ENGR 695 Advanced Topics in Engineering Mathematics Fall 2023

Lecture Outline for Monday, Nov. 13

1. Finite difference solution of the heat equation (continued)

$$c\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \quad \text{where } c = \text{thermal diffusivity}$$

a. Heat equation expressed using finite differences

$$c\frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2} = \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t}$$

b. Define discrete points in space and time at which the dependent variable u is calculated. The arrays of points are called spatial and time *grids* or *meshes*.
   i. Solution space is along x-axis between boundaries $x = a$ and $x = b$.
   ii. Calculation time begins at $t = 0$.
   iii. Space and time are discretized into N_x and N_t points, respectively:

$$x_i = a + (i-1)\Delta x, \quad i = 1, 2, 3, \ldots, N_x \quad \text{where} \quad \Delta x = \frac{b-a}{N_x - 1}$$

$$t_j = j\Delta t, \quad j = 0, 1, 2, 3, \ldots, (N_t - 1)$$

c. There is a constraint on Δt (examined soon).
d. Finite difference subscript notation:

$$u(x,t) = u_{i,j} \qquad u(x+\Delta x,t) = u_{i+1,j} \qquad u(x-\Delta x,t) = u_{i-1,j} \qquad u(x,t+\Delta t) = u_{i,j+1}$$

$$c\frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2} = \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t}$$

$$\rightarrow \quad c\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} = \frac{u_{i,j+1} - u_{i,j}}{\Delta t}$$

e. Four of the five terms in FD form of equation are defined at time t (index j), but one is defined at time $t + \Delta t$ (index $j + 1$). Isolate that term on the left-hand side and move the rest to the right-hand side to form an *update equation*:

$$u_{i,j+1} - u_{i,j} = \frac{c\Delta t}{\Delta x^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right] \quad \rightarrow \quad u_{i,j+1} = \frac{c\Delta t}{\Delta x^2}u_{i+1,j} + \left(1 - 2\frac{c\Delta t}{\Delta x^2}\right)u_{i,j} + \frac{c\Delta t}{\Delta x^2}u_{i-1,j}$$

(*continued on next page*)

f. This is an *explicit* FD method. The newest value of u at location i depends only on previous values and no values at other locations at the new time. That is, there is only one term at time index $j + 1$. A system of simultaneous equations is not required to find u everywhere.
g. To improve computational efficiency (i.e., to minimize floating-point operations):
   i. Group like terms (for particular space and time indices) together.
   ii. Pre-calculate the coefficients and store them.

$$u_{i,j+1} = c_1 u_{i+1,j} + c_2 u_{i,j} + c_3 u_{i-1,j}, \qquad \text{where} \quad c_1 = c_3 = \frac{c\Delta t}{\Delta x^2} \quad \text{and} \quad c_2 = 1 - 2\frac{c\Delta t}{\Delta x^2}$$

2. Boundary and initial conditions

   a. Dirichlet BCs are simple:

$$u(a,t) = u_{1,j} = u_a \quad \text{and} \quad u(b,t) = u_{N_x,j} = u_b,$$

   where u_a and u_b are constants (zero for homogeneous BCs)
   b. Neumann BCs are more challenging (later)
   c. Initial condition: $u(x,0) = u_{i,0} = f(x_i)$

3. Problem set-up and stability condition

   a. Define grid of solution points: $x_i = a + (i-1)\Delta x, \quad i = 1, 2, 3, \ldots, N_x$
   b. Boundaries at $i = 1$ and $i = N_x$
   c. Define u vector to hold solution at each time step. In *Matlab*, `u = zeros(1:Nx)`
   d. Initial condition: $u_i = f(x_i), \quad i = 1, 2, 3, \ldots, N_x$.
   In *Matlab*, `u = f(a + Dx*((1:Nx) - 1))`
   e. Stability requirement (from von Neumann stability analysis, not covered in this course):

$$\frac{c\Delta t}{\Delta x^2} \le \frac{1}{2} \quad \rightarrow \quad \Delta t \le \frac{\Delta x^2}{2c}$$

   f. Limitation of explicit methods: Stability requirement places an upper limit on Δt, which could cause excessively long execution times
   g. Algorithm:
      i. Apply update equation for u at every interior solution point (i.e., all x locations except the boundaries) to calculate u everywhere at next time step. Most mathematical software, including *Matlab*, has "vectorized" arithmetic operations that can do this more efficiently than a loop. See example below.
      ii. Advance time by Δt and compute new values for u everywhere. Repeat every Δt and continue until $j = N_t$ (last time step).
      iii. Store and/or display u at each time step or at reasonable intervals.

(*continued on next page*)

h. Comparison of vectorized and nonvectorized algorithms (*Matlab*) to implement update equation

$$u_{i,j+1} = c_1 u_{i+1,j} + c_2 u_{i,j} + c_3 u_{i-1,j}\,, \qquad \text{where} \quad c_1 = c_3 = \frac{c\Delta t}{\Delta x^2} \quad \text{and} \quad c_2 = 1 - 2\frac{c\Delta t}{\Delta x^2}$$

Nonvectorized

```
for j = 1:Nt
     for i = 2:(Nx-1)
          u(i) = c1*u(i+1) + c2*u(i) + c3*u(i-1);
     end
end
```

Vectorized

```
for j = 1:Nt
     u(2:(Nx-1)) = c1*u(3:Nx) + c2*u(2:(Nx-1)) + c3*u(1:(Nx-2))
end
```