CS601, Lecture 13/10/2022 – Computing Numerical Solution

In the previous lecture, we saw how first- and second-order derivatives can be approximated using difference equations. These equations are at the heart of the *finite difference method* (FDM) of solving PDEs via a computer. In FDM, all partial derivatives are approximated with the help of difference equations. Many other techniques exist to solve PDEs and *finite element method* (FEM) is another example. In FEM, the partial derivatives appearing the PDEs are approximated differently. We also saw in the previous lecture the truncation error and got to know terms like first-order accurate and second-order accurate. In this lecture we consider two example problems and see how a numerical solution is computed while approximating PDEs.

Computing Numerical Solution:

Example 1: 1D heat conduction equation for fins.

Recall that the first step in computing the numerical solution is to discretize the domain and represent using grid points. Assume that the 1D domain of length l is subdivided into N sub-domains. So, we have Δx, the spacing between grid points, = l/N and there exist N+1 grid points.

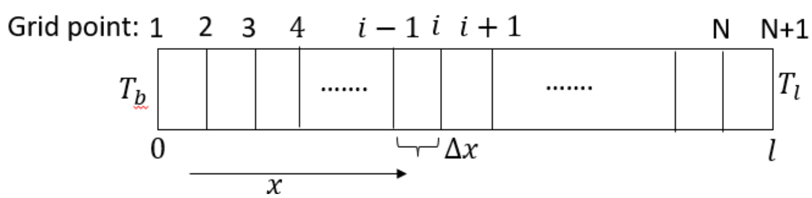
This problem can be modeled using the following PDE:

$$\frac{\partial^2 T}{\partial x^2} - hP/kA(T - T_f) = 0, \qquad \textbf{(1)}$$

where $\frac{hP}{\kappa A}$ is a constant and T_f= temperature at all grid points is known.

The above PDE describes the temperature variation along different points on the fin.

We have the boundary conditions as: temperature at $x = 0$ as T_b and at $x = l$ as T_l



Rewriting equation (1) using the difference equation for second order derivative (from previous lecture) to compute the temperature at grid point i:

$(T_{i+1} - 2T_i + T_{i-1})/(\Delta x)^2 - hp/kA(T_i - T_f) = 0$

= $\boldsymbol{(T_{i+1} - 2T_i + T_{i-1}) - \beta(T_i - T_f) = 0}$, where $\beta = \frac{hP}{\kappa A}(\Delta x)^2$ **(2)**

For grid point 2 (i=2), equation (2) becomes:

$$(T_3 - 2T_2 + T_1) - \beta(T_2 - T_f) = 0$$

$$= (T_1 - (2 + \beta)T_2 + T_3) = -\beta T_f \text{ (we know that } T_1 = T_b)$$

Similarly, for grid point 3 (i=3), equation (2) becomes:

$$(T_2 - (2+\beta)T_3 + T_4) = -\beta T_f$$

, for grid point N (i=N), equation (2) becomes:

$$(T_{N-1} - (2+\beta)T_N + T_{N+1}) = -\beta T_f \text{ (we know that } T_{N+1} = T_l\text{)}$$

So we have N equations from N grid points. These equations are written in matrix form as follows:

$$\begin{bmatrix} -(2+\beta) & 1 & 0 & . & 0 \\ 1 & -(2+\beta) & 1 & . & 0 \\ 0 & 1 & -(2+\beta) & 1 & . & 0 \\ . & 0 & 1 & -(2+\beta) & 1 \ldots & 0 \\ 0 & 0 & 0 & 1 & -(2+\beta) \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \\ . \\ T_{N-1} \\ T_N \end{bmatrix} = \begin{bmatrix} -\beta T_f - T_b \\ -\beta T_f \\ . \\ -\beta T_f \\ -\beta T_f - T_l \end{bmatrix}$$

The above is in Ax=B form, where A is a NxN matrix, x and B are vectors of size N. x is the vector unknowns. These unknowns are the temperatures at grid points. We can now use any method to solve this system of equations. Gaussian-elimination, LU decomposition, computing inverse of the matrix are some examples of such methods.

Example 2: *time-marching problem*. Such problems involve time as an independent variable and the computation progresses over time. The idea is to also divide the time domain into sub-domains (in addition to dividing the space domain into sub-domains). Consider the problem of analyzing the conduction of heat through a rod modeled as a 1D structure. The problem can be modeled using the PDE:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \qquad \text{———} \quad \textbf{(3)}$$

where α is thermal diffusivity, a constant if the material is homogeneous and *isotropic*. Goal is to find the temperature at different points on the rod at different times. We now have initial and boundary conditions specified as:

$u(0,t) = u_L$, //temperature at the left end of the rod (at distance=0) at any time is a constant u_L

$u(l,t) = u_R$, //temperature at the right end of the rod (at distance=l) at any time is a constant u_R

$u(x,0) = f(x)$ $0 < x < l$ //temperature at any point on the rod at *time=0* is some *given* function f of the distance of the point from the left end of the rod.

When we want to discretize the time domain i.e. divide the time domain into sub-domains, we chose a small step size Δt to march forward in time. Suppose we divide time t into N sub-domains, we have grid points on the time domain as: t=0, t=1, t=2, …. and t=N+1. The spacing between each of these grid points is Δt time units.

We can use forward difference equation to write $\frac{\partial u}{\partial t}$ as $(u^{i+1} - u^i)/\Delta t$. Note: e.g. u^i denotes the value function u at time $step\ t = i$. The time variable is used in the superscript. As a notation, I'll use the subscripts to denote the spatial grid point e.g. u_i denotes the value of function u at grid point i. Combining these two notations, u_i^n denotes the value of function u at grid point i and at time step $t = n$.

Substituting the difference formula in the PDE (3)

$$\boldsymbol{\frac{u_i^{n+1}-u_i^n}{\Delta t} = \alpha(u_{i+1}^n - 2u_i^n + u_{i-1}^n)/(\Delta x)^2} \qquad \text{———} \qquad \textbf{(4)}$$

The above approximation for the PDE of equation (3) is called *explicit time integration method. Explicit* because the expression on the RHS is computed at the previous time step and is known in any time step. Note that u_i^{n+1} is the only unknown in equation (4). Crank-Nicholson suggested that instead of considering only the computations at the previous time step, the computations at the *current* time step also should be considered. In other words, u^{n+1} is now a function of u^{n+1} and u^n . As per this suggestion, we would have:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1} + u_{i+1}^n - 2u_i^n + u_{i-1}^n)/2(\Delta x)^2$$

Note that the unknowns are now on the LHS and RHS of the equation and there are more than one unknown. Such a scheme, where u^{n+1} is a function of u^{n+1} and u^n is called implicit time integration method. When we use the implicit time integration method for this particular problem, we have N+1 equations (one at every grid point) at time step t=1 to begin with. Since the values at all grid points at time step t=0 are all given, we will have a system of equations, which can be solved as mentioned in the earlier problem. As you may see, the computation involved in implicit time integration step is significantly more than that in the explicit time integration step: as per equation 4, in every time step, we end up computing the temp values for N-2 grid points (N-2 because the grid points at the boundary are at a constant, given temp. In comparison, in the implicit time integration scheme, in every time step, we have to solve a system of equations to find the N-2 unknowns.