Heat Transfer - Homework 6

Rose Gebhardt, Hannah Quirk, and Harris Paspuleti

March 4, 2020

1 Estimate Solution to Heat Equation

Problem Statement: Compare the exact solution of a suddenly, symmetrically cooled wall to a multi-node equivalent circuit model. Set the initial temperature of the entire wall to be 37°C and the temperature of the boundaries to be 25°C. Assume that the wall is made of 0.5 m thick stainless steel and has a height that is much larger than the thickness.

(a) Start with a small number of nodes and find the impedance matrix. Calculate the estimated temperature as a function of time for each element.

First, consider a system with three nodes. It can be modelled with the following circuit:

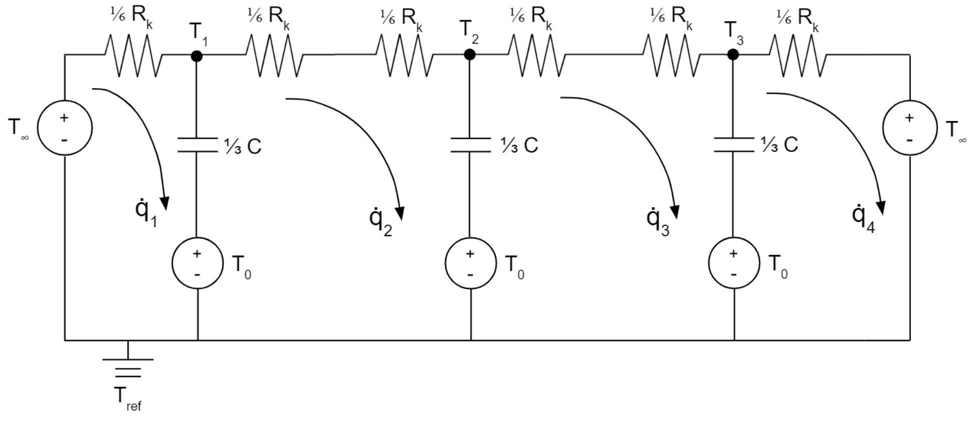


Figure 1: Three Node Model

Mesh analysis leads to the following system of equations in the Laplace domain:

$$\begin{pmatrix} \frac{R_k}{6}+\frac{3}{sC} & \frac{-3}{sC} & 0 & 0 \\ \frac{-3}{sC} & \frac{R_k}{3}+\frac{6}{sC} & \frac{-3}{sC} & 0 \\ 0 & \frac{-3}{sC} & \frac{R_k}{3}+\frac{6}{sC} & \frac{-3}{sC} \\ 0 & 0 & \frac{-3}{sC} & \frac{R_k}{6}+\frac{3}{sC} \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{pmatrix} = \begin{pmatrix} \frac{T_\infty - T_0}{s} \\ 0 \\ 0 \\ \frac{T_0 - T_\infty}{s} \end{pmatrix}$$

Once the heat transfer rates are calculated, the temperatures of each node can be found using Ohm's Law:

$$T_1 = T_\infty - \dot{q}_1\left(\frac{R_K}{6}\right) \quad , \quad T_2 = T_1 - \dot{q}_2\left(\frac{R_K}{3}\right) \quad , \quad T_3 = T_2 - \dot{q}_3\left(\frac{R_K}{3}\right)$$

MATLAB's impulse function can then be used to find the temperature of each node as a function of time.

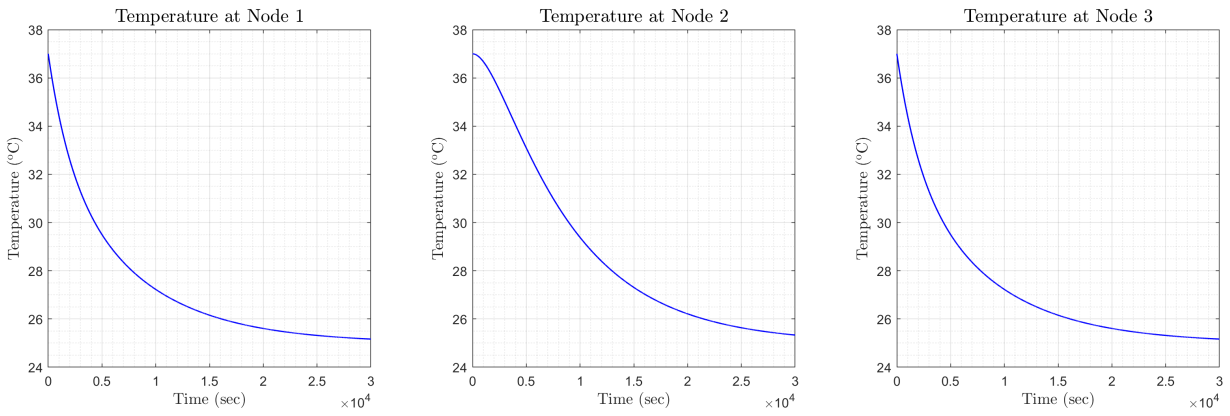


Figure 2: Temperature at Each Node for Three Node Model

(b) Using the exact solution derived in class, calculate the temperature as a function of position and time. The heat equation is

$$\frac{\partial^2 T}{\partial x^2} - \frac{\rho c}{k}\frac{\partial T}{\partial t} = 0 \qquad \text{or} \qquad \frac{\partial^2 T}{\partial x^2} - \frac{1}{\alpha}\frac{\partial T}{\partial t} = 0$$

where k is the conduction coefficient, ρ is the density, c is the specific heat capacitance, and $\alpha = \frac{k}{\rho c}$ is the thermal diffusivity of the material. The boundary conditions for this case are $T(0,t) = T_\infty$, $T(L,t) = T_\infty$, and $T(x,0) = T_0$. The solution to the heat equation is then

$$T(x,t) = \frac{T_0 - T_\infty}{\pi}\left[\sum_{n=0}^{\infty}\left(\frac{1-(1)^n}{n}\right)\left(\exp\left(-\alpha t\left(\frac{n\pi}{L}\right)^2\right)\right)\left(\sin\left(\frac{n\pi x}{L}\right)\right)\right] + T_0$$

It should be noted that, when implemented in code, the sum cannot go up to infinity. Instead, the sum goes to 100. This is only an issue for times very close to zero.

(c) Plot both the estimated temperature and the exact temperature for several different points in time. How well does the estimated temperature match the exact temperature?

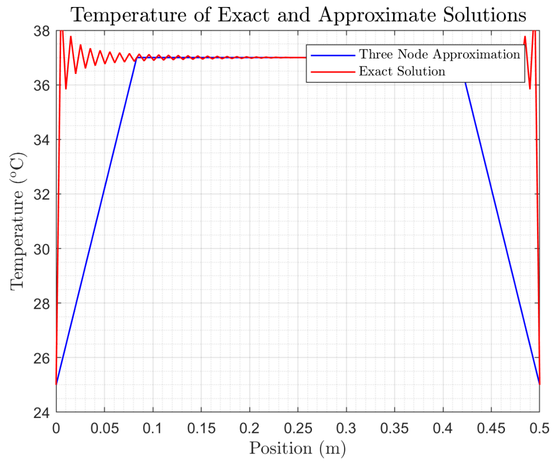


Figure 3: Temperature at Time = 0 seconds

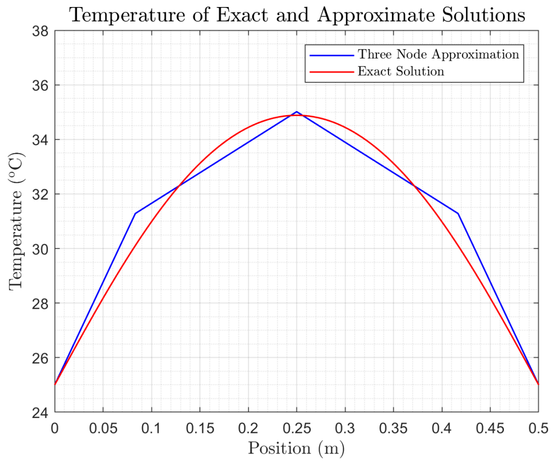


Figure 4: Temperature at Time = 3000 seconds

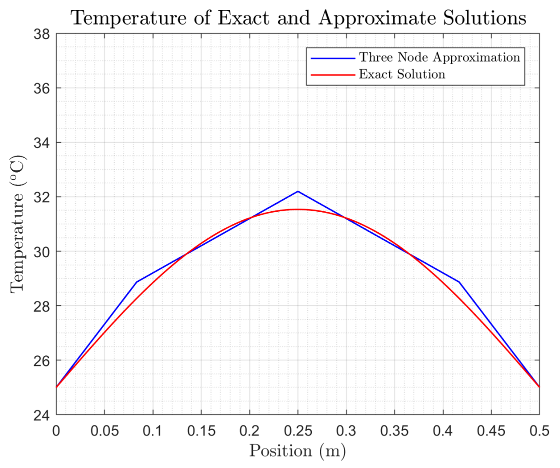


Figure 5: Temperature at Time = 6000 seconds

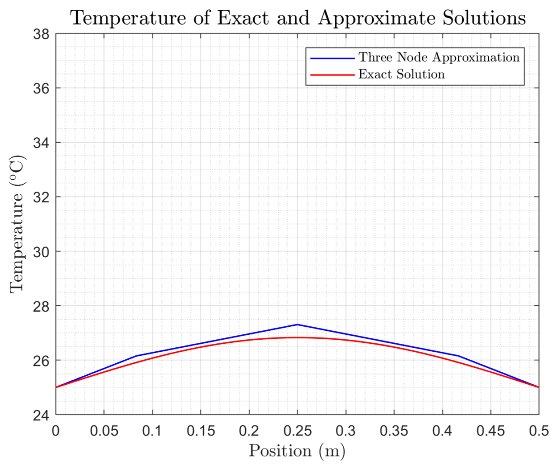


Figure 6: Temperature at Time = 15000 seconds

The general behavior of the system is approximated fairly well by using three nodes. The rate at which the temperature at each point reaches 25°C and the overall shape of the temperature over distance is the same for the approximated and exact solutions. However, the three node solution is not smooth so it is very different from the exact solution at certain points.

(d) Come up with an algorithm for generating the impedance matrix and forcing vector for an arbitrary number, n, of nodes. Increase n until you are satisfied that you have a sufficient number of nodes to adequately estimate the exact solution. How many nodes did you use?

A clear pattern arises in the set up of the mesh equations as the number of nodes, n, increases. The impedance matrix ends up being tridiagonal. All the subdiagonal and superdiagonal entries are $\frac{-n}{sC}$, the first and the last diagonal entries are $\frac{R_k}{2n} + \frac{n}{sC}$, and all other diagonal terms are $\frac{R_k}{n} + \frac{2n}{sC}$. The forcing vector ends up only having two non-zero entries. The first term is $\frac{T_\infty - T_0}{s}$ and the last term is $\frac{T_0 - T_\infty}{s}$.

$$\begin{pmatrix} \frac{R_k}{2n} + \frac{n}{sC} & \frac{-n}{sC} & & & & \\ \frac{-n}{sC} & \frac{R_k}{n} + \frac{2n}{sC} & \frac{-n}{sC} & & & \\ & \frac{-n}{sC} & \ddots & \ddots & & \\ & & \ddots & \ddots & \frac{-n}{sC} & \\ & & & \frac{-n}{sC} & \frac{R_k}{n} + \frac{2n}{sC} & \frac{-n}{sC} \\ & & & & \frac{-n}{sC} & \frac{R_k}{2n} + \frac{n}{sC} \end{pmatrix} \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \vdots \\ \vdots \\ \dot{q_n} \\ \dot{q}_{n+1} \end{pmatrix} = \begin{pmatrix} \frac{T_\infty - T_0}{s} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \frac{T_0 - T_\infty}{s} \end{pmatrix}$$

Once the heat transfer rates are found, the temperature at the nodes can be found iteratively. The temperature at the first node in the s-domain is defined with the equation

$$T_1 = T_\infty - \dot{q_1}\left(\frac{R_k}{2n}\right)$$

After this, the iterative equation is,

$$T_{(i)} = T_{(i-1)} - \dot{q_{(i)}}\left(\frac{R_k}{n}\right)$$

The number of nodes that can be used for this problem are limited due to numerical instability arising, but after about 9 nodes, the approximate solution estimates the exact solution fairly well. This can be seen in the video submitted.

Final Results

- The behavior of the system can be found using an exact solution to the heat equation or by approximating the system as many nodes.
- The two behaviors have similar steady-state tendencies.
- As the number of nodes is increased, the approximate solution better matches the exact solution.
- The solution as the number of nodes increases can be found algorithmically.

Concluding Statements This problem shows how the solution to the heat equation can be approximated by breaking a continuous path into discrete points and treating each as an element capable of storing heat. This enables one to find the behavior of the system without finding an exact equation, which is computationally much more difficult. It also demonstrates how increasing the number nodes used can improve the predicted behavior but can also lead to numerical instabilities.

2 Steel Piston

Problem Statement: A steel piston cylinder wall is subjected to an oscillating surface temperature which we approximate as $T = 650°C + (300°C)\cos\omega t$. The piston cycles 8 times per second.

(a) Plot the amplitude of the temperature variation in the steel as a function of depth.

First, the piston is modeled as shown below. It is assumed that the ambient temperature $T_\infty = 25°$ C, and the initial temperature of the wall is $T_0 = 37°$ C.

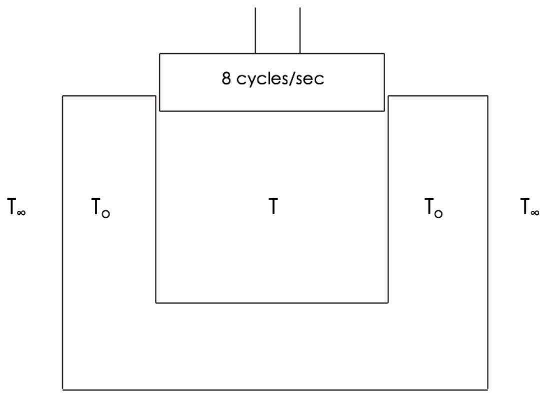


Figure 7: The steel piston cylinder

Assuming the temperature on the inside of the piston is uniform and the geometry is symmetric, the problem can be simplified by looking at only one wall of the steel piston cylinder. An arbitrary number n nodes divides the wall. The figure below shows the circuit diagram for $n = 3$ nodes.

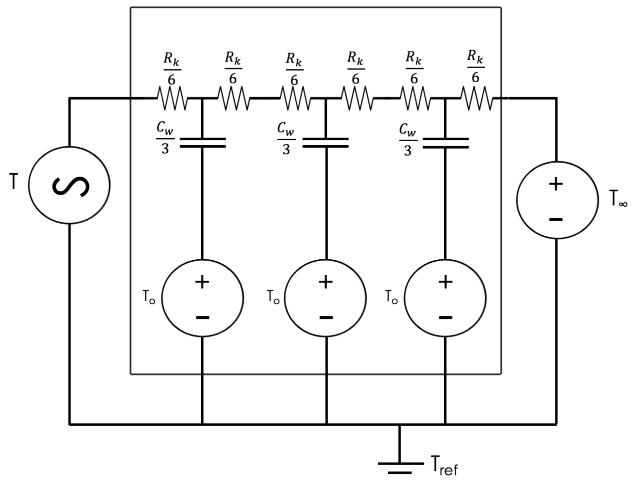


Figure 8: Three node approximation of circuit model

For this problem, the boundary temperature in the time domain is $T(t) = 650°\text{C} + (300°\text{C})\cos\omega t$.

The piston cycles 8 times per second, so $\omega = 16\pi$. In the frequency domain, the boundary temperature is,

$$T(s) = \frac{923.15}{s} + \frac{573.15s}{s^2 + (16\pi)^2}$$

The form of the impedance matrix for an arbitrary number of nodes will remain the same as question 1. Because only the boundary conditions changes between question one and two, only the forcing vector changes in the mesh equations. The equations are as follows:

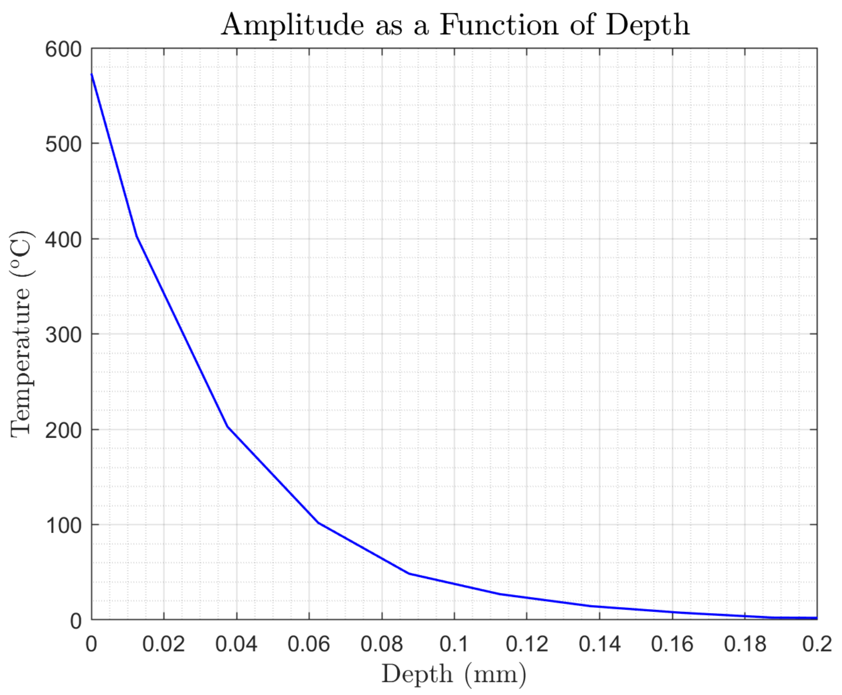
$$\begin{pmatrix} \frac{R_k}{2n} + \frac{n}{sC} & \frac{-n}{sC} & & & & \\ \frac{-n}{sC} & \frac{R_k}{n} + \frac{2n}{sC} & \frac{-n}{sC} & & & \\ & \frac{-n}{sC} & \ddots & \ddots & & \\ & & \ddots & \ddots & \frac{-n}{sC} & \\ & & & \frac{-n}{sC} & \frac{R_k}{n} + \frac{2n}{sC} & \frac{-n}{sC} \\ & & & & \frac{-n}{sC} & \frac{R_k}{2n} + \frac{n}{sC} \end{pmatrix} \begin{pmatrix} \dot{q_1} \\ \dot{q_2} \\ \vdots \\ \vdots \\ \dot{q_n} \\ \dot{q_{n+1}} \end{pmatrix} = \begin{pmatrix} T(s) - \frac{T_0}{s} \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \frac{T_0 - T_\infty}{s} \end{pmatrix}$$

Once again, the temperature at the nodes can be found iteratively. The iterative equation is

$$T_{(i)} = T_{(i-1)} - \dot{q_{(i)}} \left(\frac{R_k}{n} \right)$$

The number of nodes that can be used for this problem are limited due to numerical instability arising, but 8 nodes produce a sufficient approximation for the temperature variation as a function of depth.

At each node, the maximum and minimum temperature are found once the system reaches steady-state. The difference between these values divided by two gives the amplitude of the oscillations at each point. A plot of the amplitude as the depth of the cylinder increases is shown below.



(b) If the cylinder is 1cm thick, can we view it as having infinite depth? Why or why not?

If the cylinder is 1cm thick, it can be seen as having infinite depth. From the plot in figure above, it can be seen that the amplitude of the temperature variation is very close to zero by 2 mm. This means that, after this point, there is almost no variation in the amplitude of the temperature. So, the system will act the same at a depth of 1 cm as it would at infinite depth.

Final Results

- The introduction of an oscillating boundary temperature at one side of the wall only affects the forcing vector and not the form of the impedance matrix.
- The amplitude of the temperature variation is very close to zero for thicknesses greater than 2mm.
- The cylinder can be viewed as having infinite depth for 1cm thickness.

Concluding Statements This problem shows how a continuous system can be effectively solved as a discrete system. The piston system was simplified into a heating and cooling wall problem. The practice of taking a more complex system and analyzing it as a more simple one is an important skill in engineering, and this problem provides practice for that skill. Part B also highlights how the validity of assumptions, such as the depth being infinite, can be checked numerically.

Contents

Homework 06 - ME342 - March 4, 2020

```
clear all; close all; clc;
```

Define parameters

```
s = tf('s');

% Boundary conditions
T_0 = 37 + 273.15; % K
T_inf = 25 + 273.15; % K

% Geometry
delta_x = 0.5; % m
A = 0.1; % m^2

% Thermal properties
k = 14.4; % W/mK
c = 502.416; % J/kgK
rho = 8000; % kg/m^3
C = rho*delta_x*A*c; % J/K

% Stainless steel 304
% Conductivity: https://www.engineeringtoolbox.com/thermal-conductivity-metals-d_858.html
% Specific Heat Capacitance:
  https://www.engineersedge.com/materials/specific_heat_capacity_of_metals_13259.htm
% Density: http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=mq304a

% Define impedances
z_r = delta_x/(k*A);
z_c = 1/(s*C);

% Define thermal diffusivity
alpha = k/(rho*c);
```

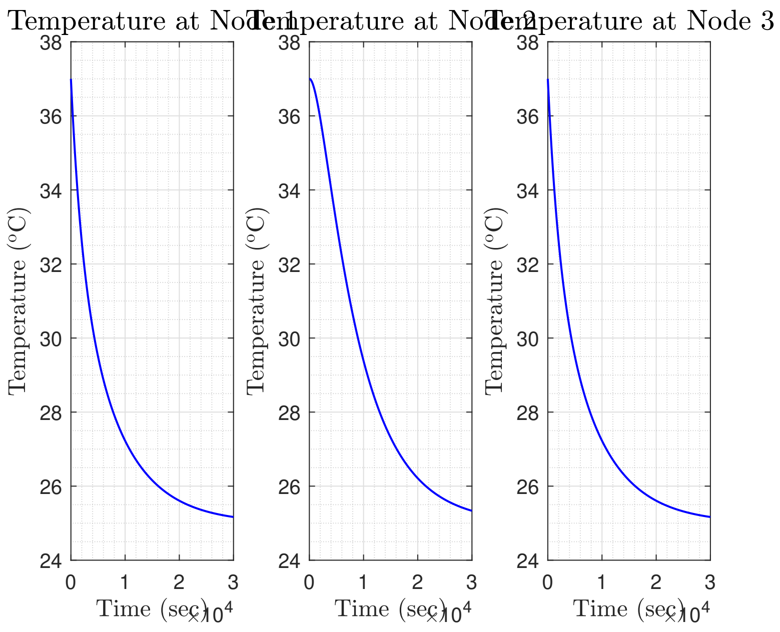
Part A Calculations

```
% Define terms used in impedance matrix
z3_11 = (z_r/6) + (3*z_c);
z3_12 = -3*z_c;
z3_22 = (z_r/3) + (6*z_c);

% Mesh analysis equations
z3 = [z3_11, z3_12, 0, 0;
      z3_12, z3_22, z3_12, 0;
      0, z3_12, z3_22, z3_12;
      0, 0, z3_12, z3_11];
f3 = [(T_inf-T_0)/s; 0; 0; (T_0-T_inf)/s];
q_dot3 = z3\f3;

% Find temperature at nodes
T3_1 = (T_inf/s) - q_dot3(1)*(z_r/6);
T3_2 = T3_1 - q_dot3(2)*(z_r/3);
T3_3 = T3_2 - q_dot3(3)*(z_r/3);

% Convert to time domain
time = linspace(0,30000,10000);
y3_1 = impulse(T3_1,time);
y3_2 = impulse(T3_2,time);
y3_3 = impulse(T3_3,time);
t3_1 = y3_1 - 273.15;
t3_2 = y3_2 - 273.15;
t3_3 = y3_3 - 273.15;
```

Part A Plots

```
figure(1)
subplot(1,3,1)
plot(time,t3_1,'blue','LineWidth',1)
title('Temperature at Node 1','interpreter','latex','FontSize',14)
xlabel('Time (sec)','interpreter','latex','FontSize',12)
ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
grid on
grid minor
xlim([0,30000]);
ylim([24,38]);

subplot(1,3,2)
plot(time,t3_2,'blue','LineWidth',1)
title('Temperature at Node 2','interpreter','latex','FontSize',14)
```

```
xlabel('Time (sec)','interpreter','latex','FontSize',12)
ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
grid on
grid minor
xlim([0,30000]);
ylim([24,38]);

subplot(1,3,3)
plot(time,t3_3,'blue','LineWidth',1)
title('Temperature at Node 3','interpreter','latex','FontSize',14)
xlabel('Time (sec)','interpreter','latex','FontSize',12)
ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
grid on
grid minor
xlim([0,30000]);
ylim([24,38]);
```



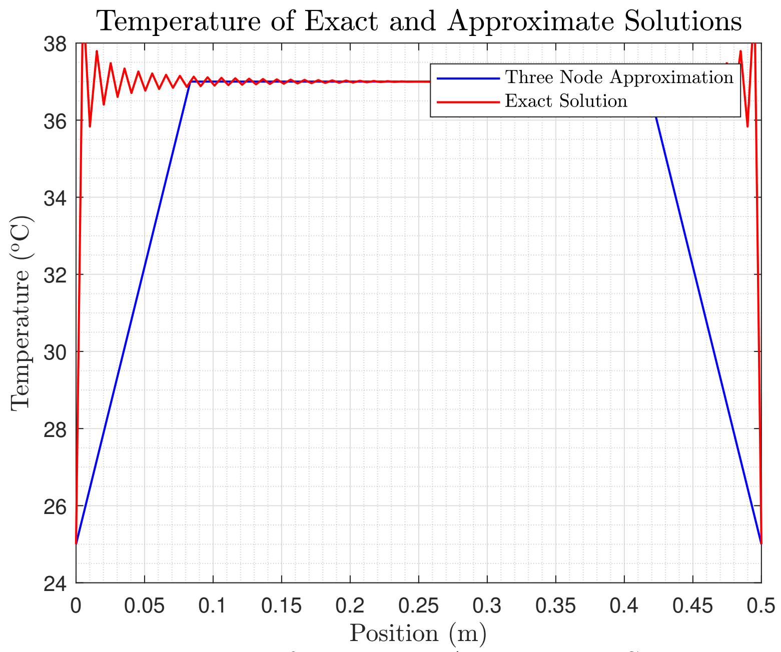
Part B Calculations

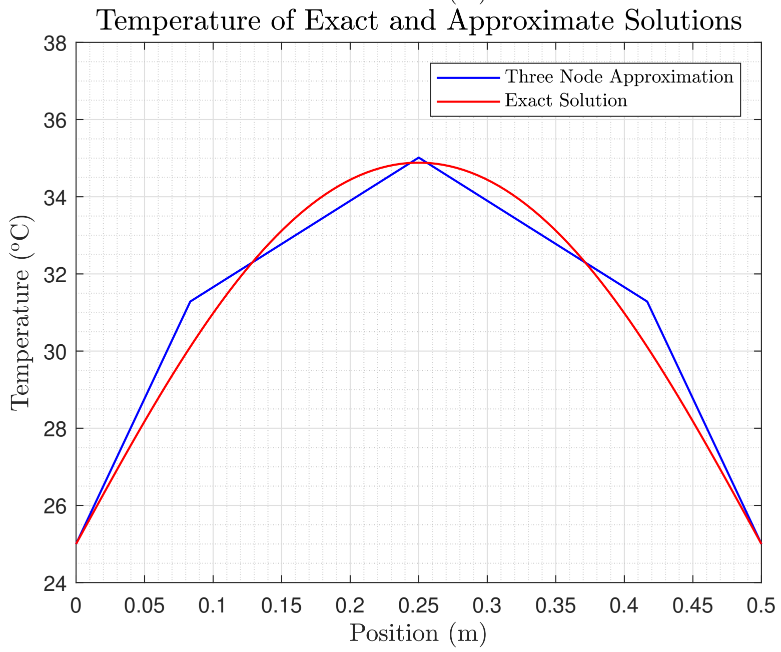
```
% Define range of position and time to consider
t_range = linspace(0,30000,10000);
x_range = linspace(0,0.5,100);

% Create a matrix with the Cartesian product of time and position
[T_range,X_range] = meshgrid(t_range,x_range);
TX = [T_range(:),X_range(:)];

% Initialize temperature vector
U = zeros((length(t_range)*length(x_range)),1);
```

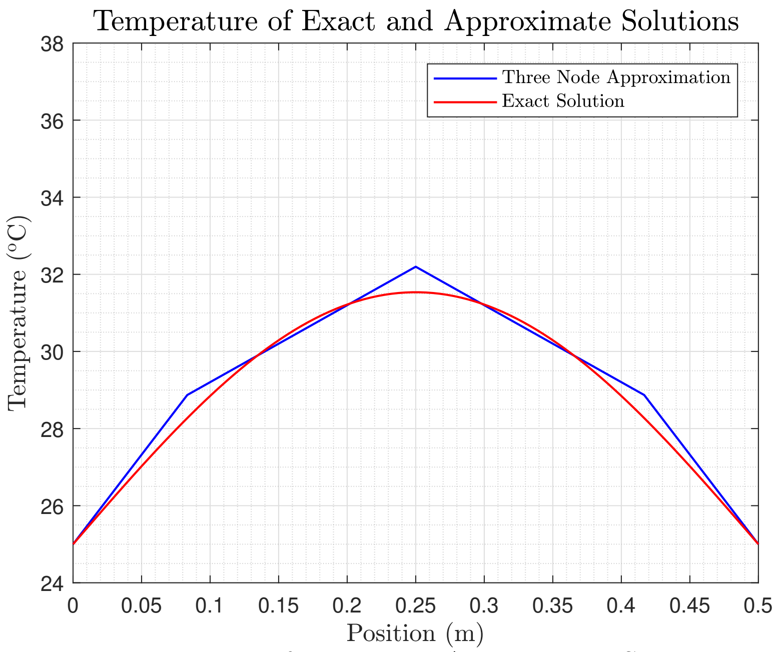
```
% Get "infinite" sum
for n = 1:100
    c_n = (2*(T_0-T_inf)*(1-((-1)^n)))/(n*pi);
    e_n = exp(((-1*alpha*n^2*pi^2)/(delta_x^2))*TX(:,1));
    s_n = sin(n*pi*TX(:,2)/delta_x);
    U = U + c_n.*e_n.*s_n;
end

% Convert to Celcius
U = U + T_inf - 273.15;

% Reshape vector into matrix
u = reshape(U,length(x_range),length(t_range));
```

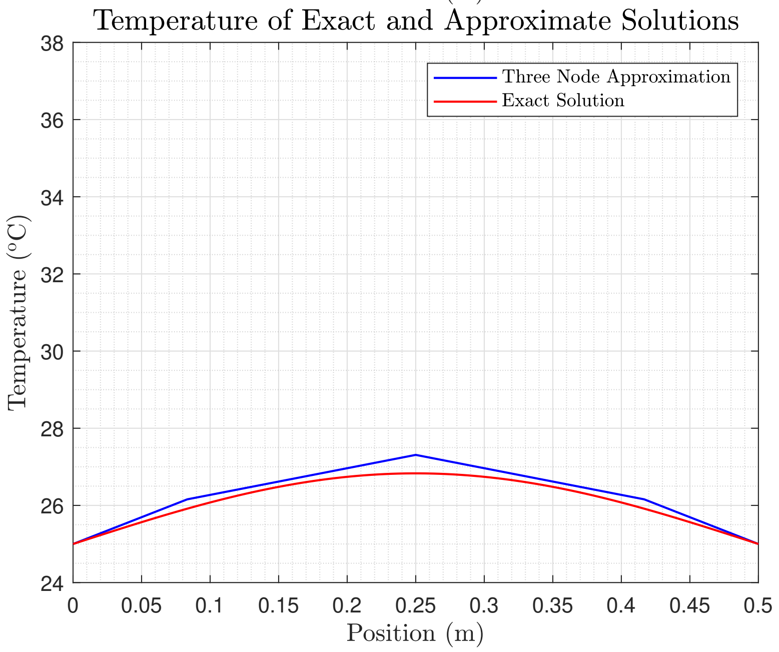
Part C Plots

```
% Position of the three nodes
x_range3 = [0, 1/12, 1/4, 5/12, 1/2];

% Plot temperature using both solutions at various times
for index = [1,1001,2001,5001]
    apx3 = [25, t3_1(index), t3_2(index), t3_3(index), 25];
    figure(index + 1)
    plot(x_range3,apx3,'blue','LineWidth',1);
    hold on
    plot(x_range, u(:,index),'red','LineWidth',1)
    hold off
    title('Temperature of Exact and Approximate Solutions','interpreter','latex','FontSize',14)
    xlabel('Position (m)','interpreter','latex','FontSize',12)
    ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
    legend_animation = legend('Three Node Approximation','Exact Solution');
    set(legend_animation,'Interpreter','latex');
    grid on
    grid minor
    xlim([0,delta_x]);
    ylim([24,38]);
    pause(1);
end
```


Temperature of Exact and Approximate Solutions
Three Node Approximation
Exact Solution
Temperature (°C)
Position (m)
38
36
34
32
30
28
26
24
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


Temperature of Exact and Approximate Solutions
Three Node Approximation
Exact Solution
Temperature (°C)
Position (m)
38
36
34
32
30
28
26
24
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5





Part D Calculations

```
% Number of nodes
n = 9;

% Define impedances that repeat
Z_R = z_r/(2*n);
Z_C = n*z_c;

% Define impedance matrix
Z(1,1) = Z_R + Z_C;
Z(1,2) = -1*Z_C;
for index = 2:n
```

```
    Z(index,index-1) = -1*Z_C;
    Z(index,index) = 2*Z_R + 2*Z_C;
    Z(index,index+1) = -1*Z_C;
end
Z(n+1,n) = -1*Z_C;
Z(n+1,n+1) = Z_R + Z_C;

% Define forcing vector
F(1,1) = (T_inf-T_0)/s;
F(n+1,1) = (T_0-T_inf)/s;

% Find flow rates
Q_dot = Z\F;

% Define temperature of nodes
T(1) = (T_inf/s) - (Q_dot(1)*Z_R);
for index = 2:n
    T(index) = T(index-1) - (2*Q_dot(index)*Z_R);
end

% Create matrix over time and distance
time_steps = 1000000;
time = linspace(0,30000,time_steps);
t = zeros(time_steps,n+2);
t(:,1) = T_inf;
t(:,n+2) = T_inf;
for index = 1:n
    t(:,index+1) = impulse(T(index),time);
end

% Define position vector
pos_frac = zeros(n+2,1);
pos_frac(1) = 0;
pos_frac(n+2) = 1;
for index = 1:n
    pos_frac(index+1) = (2*index - 1)/(2*n);
end
pos = pos_frac*delta_x;
```

Animation

```
% Plot temperature over distance
for index = 1:100:10000
    figure(4)
    plot(pos,t(index*100,:)-273.15,'blue',x_range, u(:,index),'red','LineWidth',1);
    title('Temperature of Exact and Approximate Solutions','interpreter','latex','FontSize',14)
    xlabel('Position (m)','interpreter','latex','FontSize',12)
    ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
    legend_animation = legend('Approximate Solutions','Exact Solution');
    set(legend_animation,'Interpreter','latex');
    grid on
    grid minor
    xlim([0,delta_x]);
    ylim([24,38]);
    video(((index-1)/100)+1) = getframe(gcf);
    drawnow
end

% Create video from plots
writerObj = VideoWriter('plots.avi');
writerObj.FrameRate = 10;

open(writerObj);
for index=1:length(video)
    % convert the image to a frame
    frame = video(index);
    writeVideo(writerObj, frame);
end
close(writerObj);
```

Contents

Homework 06 - ME342 - March 4, 2020

```
clear all; close all; clc;
```

Define parameters

```
s = tf('s');

% Boundary conditions
T_0 = 37 + 273.15; % K
T_inf = 25 + 273.15; % K

% Geometry
delta_x = 0.002; % m
A = 0.01; % m^2

% Thermal properties
k = 14.4; % W/mK
c = 502.416; % J/kgK
rho = 8000; % kg/m^3
C = rho*delta_x*A*c; % J/K

% Stainless steel 304
% Conductivity: https://www.engineeringtoolbox.com/thermal-conductivity-metals-d_858.html
% Specific Heat Capacitance: https://www.engineersedge.com/materials/specific_heat_capacity_of_metals_13259
% Density: http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=mq304a

% Define impedeances
z_r = delta_x/(k*A);
z_c = 1/(s*C);
```

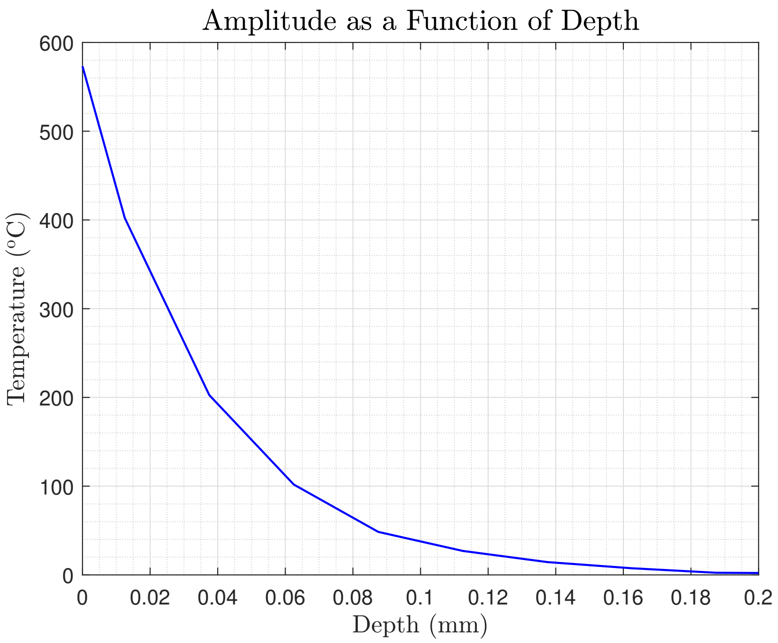
Calculations

```
% Define forcing temperature in s domain
T_osc = ((650+273.15)/s) + ((300+273.15)*s)/(s^2 + 256*pi^2);

% Number of nodes
n = 8;

% Define impedeances that repeat
```

```
Z_R = z_r/(2*n);
Z_C = n*z_c;

% Define impedeance matrix
Z(1,1) = Z_R + Z_C;
Z(1,2) = -1*Z_C;
for index = 2:n
    Z(index,index-1) = -1*Z_C;
    Z(index,index) = 2*Z_R + 2*Z_C;
    Z(index,index+1) = -1*Z_C;
end
Z(n+1,n) = -1*Z_C;
Z(n+1,n+1) = Z_R + Z_C;

% Define forcing vector
F(1,1) = T_osc- (T_0/s);
F(n+1,1) = (T_0-T_inf)/s;

% Find flow rates
Q_dot = Z\F;

% Define temperature of nodes
T(1) = (T_osc) - (Q_dot(1)*Z_R);
for index = 2:n+1
    T(index) = T(index-1) - (2*Q_dot(index)*Z_R);
end

% Create matrix over time and distance
time_steps = 10000;
time = linspace(0,1,time_steps);
t = zeros(time_steps,n+2);
t(:,1) = impulse(T_osc,time);
for index = 1:(n+1)
    t(:,index+1) = impulse(T(index),time);
end

% Define position vector
pos_frac = zeros(n+2,1);
pos_frac(1) = 0;
pos_frac(n+2) = 1;
for index = 1:n
    pos_frac(index+1) = (2*index - 1)/(2*n);
end
pos = pos_frac*delta_x;
```

Find Amplitude

```
% Initialize amplitude vector
A = zeros(length(pos),1);

% Find amplitude at steady-state
for index = 1:length(pos)
    A(index) = (max(t(5000:6000,index)) - min(t(5000:6000,index)))/2;
end

figure(1)
plot(pos*100,A,'blue','LineWidth',1)
title('Amplitude as a Function of Depth','interpreter','latex','FontSize',14)
xlabel('Depth (mm)','interpreter','latex','FontSize',12)
ylabel('Temperature ($^{\mathrm o}$C)','interpreter','latex','FontSize',12)
grid on
grid minor
xlim([0,0.2]);
ylim([0,600]);
```


Amplitude as a Function of Depth
Temperature ($^{\mathrm o}$C)
Depth (mm)