Lee Sutton - 48507107

Mech 358 – Computer Lab 4

Problem 1

The m-file for this problem is shown below. Please note that it was taken from computer lab 3 so all of the analytical solutions were commented out. A plot of the unstable solution is shown below in Figure 1.

```
%This matlab code will calculate the solution to the heat equation using
%finite differences
clear
clc

%define x vector from 0 to 1 and we want 11 points
x = 0:0.1:1;

%define the x index counter
j = 1;

%define time step and x step
dt = .006;
dx = 0.1;

%number of points in x vector
N = 11;

%define constants
time = 1;

%define temporary temperatures so we dont have store the temperature
%profile for 1000 timesteps
T_temp1 = zeros(1,N);
T_temp2 = zeros(1,N);

%create the matrix to hold the Temperature profile
T = zeros(5,N);

%Set initial conditions T = 1, at time = 0, for all x
T(1,:) = 1;

%set boundary conditions T = 0, at x=0 and x=1, for t>0
T(2,1) = 0;
T(2,N) = 0;

%initialize the temporary temperature with the next time step
for j = 2:N-1

T_temp1(j) = T(1,j) + dt/(dx^2)*( T(1,j+1) - 2*T(1,j) + T(1,j-1)  );
```

```
end

%create a for loop to calculate the temp at the next time steps, for 1000
%timesteps
for timestep = 1:15


    %we have to calculate the T_temp 2 for all x
    for j = 2:N-1

        T_temp2(j) = T_temp1(j) + dt/(dx^2)*( T_temp1(j+1) - 2*T_temp1(j) +
T_temp1(j-1)  );

    end

    %check to see if we should store this data, we only store it 5 times
    if timestep == 4 || timestep == 5 || timestep == 6 || timestep == 10

        %add 1 to time so we can store the next value for temperature
        time = time +1;

        %store this temperature profile at the specified time
        T(time,:) = T_temp2(:)

    end

    %now we replace T_temp1 with T_temp2 and run the loop agian so that we
    %only have to store a limited number of temperature profiles
    T_temp1 = T_temp2;


end


%now we plot the temperature profile vs x at various times with stars
%at time =0
% plot(x,T(1,:),'k')


%at time = .024
plot(x,T(2,:),'r')
hold on

%increase the y limit of our plot
axis([0 1 0 1.2])

%at time = 0.030
plot(x,T(3,:))

%at time = 0.036
plot(x,T(4,:),'g')

%at time = .040s
```

```
plot(x,T(5,:),'m')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %Now we plot the exact solutions found using the fourier series
%
% %define a larger x interval to get a smoother fourier curve
% x = linspace(0,1);
%
% %define a sum for the partial sum initially equal to zero
% Sum_1 = 0;
%
% %use a for loop to evaluate the first 100 terms first at t = 0
% t=0;
%
% for n = 1:100
%
%     Sn = 4/pi* exp( -1*(2*n-1)^2*pi^2*t ) * sin( (2*n-1)*pi*x ) / (2*n-1);
%
%     Sum_1 = Sum_1 + Sn;
%
% end
%
% %plot this partial sum vs. x as a solid black line
% plot(x,Sum_1,'k','linewidth',2)
%
%
%
% %create another fourier series for t=0.1
% %set Sn = 0 again
% Sn=0;
%
% %define the sum for the partial sum initially equal to zero
% Sum_2 = 0;
%
% %use a for loop to evaluate the first 100 terms first at t = 0.05
% t=0.05;
%
% for n = 1:100
%
%     Sn = 4/pi* exp( -1*(2*n-1)^2*pi^2*t ) * sin( (2*n-1)*pi*x ) / (2*n-1);
%
%     Sum_2 = Sum_2 + Sn;
%
% end
%
%
% plot(x,Sum_2,'r','linewidth',2)
%
%
%
% %create another fourier series for t=0.1
% %set Sn = 0 again
% Sn=0;
%
```

```
% %define the sum for the partial sum initially equal to zero
% Sum_3 = 0;
%
% %use a for loop to evaluate the first 100 terms first at t = 0.05
% t=0.1;
%
% for n = 1:100
%
%     Sn = 4/pi* exp( -1*(2*n-1)^2*pi^2*t ) * sin( (2*n-1)*pi*x ) / (2*n-1);
%
%     Sum_3 = Sum_3 + Sn;
%
% end
%
%
% plot(x,Sum_3,'linewidth',2)
%
%
%
%
% %create another fourier series for t=0.2
% %set Sn = 0 again
% Sn=0;
%
% %define the sum for the partial sum initially equal to zero
% Sum_4 = 0;
%
% %use a for loop to evaluate the first 100 terms first at t = 0.05
% t=0.2;
%
% for n = 1:100
%
%     Sn = 4/pi* exp( -1*(2*n-1)^2*pi^2*t ) * sin( (2*n-1)*pi*x ) / (2*n-1);
%
%     Sum_4 = Sum_4 + Sn;
%
% end
%
%
% plot(x,Sum_4,'g','linewidth',2)
%
%
%
% %create another fourier series for t=1
% %set Sn = 0 again
% Sn=0;
%
% %define the sum for the partial sum initially equal to zero
% Sum_5 = 0;
%
% %use a for loop to evaluate the first 100 terms first at t = 0.05
% t=1;
%
% for n = 1:100
%
%     Sn = 4/pi* exp( -1*(2*n-1)^2*pi^2*t ) * sin( (2*n-1)*pi*x ) / (2*n-1);
%
```

```
%      Sum_5 = Sum_5 + Sn;
%
% end

% plot(x,Sum_5,'m','linewidth',2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




%add titles and labels to the graph along with a legend
xlabel('X')
ylabel('T')
title('Temperature vs. X For the Transient Heating Problem with an Unstable
Time Step')

%legend
legend('t=0.024','t=0.03','t=0.036','t=.04')
```

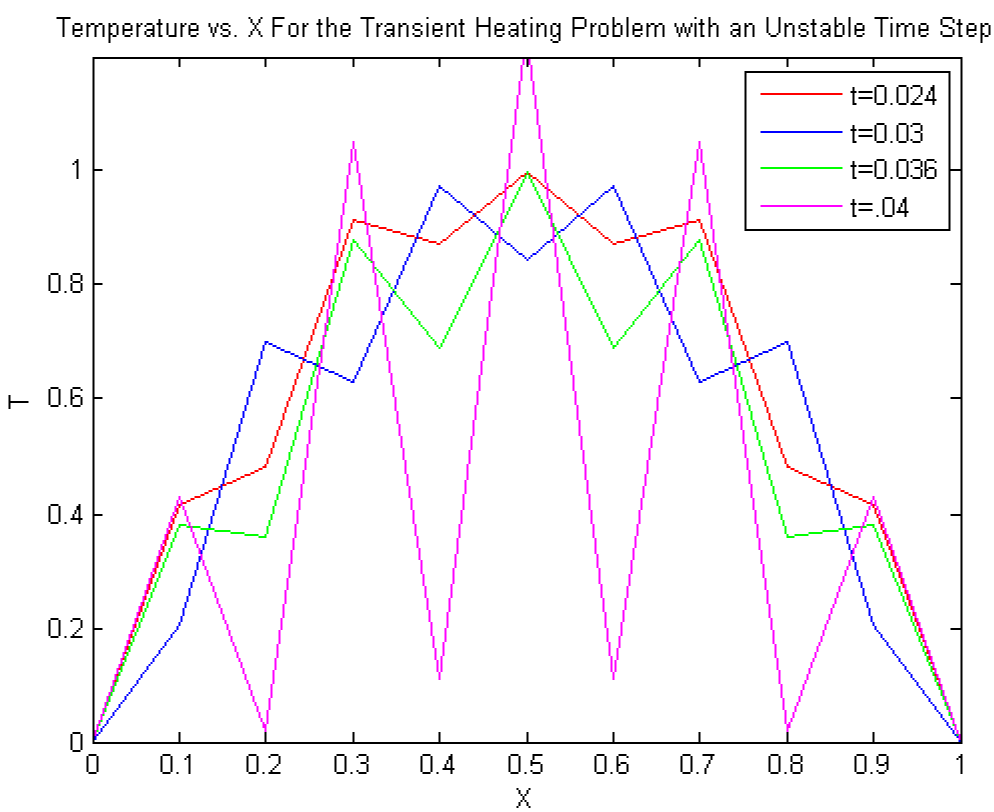

Figure 1 Transient heating solution with an unstable time step

Lee Sutton - 48507107

Problem 2

Case 1:

The plot of temperature vs. x for various times is shown below in Figure 2. A plot of the temperature vs. time is shown below in Figure 3. It shows that the oscillation dies out with distance. We can see from Figure 3 that the amplitude of oscillation is much lower for x=0.5 and x=0.8 when compared with the amplitude at x=0 which is 0.7.

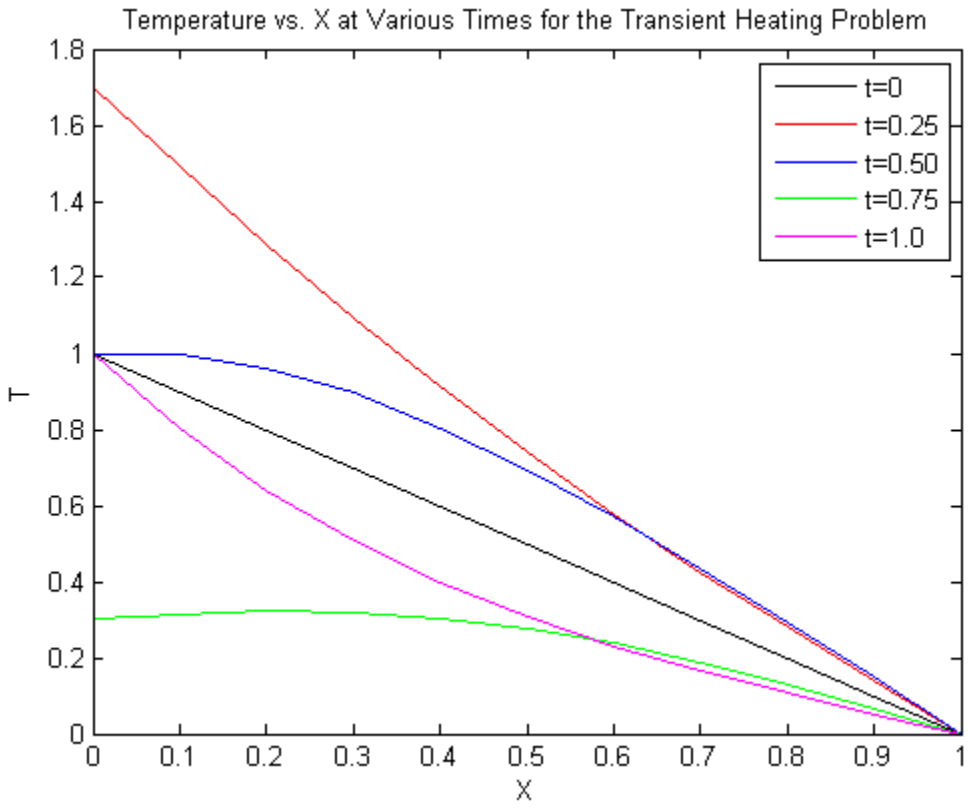

Figure 2 Temperature vs. X for various times

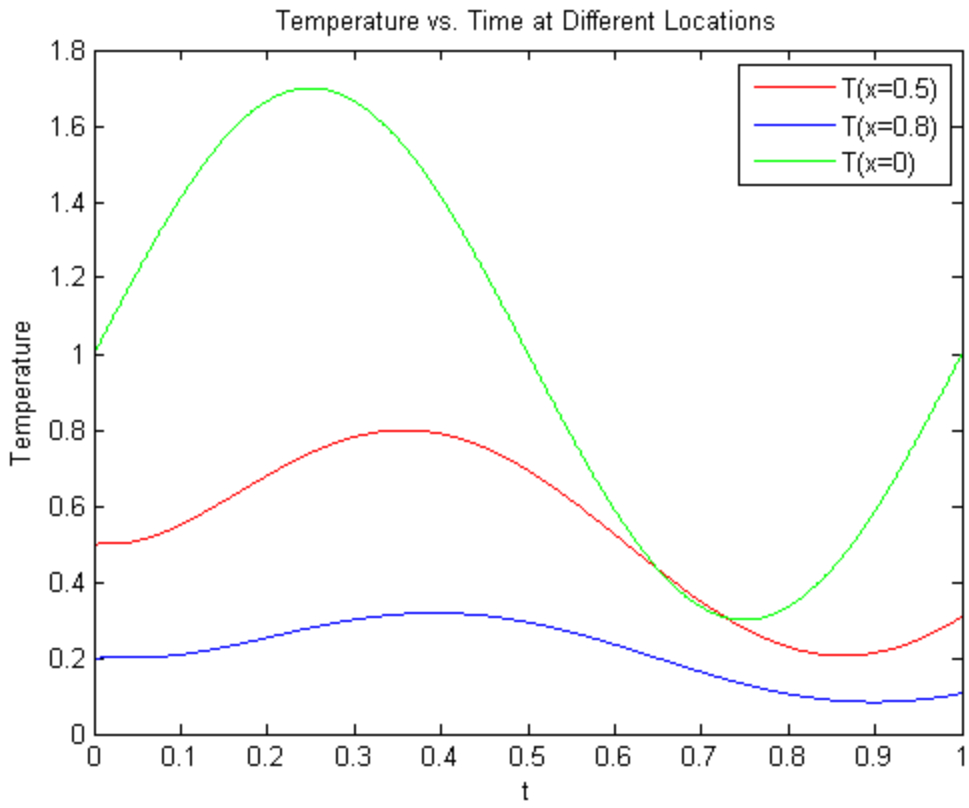

Figure 3 Temperature vs. Time at various locations

The Matlab code used to create these figures is shown below.

```
%This matlab code will calculate the solution to the heat equation using
%finite differences
clear
clc

%define x vector from 0 to 1 and we want 11 points
x = 0:0.1:1;

%define a vector to save the temperatures at x=0.5 and x=0.8
T_x5 = zeros(1,1001);
T_x8 = zeros(1,1001);

%define the x index counter
j = 1;

%define time step and x step
dt = .001;
dx = 0.1;

%number of points in x vector
```

```
N = 11;

%define constants
time = 1;

%define temporary temperatures so we dont have store the temperature
%profile for 1000 timesteps
T_temp1 = zeros(1,N);
T_temp2 = zeros(1,N);

%create the matrix to hold the Temperature profile
T = zeros(5,N);

%Set initial conditions T = f(x) = 1-x, at time = 0, for all x
T(1,:) = 1-x;

%save this to Temperature at x=0.5 and x=0.8 so we can plot this later
T_x5(1) = T(1,6);
T_x8(1) = T(1,9);

%initialize the temporary temperature with the next time step
for j = 2:N-1

T_temp1(j) = T(1,j) + dt/(dx^2)*( T(1,j+1) - 2*T(1,j) + T(1,j-1)  );

end

%set boundary conditions T = g(t) at x=0; T = 0 x=1, for t>0
T_temp1(1) = 1 + 0.7*sin(2*pi*dt);
T_temp1(N) = 0;

%save this to Temperature at x=0.5 and x=0.8 so we can plot this later
T_x5(2) = T_temp1(6);
T_x8(2) = T_temp1(9);

%create a for loop to calculate the temp at the next time steps, for 1000
%timesteps we have already taken 1 timestep so we take 999 more

for timestep = 2:1000


    %we have to calculate the T_temp 2 for all x
    for j = 2:N-1

        T_temp2(j) = T_temp1(j) + dt/(dx^2)*( T_temp1(j+1) - 2*T_temp1(j) +
T_temp1(j-1)  );

    end

    %set boundary conditions T = g(t) at x=0; T = 0 x=1, for t>0
    T_temp2(1) = 1 + 0.7*sin(2*pi*timestep*dt);
    T_temp2(N) = 0;
```

```
    %save the temperature at x = 0.5 and x = 0.8 so this can be plotted later
    T_x5(timestep+1) = T_temp2(6);
    T_x8(timestep+1) = T_temp2(9);


    %check to see if we should store this data, we only store it 5 times
    if timestep == 249 || timestep == 499 || timestep == 749 || timestep ==
999

        %add 1 to time so we can store the next value for temperature
        time = time +1;

        %store this temperature profile at the specified time
        T(time,:) = T_temp2(:);

    end

    %now we replace T_temp1 with T_temp2 and run the loop agian so that we
    %only have to store a limited number of temperature profiles
    T_temp1 = T_temp2;


end


%now we plot the temperature profile vs x at various times with stars
%at time =0
plot(x,T(1,:),'k')
hold on

%at time = .25
plot(x,T(2,:),'r')

%at time = 0.5
plot(x,T(3,:))

%at time = .75
plot(x,T(4,:),'g')

%at time = 1.0
plot(x,T(5,:),'m')


%add titles and labels to the graph along with a legend
xlabel('X')
ylabel('T')
title('Temperature vs. X at Various Times for the Transient Heating Problem')

%legend
legend('t=0','t=0.25','t=0.50','t=0.75','t=1.0')


%plot the temperature at x = 0.8 on a new graph
```

```
figure
%define a time vector to plot against
t = 0:dt:1;

%plot the temperature at x = 0.5 in red
plot(t,T_x5,'r')
hold on

%plot the temperature at x = 0.8
plot(t,T_x8)

%compare these to the temp profile at x = 0
T_x0 = 1 + 0.7*sin(2*pi*t);
%apply initial condition
T_x0(1) = 1;

%plot this as well
plot(t,T_x0,'g')

%add labels to the plot
xlabel('t')
ylabel('Temperature')
legend('T(x=0.5)','T(x=0.8)','T(x=0)')
title('Temperature vs. Time at Different Locations')
```

Case 2

The plot of the temperature vs. x is shown below in Figure 4. We can see from Figure 5 that the temperature at the centre of the bar has reached steady state. The percentage heat loss at the centre of the rod is shown below in Table 1. The m-file used to solve this problem is shown below after the figures.

Table 1 Heat loss at the centre of the rod

Heat Loss Percentage	Temperature at the Centre
26.80%	0.7320

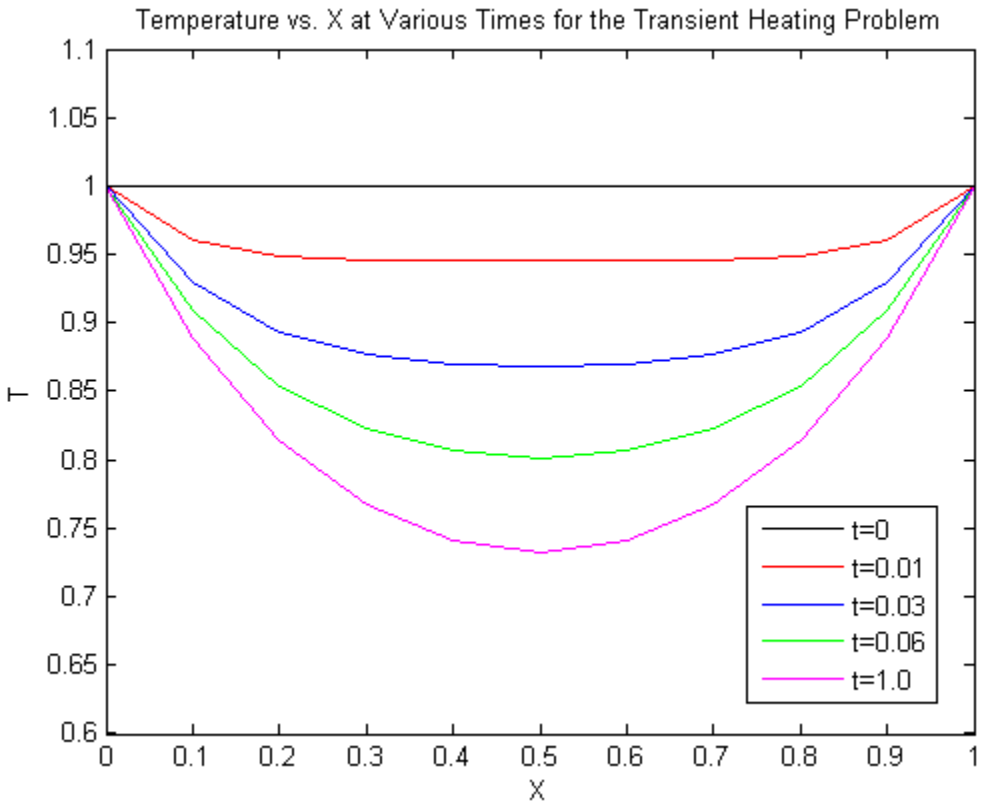

Figure 4 Transient heating problem for case 2

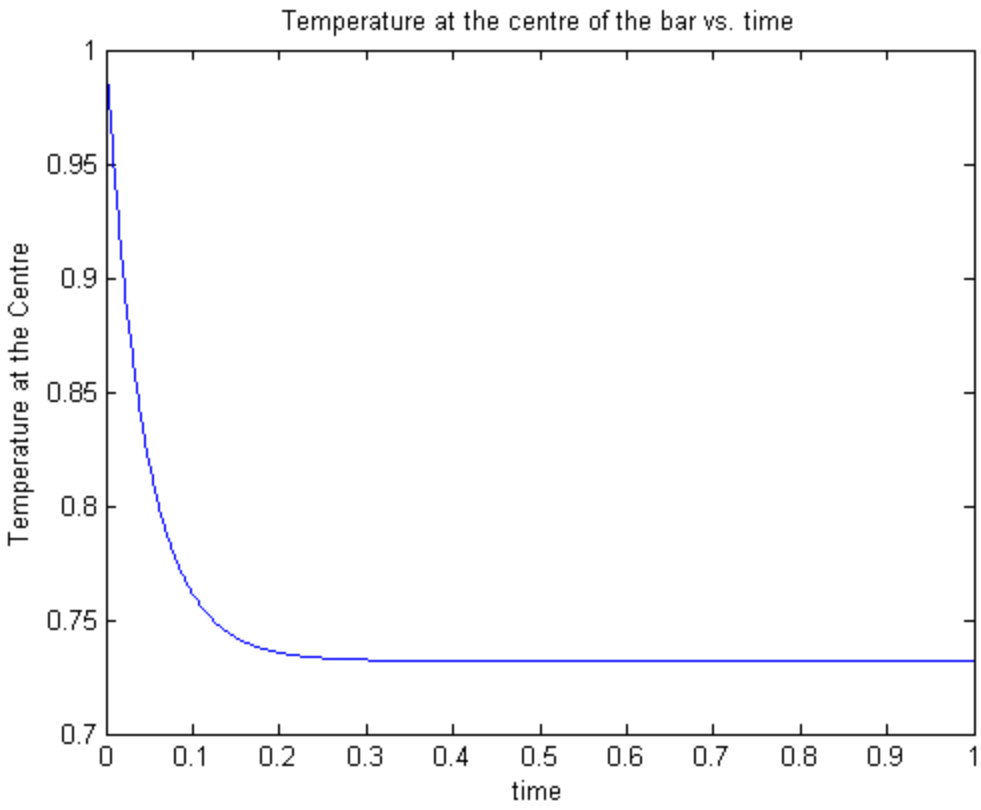

Figure 5 Temperature at the centre of the bar vs. time

```
%This matlab code will calculate the solution to the heat equation using
%finite differences
clear
clc

%define x vector from 0 to 1 and we want 11 points
x = 0:0.1:1;

%define the x index counter
j = 1;

%define time step and x step
dt = .001;
dx = 0.1;

%number of points in x vector
N = 11;

%define constants
time = 1;

%define temporary temperatures so we dont have store the temperature
```

```
%profile for 1000 timesteps
T_temp1 = zeros(1,N);
T_temp2 = zeros(1,N);

%create the matrix to hold the Temperature profile
T = zeros(5,N);

%define a matrix to hold the temperature at the centre of the bar for 1
%intial condition and then 1000 time steps ahead
T_centre = zeros(1001,1);

%Set initial conditions T = f(x) = 1, at time = 0, for all x
T(1,:) = 1;

%save the temperature at the centre of the bar
T_centre(1) = T(1,6);
%initialize the temporary temperature with the next time step
for j = 2:N-1

T_temp1(j) = T(1,j) + dt/(dx^2)*( T(1,j+1) - 2*T(1,j) + T(1,j-1)  ) + dt*-
6*T(1,j)^4;

end

%save the temperature at the centre of the bar
T_centre(2) = T_temp1(6);

%set boundary conditions T = g(t) at x=0; T = 0 x=1, for t>0
T_temp1(1) = 1;
T_temp1(N) = 1;

%create a for loop to calculate the temp at the next time steps, for 1000
%timesteps we have already taken 1 timestep so we take 999 more

for timestep = 2:1000


    %we have to calculate the T_temp 2 for all x
    for j = 2:N-1

        T_temp2(j) = T_temp1(j) + dt/(dx^2)*( T_temp1(j+1) - 2*T_temp1(j) +
T_temp1(j-1)  ) + dt*-6*T_temp1(j)^4;

    end

    %set boundary conditions T = g(t) at x=0; T = 0 x=1, for t>0
    T_temp2(1) = 1;
    T_temp2(N) = 1;

    %save the temperature at the centre of the bar
    T_centre(timestep+1) = T_temp2(6);
```

```
    %check to see if we should store this data, we only store it 5 times
    if timestep == 10 || timestep == 30 || timestep == 60 || timestep == 1000

        %add 1 to time so we can store the next value for temperature
        time = time +1;

        %store this temperature profile at the specified time
        T(time,:) = T_temp2(:);

    end

    %now we replace T_temp1 with T_temp2 and run the loop agian so that we
    %only have to store a limited number of temperature profiles
    T_temp1 = T_temp2;


end


%now we plot the temperature profile vs x at various times with stars
%at time =0
plot(x,T(1,:),'k')
hold on

%at time = .25
plot(x,T(2,:),'r')

%at time = 0.5
plot(x,T(3,:))

%at time = .75
plot(x,T(4,:),'g')

%at time = 1.0
plot(x,T(5,:),'m')


%add titles and labels to the graph along with a legend
xlabel('X')
ylabel('T')
title('Temperature vs. X at Various Times for the Transient Heating Problem')

%legend
legend('t=0','t=0.01','t=0.03','t=0.06','t=1.0')

%adjust the axis to make it fit
axis([0 1 0.6 1.1])

%for the percentage of temperature lost at its centre
Temp_centre = T(5,6)

%for percentage lost
percentage_loss = 1-Temp_centre
```

```
%plot the temperature at the centre of the bar vs. time
%define a time vector from 0 to 1
t = 0:dt:1;

%plot the temp at the centre vs time in a new figure
figure
plot(t,T_centre)

%add labels
ylabel('Temperature at the Centre')
xlabel('time')
title('Temperature at the centre of the bar vs. time')
```

Problem 3

A plot of the finite difference solution, the Gauss Seidel solution, and the Fourier series solution is shown below in Figure 6. The Matlab code used to solve this problem is shown on the following page.

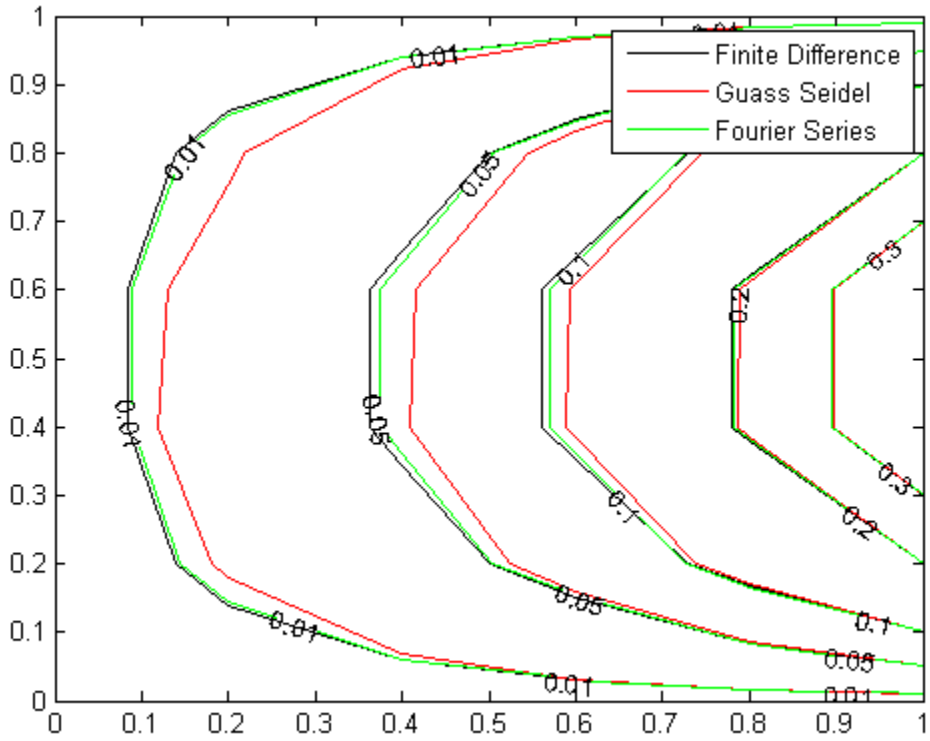

Figure 6 Plot of the solution Laplace's equation using three different methods to solve

We can see from Figure 6 that only a small discrepancy exists between the finite difference solution solved using backslash (black line) and the analytical solution found using the Fourier Series (green line). This small error is likely due to the truncation error associated with the finite difference method. The finite differences have error that is 2^{nd} order in y and 2^{nd} order in x. This discrepancy is likely due to this truncation error.

However, we notice a larger discrepancy between the Gauss Seidel method and the other solutions. Since the iteration was started by assuming that all interior points were 0, the iterative method of Guass Seidel was not given enough iterations to reach the exact solution. Since only 7 iterations were conducted, this error is expected and in the future this error can easily be reduced by using more iterations or by using a relaxation method.

Lee Sutton - 48507107

```
%this m-file will create the matrix that will solve the finite difference
%solution using backlash and using Gauss seidel methods, it will also solve
%the problem using the exact solution produced by forming a fourier series
%solution
clear all
close all
clc


%backslash method
%set up our mesh
x = linspace(0,1,6);
y = linspace(0,1,6);

%define the number of points in each row
N = 6;

%set up the solution vector
b= zeros(36,1);

%set up a vector for the interior points
phi_vector = zeros(36,1);

%create the temperature matrix
temp_matrix = zeros(6,6);

%define the non-zero terms
b(12) = y(2);
b(18) = y(3);
b(24) = 1-y(4);
b(30) = 1-y(5);

%load the matrix A
load A.mat;

%solve the system using backslash
phi_vector = A\b;

%define a counter for the vector
k=1;

%convert this into the matrix moving from left to right, bottom to top
for i = 1:N

    %take 6 rows of the phi vector and put it into the matrix row
    temp_matrix(i,1:N) = phi_vector(k:k+N-1);

    %add an interaval to the k value
    k = k+N;

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%now for the guass seidel method

%create the phi matrix
phi_gauss = zeros(6,6);

%apply boundary conditions
%only non-zero terms occur at x = 1
phi_gauss(N-1,N) = y(2);
phi_gauss(N-2,N) = y(3);
phi_gauss(N-3,N) = 1 - y(4);
phi_gauss(N-4,N) = 1 - y(5);

%apply the gauss seidel method 7 times
for guass = 1:7

    %sweep from top to bottom
    for i = 1:N-2

        %sweep from left to right
        for j = 2:N-1

           phi_gauss(N-i,j) = 1/4*( phi_gauss(N-(i+1),j) + phi_gauss(N-(i-
1),j) + phi_gauss(N-i,j+1) + phi_gauss(N-i,j-1)   );

        end

    end




end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%now for the exact solution using a fourier series

%create a meshgrid for x and y
[X,Y] = meshgrid(x,y)

%define a vector to hold the partial sums
sum = zeros(6,6)

%define a vector to temporarily hold the current partial sum
phi_fourier = zeros(6,6)

%sum the first 20 terms of the fourier series
for n = 1:20

    bn = 4/( (n^2*pi^2)*sinh(n*pi) ) * sin(n*pi/2);
```

```
    phi_fourier = bn*sin(n*pi*Y).*sinh(n*pi*X);

    sum = sum + phi_fourier;
end


%Plotting the contours
C=[0.01 0.05 0.1 0.2 0.3];

%create a for loop to plot it at each interval
for i = 1:5

    %plot the temperature matrix at the current contour level in black
    contour(x,y,temp_matrix,C(i),'k');

    %keep the plot open
    hold on

    %plot the gauss seidel method
    contour(x,y,phi_gauss,C(i),'r');

    %plot the fourier method in gree stars
    [a,b] = contour(x,y,sum,C(i),'g');

    %add the labels
     clabel(a,b);

end


%add a legend
legend('Finite Difference','Guass Seidel','Fourier Series')
```