Boston University

EC500: Final Project

Ariya Shajii, Huy Le, Winston Chen

April 26, 2016

1 Introduction

In this project, we solve in two dimensions the heat equation

$$\rho c_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = \dot{q}, \tag{1.1}$$

where

- k is a constant taken to be 1,
- ρ is the density of the medium (assumed to be constant),
- c_p is the specific heat of the medium (assumed to be constant),
- $\dot{q}$ is the volumetric heat flux as a function of spacial coordinates (x, y), and
- T is the temperature of the material as a function of spatial coordinates (x, y) and of time t.

In discrete form, the equation can be written as follows:

$$\rho c_p (T_{x,y,t+1} - T_{x,y,t}) - k(T_{x+1,y,t} + T_{x-1,y,t} + T_{x,y+1,t} + T_{x,y-1,t} - 4T_{x,y,t}) = \dot{q}. \tag{1.2}$$

The time step and spatial step have both been taken to be 1 in the finite difference approximations.

We now solve this problem using three different approaches:

- Red-black iteration, parallelized with OpenMP and MPI
- Conjugate gradient method
- Using a triangular lattice instead of a conventional square lattice

2 Parallelized Red-Black

3 Conjugate Gradient Method

In order to use conjugate gradient (CG), we must write our problem in the form $\mathbf{A}\vec{x} = \vec{b}$ for vectors $\vec{x}$, $\vec{b}$ and matrix $\mathbf{A}$. In this case, $\vec{x}$ will be our temperature T, $\vec{b}$ will be our volumetric heat flux $\dot{q}$ and $\mathbf{A}$ will encode the left-hand side of Equation 1.1. Note that we take $\frac{\partial T}{\partial t} = 0$ since we are only interested in the steady-state solution when using CG. We are therefore left with:

$$\mathbf{A}T_{x,y} = 4T_{x,y} - (T_{x+1,y} + T_{x-1,y} + T_{x,y+1} + T_{x,y-1}) = \vec{b} = \dot{q}. \tag{3.1}$$

Note that we make the time variable t implicit, since we no longer need it directly. The CG algorithm can now be applied as follows [1]:

Require: $\mathbf{A}$, $\vec{b}$, ϵ
Ensure: T
  $T \leftarrow \vec{0}$
  $\vec{r} \leftarrow \vec{b} - \mathbf{A}T$ {residual}
  $\vec{p} \leftarrow \vec{r}$
  loop
    $\alpha \leftarrow \frac{|\vec{r}|}{\vec{p} \cdot \mathbf{A}\vec{p}}$
    $T \leftarrow T + \alpha\vec{p}$
    $\vec{r}_{\text{new}} \leftarrow \vec{r} - \alpha\mathbf{A}\vec{p}$
    if $|\vec{r}_{\text{new}}| < \epsilon$ **then**
      return T
    $\beta \leftarrow \frac{|\vec{r}_{\text{new}}|}{|\vec{r}|}$
    $p \leftarrow \vec{r}_{\text{new}} + \beta\vec{p}$
    $\vec{r} \leftarrow \vec{r}_{\text{new}}$

Algorithm 1: Conjugate gradient algorithm

We take our initial guess T_0 to be zero everywhere. All that Algorithm 1 requires us to have is a way to perform dot products and a way of applying $\mathbf{A}$ to a vector. The former is fairly trivial and the latter is given by Equation 3.1. With these fundamental operations at hand, it is straightforward to implement Algorithm 1 in code.

[1] Adapted from `https://en.wikipedia.org/wiki/Conjugate_gradient_method#The_resulting_algorithm`

Quick note on boundary conditions: We incorporate our boundary conditions in $\vec{b}$ and apply $\mathbf{A}$ only to the interior points of T – that is, points that have exactly four neighbors.

3.1 Results

4 Triangular Lattice