Numerical Analysis

Group 10: Forbelets

Imperial College, March 2017

Finite Differences for PDE

Comments on implementation:

We first create a matrix to implement our finite difference method on. The two dimensions of the matrix and time and displacement.

The first row of the matrix (in other words at time t = 0) is populated with the initial conditions.

The rest of the matrix is populated in a loop from the intial conditions and the boundary conditions.

In the loop, for each value of m, the time itterator of the matrix, we set the the first and last possible displacement index's to our desired boundary conditions. We then we itterated trough each displacement step and apply the central algorithm to it. The exact implementation of this loop can be seen in the included code.

Chosing h and k

Chosing h (the displacement step) and the appropirate k (time step) is not a trivial task. Picking a very small time step is computationally expensive (requires a big matrix and a computer with more memory), and we generally want to compare the temperature distribution at larger time increments (as shown in the instruction sheet).

The Von Neumann stability anlaysis shows that in order to produce a stable results it is required:

$$\frac{k}{h^2} \leq \frac{1}{2}$$

Since we want to produce long simulations with a small distance step, but don't want to see extremely small steps in the graph, we set the time step to satisfy the equality given a value for h. Even then, the steps are hardly small so when graphing we only a fraction of the time steps from the whole range. In our implementation we print 10 equaliy spaced lines from the time range.

Tent Function

$$y_0(x) = \begin{cases} 2x & \text{for } x \in [0, 0.5] \\ 2 - 2x & \text{for } x \in [0.5, 1] \end{cases}$$

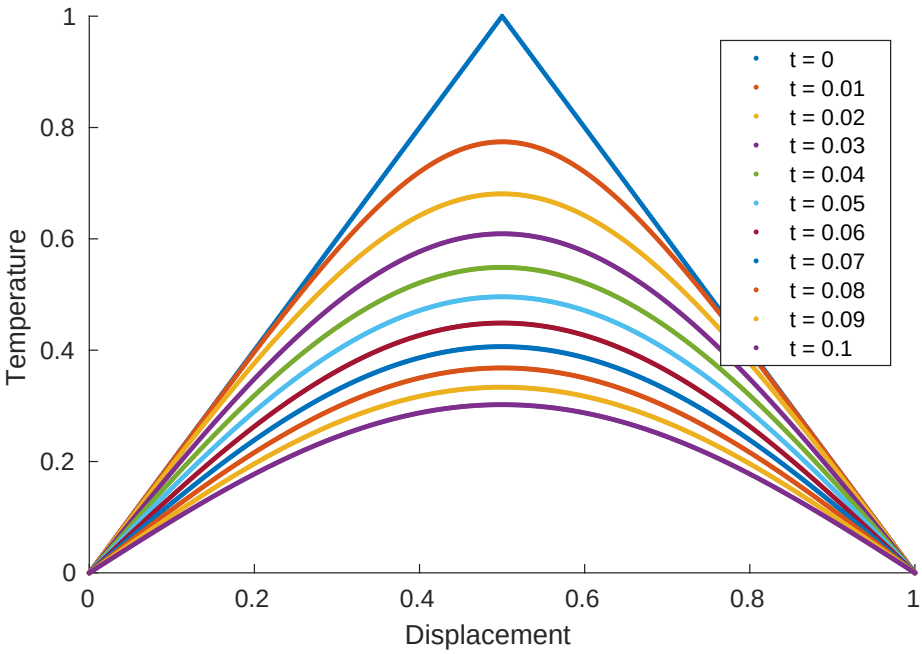

Figure 1: Tent Function

In figure 1 we can see the tent function applied as the initial condition to a one dimensional rod of length 1. The coloured lines represent the approximated temperature at times increments of 0.1, all the way up to 1.0. Because of the boundary conditions the end points are bound to zero degrees and as a result the temperature of the rod decreases. As $\mathbf{t}$ approaches infinity the temperate of the rod will approach zero, where the center point which started as the highest temperature and is furthest away from the boundaries maitains the highest temperature of all points.

Sinusoidal Function

$$y_0(x) = \sin 2\pi x$$

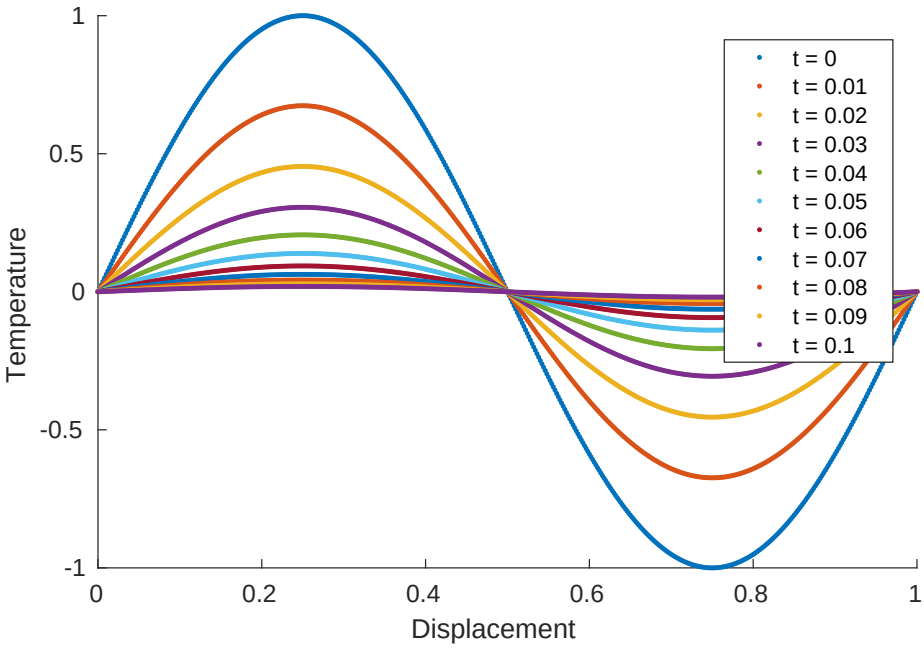

Figure 2: Sinusoidal Function A

In figure 2 we perform the finite difference method on to a sinusoid. The result yet again is decreasing temperature as the boundaries are zero. The temperature decreases such that the resulting distribution continues to be sinusoidal, only with a different amplitude. The left and right side fo the rod have identical but opposite amplitudes, and due to this equilibrium the center point at 0.5 displacement stays at 0 temperature.

$$y_0(x) = |\sin 2\pi x|$$

In figure 3 we use the same initial condition function, only this time we take the absolute value of it so what was negative before is now positive. It is interesting to note that the resulting distribution at $t \neq 0$ is very different from the absolute value of that in 3. Point 0.5 no longer sits at 0 and is instead heated due to neighboring elements until it becomes the hottest point.

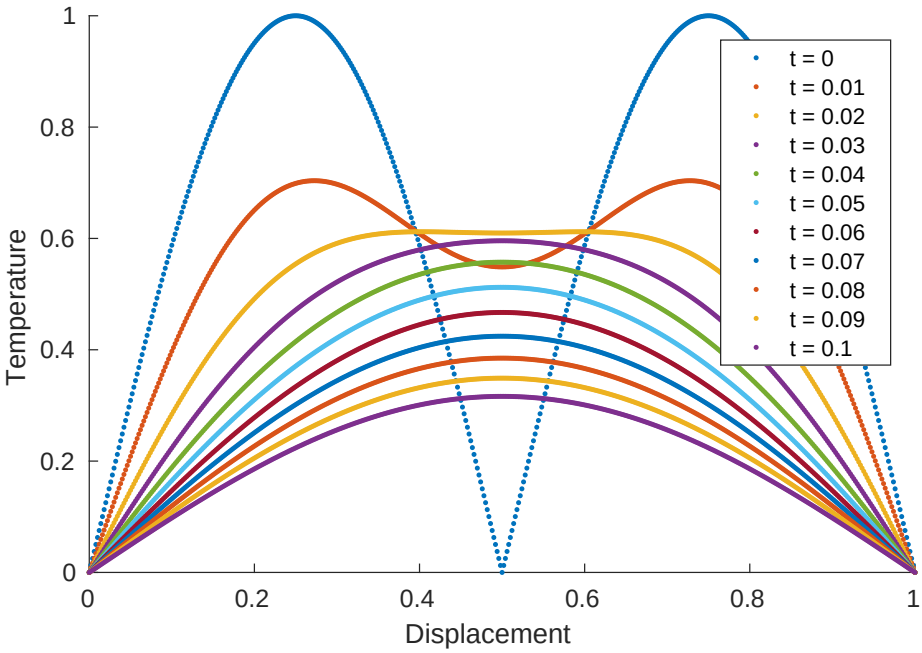

Figure 3: Sinusoidal Function B

Square Function

$$y_0(x) = \begin{cases} 0 & \text{for } x \in [0, 0.25] \cup [0.75, 1] \\ 1 & \text{for } x \in [0.25, 0.75] \end{cases}$$

Applying the "square" function in figure 4 produces interesting results. Where as in figure 3 the center heated up while the corners cooled down the opposite happens initially, as the curve straightens out. This happens because of the extreme temperature difference that we introduced.

Double Tent function

!Double Tent](cond4.pdf)

Making two symmetric tent functions with oposite signs like in figure ,shows a result like that in figure 1, with an artificial boundary of zero in the centre due to the symmetry.

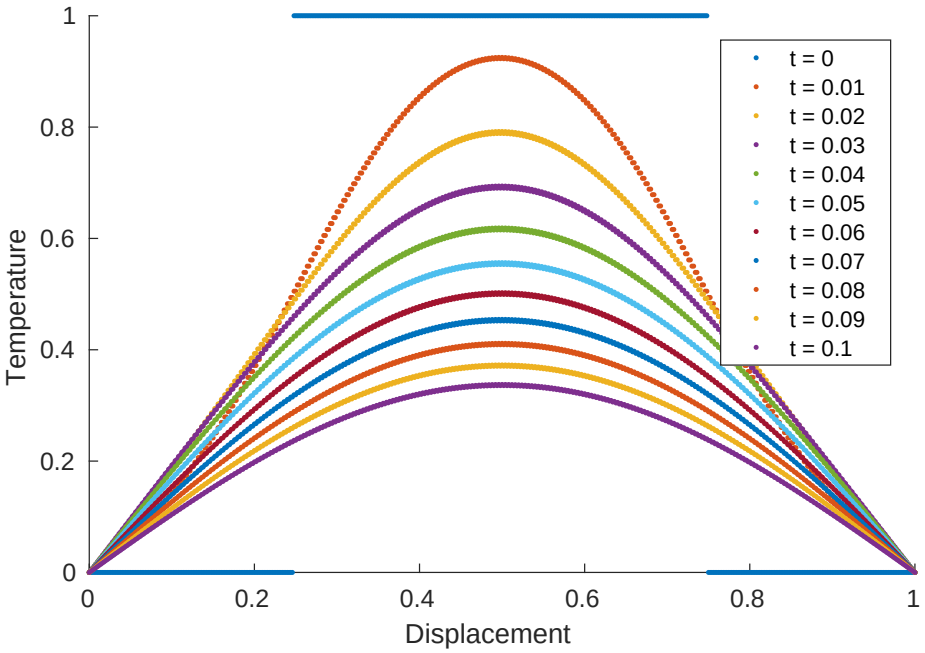

Figure 4: Square Function

Varying boundary conditions

It is possible to give the algorithm an initial condition which does not match the boundary conditions. In figure 5 we can see the initial conditions we used in figure 2, but we instead set the left boundary to 0.5 and the right boundary to 1. As the boundaries are forced to assume the temperature given they quickly pull the graph away from the original initial conditions and towards themselves. As **t** approaches infintity 5 will become a straight line between the two boundaries at 0.5 and 1.

In 6 we ...

See figure 7 for more information.

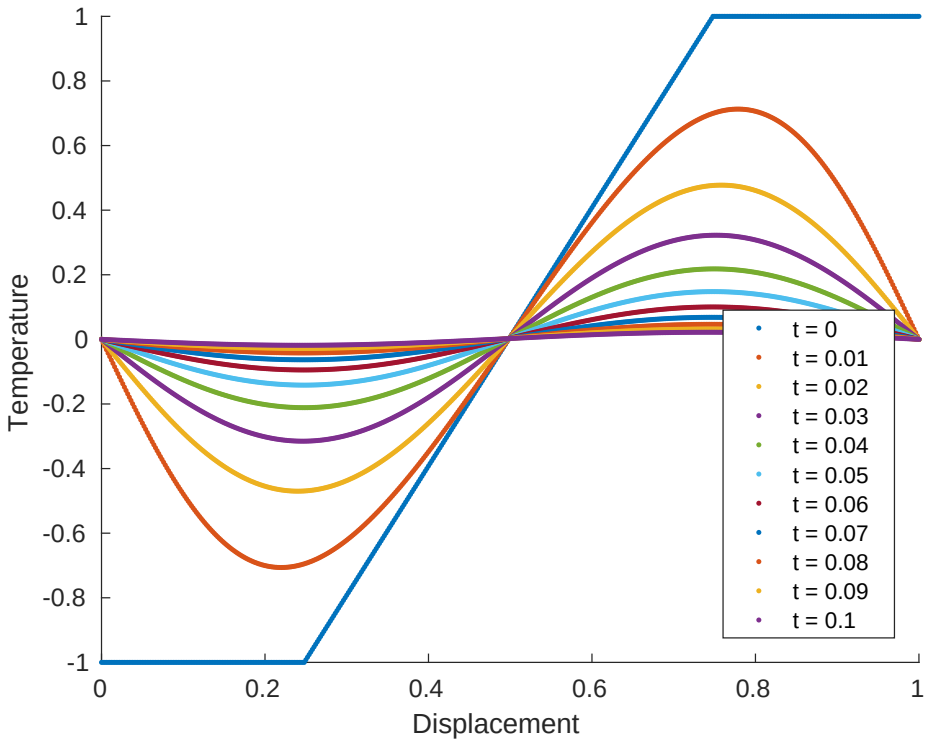

Figure 5: Non-matching initial and boundary conditions

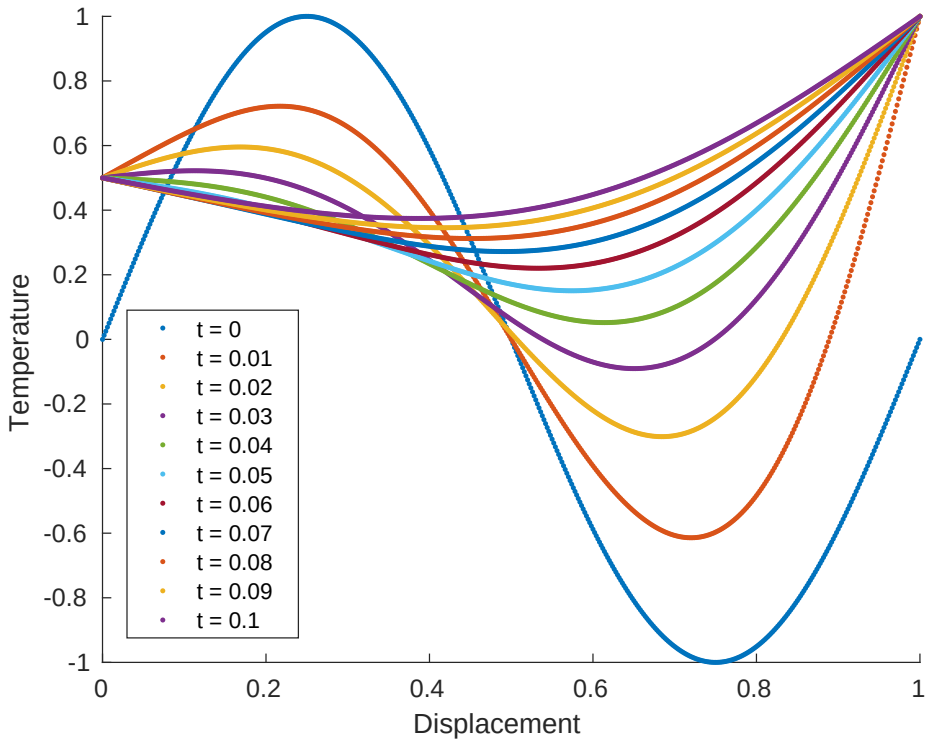

Figure 6: Non Zero Boundary Conditions

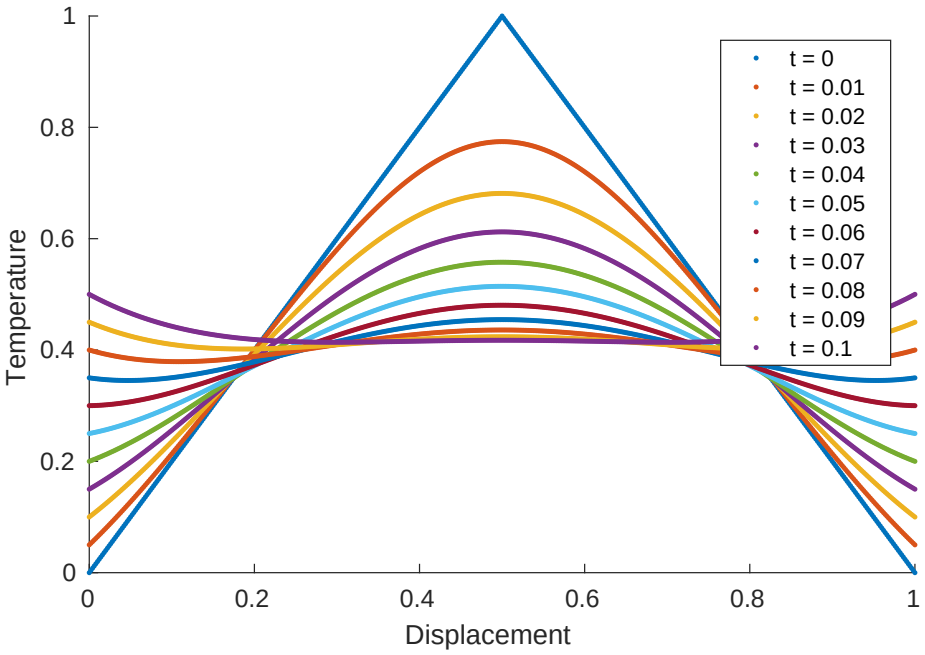

Figure 7: Varying Boundary Conditions