ENGR 695 Advanced Topics in Engineering Mathematics Fall 2024

Lecture Outline for Wednesday, Nov. 13, 2024

1. Boundary 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)

   $$\left.\frac{\partial u}{\partial x}\right|_{x=a} = u_{xa} \quad \text{and} \quad \left.\frac{\partial u}{\partial x}\right|_{x=b} = u_{xb} ,$$

   where u_{xa} and u_{xb} are usually constants but could be time varying; they could also be zero (homogeneous).

   c. Robin BCs are still more challenging (later)

   $$\left.\frac{\partial u}{\partial x}\right|_{x=a} + A\,u(a,t) = C_{xa} \quad \text{and} \quad \left.\frac{\partial u}{\partial x}\right|_{x=b} + B\,u(b,t) = C_{xb} ,$$

   where A and B are constants and C_{xa} and C_{xb} are usually constants but could be time varying; C_{xa} and C_{xb} could also be zero (homogeneous). Example: Outward heat flux is proportional to difference between temperature at boundary and temperature u_m of surrounding medium (h is a constant):

   $$\left.\frac{\partial u}{\partial x}\right|_{x=b} = -h\left[u(b,t) - u_m\right] \quad \rightarrow \quad \left.\frac{\partial u}{\partial x}\right|_{x=b} + h\,u(b,t) = h\,u_m$$

2. 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 $x = a$ (corresponding to $i = 1$) and $x = b$ (corresponding to $i = N_x$)
   c. Define u vector to hold solution at each time step. In *Matlab*, `u = zeros(1:Nx)`
   d. Initial condition: $u(x,0) = u_{i,0} = f(x_i) , \quad i = 1, 2, 3, \ldots, N_x .$

(*continued on next page*)

e. Stability requirement (from von Neumann stability analysis; derivation 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. There are $N_x - 2$ interior points. 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.

h. Comparison of vectorized and nonvectorized algorithms (in *Matlab*) to implement the 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 ($N_x - 2$ interior points)

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

Vectorized ($N_x - 2$ interior points)

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

   i. *Matlab* simulation

3. Next: Finite difference solution of the wave equation