Math 447

Due date: 2015 March 18 (Wednesday)

Project 2: Iterative Methods and WiFi Routers

1. Introduction

Where should you put a wifi router to get the best signal? This question was explored in an entertaining blog, which can be found at:

http://jasmcole.com/2014/08/25/helmhurts/

There is no requirement to read the article there, but it may give you some context. Some parts of the article are reproduced here for purposes of illustration.

We can approach this problem using some of the tools we have learned in class. You don't have to understand the physics or the underlying differential equations for this project (although they are fascinating, and may grab your curiosity). For now, let us concentrate on the following issue: We will have a linear system of size $39,204 \times 39,204$ to solve! This system is enormous. Since the numbers we use (double precision floating-point numbers) take up 8 bytes (8B), just to *store* the matrix takes

$$39204 \times 39204 \times 8\text{B} = 12295628928\text{B} \approx 12.3 \text{ GB !}$$

If we were to use Gaussian elimination on this problem, it would take about

$$(2/3)N^3 = (2/3) \times 39204^3 = 40,169,819,707,776 \approx 40 \text{ trillion}$$

operations to perform! However, this matrix has only 195228 non-zero elements (which can be store in about 1.5 MB, over a 99.98% reduction in size!), and therefore is very *sparse*. This means it is an excellent candidate for modern iterative schemes.

2. Background

This part[1] tells you (a little bit) about where the matrix comes from. You can skip it if you like.

Wifi signals are just electromagnetic waves. Examples of electromagnetic waves include light (infrared, visible, and UV), radio waves, microwaves, X-rays, gamma rays, and many others. Waves are governed by a partial differential equation known as the wave equation, which looks like $E_{tt} = c^2\nabla^2 E + f$, where $\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2}$ (in 2D), c is the wave speed, $E = E(x, y, t)$ is the electromagnetic field (i.e., the waves carrying the wifi), and $f = f(x, y)$ is the input (i.e., the wifi signal from the router). Waves from wifi signals can be modeled as "standing waves," which we can think of as eigenfunctions of the steady-state of the equation. Steady-states have no time-dependence ($E = E(x, y)$), so $E_{tt} = 0$. The eigenfunction equation for the steady-states (with $f = 0$) would then be $c^2\nabla^2 E = \lambda\nabla^2 E$. Rewriting in more relevant variables, and adding back in a source term we arrive at

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} + \frac{k^2}{n^2} E = f \tag{2.1}$$

where k is the wave-number ($k = 2\pi/\ell$ where ℓ is the wave-length), and $n = n(x, y)$ is the refractive index of the material the wifi wave is in (i.e., the air or a wall or something). Equation (2.1) is called the Helmholtz equation.

[1] Acknowledgement: Much of this is taken from the article at http://jasmcole.com/2014/08/25/helmhurts/.

OK, but how do we solve an equation like this? In general, it can be quite hard, since n is a non-constant coefficient. Therefore, let us try to solve it in the computer! First, we need a numerical scheme. Let's consider a square room, 5×5 meters, where the wifi signal is, and imagine a rectangular grid over the room, with each point on the grid indexed by i,j. Let us write $E(i,j)$ for the value of E at the point i,j, and let Δx and Δy be the distance between grid points. We can approximate a derivative of E in two ways:

$$\frac{\partial E}{\partial x} \approx \frac{E(i+1,j) - E(i,j)}{\Delta x} \text{ or } \frac{\partial E}{\partial x} \approx \frac{E(i,j) - E(i-1,j)}{\Delta x}$$

Since the second derivative is the derivative of the derivative, we can approximate it by combining the above two formulas:

$$\frac{\partial^2 E}{\partial x^2} \approx \frac{\frac{E(i+1,j)-E(i,j)}{\Delta x} - \frac{E(i,j)-E(i-1,j)}{\Delta x}}{\Delta x} = \frac{E(i+1,j) - 2E(i,j) + E(i-1,j)}{(\Delta x)^2}$$

This is called the "centered difference" approximation of the second derivative, and it turns out to be a good approximation to use in many cases. We can do the same thing with the derivative in y, except that the i index will be fixed and j will vary. Equation (2.1) becomes

$$\frac{E(i+1,j) - 2E(i,j) + E(i-1,j)}{(\Delta x)^2} + \frac{E(i,j+1) - 2E(i,j) + E(i,j-1)}{(\Delta x)^2} + \frac{k^2}{(n(i,j))^2} E(i,j) = f(i,j)$$

Notice that this is a linear system in $E(i,j)$! The system can be written like this:

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots \\ \cdots & \frac{1}{(\Delta x)^2} & \cdots & \frac{1}{(\Delta y)^2} & \frac{-2}{(\Delta x)^2} + \frac{-2}{(\Delta y)^2} + \frac{k}{(n(i,j))^2} & \frac{1}{(\Delta y)^2} & \cdots & \frac{1}{(\Delta x)^2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots \\ E(i-1,j) \\ \vdots \\ E(i,j-1) \\ E(i,j) \\ E(i,j+1) \\ \vdots \\ E(i+1,j) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ f(i-1,j) \\ \vdots \\ f(i,j-1) \\ f(i,j) \\ f(i,j+1) \\ \vdots \\ f(i+1,j) \\ \vdots \end{pmatrix}$$

If we have a grid of size $N \times N$, then we have N^2 grid points. We have not discussed boundary conditions (and we will avoid that here), but this means that E is already known to be zero at some of the grid points, but we end up with $(N-2)^2$ points at which E is unknown. Therefore, the size of the matrix is $(N-2)^2 \times (N-2)^2$ for a total of $((N-2)^2)^2 = (N-2)^4$ entries! For this problem, we need a resolution of at around $N = 200$, and $(200-2)^2 = 39204$, which is where the number came from in the introduction. Note that most of the entries in the matrix are zero; there are at most five non-zero entries in each row!

In real-world problems, a resolution of 200×200 is not very large, but it already gives rise to a matrix so large that it would be impossible to solve without a computer *and* modern numerical methods. Note that for 3D problems, the number of unknowns gets even worse. We would have $(200-2)^3$ unknowns, which is over 7 million!

3. Objective

Your task to solve this system (or try to) using three methods:

- QR-factorization (actually, this will fail, but we will try it to see what happens)
- Gauss-Siedel iteration
- Conjugate Gradient

I already wrote a code, called `Helmholtz.m`, which does everything except for solving the linear system. (Coding the matrix is slightly tricky, but Matlab has some nice tools for doing this.) Take my code, insert your solvers, and time them using the Matlab functions `tic` and `toc`. **Briefly report your results** on paper, comparing the run times as the resolution N increases (eventually, for large enough N, the code will not run very fast on your computer, so don't increase resolution too close to that point). It is up to you to decide how to make the comparison, but for example, a log-log graph comparing run-time vs. N would be appropriate. (Hint: Make Matlab compute the run times for you in a loop over N, and use the function `loglog` instead of `plot`.)

4. Develop and Test your code

You already wrote a QR-solver in your last project, so just update your `linearQRsolver.m` so that it can be called as a function.

You need to write a Gauss-Siedel solver and a Conjugate Gradient solver. Each one should have its own testing code, just like in project 1, which checks the error for a manufactured solution. By this, I mean randomly generate `A` and `x` (let `A` be pretty small, like 5×5 or so), compute `b=A*x`, then use your linear solver to solve for an approximate `x`, and compute `norm(x - x_approx)`. You will know you are successful when the error is sufficiently small.

5. Challenge (not for the faint-of-heart)

Try to modify the Helmholtz.m code. Can you move the "router signal" around and draw any firm conclusions about where you should put the wifi signal? Can you find a better way to add in walls? Is the physics solid, or should it be done differently? This part is not required, but if you have some good findings (not just minor tinkering), or can significantly improve the code, it can be worth up to an additional 20 bonus points above the usual 100.

6. Instructions for turning in the project

(1) **Make sure your code runs!** Also, try as hard as possible to clear our the warnings and errors. You can see these by hovering over the little orange and red marks in Matlab's scroll bar (just to the right of your m-file). When they are all clear, a green box will appear at the top of your scroll bar. Strictly speaking, you don't have to have a green box to turn in your code, but trying for it is a good idea. To get credit for the project, your code must run!
(2) **Properly indent your code.** To do this, on all your `*.m` files, **hit `CTRL+A` (to select all the text) and then `CTRL+I`** (to get your code properly indented). If you skip this step, I will ask you to resubmit your code.
(3) Send me a very brief email from your university account with:
    (a) The **title:** Math 447 Project 1 Submission.
    (b) Your **name**, and the names of any collaborators.
    (c) Your `*.m` **files** attached.
    (d) ***Brief* instructions on how to run your code** (only if necessary, I know how to click "play (▶)", and so on).
    (e) **Nothing else.** If you have a question for me, send it in a separate email with a different title.
(4) Just to be clear, the materials you should submit for this project are:
    (a) `cg.m` (emailed in zip file)
    (b) `cgTest.m` (emailed in zip file)
    (c) `gs.m` (emailed in zip file)
    (d) `gsTest.m` (emailed in zip file)
    (e) Possibly some other code for additional testing, the challenge problem, etc.
    (f) A **paper** copy of a write-up which *briefly* discusses your findings on using your solvers in Helmholtz.m. Use scientific language and reasoning. This is not the place for hand-waving or Please make it one-page, one-side. I would like it in paper so I have something to hand back to you.

Please send all the code in a **single email**, compressed zip file, with your last name (family name) as the title of the zip file. For example, John Smith would submit `Smith.zip`.

6.1. **Collaboration.** It is OK to work with somebody else, but if you do so, **you must state it clearly in your submission email**. In general, work together to get ideas and check each other's code, but write the code yourself. This is a great way to learn. Turning in somebody else's work (whether another classmate's, something you found online, etc.) will be dealt with according to the university's academic dishonesty policy. Plus, you will miss out on learning some awesome computing skills, which would be no fun, and getting a computer to solve hard matrix problems should be fun! (I guess it is no fun for the computer, but computers are machines and have no emotions, so we probably shouldn't feel bad about making them do our computations.)

Good luck and have fun!