# ENGR 695 Advanced Topics in Engineering Mathematics Fall 2023

## Lab #9: Explicit Finite Difference Solution of Heat Equation

Introduction

We have seen that many partial differential equations (PDEs) can be solved relatively easily via the separation of variables (SOV) method, especially if their boundary conditions align with the applicable coordinate system. Other problems, however, are very difficult or even impossible to solve in this way. Examples include complicated nonhomogeneous problems and cases in which the problem geometry is highly irregular and does not conform to an orthogonal coordinate system. In these situations, it is often necessary to turn to some type of numerical method. While numerical approaches are usually more versatile and often have simpler formulations than analytical approaches like the SOV method, the downside is that they usually provide less physical insight. Numerical solutions can have other practical limitations as well such as requiring significant computational resources.

In this lab session, you will examine the solution of the heat equation using an explicit finite difference method. You will have the opportunity to write the key parts of the code that implement the algorithm and to observe how choices of various solution parameter values affect its accuracy and stability.

Theoretical Background

A widely used numerical approach is the finite difference (FD) method, which approximates the derivatives in a PDE using backward, forward, or centered finite differences. The FD method can be either explicit, in which the dependent variable at each point in the solution space at each time step is sequentially calculated using known quantities from the previous time step, or implicit, in which the dependent variable at all points in the solution space are calculated simultaneously via a system of equations (i.e., a matrix solution).

For example, an explicit FD solution of the one-dimensional heat equation described by

$$c\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \qquad a \le x \le b \quad \text{and} \quad t \ge 0,$$

is

$$u_{i,\,j+1} = C\,u_{i+1,\,j} + \left(1-2C\right)u_{i,\,j} + C\,u_{i-1,\,j}, \qquad \text{where} \quad C = \frac{c\Delta t}{\Delta x^2},$$

and where $u_{i,j}$ is the value of the dependent variable (heat or temperature) at the location $x_i = a + (i - 1)\Delta x$ for $i = 1, 2, 3, \ldots, N_x$, where $N_x$ is the number of discrete points within the solution space. Index $j$ specifies the discrete moment in time $t = j\,\Delta t$, and constant $c$ is the thermal diffusivity of the material.

Since the dependent variable $u$ is evaluated only at a finite number ($N_x$) of discrete points in the solution space, the values of $u$ over the whole space at a given moment in time are stored in a software vector of length $N_x$. The solution at all points within the space is then "updated" at each

new time step in the numerical routine using the expression for $u_{i,j+1}$ given above, which is why it is often referred to as an *update equation*. Note that the value of $u$ at location $i$ and time $(j + 1)\Delta t$ depends only on the values of $u$ at the same or adjacent locations at the previous time step ($j\,\Delta t$), thus confirming that this is an explicit method.

Dirichlet boundary conditions such as

$$u(a,t) = u_a \quad u(b,t) = u_b$$

are treated in the FD solution simply by maintaining the values

$$u_{1,j} = u_a \quad u_{N_x,j} = u_b$$

at all times, where again $N_x$ is the number of spatial locations in the solution space. That is, the first and last elements of the dependent variable vector are set equal to the values specified by the boundary conditions at the beginning of the execution of the routine and are then left unchanged. The update equation is not applied at those locations. The initial condition

$$u(x,0) = f(x)$$

is accommodated by filling the solution vector at time $t = 0$ (corresponding to $j = 0$) with the values of $f(x)$ evaluated at each location within the solution space. That is,

$$u_{i,0} = f\left[a + (i-1)\Delta x\right] \quad i = 1, 2, 3, \ldots, N_x.$$

A key disadvantage of most explicit methods is that they can become unstable if the relationship between the spatial step size $\Delta x$ and the temporal step size $\Delta t$ is not constrained. An unstable solution is one that grows unnaturally without bound and therefore represents nonphysical behavior. For example, for the explicit FD update equation given above, the condition

$$\frac{c\Delta t}{\Delta x^2} \le 0.5 \quad \rightarrow \quad \Delta t \le \frac{\Delta x^2}{2c}$$

must be satisfied to obtain a stable solution. The proof of the stability condition is beyond the scope of this course but can be found in good textbooks on numerical methods.

Procedure

- Your assignment is to add a few lines of code to an incomplete *Matlab* m-file to solve a 1-D heat distribution problem like the one described above using an explicit finite difference method. Download the following *Matlab* m-files, which are available at the course Moodle site. You should set up a separate folder to contain them.

  `HeatEqnFDExample_lab.m` – primary script containing most of the required code
  `f_lab.m` – function that defines the initial temperature distribution $f(x)$

Note that you *must* change the file name of `f_lab.m` to `f.m` in order for the simulation to work properly. You may also change the name of `HeatEqnFDExample_lab.m` if you wish. (You might want to shorten it.)

The m-files are heavily commented, and the places where you need to add code are clearly indicated. Edit the m-files, and then run a test case with the following conditions:

$$u(0,t)=0 \qquad u(b,t)=0 \qquad u(x,0)=f(x)=100\sin^2\left(\frac{2\pi x}{b-a}\right)$$

The other important parameters such as the boundary locations $a$ and $b$, the thermal diffusivity of the material, the number of spatial steps and time steps, etc. are already coded near the top of the m-file. Note that this problem is the same as one of the early heat problems that we solved analytically when we began covering the SOV method.

- Run your modified scripts to verify that the routine is stable and producing accurate results. If you run the code for 4000 time steps with the largest allowable time step for a stable solution, the last displayed solution should closely match the SOV solution of the same problem shown in Fig. 1.

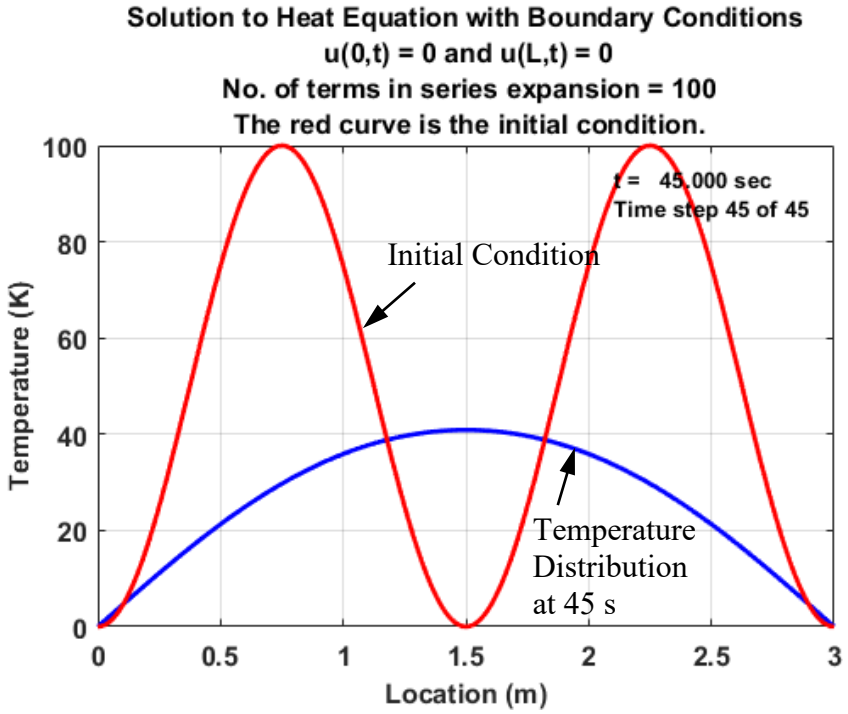


**Figure 1.** Separation-of-variables solution of the homogeneous 1-D heat equation problem at time $t$ = 45 s for the indicated boundary conditions and initial condition.

- After you are confident that the *Matlab* script is working, execute it again with the new boundary conditions and initial condition $f(x)$ given below with $c = 0.01$ m$^2$/s.

$$u(1,t) = u_a = 300 \qquad u(5,t) = u_b = 400 \qquad f(x) = u_a + \frac{u_b - u_a}{b-a}(x-a) + 30(x-a)(x-b)^2,$$

where $a = 1.0$ m, $b = 5.0$ m, $u_a = 300$ K, and $u_b = 400$ K. Remember that the *Matlab* function that implements $f(x)$ has to use element-by-element arithmetic operations (e.g., .* instead of * for multiplication; .^ instead of ^ for exponentiation). A plot of $f(x)$ is shown in Fig. 2. Check that the solution evolves over time properly.

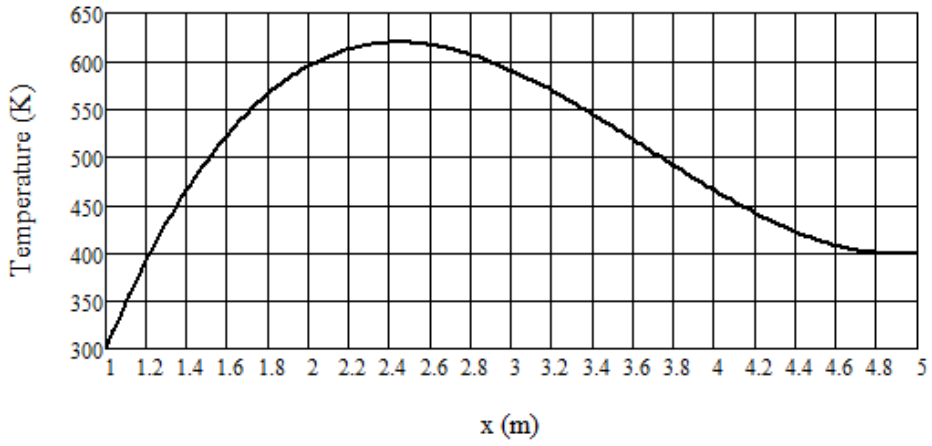


**Figure 2.** Initial condition $f(x)$ for the second heat equation problem.

- Execute your code with the new boundary conditions and initial condition $f(x)$ for a sufficient number of time steps so that the simulation time ends at $t = 600$ s. (Determine the number of time steps deterministically, not using trial-and-error.) This should be enough time for the solution to closely approach the steady-state condition. You may execute the code again with a greater number of time steps if you wish. Explain in comments added to your `f.m` file how you determined the required number of time steps to reach $t = 600$ s, and explain the physical basis for the steady-state temperature distribution at 600 s that you observe. That is, explain why the solution converges to that particular state.

- Set the number of time steps back to 3000 or so, and change the time step size so that it is barely above the stability limit described in the "Theoretical Background" section. To begin with, set the time step so that

$$\frac{c\Delta t}{\Delta x^2} = 0.505 \quad \rightarrow \quad \Delta t = 0.505\frac{\Delta x^2}{c},$$

and observe the results. Set the value of $\Delta t$ a little higher (say, to 0.55) and observe the results again. In comments added to your `f.m` file, briefly describe what you observe and whether the solution you obtain represents physical behavior.

Lab Work Submission and Scoring

Save copies of your edited `HeatEqnFDExample_lab.m` and `f.m` scripts. Change the names to `LName_Lab9main_fa23.m` and `LName_Lab9f_fa23.m`, where `LName` is your last name and "main" or "f" refers to the main script or `f.m`. Add comments to the `f.m` script in response to the various prompts above, which are repeated below:

- how you determined the required number of time steps to reach $t = 600$ s
- the physical basis for the observed steady-state temperature distribution
- observations of solution behavior with a time step that exceeds the stability limit
- comments on whether the solution you obtain for an unstable time step represents physical behavior

E-mail your edited scripts with comments to me.

Your score will be assigned according to the lab scoring rubric posted on the Laboratory page at the course web site.

If you do not complete the exercises during the lab session, then you may submit your documentation as late as 11:59 pm on Friday, November 17. If the file is submitted after the deadline, a 5% score deduction will be applied for every 24 hours or portion thereof that the item is late (not including the days over Thanksgiving break) unless extenuating circumstances apply. No credit will be given five or more semester days after the deadline.