# Comp 670 – Problems in Computational Science

H.W. # 1

## 1. Introduction

The Heat Equation is a Partial differential equation (PDE) which model the distribution of heat over a region. Heat Equation can be solved using separation of variables. However, many PDE's cannot be solved exactly and one needs to turn to numerical solutions to get the exact solution at every point in time/space.

In this assignment, I have solved 1D Heat Equation using Numerical scheme. This report includes brief introduction to the methodology used in solving heat equation numerically, Matlab code, visual representation of heat equation using plots and graphs and lastly, some observations and conclusion.

## 2. Methodology

The heat equation is a parabolic PDE and is a time dependent problem. Here we will use a scheme called FTCS scheme (Forward-Time Central-Space). In FTCS, we use Forward difference (Right difference) and Central difference to calculate the value at given point in the grid.

$$u(t) = cu(xx)$$

Since, heat equation is a second order in space and first order in time. We solve second order factor in space using forward difference first and then backward difference (which is also called Central approx.). For the first order factor in time, we simply use forward difference method.

Solving and combining these two factors we get an equation,

u(i,n+1) = u(i,n) + c*k/h^2 (u(i+1,n) + u(i-1,n) – 2*u(i,n))

Using this equation we can easily find value at any point u(i,n+1) provided we know value of u(i,n), u(i-1,n) and u(i+1,n).

I then used Matlab to write solution for this equation, which can model and simulate 1D heat equation.

## 3. Matlab Code

```
% 1D Heat equation using FTCS scheme
% (Forward-Time Central-Space)

% Initializing given variables/ data
% stability constant (c), dt = timesteps (k),
% dx = space steps (h),
% cfl = stability factor, it should be <= 1/2

clear;                    % clear the screen
c = .25;
dx = 0.1;
dt = 0.02;
cfl = c * dt/dx^2;

x = 0:dx:1;
t = 0:dt:2-dt;
row = length(t);          % number of rows in matrix u
col = length(x);           % number of col in matrix u

% Matrix with 100 rows and 10 col
% All elements are zero initally

u = zeros(row,col);
```

```
% Initial Condition, u(x,0) = f(x) = 100
% All elements in 1st row are 100

for k = 1:length(x)
     u(1,k) = 100;
end

% Boundary Conditions, u(t,0) = 0, u(t,1) = 0

u(:,1) = 0;
u(:,end) = 0;

% Loop to calculate u(n+1,i) given three values u(i,n),
% u(n,i+1) and u(n,i-1) from previous row.

for n = 1:row-1                              % row = 100
    for i = 2:col-1                          % col = 10
        u(n+1,i) = u(n,i) + cfl*u(n,i+1) + cfl*u(n,i-1) - 2*cfl*u(n,i);
    end
end


% Plotting the graph for every step using for loop.

steps = row;
for step = 1:steps
    plot(x,u(step,:))
    xlim([0 1])                                  % limits for x-axis
    ylim([0 2])                                  % limits for y-axis
    title('Numerical Solution for 1D Heat Equation')  % Title
    xlabel('Space')                               % x-axis label
    ylabel('Time')                                % y-axis label
    grid on
    drawnow
end

imagesc(u)                   % Image with scaled colors in xy plane
% surf(u)                    % Surface plot
```

# 4. Results

## 4.1. Plot

Following three plots show diffusion of heat over a region, where time (y-axis) goes from 0 to 2 and distance (x-axis) goes from 0 to 1. A continuous diffusion behavior can be seen by running Matlab code.

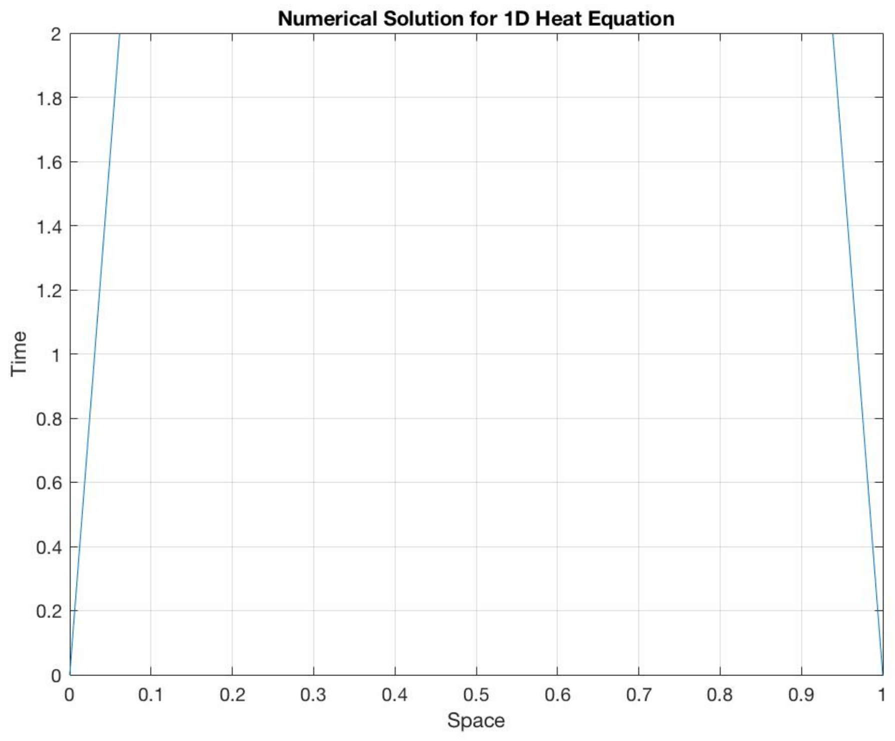

Figure 1: Heat equation numerical solution plot 1.

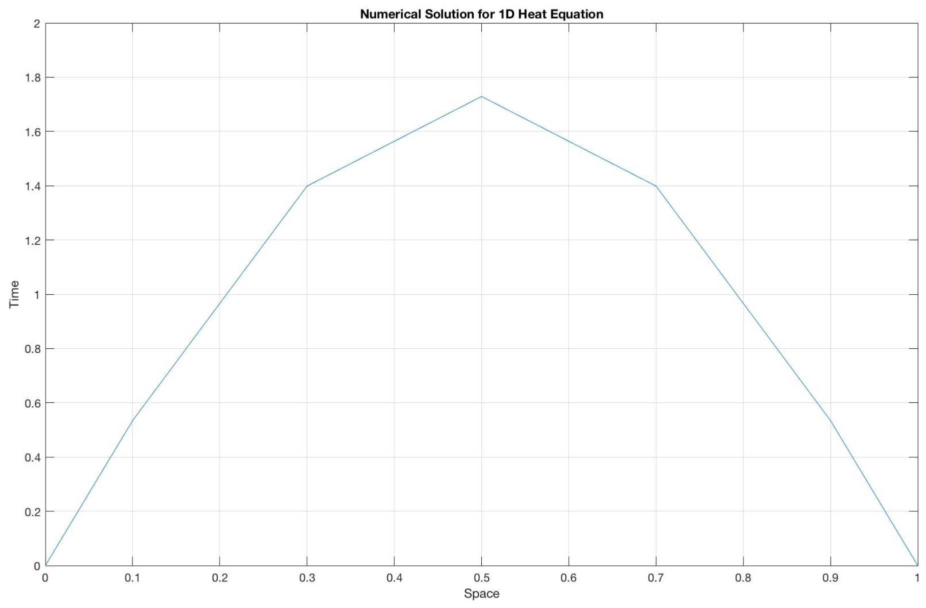

Figure 2: Heat equation numerical solution plot 2.

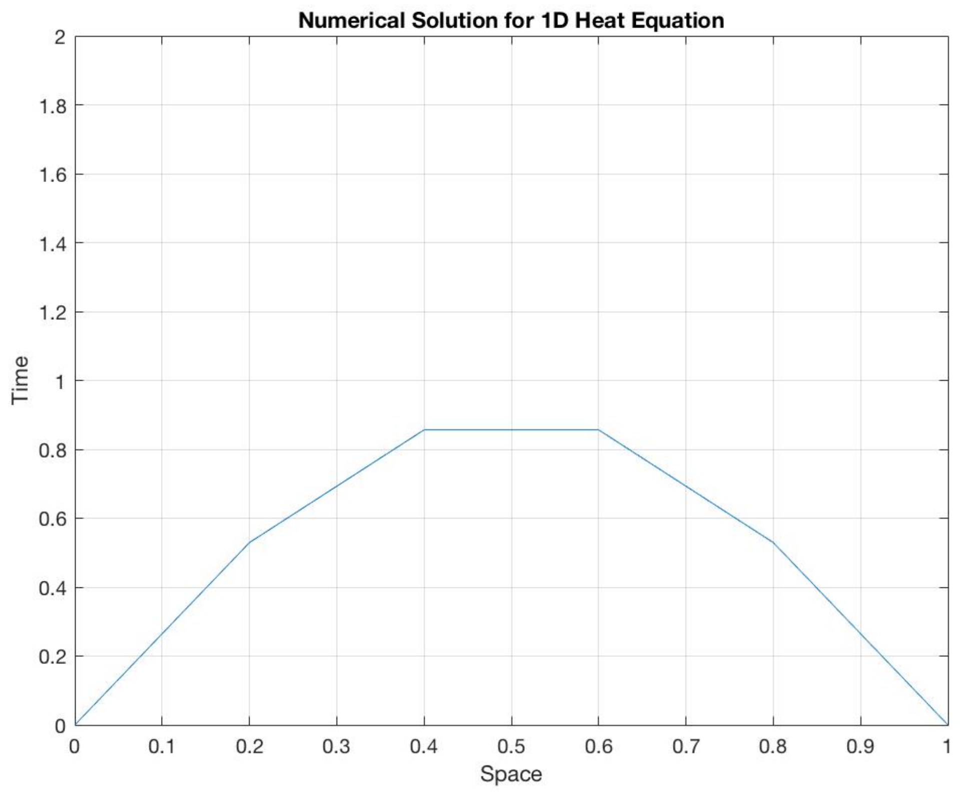

Figure 3: Heat equation numerical solution plot 3.

### 4.2. Heat equation image with scaled colors

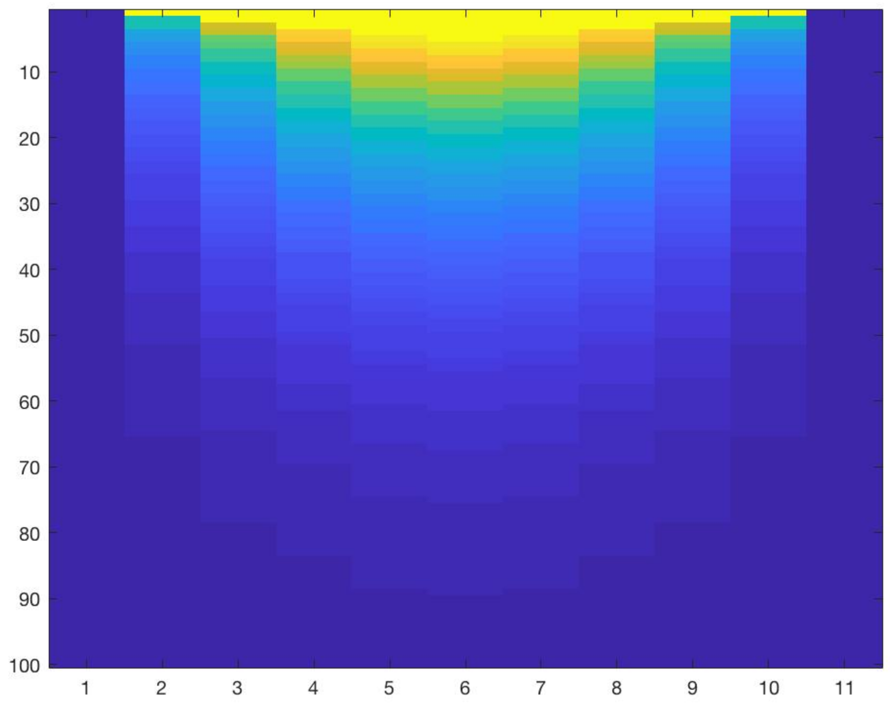

Figure 4: Heat equation image with scaled colors.

## 5. Observations

We know stability of heat equation depends on a factor: **c*k/h^2**, it's also known as CFL condition. For our system/ equation to be stable, this factor should be less than or equal to ½. I have taken care of CFL stability condition in my code and as result the plots that I got represents a stable equation/ system.

But, when I tried solving the equation without satisfying CFL condition, the plots that I got were completely unstable and are shown below.

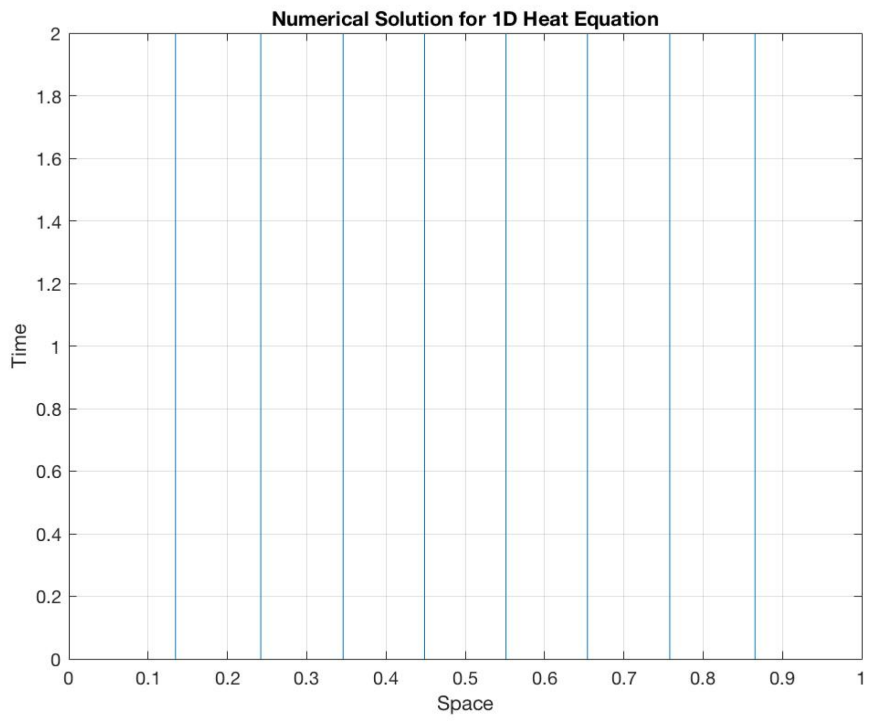

Figure 5: Heat equation not satisfying CFL condition.

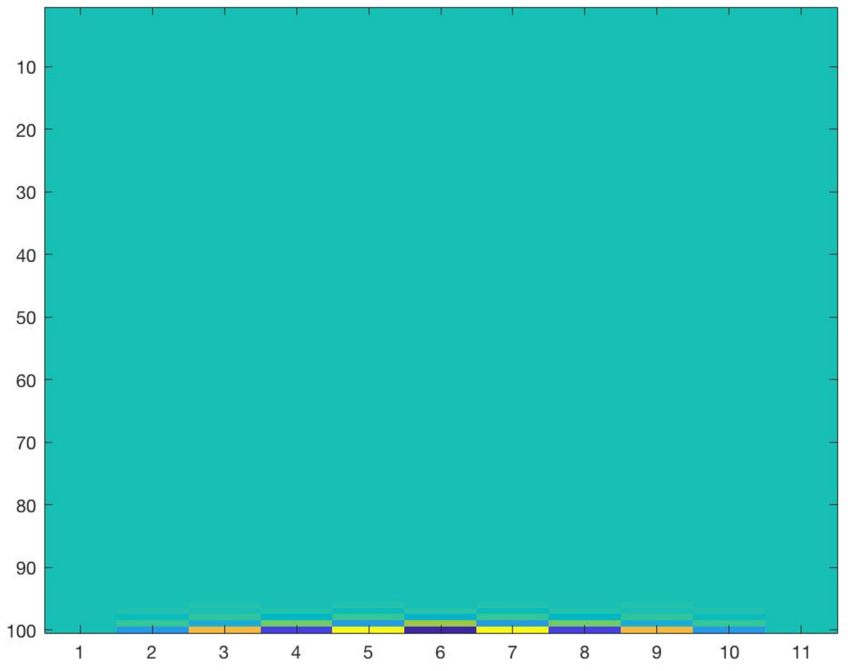

Figure 6: Heat equation image with scaled colors with unstable system

Also, I observed that when grid points (number of elements in matrix) was increased I got much better results in terms of plots and scaled color images. Plots were much more finer and clear.

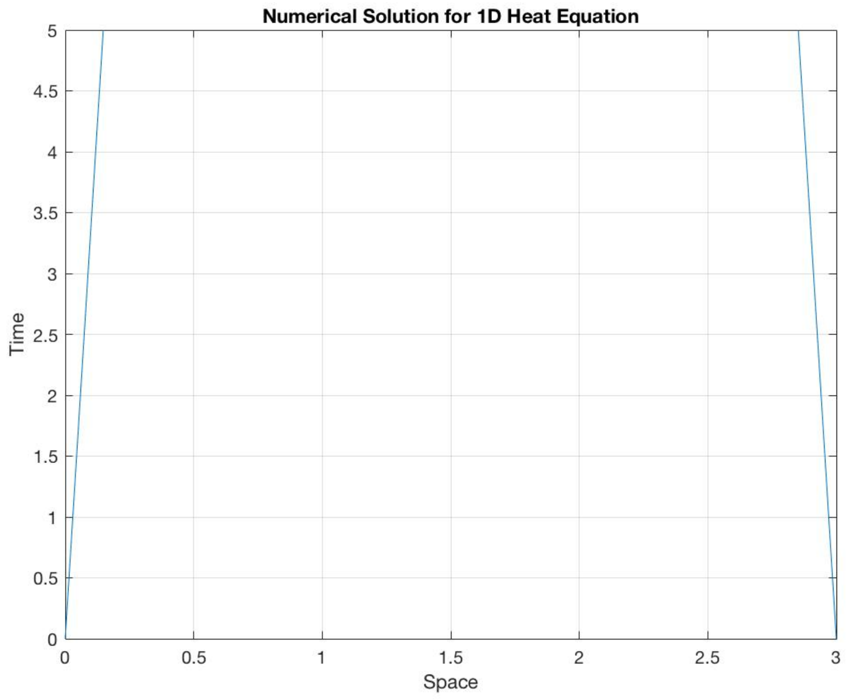

Figure 6: Heat equation plot with more grid points.

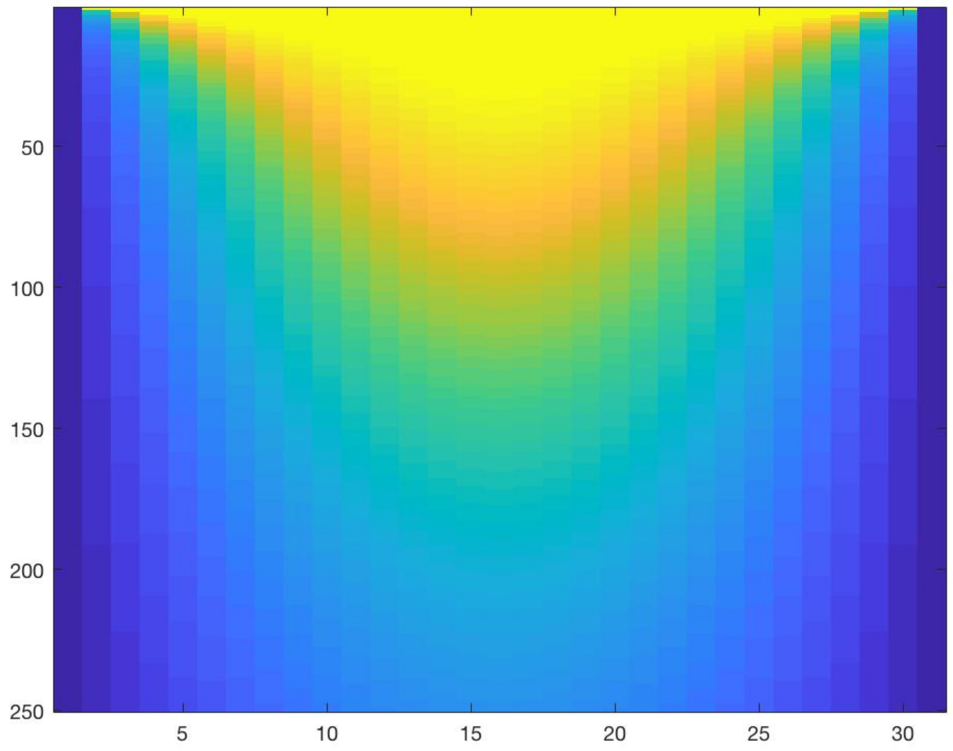

Figure 8: Heat equation image with scaled colors with more grid points

## 6. Conclusion

Many partial differential equations cannot be solved exactly and one needs to turn to numerical solutions. The heat equation is a simple test case for using numerical methods. Numerical methods enable us to find value at every point/ instance given we have some prior information about that system/ equation.

Also, solution of equations can be improved by increasing grid points. Increasing grid points increase the accuracy of the system.