therefore the partial derivative with respect to θ goes to zero. The resulting governing differential equation for the entire model is

$$\frac{\delta T}{\delta t} = \alpha \nabla^2 T = \frac{\delta^2 T}{\delta r^2} + \frac{1}{r}\frac{\delta \quad T}{\delta r} + \frac{\delta^2 T}{\delta z^2} \tag{4}$$

Using equation 4 and parameters given in the problem statement, the following boundary conditions were created to use in the numerical solution. The medium is considered to be semi-infinite, which allows the outer and bottom edges to be considered insulated to the surroundings. Therefore the temperature gradient at these locations is zero, and this is shown in equations 5 and 6.

$$\left.\frac{\delta T}{\delta r}\right|_{r=R} = 0 \tag{5}$$

$$\left.\frac{\delta T}{\delta z}\right|_{z=Z} = 0 \tag{6}$$

The system is symmetrical along the z-axis, which allows for an insulated boundary condition because there is no temperature change across r=0, seen in equation 7.

$$\left.\frac{\delta T}{\delta r}\right|_{r=0} = 0 \tag{7}$$

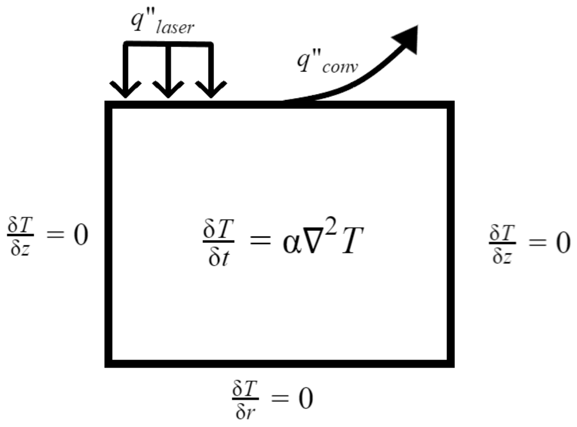
The entire surface at z=0 is exposed to convection cooling and a heat flux at the center or r=0. Equation 8 gives the convection boundary condition applied to the surface.

$$\frac{\delta T}{\delta z} = \frac{h(T - T_a)}{k} \tag{8}$$

Energy is going into the medium via a laser with diameter *d*. Therefore, at 0<r<d/2, a constant heat flux is given as a boundary condition in equation 9.

$$\frac{\delta T}{\delta z} = \frac{q''_{laser}}{k} \tag{9}$$

Also, when and where there is no heat flux into the medium from the top surface, there is only convection. Figure 2 shows the system with the boundary conditions applied to the proper edges.



q''_{laser}
q''_{conv}
$\frac{\delta T}{\delta z} = 0$
$\frac{\delta T}{\delta t} = \alpha \nabla^2 T$
$\frac{\delta T}{\delta z} = 0$
$\frac{\delta T}{\delta r} = 0$

Figure 2: Boundary Conditions

Numerical Procedure

For the numerical solution, a simple explicit forward difference approximation (FDA) was used to march in time, and a second order accurate central difference approximation (CDA) scheme was used for spatial derivatives. Equations 10 and 11 show general equations for the FDA and CDA, respectively.

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h) \tag{10}$$

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2) \tag{11}$$

Also a first order CDA was used to evaluate the middle term of equation 4,$\frac{1}{r}\frac{\delta\ T}{\delta r}$. Evaluating$\frac{1}{r}$at r = 0 is undefined, and results in a singular matrix. Therefore,$\frac{\delta\ T}{\delta r}$=0 at r=0 because the slope of the temperature gradient at that value is zero. The first order CDA is shown in equation 12.

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + O(h^2) \tag{12}$$

Equations 10-12 give estimations of derivatives that can be applied to equation 4 to estimate the temperature of the inside nodes of the system. This results in equation 13,

$$\frac{T_{ij}^{n+1} - T_{ij}^{n}}{\alpha \Delta t} = [(\frac{T_{i+1,j}^{n} - 2T_{ij}^{n} + T_{i-1,j}^{n}}{\Delta r^2}) + \frac{1}{r_{ij}}(\frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{2\Delta r}) + (\frac{T_{i,j+1}^{n} - 2T_{ij}^{n} + T_{i,j-1}^{n}}{\Delta z^2})] \tag{13}$$

and with further simplifying and separating terms gives equation 14.

$$\frac{1}{\alpha \Delta t}[T_{ij}^{n+1} - T_{ij}^{n}] = (\frac{-2}{\Delta r^2} + \frac{-2}{\Delta z^2})T_{ij}^{n} + (\frac{1}{\Delta r^2} + \frac{1}{2\Delta r r_{ij}})T_{i+1,j}^{n} + (\frac{1}{\Delta r^2} + \frac{-1}{2\Delta r r_{ij}})T_{i-1,j}^{n} + (\frac{1}{\Delta z^2})T_{i,j+1}^{n} + (\frac{1}{\Delta z^2})T_{i,j-1}^{n} \tag{14}$$

To explicitly solve this differential equation, T_{ij}^{n+1} must be isolated as shown in equation 15.

$$T_{ij}^{n+1} = T_{ij}^{n} + \alpha \Delta t[(\frac{-2}{\Delta r^2} + \frac{-2}{\Delta z^2})T_{ij}^{n} + (\frac{1}{\Delta r^2} + \frac{1}{2\Delta r r_{ij}})T_{i+1,j}^{n} + (\frac{1}{\Delta r^2} + \frac{-1}{2\Delta r r_{ij}})T_{i-1,j}^{n} + (\frac{1}{\Delta z^2})T_{i,j+1}^{n} + (\frac{1}{\Delta z^2})T_{i,j-1}^{n}] \tag{15}$$

Equation 15 is the explicit finite difference method that will be used to solve the mathematical model. The radius at the evaluated node is r_{ij} and, $r_{ij} = i * \Delta r$, which gives $2\Delta r r_{ij} = 2i\Delta r^2$. If $\Delta r = \Delta z$ then let $\lambda = \frac{\alpha \Delta t}{\Delta r^2}$ and let $\gamma = \frac{\alpha \Delta t}{2i\Delta r^2}$.

By defining $\lambda$and γ, equation 15 can be computationally simplified yielding equation 16,

$$T_{ij}^{n+1} = T_{ij}^{n} + (1-4\lambda)T_{ij}^{n} + (\lambda+\gamma)T_{i+1,j}^{n} + (\lambda-\gamma)T_{i-1,j}^{n} + (\lambda)T_{i,j+1}^{n} + (\lambda)T_{i,j-1}^{n} \quad (16)$$

which is the equation that is applied individually at every node in the system to yield the temperature at time $t = n + 1$.

Before equation 16 can be applied, each node on the surface must be evaluated using the appropriate boundary condition. Evaluating these edge nodes with equation 16 will result in an error since equation 16 is referencing a neighboring node that is out of the system. To fix this, every node outside the system is replaced using the boundary condition for that node.

The boundary conditions for the insulated surfaces are shown in equations 5 and 6. Applying equation 12 to these boundary conditions results in the following equations

$$\frac{\delta T}{\delta z} = \frac{T_{i,j+1}-T_{i,j-1}}{2\Delta z} = 0 \Rightarrow T_{i,j+1} = T_{i,j-1} \qquad \text{z=Z} \quad (17)$$

$$\frac{\delta T}{\delta r} = \frac{T_{i+1,j}-T_{i-1,j}}{2\Delta r} = 0 \Rightarrow T_{i+1,j} = T_{i-1,j} \qquad \text{at r=0} \quad (18)$$

$$\frac{\delta T}{\delta r} = \frac{T_{i+1,j}-T_{i-1,j}}{2\Delta r} = 0 \Rightarrow T_{i+1,j} = T_{i-1,j} \qquad \text{r=R} \quad (19)$$

The boundary conditions for the top surface are shown in equation 8 and 9. Applying equation 12 for these two boundary conditions result in the following equations.

$$-k\frac{\delta T}{\delta z} = q''_{laser} = -k\frac{T_{i,j+1}-T_{i,j-1}}{2\Delta z} \Rightarrow T_{i,j+1} = \frac{-2\Delta z}{k}q''_{laser} + T_{i,j-1} \quad (20)$$

$$-k\frac{\delta T}{\delta z} = h[T_{i,j} - T_{amb}] = \frac{T_{i,j+1}-T_{i,j-1}}{2\Delta r} = \Rightarrow T_{i,j+1} = \frac{2\Delta zh}{k}[T_{amb} - T_{ij}] + T_{i,j-1} \quad (21)$$

Before equation 16 can be applied to the numerical model, the boundary conditions have to be applied across all edges of the model. Each edge will have one false node that is replaced by the appropriate interior node and each corner will have two false nodes that are replaced by the appropriate interior nodes. After these false nodes are replaced in the numerical model, the boundary conditions shown in equation 17 through 21 are applied. Once this is complete, equation 16 can be applied to all nodes of the system. The nodal setup is shown below in figure 3, and the top left node is chosen to be the starting node and the bottom right node is chosen to be the end. This nodal indexing was chosen since it follows the same matrix indexing that MatLab uses for all matrices.

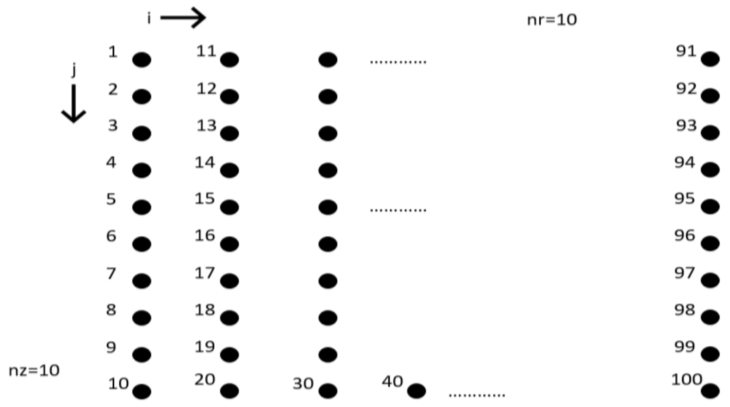


Figure 3: Node Layout

The Matlab code is shown in the appendix. The problem parameters are first defined for this model, and all constants are chosen to be 1 for mathematical simplicity. The system parameters are then defined and the height and diameter of the cylinder and the number of nodes for each was chosen to be equal in order to simplify the governing equation as shown in equation 16. The initial condition is then defined as Tmat and the new temperature matrix for T_{ij}^{n+1} is preallocated as T_new. The final time that the model is to be evaluated for is defined as ftime and the time step size is defined as dt. The number of time steps are then calculated in order to evaluate equation 16 at every time step until the resulting temperature matrix is the temperature of the system at the final time.

Before the governing differential equation can be evaluated at all nodes, the stability of the model has to meet the Von Neumann stability criteria as shown below,

$$\Delta t\left(\frac{1}{\Delta r^2}+\frac{1}{2\Delta r}+\frac{1}{\Delta z^2}\right)\leq\frac{1}{2} \tag{22}$$

If this equation is not satisfied, then the solution will not be valid since error at each iteration would be magnified instead of attenuated. If this equation fails, the code will stop and a smaller Δt will have to be chosen. This is a drawback of the explicit method since it is not unconditionally stable.

Once equation 22 is satisfied, a for loop for the number of time steps is created and a nested for loop for all i and j values are created. The node number is calculated using node = (i-1)*nr+j at each iteration, and if any neighboring nodes are false nodes, then a series of if statements determines which internal node will replace the false node at that point. After the false nodes are replaced, all appropriate boundary conditions are applied for all edge nodes and finally, equation 16 is used to find T_new. An entire loop of all the nodes is then evaluated, and the current temperature matrix is redefined as new temperature matrix calculated in the previous step. Lastly, after each new temperature matrix is found, a contour plot is created to show the temperature gradient in the medium.

Results and Discussion

Figures 4-7 demonstrate the results of the Matlab code at time=1s, 10s ,30s, and 60s, respectively.

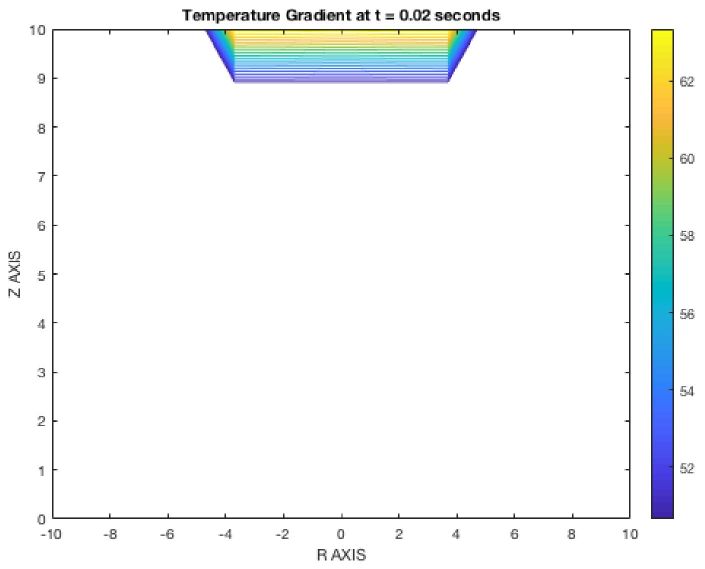


Figure 4: Results at 1st Time Step

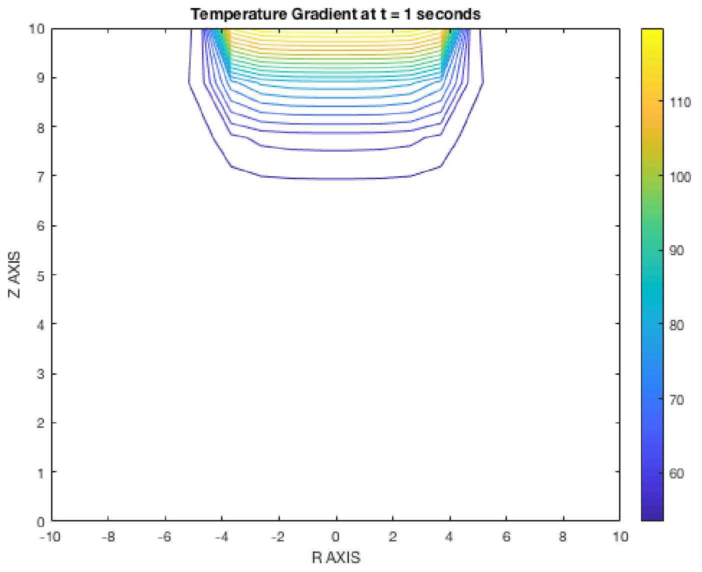


Figure 5: Results at Time = 1s

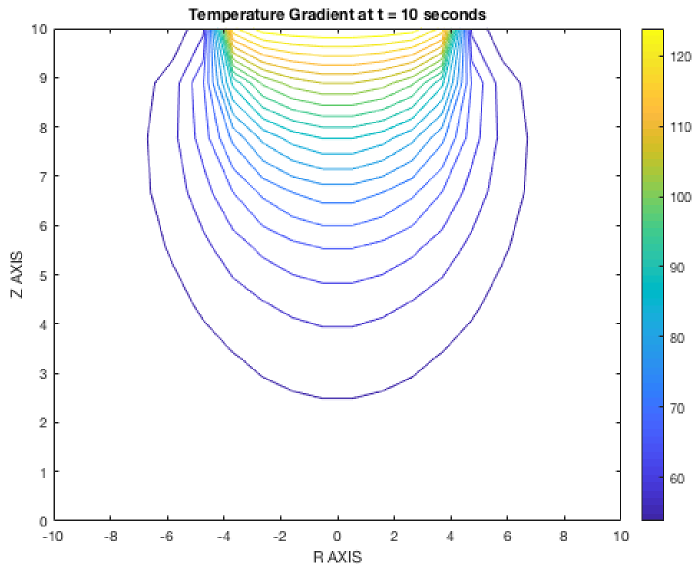


Figure 6: Results at Time = 10s

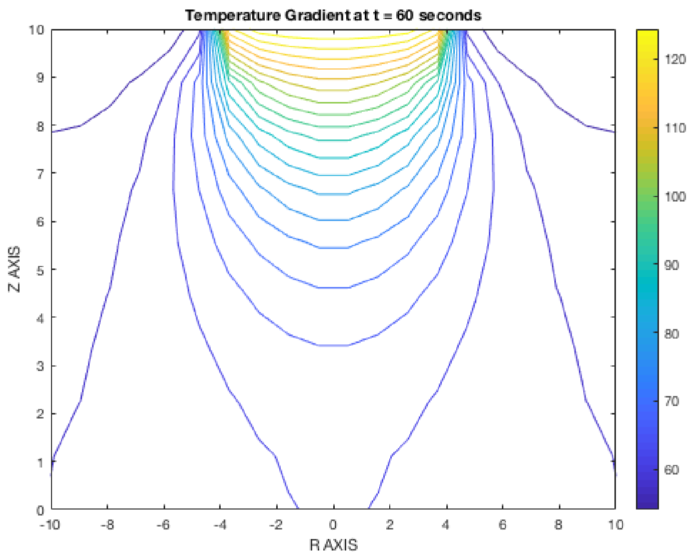


Figure 7: Results at Time = 60s

It can be seen that the temperature starts at ambient temperature, then immediately rises in the middle where the heat flux from the laser is applied. From that time on, the heat slowly moves from the center of the laser towards the outer edges as time progresses. Convection on the surface slows the heating of the medium, and allows for some of the energy from the laser to leave the system.

Appendix

Problem Parameters

```
qlaser = 100;     % Laser Heat Flux {W/m^2}
h      =  1 ;     % Free convection {W/m^2 K}
k      =  10;   % Thermal Conductivity {W/m K}
Tamb   =  50;     % Ambient Temperature {K}
alpha  =  1 ;     % Thermal Diffusivity {m^2/s}
```

System Parameters

```
R   = 10; % Radius of Cylinder
Z   = 10; % Depth of Cylinder
r_l = 4 ; % Radius of Laser

nodes = 10; % Number of nodes

nr = nodes;     % Nodes in r-direction
nz = nodes;     % Nodes in z-direction
dr = R/(nr-1); % Create step size for r-direction
dz = Z/(nz-1); % Create step size for z-direction

r = linspace(0,R,nodes)'; % Create radius vector
z = linspace(0,Z,nodes)'; % Create z vector
```

Preallocate Temp Matrix

```
Tmat  = ones(nr,nz)*Tamb;  % Initial Conditions based on model
T_new = zeros(nr,nz);      % Preallocate matrix for T at time n+1
```

Add Time Element

```
time  = 120;        % Total time in seconds
dt    = .02;        % Delta time for each iteration
```

```
ntime = time/dt;   % Number of steps in time
```

Check For Stability

```
stability = alpha* dt* ((1/dr^2) + (1/dz^2));  % Von Neumann Stability
 Analyis

if stability > 0.5
    fprintf('SOLUTION IS UNSTABLE!!\n')
    fprintf('Stability Criteria is %g\n',stability)
    return                                      % End if solution is
 unstable
else
    fprintf('Solution is Stable\n')
end
```

```
Solution is Stable
```

Loops

```
for n = 1:ntime

    for i = 1:nr
        for j = 1:nz
            node = (i-1)*nr+j;

%------------Bottom Edge Insulated---------------
            if j == nz && i > 1 && i < nr
            nodeleft  = node - nz;
            noderight = node + nz;
            nodeabove = node - 1 ;
            nodebelow = nodeabove; %false node

%------------Left Edge--------------------------
            elseif i == 1 && j > 1 && j < nz
            noderight = node + nz;
            nodeleft  = noderight;  %false node
            nodeabove = node - 1 ;
            nodebelow = node + 1 ;

%------------Right Edge Insulated----------------
            elseif i == nr && j > 1 && j < nr
            nodeleft  = node - nz;
            noderight = nodeleft ; %false node
            nodeabove = node - 1 ;
            nodebelow = node + 1 ;

%------------Top EdgeConvection-------------------
            elseif j == 1 && i > 1 && i < nr
            nodeleft  = node - nz;
            noderight = node + nz;
            nodebelow = node + 1 ;


         nodeabove = nodebelow; %false node
         % Boundary Conditions
          if i <= r_l
              Tmat(node) = Tmat(node) + (2* qlaser)/(k* dz);
              Tmat(node) = Tmat(node) + ((2* h)/(dz* k) * (Tamb -
Tmat(node)));
          else
              Tmat(node) = Tmat(node) + ((2* h)/(dz* k) * (Tamb -
Tmat(node)));
          end
```

```
%------------Top Left Corner w/ Laser-----------------
            elseif i == 1 && j == 1
            noderight = node + nz;
            nodeleft  = noderight; %false node
            nodebelow = node + 1 ;
            nodeabove = nodebelow; %false node
            % Boundary Conditions
             Tmat(node) = Tmat(node) + (2* qlaser)/(k* dz);
             Tmat(node) = Tmat(node) + ((2* h)/(dz* k) * (Tamb -
 Tmat(node)));

%------------Top Right Corner w/ Convection-------------
            elseif i == nr && j == 1
            nodeleft  = node - nz;
            noderight = nodeleft ;  %false node
            nodebelow = node + 1 ;
            nodeabove = nodebelow; %false node
            % Boundary Conditions
             Tmat(node) = Tmat(node) + ((2* h)/(dz* k) * (Tamb -
 Tmat(node)));

%------------Bottom Left Corner------------------------
            elseif i ==1 && j == nz
            noderight = node + nz;
            nodeleft  = noderight; %false node
            nodeabove = node - 1 ;
            nodebelow = nodeabove; %false node

%------------Bottom Right Corner------------------------
            elseif i == nr && j == nz
            nodeleft  = node - nz;
            noderight = nodeleft ;
            nodeabove = node - 1 ;
            nodebelow = nodeabove; %false node

%------------Internal Nodes-----------------------------
            else % i > 1 && i < nr && j > 1 && j < nz
            noderight = node + nz;
            nodeleft  = node - nz;
            nodeabove = node - 1 ;
            nodebelow = node + 1 ;
            end
```

```
%---------------------- Equations Constants----------------------------
    if dr == dz
        l = (alpha* dt)/(dr^2);  % lambda
        if i == 1
            g = 0; % At radius r=0, temperature gradient is zero
        else
            g = (alpha* dt)/(2* (i-1)* dr^2); % gamma
        end
    else
        fprintf('ERROR IN SIMPLIFIED DIFFERENCE APPROXITION DUE TO
UNEQUAL STEP SIZES\n')
    end

%----------------------Implicit  Equations-----------------------------
            T_new(node) = (1-4*l)*Tmat(node)...
                    + (l + g)*Tmat(noderight) + (l -
g)*Tmat(nodeleft)...
                    + (l)*Tmat(nodeabove) + (l)*Tmat(nodebelow);
         end
    end

    Tmat = T_new; % Temp at time n+1 becomes Temp at Time n for next
iteration
    time = n*dt;  % Time at current iteration
```

Plot Results

```
        if (mod(n,500)==0)  % plot at every 100 time steps

            Tmatr1 = Tmat;
            Tmatr2 = fliplr(Tmat); % Reflect Temperature matrix along z-
axis

            Tmatr  = [Tmatr2(:,1:end) Tmatr1]; % Combine reflected and
 original Tmat

            Rr = linspace(-R,R,2*nodes)'; % Create new R axis for plotting

            [c1, h1] = contour(Rr,z,flipud(Tmatr),20);
            colorbar
            title(sprintf('Temperature Gradient at t = %g seconds',time))
            xlabel('R AXIS')
            ylabel('Z AXIS')
            drawnow
        end
```