TCNJ THE COLLEGE OF NEW JERSEY

School of Engineering

ENG342-02 **Fall 2020**

Matlab Assignment 2

Date: 10/29/2020

Submitted by: Jeffrey Blanda

Instructor: Wieslaw Marszalek

Introduction: Matlab was used to create multiple graphs to display the solution to the heat equation as per problem 3 on page 718 for the 6th edition Advanced Engineering Mathematics textbook. This assignment was an exercise to show the applications of solving the heat equation PDE and displaying the solution as real world data.

Temperature *u* as a function of time, *t*, and distance, *x*:

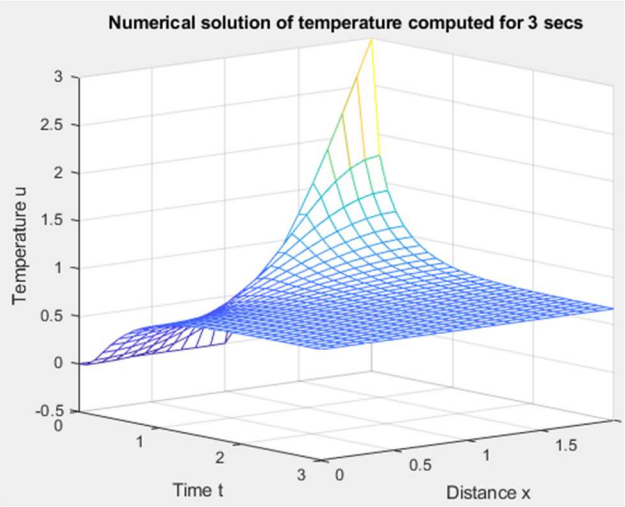

Figure 1. 3d MatLab Plot for u(x,t)

We can see from the plot that the boundary conditions were satisfied as when $0 < x < 1$ the value of the function is 0 and when $1 < x < 2$ the value of the function is $x^2 - 1$. It can also be observed from the plot that the temperature quickly nears and as t approaches 2.5, the difference in temperature from one end of the rod to the other is less than 0.1.

Temperature as a function of x at various fixed times:

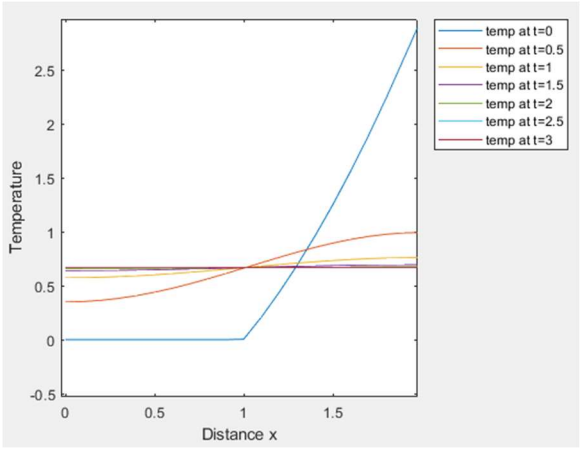

Figure 2. MatLab Plot for Temp as a function of x at different times

This plot displays the temperature of the rod over 0.5 second intervals. Once again, we can observe that at $t = 0$ the function exactly matches $f(x)$ and that the temperature of the rod equalizes as time passes.

Temperature as a function of t at various fixed positions:

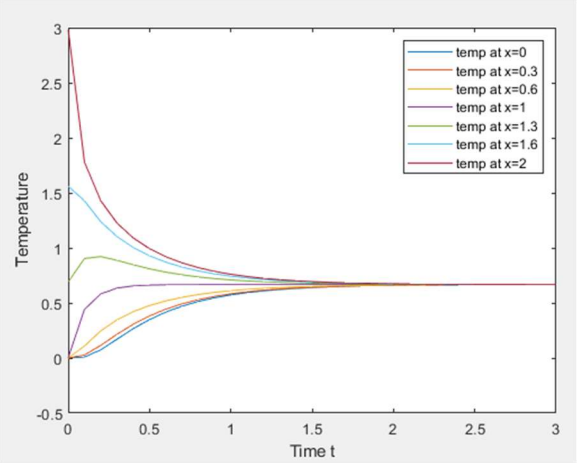

Figure 3. MatLab Plot for Temp as a function of t at various positions

This plot shows the equalization of temperatures of specific positions over time. Notable that at $t = 2.5$ the lines are indistinguishable.

Impact of resolution and number of partial sums on the graph:

Increasing the resolution of time or distance, we can increase the accuracy of the graph at the expense of runtime. For example, when $n = 100, S_x = 0.1, S_t = 0.1$ (where S represents the increase of its subscript from its initial value to its final value) The runtime of the script is 0.082442 seconds. If we were to decrease $S_x = 0.001$, we gain 10x the resolution when it comes to distance, but the runtime increases to 0.126016 seconds. Likewise, if we increase S_x back to 0.1 and decrease S_t to 0.001, the script now takes 0.096687 seconds to run. The benefits of decreasing the magnitude of these increments are that we can observe more values on the plot.

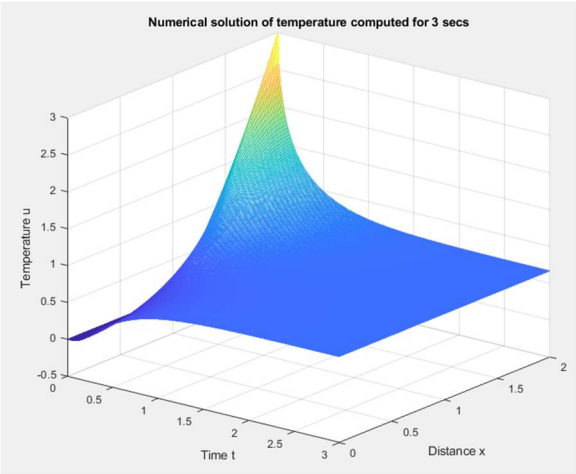

Figure 4. MatLab Plot for Temp with $S_x = 0.01 \; and \; S_t = 0.01$

Compared to the original plot, this one has a great magnitude more points. Doing this however does not increase the accuracy at the points previously computed (ex. $t = 1, x = 1.5$ has the same value). In order to increase the accuracy, the max value of n must be increased. Recall that when writing the solution to the PDE an infinite summation is used. This is because the function only matches the solution exactly when that function is computed all the way to $n = \infty$. Therefore, increasing the max value of n brings the computed value closer to the actual solution.

At $n = 100, S_x = 0.01 \; and \; S_t = 0.01$, the runtime of the program was 0.092197 seconds while at $n = 10000$ with no other changes, runtime increased to 29.634093 seconds.

Appendix:

```
%This script is meant to simulate the temperature on a thin rod subject to
%the conditions given in exercise 3 on page 718
clear; %clear saved variables
tic %start time counter
t = 0:0.1:3; %set time matrix
x = (0:0.1:2)'; %set position matrix
u = zeros(length(x), length(t)); %give initial size of u to match x by t
L = 2; %Given
k = 1; %Given
A0 = 2/3;
u = u+A0;
%Using for loop to replace infinite summation.
for n=1:100
    %Using MatLab to calculate the value of An
    S = @(x) ((x.^2)-1).*cos(n*pi*x/L);
    An = integral(S,1,2);
   a = exp(-k*pi*pi*n*n*t/(L*L));
   b = cos(pi*n*x/L);
   u = u + An*a.*b;
end
%3d Plot (x=t)(y=x)(z=u)
figure(1);
mesh(t,x,u)
title('Numerical solution of temperature computed for 3 secs')
xlabel('Time t')
ylabel('Distance x')
zlabel('Temperature u')
%Worth noting: For the following, u(x,:) x = 1+value/step where step is 0.1
%in this case.
%Output 2d plot for Temperature in respect to x for different times
figure(2);
plot(x,u(:,1),x,u(:,6),x,u(:,11),x,u(:,16),x,u(:,21),x,u(:,26),x,u(:,31));
legend('temp at t=0','temp at t=0.5','temp at t=1','temp at t=1.5','temp at t=2','temp at t=2.5','temp at t=3')
xlabel('Distance x');
ylabel('Temperature');
%Output 2d plot for Temperature in respect to t for different positions
figure(3);
plot(t,u(1,:),t,u(4,:),t,u(7,:),t,u(11,:),t,u(14,:),t,u(17,:),t,u(20,:));
legend('temp at x=0','temp at x=0.3','temp at x=0.6','temp at x=1','temp at x=1.3','temp at x=1.6','temp at x=2')
xlabel('Time t');
ylabel('Temperature');
toc %output how long the execution took
```

Appendix A. Code for the function to display the 3d plot and two 2d plots of temperature in respect to time and distance, respectively.

References

Zill, Dennis G. Advanced Engineering Mathematics (6th Edition), ISBN: 978-1-284-10590-2, Jones and Bartlett Learning, 2017.