3D Thermal Modeling using Finite Differences

Earlier in the course we covered 1D and 2D thermal modeling. Here we will briefly look at the 3D thermal modeling using the finite difference method. The 3D version of the transient heat conduction equation in a uniform medium without any heat generating sources is:

$$\frac{\partial T}{\partial t} = \kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right), \tag{1}$$

$$= \kappa \nabla^2 T. \tag{2}$$

As usual for the finite difference method we divide up the solution space in (x, y, z) into a grid of points with spacings $\Delta x, \Delta y, \Delta z$ in the x, y, z directions, respectively. The temperature T is then represented using a 3D array. We also divide up time t into a grid of points in time with spacing Δt. Thus, the temperature at time t_n at grid location x_i, y_j, z_k is represented as $T^n_{i,j,k}$. We then use the forward in time, center in space (FTCS) finite difference method to solve this equation for the transient temperature variations across the grid of points. The method in 3D is a straightforward extension of the 1D and 2D solutions you have already seen and the details are briefly shown below.

The second order spatial derivatives in the thermal conduction equation have central finite difference approximations:

$$\frac{\partial^2 T^n_{i,j,k}}{\partial x^2} \approx \frac{T^n_{i+1,j,k} - 2T^n_{i,j,k} + T^n_{i-1,j,k}}{\Delta x^2}, \tag{3}$$

$$\frac{\partial^2 T^n_{i,j,k}}{\partial y^2} \approx \frac{T^n_{i,j+1,k} - 2T^n_{i,j,k} + T^n_{i,j-1,k}}{\Delta y^2}, \tag{4}$$

$$\frac{\partial^2 T^n_{i,j,k}}{\partial z^2} \approx \frac{T^n_{i,j,k+1} - 2T^n_{i,j,k} + T^n_{i,j,k-1}}{\Delta z^2}. \tag{5}$$

Notice how each of these is nearly the same, but that the differences on the right side of the equation operate across the first, second and third dimensions of the 3D array T, respectively.

The time derivative on the left hand side of equation 2 is approximated using a forward

finite difference formula:

$$\frac{\partial T_{i,j,k}}{\partial t} \approx \frac{T^{n+1}_{i,j,k} - T^{n}_{i,j,k}}{\Delta t}. \tag{6}$$

Inserting the finite difference approximations into equation 2 gives:

$$\begin{aligned} \frac{T^{n+1}_{i,j,k} - T^{n}_{i,j,k}}{\Delta t} &= \kappa \frac{T^{n}_{i+1,j,k} - 2T^{n}_{i,j,k} + T^{n}_{i-1,j,k}}{\Delta x^2} \\ &+ \kappa \frac{T^{n}_{i,j+1,k} - 2T^{n}_{i,j,k} + T^{n}_{i,j-1,k}}{\Delta y^2} \\ &+ \kappa \frac{T^{n}_{i,j,k+1} - 2T^{n}_{i,j,k} + T^{n}_{i,j,k-1}}{\Delta z^2}. \end{aligned} \tag{7}$$

This equation is then rearranged to yield the time stepping update equation that solves for temperature at the next time T^{n+1} using the temperature results from the current time T^n:

$$\begin{aligned} T^{n+1}_{i,j,k} &= T^{n}_{i,j,k} + \\ &+ \kappa \frac{\Delta t}{\Delta x^2} \left(T^{n}_{i+1,j,k} - 2T^{n}_{i,j,k} + T^{n}_{i-1,j,k} \right) \\ &+ \kappa \frac{\Delta t}{\Delta y^2} \left(T^{n}_{i,j+1,k} - 2T^{n}_{i,j,k} + T^{n}_{i,j-1,k} \right) \\ &+ \kappa \frac{\Delta t}{\Delta z^2} \left(T^{n}_{i,j,k+1} - 2T^{n}_{i,j,k} + T^{n}_{i,j,k-1} \right). \end{aligned} \tag{8}$$

For stability of the finite difference method, the grid spacings need to satisfy:

$$\kappa \frac{\Delta t}{\Delta x^2} \leq \frac{1}{8}, \tag{9}$$

$$\kappa \frac{\Delta t}{\Delta y^2} \leq \frac{1}{8}, \tag{10}$$

$$\kappa \frac{\Delta t}{\Delta z^2} \leq \frac{1}{8}. \tag{11}$$

Note that the term on the right hand side of the inequality is 8 now, whereas it was 2 for 1D modeling and 4 for 2D diffusion. For the FTCS modeling of the diffusion equation, this term has form 2^d where d is the spatial dimension (1, 2 or 3).

In python, we can implement the update equation using either three nested for-loops (over the i, j and k indices), but as we saw for the 1D and 2D problems, that will run quite slowly. It is much more efficient to use array slices, since then *numpy* will use array operations for faster calculations. The array slice version of the update equation is:

```
Tnew[1:nx-1,1:ny-1,1:nz-1] = T[1:nx-1,1:ny-1,1:nz-1]  \
                           + kx*(T[2:nx,1:ny-1,1:nz-1]-2*T[1:nx-1,1:ny-1,1:nz-1]+T[0:nx-2,1:ny-1,1:nz-1]) \
                           + ky*(T[1:nx-1,2:ny,1:nz-1]-2*T[1:nx-1,1:ny-1,1:nz-1]+T[1:nx-1,0:ny-2,1:nz-1]) \
                           + kz*(T[1:nx-1,1:ny-1,2:nz]-2*T[1:nx-1,1:ny-1,1:nz-1]+T[1:nx-1,1:ny-1,0:nz-2])
```

where $kx = \kappa\Delta t/\Delta x^2$, $ky = \kappa\Delta t/\Delta y^2$ and $kz = \kappa\Delta t/\Delta z^2$.

Previously in this course we have only used 1D and 2D arrays. The 3D array used above represents temperature over the volume of space in x, y, z and thus you can now make volume plots of the temperature or you can extract 2D slices from the 3D array and use those to make surface or pcolormesh plots fo temperature in, for example, the x, y plane at a certain value of z.