Project I: Serial Solution of the 2D Poisson Equation

CS1645/2045

Due: September 27, 2018

September 7, 2018

1 Introduction

The two dimensional Poisson equation is a partial differential equation as follows:

$$\Delta T = S(x, y) \tag{1}$$

where S is an arbitrary source term. A practical application of Poisson's equation is to model thermal conduction on a plate with an added heat source. In this case, the variable T corresponds to the temperature of a point on the plate. In this project, only Dirichlet (fixed temperature) boundary conditions will be considered on a rectangular plate.

2 Algorithm Development

Develop an algorithm to solve Poisson's equation. Recall that second derivatives can be approximated by a finite difference to second order accuracy as follows (example is written for the x direction):

$$\frac{\delta^2 T}{\delta^2 x} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{(\Delta x)^2} + \mathcal{O}((\Delta x)^2) \tag{2}$$

2.1 Norms

The purpose of a norm is to measure the "length" of some arbitrary vector. In our project, two vectors that we will measure will be the difference in the temperature solution from one iteration to the next ($\Delta T_k = T_{k+1} - T_k$), and the difference between our computed solution and the exact solution ($T_{conv} - T_{exact}$). The former is to be used to determine when to stop the iterative solution of the matrix system, and the latter is to be used to determine how close our computed solution is to the exact solution, if we are fortunate enough to know what that exact solution is. A p-norm is defined as

$$||v||_p = (\Sigma |v_i|^p)^{(1/p)} \tag{3}$$

There are a few ways to define a norm, but for this project, we will always use an infinity norm (also called a max norm). If one chooses $p = \infty$:

$$||v||_\infty = max(|v_i|) \tag{4}$$

Note that the norm is defined over all entries of the vector of interest.

3 Implement

Write source code, from scratch, to implement the algorithm stated from Part 1. Your code should be written to take as input the number of points to use in the x and y directions. You may use any numerical algorithm that you choose to solve the discretized system, but this algorithm must be written by you (i.e., do not download a canned linear solver). It is strongly suggested to use an iterative relaxation algorithm, such as a point Jacobi or Gauss-Seidel. Please take care to write

your code in a careful and coherent manner. You will reuse this code in subsequent parallelization projects using different paradigms. You should write this code in C, C++.

4 Verification

Verify your implementation by solving Poisson's equation on the domain

$$0 \leq x \leq 2 \tag{5}$$
$$0 \leq y \leq 1$$

subject to the following boundary conditions

$$T(0, y) = 0 \tag{6}$$
$$T(2, y) = 2e^y$$
$$T(x, 0) = x$$
$$T(x, 1) = xe \tag{7}$$

with the following source term

$$S(x, y) = xe^y \tag{8}$$

Note that the exact solution to the PDE is $T = xe^y$.

- Verify that the discretization is second order accurate by running on successively finer grids and plotting the normed difference between the calculated and exact solution. Suggestion: use 5x5, 11x11, 21x21, 41x41, 51x51, 101x101, and 201x201 grids. Choose a target convergence level (such as $10e - 12$) and use this consistently for each run.
- Verify that your code works with grids that do not have the same number of points in the x and y directions. Repeat part a of this section with grids of 11x5, 21x11, etc.

5 Problem Solving

Solve Poisson's equation on the domain

$$0 \leq x \leq 1 \tag{9}$$
$$0 \leq y \leq 1$$

subject to the following boundary conditions

$$T(0, y) = 0 \tag{10}$$
$$T(1, y) = 0$$
$$T(x, 0) = 0$$
$$T(x, 1) = 0 \tag{11}$$

with the following source term

$$S(x, y) = 0.2 \tag{12}$$

Using a sequence of finer grids, show that your algorithm attains grid convergence (select a consistent location such as x = 1/2, y = 1/2; record the computed temperature of this location as the grid resolution increases). Plot the contours for at least one of the solutions. You can also use l_2 or l_∞ norm to monitor the convergence.

6 Performance

Time your source code to show how the computational cost grows with time. Push the limits of the grid size as much as you can.

7 Bonus Points

If you implement both Jacobi and Gauss-Seidel algorithms and compare their time to solution for various grid sizes you will get an extra 10% bonus.

8 Administrative Information

Submit at least the following for your project:

- presentation of the algorithm
- source code
- include all the requested results in this project

Please do not include the object files or the executable in your email. Needless to say the report should be in pdf.