MAE 267 – Project 4
Parallel, Multi-Block, Finite-Volume Methods For Solving 2D Heat Conduction

Logan Halstrom
PhD Graduate Student Researcher
Center for Human/Robot/Vehicle Integration and Performance
Department of Mechanical and Aerospace Engineering
University of California, Davis
Davis, California 95616
Email: ldhalstrom@ucdavis.edu

1 Statement of Problem

This analysis demonstrates the fundamentals of parallel computing through the numerical solution of the steady-state, two-dimensional temperature distribution of a 1m x 1m steel block with properties listed in Table 1.

Table 1: Steel Block Properties

Dimensions	1m x 1m
Thermal Conductivity	$k = 18.8 \frac{W}{m \cdot K}$
Density	$\rho = 8000 \frac{kg}{m^3}$
Specific Heat Ratio	$c_p = 500$

The demonstration of parallel computing techniques was accomplished in stages, starting with a serial (single-processor) solution of a single grid of dimensions 101x101 and 501x501, which serves as a solver basis and performance benchmark for later parallel codes.

The next stage was to divide the grid into NxM subdomains (blocks), on each of which the solution for a given iteration was calculated independently. 5x4 and 10x10 block decompositions of both previous grid dimensions were solved to demonstrate compartmentalization of solver processes, which is a necessary step for distributing processes in parallel computing.

Finally, the code will be adapted to solve multi-block decompositions on multiple processors for the 501x501 grid decomposed into 10x10 blocks running on 4 to 8 processors. This stage steps closer to that solution by performing domain decomposition and processor distribution for 5x4 and 10x10 blocks on 4 and 6 processors, and saving the decompositions to restart files to be loaded by the parallel solver.

2 Methods and Equations

The core of this demonstration code is the heat transfer solver developed in the first project, but a number of domain decomposition functions have since been included, as will be detailed in this section.

2.1 Grid Initialization

The numerical solution is initialized with the Dirichlet boundary conditions (Eqn 1) using a single processor.

$$T = \begin{cases} 5.0\left[\sin\left(\pi x_p\right) + 1.0\right] & \text{for } \; j = j_{max} \\ \left|\cos\left(\pi x_p\right)\right| + 1.0 & \text{for } \; j = 0 \\ 3.0 y_p + 2.0 & \text{for } \; i = 0, i_{max} \end{cases} \tag{1}$$

Square grids are generated according to Eqn 2 to create non-uniform spacing in both the x and y directions (with finer spacing at the larger indices). The "prime" system is then rotated by angle *rot* to create the final grid.

$$\begin{aligned} rot &= 30.0 \frac{\pi}{180.0} \\ x_p &= \cos\left[0.5\pi \frac{i_{max} - i}{i_{max} - 1}\right] \\ y_p &= \cos\left[0.5\pi \frac{j_{max} - j}{j_{max} - 1}\right] \\ x(i,j) &= x_p \cos(rot) + (1.0 - y_p)\sin(rot) \\ y(i,j) &= y_p \cos(rot) + x_p \sin(rot) \end{aligned} \tag{2}$$

2.2 Numerical Solver

The solver developed for this analysis utilizes a finite-volume numerical solution method to solve the transient heat conduction equation (Eqn 3).

$$\rho c_p \frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right] \tag{3}$$

To solve Eqn 3 numerically, the equation is discretized according to a node-centered, finite-volume scheme, where first-derivatives at the nodes are found using Green's theorem integrating around the secondary control volumes. Trapezoidal, counter-clockwise integration for the first-derivative in the x-direction is achieved with Eqn 4.

$$\begin{aligned}\frac{\partial T}{\partial x} = \frac{1}{2Vol_{i+\frac{1}{2},j+\frac{1}{2}}}\big[&(T_{i+1,j}+T_{i+1,j+1})Ayi_{i+1,j}\\ &-(T_{i,j}+T_{i,j+1})Ayi_{i,j}\\ &-(T_{i,j+1}+T_{i+1,j+1})Ayi_{i,j+1}\\ &-(T_{i,j}+T_{i+1,j})Ayi_{i,j}\big]\end{aligned} \tag{4}$$

A similar scheme is used to find the first-derivative in the y-direction.

Size of each block (no ghost nodes) Total mesh points in I-direction including block overlap points:

$$IMAX_{tot} = IMAX + (N-1) \tag{5}$$

Points in I-direction per block

$$IMAXBLK = \frac{IMAX_{tot}}{N} = \frac{IMAX+(N-1)}{N} = 1 + \frac{IMAX-1}{N} \tag{6}$$

For points in J-direction, replace I with J and N with M
Global starting index of each block in I-direction:

$$IMIN_{block} = IMIN_{global} + (IMAXBLK-1)(I-1) \tag{7}$$

where I counts blocks in the direction of N and $IMIN_{global} = 1$. The first block in the N-direction has a global starting index of 0, and IMAXBLK must be reduced by one to account for the single-point overlap at block boundaries.

3 Results and Discussion

All simulations in this analysis were run on a 501x501 point grid, once with a single block solver and once with at 10x10 multi-block solver. Fig **??** portrays the multi-block solution, which is comparable to that of the single block solver. Convergence histories of the two solvers are compared in Fig **??**. It can be seen that the two solvers are comparable in performance, both following a similar convergence path and converging at almost the same iteration.

Actual solver times are compared in Appendix A. The multi-block solver was found to be approximately 11 seconds (2.6%) faster than the single block solver. This may be due to more code streamlining in the later project. It can be expected that the speed of the multi-block solver will improve even further when linked-lists are employed to navigate neighbor boundary actions (this capability is currently functional in the code, but does not work on HPC1, so a logic-based approach was used for this project.)

4 Conclusion

Decomposing the domain introduced unforseen complications in adapting the single block solver. In some cases, it was as simple as adding a third loop for the block number, but in others (especially in updating the ghost nodes) considerable thought and error-checking was required. This implies that adapting the code for parallel processing will be an equally complicated step, so it is beneficial that we are adapting our codes modularly in stages.