University of Nevada, Reno

CS 791v: Topics: Parallel Computing
Spring 2013

Programming Assignment 3
Heat Distribution

Assigned Date
2/18/2013

Due Date
2/27/2013

Overview
In this assignment, we will write CUDA programs to determine the heat distribution in a space using synchronous iteration on a GPU. We will be solving Laplace's equation, which has wide application in science and engineering [1][2]. We will start with 2-dimensional space (square) and simple boundary conditions (walls at fixed temperatures). This program can then be modified to satisfy additional requirements.

Determining Heat Distribution by a Finite Difference Method.
Consider an area that has known temperatures along each of its edges. The objective is to find the temperature distribution within. The temperature of the interior will depend upon the temperatures around it. We can find the temperature distribution by dividing the area into a fine mesh of points, $h_{i,j}$. The temperature at an inside point can be taken to be the average of the temperatures of the four neighboring points, as illustrated in Figure 1. For this calculation, it

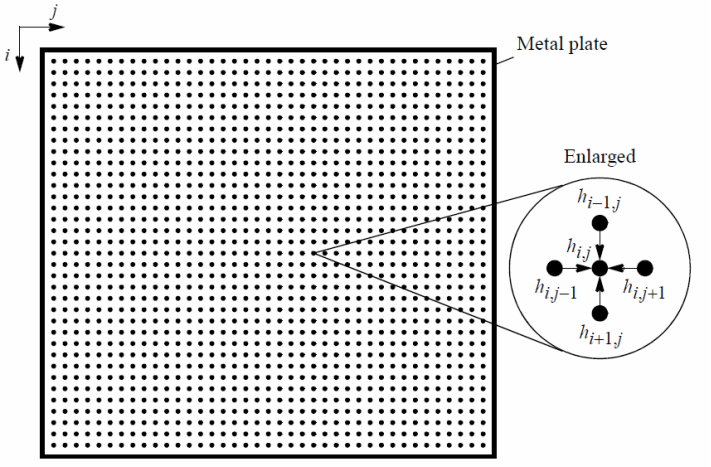


Figure 1 Heat distribution problem.

is convenient to describe the edges by points adjacent to the interior points. The interior points of $h_{i,j}$ are where 0 < i < n, 0 < j < n [(n - 1) * (n - 1) interior points]. The edge points are when i = 0, i = n, j = 0, or j = n, and have fixed values corresponding to the fixed temperatures of the edges. Hence, the full range of $h_{i,j}$ is 0 ≤ i ≤ n, 0 ≤ j ≤ n, and there are (n + 1) * (n + 1) points. We can compute the temperature of each point by iterating the equation

$$h_{i,j} = \frac{h_{i-1,j} + h_{i+1,j} + h_{i,j-1} + h_{i,j+1}}{4}$$

(0 < i < n, 0 < j < n) for a fixed number of iterations or until the difference between iterations of a point is less than some very small prescribed amount. This iteration equation occurs in several other similar problems; for example, with pressure and voltage. More complex versions appear for solving important problems in science and engineering. In fact, we are solving a system of linear equations. The method is known as the *finite difference* method. It can be extended into three dimensions by taking the average of six neighboring points, two in each dimension. We are also solving Laplace's equation.

Sequential Code:

Suppose the temperature of each point is held in an array h[i][j] and the boundary points h[0][x], h[x][0], h[n][x], and h[x][n] (0 ≤ x ≤ n) have been initialized to the edge temperatures. The calculation as sequential code could be

```
for (iteration = 0; iteration < limit; iteration++) {
     for (i = 1; i < n; i++)
          for (j = 1; j < n; j++)
               g[i][j] = 0.25 * (h[i-1][j] + h[i+1][j] +
          h[i][j-1] + h[i][j+1]);
     for (i = 1; i < n; i++) /* update points */
          for (j = 1; j < n; j++)
               h[i][j] = g[i][j];
}
```

using a fixed number of iterations. Notice that a second array g[][] is used to hold the newly computed values of the points from the old values. The array h[][] is updated with the new values held in g[][]. This is known as Jacobi iteration. Multiplying by 0.25 is done for computing the new value of the point rather than dividing by 4 because multiplication is usually more efficient than division. Normal methods to improve efficiency in sequential code carry over to GPU code and should be done where possible in all instances. (Of course, a good optimizing compiler would make such changes.)

Note: It is possible to use the same array for the updated points, thereby using some newly computed values for subsequent points (a Gauss-Seidel iteration) - this will converge significantly faster but may be difficult to implement on the GPU as it implies a sequential calculation. However a sequential version should really use Gauss-Seidel iteration for comparison purposes when computing speedup factors

Task 1 - Sequential Program

Write a C program to compute the temperature distribution inside the room shown in Figure 2 using Jacobi iteration. The room has four walls and a fireplace. The temperature of the wall is 20C, and the temperature of the fireplace is 100C. Divide the room into N x N points (including the boundaries), where N is input and can vary. The values of the points are stored in an array.

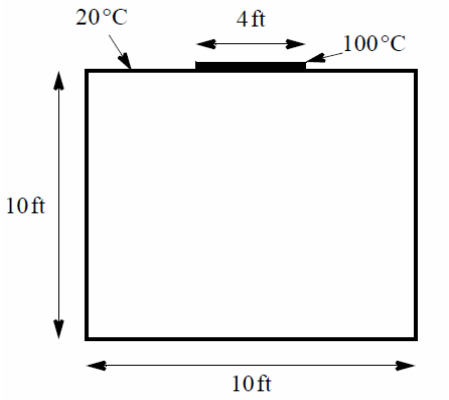


Figure 2 Room

This is due Monday 2/25

Task 2 - Basic CUDA Program

Modify the sequential program in Task 1 to be a CUDA program:

a) Use dynamically allocated memory for the data arrays (h[N][N], g[N][N]) and add keyboard input statements to be to specify N.
b) Add host code to compute the heat distribution on the host only.
c) Add code to ensure both CPU and GPU versions of heat distribution calculation produce the same correct results
d) Different CUDA grid/block structures -- Add keyboard statements to input different values for the CUDA grid/block structure:
    a. Numbers of threads in a block (T)
    b. Number of blocks in a grid (B)

    (2-D grid and 2-D blocks). Include checks for invalid input.

Task 3 Termination Detection

In the sample sequential code in Task 1, termination is set by a specific number of iterations. However the computed values may not have converged sufficiently towards the solution by that time. Re-write the CUDA code to terminate the computation when all values computed in iteration t+1 differ by those in iteration t by less than a value that is input, say e. Repeat the study in Task 2 with this CUDA program and comment on the results. Use synchronization within blocks to terminate each block separately. Note that the above does not guarantee the computed values are accurate to $\pm e$, see Figure 3. A more complex termination calculation can be done, see [3] page 176

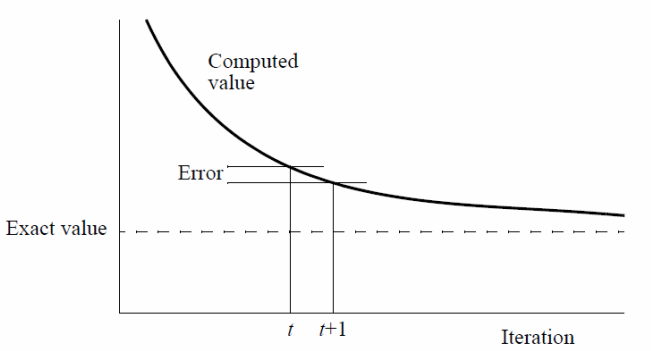


Figure 3 Convergence rate.

Grading

Every task and subtask specified will be allocated a score so make sure you clearly identify each part you did. The computational efficiency and elegance of your solutions is will be a factor in grading.

Project Requirements

- 2 versions of the code:
  - A compiled and running sequential C program
  - A compiled and running CUDA program
- Multiple timings of runs.
- Appropriate graphs

Deliverables

- Bring code and output to class for discussion next Monday.
- Have a pdf of your writeup and a zip of your source code emailed to Fred Harris and Lee Barford (DO NOT send binaries).
  - Firstname dot Lastname at … (Fred is cse, Lee is gmail)

Derivation of Jacobi Iteration Equation (from [3] page 357)

The steady-state heat distribution is governed by Laplace's equation:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

(in two dimensions). The two-dimensional solution space is "discretized" into a large number of solution points, as shown in Figure 4. If the distance between the points in the *x* and *y* directions,

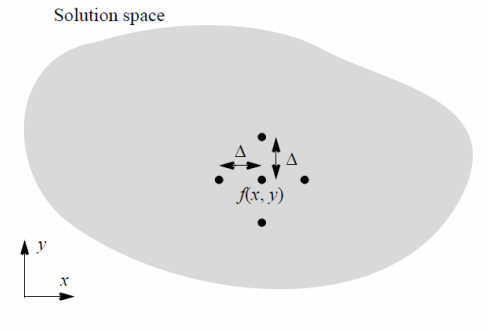


Figure 4 Finite difference method.

Δ, is made small enough, the central difference approximation of the second derivative can be used:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{1}{\Delta^2}[f(x+\Delta, y) - 2f(x,y) + f(x-\Delta, y)]$$
$$\frac{\partial^2 f}{\partial y^2} \approx \frac{1}{\Delta^2}[f(x, y+\Delta) - 2f(x,y) + f(x, y-\Delta)]$$

[See Bertsekas and Tsitsiklis (1989) for proof.] Substituting into Laplace's equation, we get

$$\frac{1}{\Delta^2}[f(x+\Delta, y) + f(x-\Delta, y) + f(x, y+\Delta) + f(x, y-\Delta) - 4f(x,y)] = 0$$

Rearranging, we get

$$f(x,y) = \frac{[f(x+\Delta, y) + f(x-\Delta, y) + f(x, y+\Delta) + f(x, y-\Delta)]}{4}$$

The formula can be rewritten as an iterative formula:

$$f^k(x,y) = \frac{[f^{k-1}(x+\Delta, y) + f^{k-1}(x-\Delta, y) + f^{k-1}(x, y+\Delta) + f^{k-1}(x, y-\Delta)]}{4}$$

where $f^k(x, y)$ is the value obtained from *k*th iteration, and $f^{k-1}(x, y)$ is the value obtained from the (*k*-1)th iteration.

By repeated application of the formula, we can converge on the solution.

Further Information

[1] Wikipedia "Laplace's equation" http://en.wikipedia.org/wiki/Laplace%27s_equation

[2] Wikipedia "Heat equation" http://en.wikipedia.org/wiki/Heat_equation

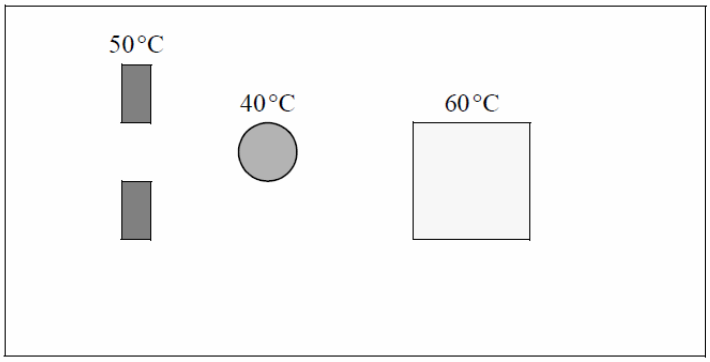
[3] Barry Wilkinson and Michael Allen, *Parallel Programming: Techniques and Application Using Networked Workstations and Parallel Computers 2nd edition*, Prentice Hall Inc., 2005.

Extra Credit

The following are currently optional - they are given for those who want to explore the topic further.

Printed Circuit Board

Repeat Task 2 but with the Figure 5. Figure 5 shows a printed circuit board with various electronic components mounted that generate heat and are at the temperatures indicated. Choose your own components and board dimensions and component placement for this problem.



Ambient temperature at edges of board = 20°C

Figure 5 Printed circuit board

3-D Computation

Repeat Task 2 but with a 3-D space, i.e. model the room in three dimensions. Take advantage of 3-D CUDA block structures.

Dynamic Changes in Heat Distribution when an Object is Inserted

Now you need to solve the Heat equation for that [2], which gives the rate of change of heat. Laplace's equation is the steady-state special case