HOWTOCFD23 : Day 2

Course Instructors: Niranjan S. Ghaisas and Anupam Gupta

Poisson Equation using Finite-Differences

Question 1: The 2D steady heat conduction equation,

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\dot{q}=0,$$

is to be solved using a second-order accurate finite difference method on the domain $[0, Lx] \times [0, Ly]$. The domain is discretized into $nx \times ny$ grid points. The grid points are $x_i = i \times h_x$ with $i = 0, 1, 2, ...nx-1$ and $h_x = L_x/(nx-1)$ in x and similarly in y.

We assume that homogeneous Dirichlet conditions are specified at the left, right and top boundaries, i.e. $T(x = 0) = T(x = Lx) = T(y = Ly) = 0$, and homogeneous Neumann condition is specified at the bottom, i.e. $(\partial T/\partial y)\,(y = 0) = 0$.

(a) Determine analytically the expression for $\dot{q}$ for the exact solution to be $T_{ex} = x(1-x)cos(\pi y)$ on the domain $Lx = 1$ and $Ly = 0.5$ and a constant diffusivity, $k = 1$. Note that the exact solution satisfies the boundary conditions prescribed above.

(b) Starting from the skeleton code given to you, write a code to solve the problem described in part (a) numerically.

(c) Do the calculations for computing the coefficients by hand for a small system, e.g. $nx = 5$ and $ny = 3$. Ensure that your hand calculations agree with the answers computed by the code. You need not report anything for this part. Once you have ensured correctness of the code, you can comment out any unnecessary 'printf' statements so that the output of the code remains clean.

(d) Run the code for about 5 or 6 grids such as 9×5, 17×9, 33×17, etc. For a few selected grids, plot the contours of the numerical and exact solutions.

(e) Calculate the error norm for different grid sizes and hence verify the order of accuracy. This should be reported as a plot of the error norm vs the number of grid points along one direction (or vs the grid spacing).

(f) Use timing function ‘gettimeofday’ to find the time taken for the entire code and for only the linear solver portion of the code. Note the fraction of time taken for the linear solver as a function of number of grid points.

(g) Introduce appropriate compiler directives to convert the code to an OpenMP parallel code. Use the same timing function ‘gettimeofday’ to find the time taken for the entire code and for the linear solver portion. Keeping the number of grid points the same, note these two times as a function of the number of threads (this is related to strong scaling). Keeping the number of threads fixed, note the times as a function of number of grid points (this is related to weak scaling).