Solution of the differential equation (Heat transfer)

Juan Bettinelli B129454

Loughborogh Uni. Dep. of Phys.

Abstract

The 1 and 2 dimensional heat equation was numerically solved using the Euler method. Different boundary conditions were applied and plots where generated to demonstrate an understandable abstraction of the data produced. The modelling was reformed using MatLab, a forth generation programing language that is specialised on mathematical modelling. The report discus the steps taken to preform such results, together with the assumptions taken and limitations occurring with this method.

Introduction

The heat equation is a differential equation of the second order, it is used in many field across engineering, science and maths. The heat equation describes the flow of heat in a material and it is formulated as

for all coordinates systems. The one-dimensional Cartesian equivalent can be formulates as

Where α and D are positive constants called the thermal diffusivity which is a property of the material. This equation can be very useful in order to model the heat flow analytically, but this can’t be done for most real scenarios. In order to model those cases one have to fall back to numerical methods. Numerical methods lose a small amount of accuracy in the result, which can be overlooked due to the fact that they can be used in computing and so provide a tremendous speed of calculation. Now to solve the heat equation numerically a certain approach has to be taken, which is for this purpose the Euler method. The Euler method estimates a curve using the three known parameters, which are the derivative of the function at a certain point, its initial condition and a step size between curve points. The principle is simple for a function with initial condition , the first point is know, together with the derivative at this point. Now gradient of this derivative is follow for a small distance h to a new point y1 this point will only differ by a very small amount to the actual curve. This process is the repeated various times to model the curve, Image. Expressed in mathematical form

or arranged slightly different using (2) and (4) as

it is simple to understand that a higher precision can be realized by using a smaller step size, as the small sections between the data points approach linearity.

The Euler method is not just able to estimate first order differential but also higher orders as required for example in the heat equation and they all follow the same principle, the second order for example is described as

More accurate methods do exist, but for the purpose of this problem the Euler method is sufficient.

The platform used to preform this numerical analysis was MatLab, which is a forth generation programing language specifically designed for mathematical proposes, and so forms a perfect environment for numerical analysis intended on this problem.

Some of the syntaxes used to active this will be shown in this report with sufficed explanation to follow the meaning.

The problem in question was to model the heat equation on a 1 dimensional bar over time, with two different sets of initial conditions, and an extension of the model to a 2 dimensional surface. The exact condition will be discus in the sections below.

Programing solution

1D Problem Initial conditions T(:,1),T(:,L)=0

The task in hand was to solve the heat equation (1) for a 1 dimensional bar, with initial conditions T(x,t0)=x2 and boundary conditions T(x0,t)=T(xL,t)=0 stating that the end of the bar are kept at 0, it is to be noted that the temperature and the length don’t have specified units, the conversion to units would be possible be choosing appropriate constants D & constant.

The first step to solve this problem was to identify an appropriate algorithm for this task, this algorithm can easily be derived from equations (1), (3) and (4), the heat equation needed to be transformed to a numerical equivalent which iwas easily done by replacing dT/dt and d2T/dx2 with (3) and (4) to generate

the rearranged version (#.b) stats that the data point at a certain time is found by processing the acquainting points and the point itself at one time instant before as describe by above. The task then was to generate a contentious process that finds the temperature of the bar at every point over multiple time steps, to do this two loop processes were required one for the spatial dimension and one for the time dimension.

The MatLab code preforming this action is shown below

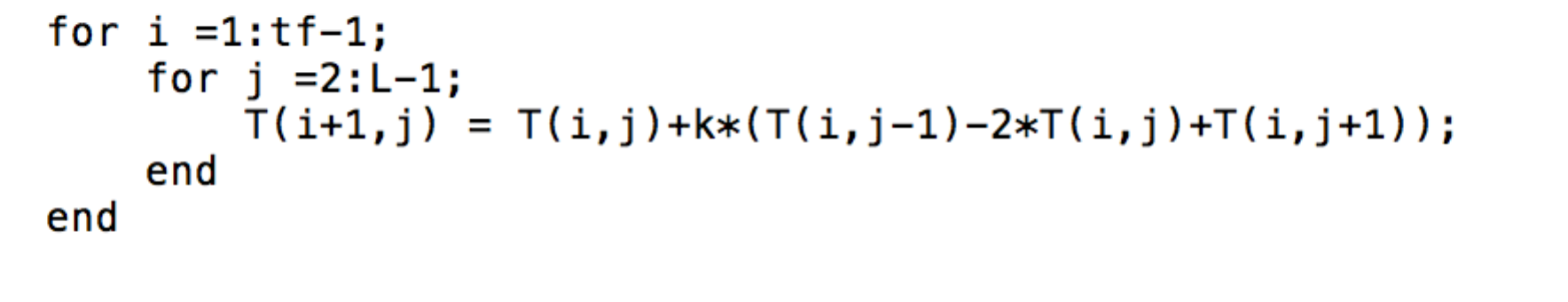


Figure MatLab syntax for Euler method 1D Heat equation, 1st loop time iteration 2nd loop spatial iteration

were k was declared beforehand, MatLab syntax is seen below

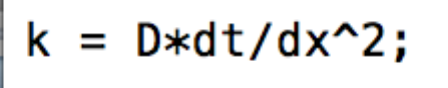


Figure MatLab syntax defining constant k

The constants D: thermal diffusivity, dt: step in time, dx: step in space, Tf: final time for iterations and L: length of bar, are declared beforehand and more details can be seen in the appendix where the full code is shown.

This function on its own was not able to preform as no initial condition are stated in the code at that instant, the conditions are described above and its Coding equivalent is shown below

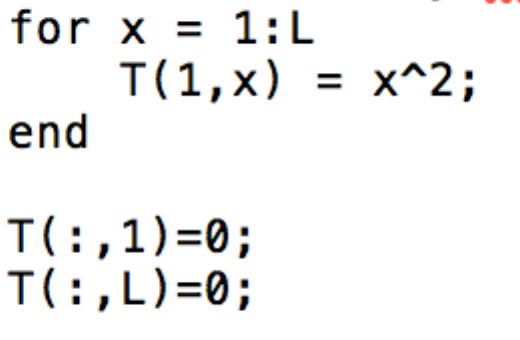


Figure MatLab syntax defining initial conditions as T(x,0)=x^2 using a loop iteration and boundary conditions keeping the ends at 0.

the final step was to display the result. So far the generated heat dispersion was stored in a 3 dimensional matrix T, to display the data so that it can be understood a mesh plot was used, and example of a plot can be seen in the appendix. The code to generate such a plot shown below.

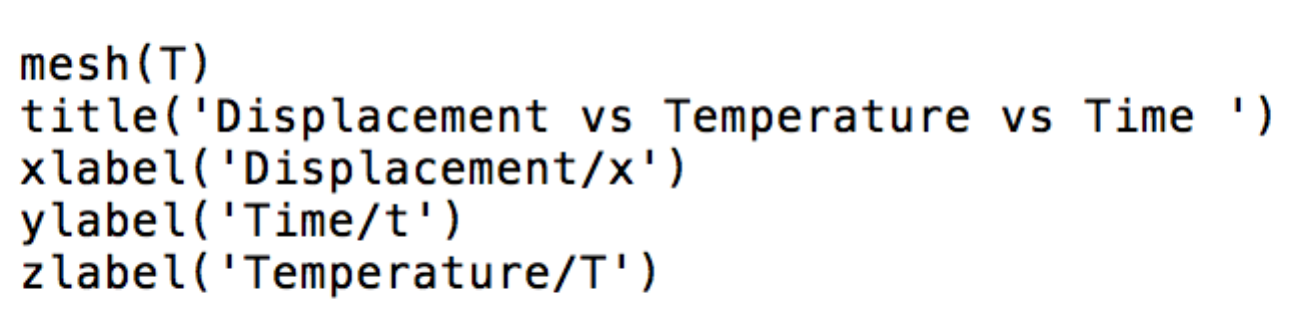


Figure MatLab syntax to display matrix T using a mesh plot, together with labelling of plot and axis.

The code produces has limitation in performance, due to the limited number of iteration that can be made in a reasonable time and the minimum step size that can be performed. Additionally the algorithm faces a different limitation which is that it only stabilizes for the conditions , this limitation could be evoided by using a different numerical approach like the Crank–Nicolson method witch will not be discus here as the accuracy is sufficient with the Euler method. An example of not stabilizing plot can be found in the appendix. REFERENCE

Testing

1D Problem Initial conditions dT/dx(:,1),dT/dx(:,L)=0

The boundary conditions of the previous problem were slightly altered, before the ends of the bar where kept at 0, now the flux the end of the bars where kept at 0.

i.e. , the interpretation of this condition was a little difficult and so the assumption was made were T(0,t)=T(1,t) and T(L,t)=T(L-1,t) interpreted as that T(1,t) was calculated first as shown before and then T(0,t) was taken to be the same. To accomplish this the initial condition where changed to

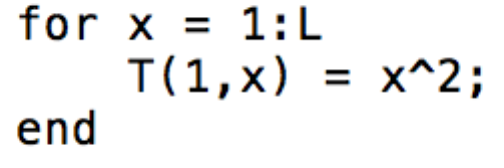


Figure MatLab syntax defining initial conditions as T(x,0)=x^2 using a loop iteration

and the section processing the heat equation to

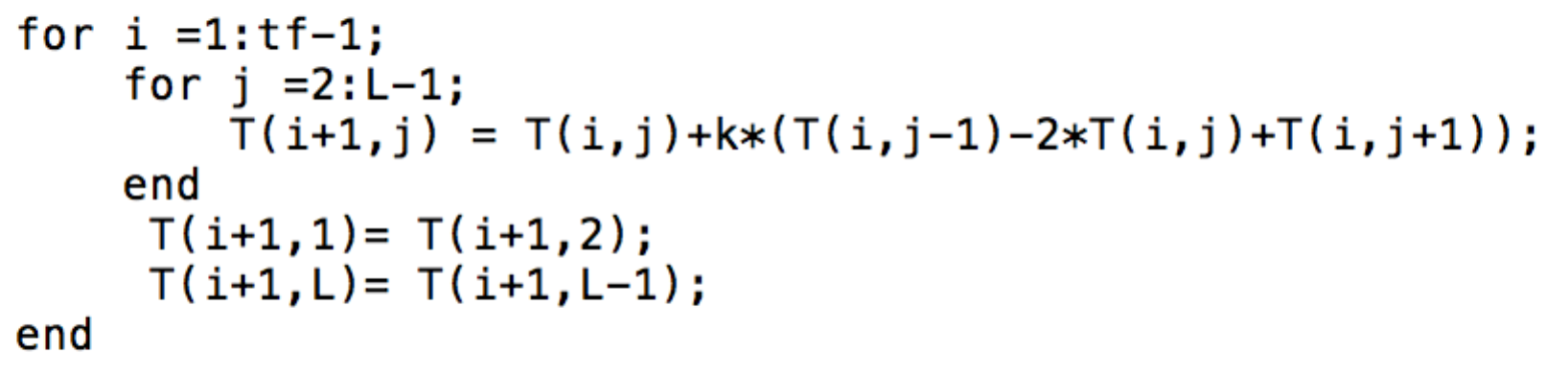
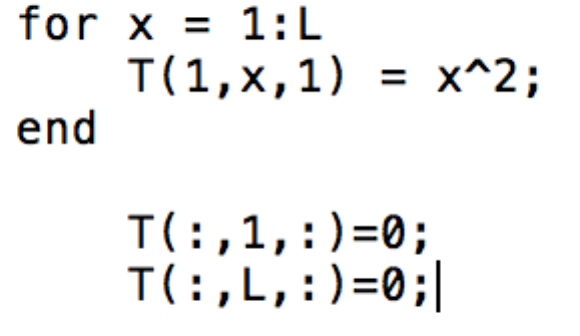


Figure MatLab syntax for Euler method 1D Heat equation, 1st loop time iteration 2nd loop spatial iteration, boundary condition ‘flux 0 at end of the bar’ are embedded in time loop.

were the coping of the temperature at the ends of the bar was taken outside the spatial loop to minimise redundant calculations and possible overwriting of values. An example of a generated mash plot and the complete code can be seen in the appendix. The same limitations apply then with the pervious initial conditions.

2D Problem Initial conditions T(:,:,1),T(:,:,L)=0

Now that the basic algorithm for the 1 directional bar was established it was easy to extend it to a 2 dimensional surface, the initial conditions are chosen so that one corner stards at temperature T(0,x,0)=x2 the remaining at 0, and boundary conditions that the ends in x direction kept at 0. The code declaring this is shown below



In the Apendix a different initial and boundary condition is demonstrated where the edges are kept a 0 and the temperature is initaly at T(x,:,:)=x^2)

The Heat equation for a 2 dimensional system is

were D and B are not necessary the same, as for example seen in some crystal sutures. By then applying the Euler method again we got

To compute each point of the surface a new iteration loop needed to be added, that dealt with the y index. The modifications made to the function dealing with the 1 dimensional is shown below

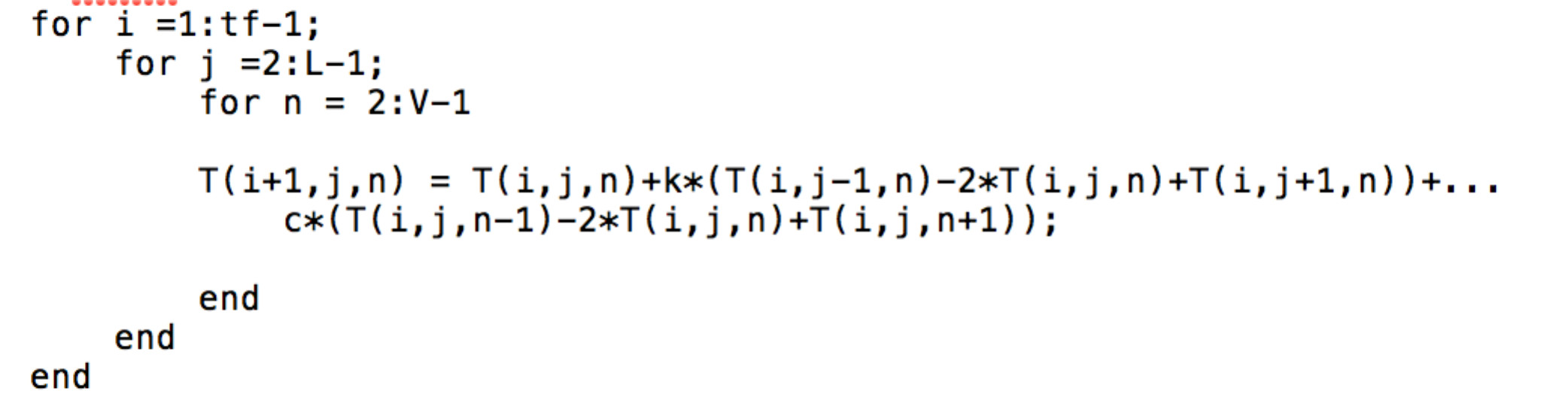


Figure MatLab syntax for Euler method 2D Heat equation, 1st loop time iteration 2nd & 3rd loop spatial iterations

were k and c were declared before as

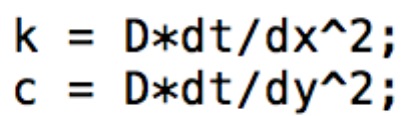


Figure MatLab syntax defining constants k & c

The stability limitation for this function has changed to .

2D Problem Movie representation

As the matrix T is now 4 dimensional it cannot be represented in a 3 dimension mash plot. In order to represent it meaningful a movie has to be created were the change is shown over time. To achieve this the matrix needed to be transformed into 3 dimensions (2 space, 1 temperature) at each time point and a plot of this matrix had to be made. The plot at each time point was saved in a 1 directional matrix, and by then using the function “movie” the individual images where merged to a movie sequence. The code used to accomplish this is shown below

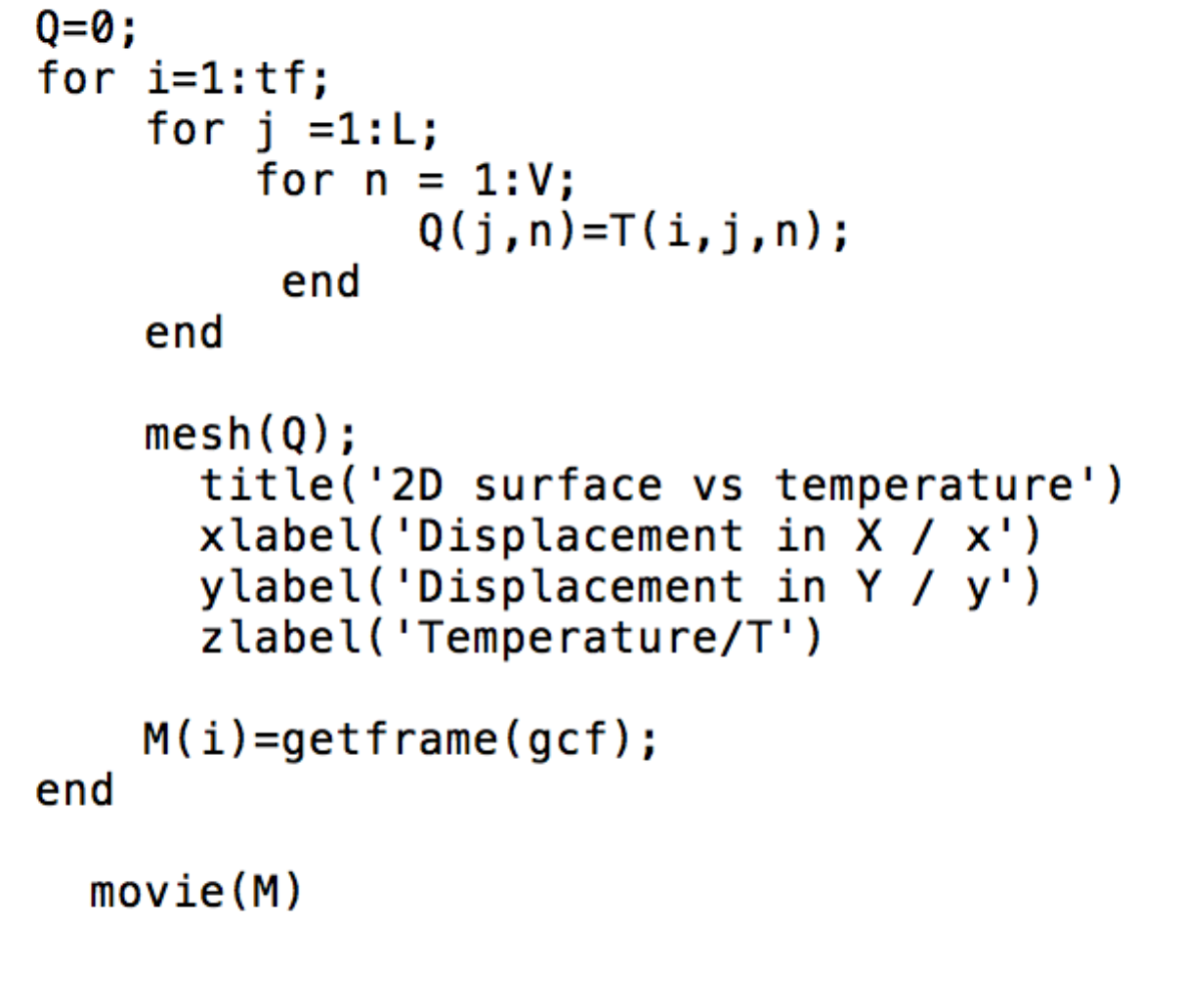


Figure MatLab syntax to generate a movie sequence of a mash plot changing in time, using loop iterations to transformed the spatial and temperature demotions to a different matrix, potting a mash plot at each time interval and saving it in a new variable, then outputting a movie sequence constructed from individual plots.

the movie sequence might ampere a bit chopped due to the jump in temperature axis, this can be stopped by fixing the axis to a constant value but this has the consequence that the flow can’t be observed at smaller values of temperaure so that it was not done for this representation. Examples of the movie sequence can be found in the appendix.

Conclusion

Reference

Appendix

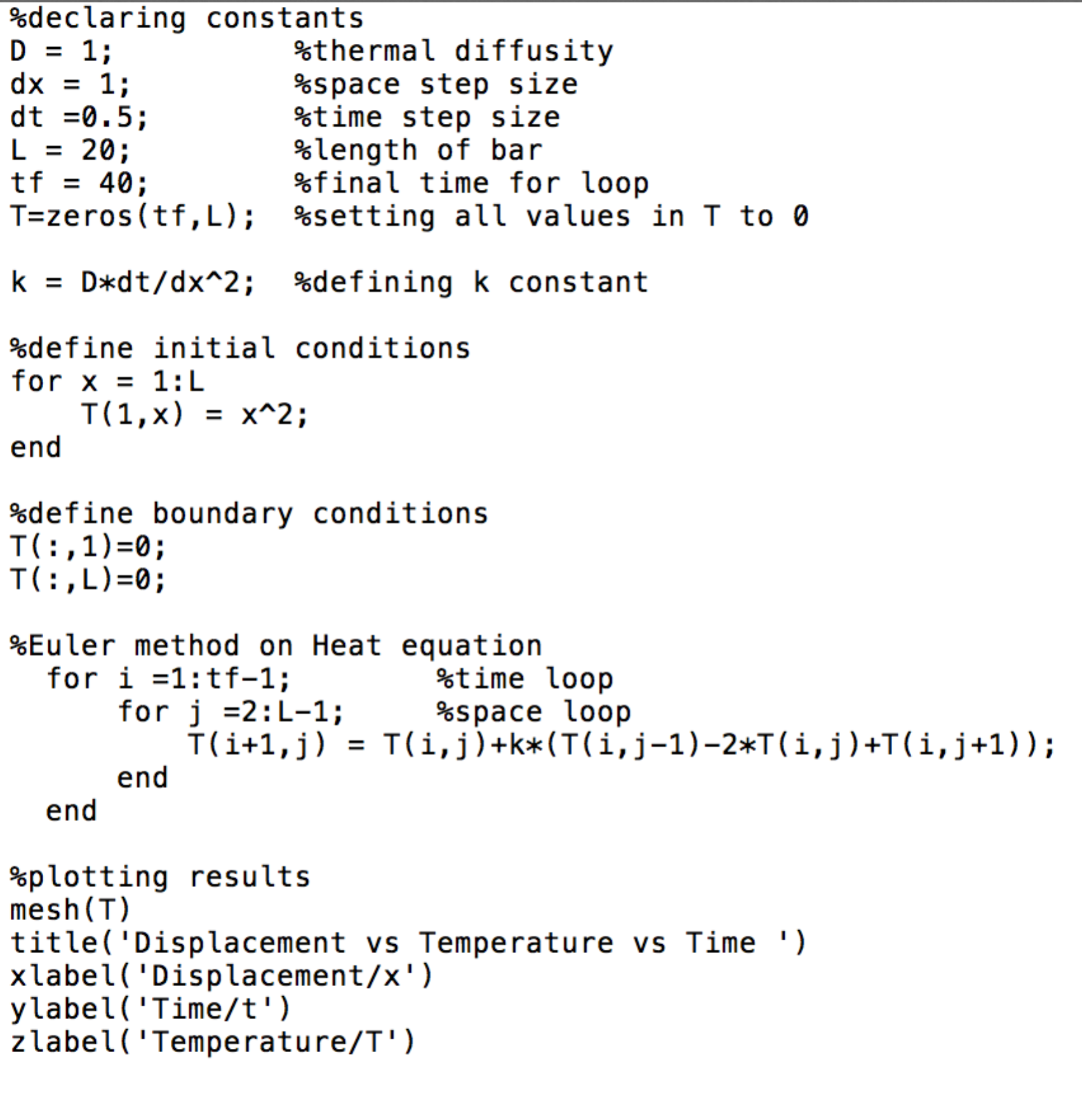


Figure Full code for 1D heat equation with i.c. T(x,t0)=x^2, b.c. T(x0,t)=T(xL,t)=0

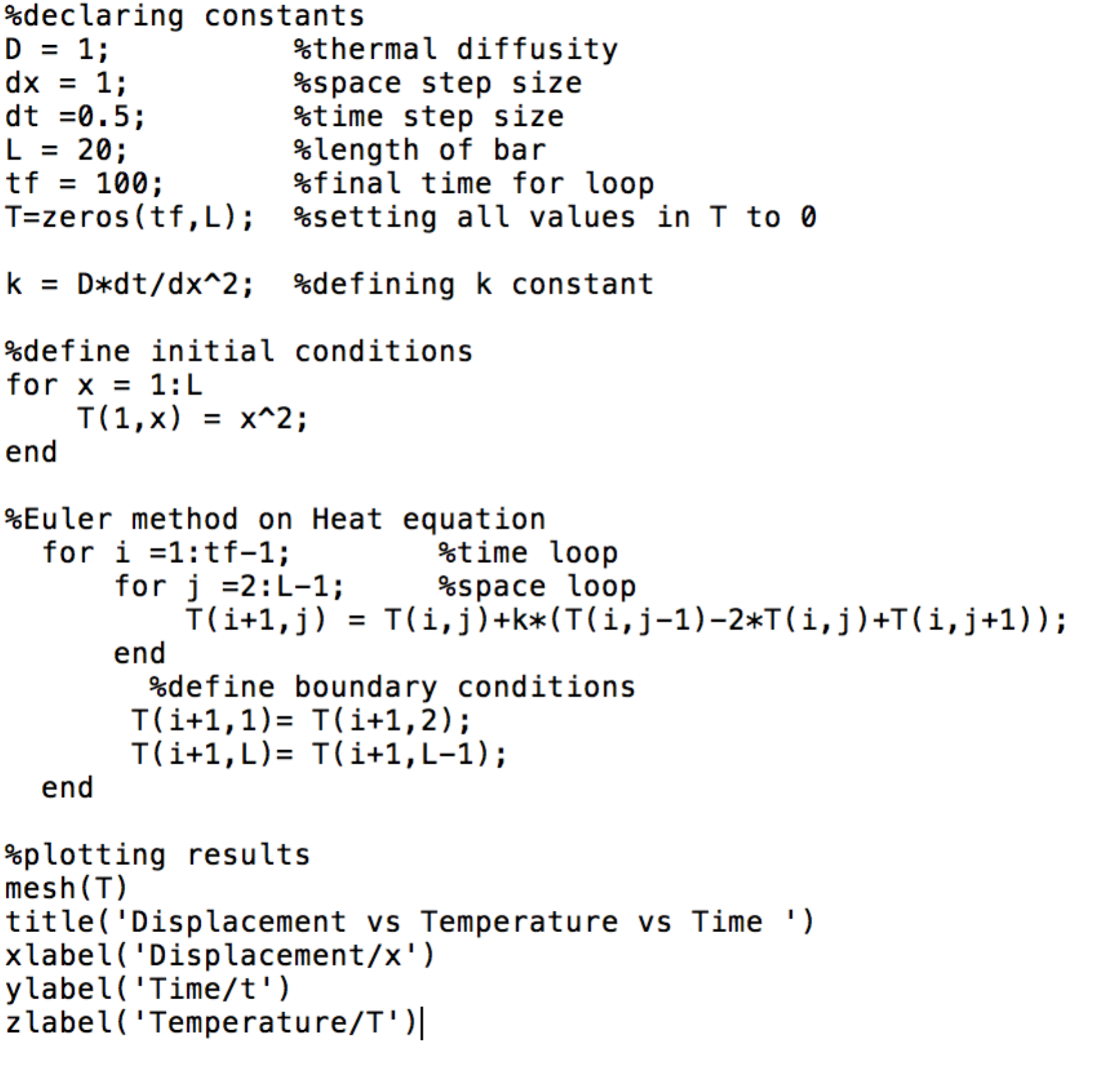


Figure Full code for 1D heat equation with i.c. T(x,t0)=x^2, b.c. T(x0,t)=T(x1,t) & T(xL,t)=T(xL-1,t)