UCB Math 128B, Spring 2016: Programming Assignment 2

Due March 17

A Linear Elasticity Model Problem

In this assignment, you will study inverse iteration for computing eigenvalues of a linear elasticity problem. Some utility functions in MATLAB are available on the course web pages, for creating the matrices and visualizing the solutions. You can essentially use these functions as black-boxes, only understanding what they do. Here is what you need to know:

- `K=assemble(n)` Creates a matrix K (the "stiffness" matrix for the problem). We will study the eigenvalue problem $Ku = \lambda u$ for the problem size `n=20`.
- `[K,f]=mkmodel(n)` Creates a matrix K and a vector f, implementing a typical boundary-value model problem. We will study the linear system of equations $Ku = f$.
- `qdplot(u)` Plots the displacement field u
- `qdanim(u)` Animates the displacement field u (only makes sense for eigenfunctions)

Please hand in short and concise MATLAB codes. You do not have to hand in code for plotting or setting up calculations, or any displacement plots with `qdplot`.

Inverse Iteration and Rayleigh Quotient Iteration for Eigenvalues

1. Write MATLAB code that performs inverse iteration on a matrix K with shift μ. Initialize $\boldsymbol{v}^{(0)}$ with `randn`, and compute and store all the Rayleigh quotients $\lambda^{(k)}$. Iterate until the difference between $\lambda^{(k+1)}$ and $\lambda^{(k)}$ is less than 10^{-14}.

You can create the shifted matrix by `K-mu*speye(N,N)`, where `N=size(K,1)`, and you can solve the linear systems using the MATLAB \ command. You might also want to plot the eigenfunctions in each iteration using `qdplot(v); drawnow;`.

2. Run the example in the help-text for `assemble`. The true eigenvalues are on the diagonal of `D`, or `d=diag(D)`. Based on the true eigenvalues choose four shifts μ according to below. For each shift, run the code you wrote in **1** and plot the error $e^{(k)} = \lambda^{(k)} - \lambda$ versus the iteration number k using `semilogy` (you can plot all four curves in the same graph). For the true eigenvalues λ you can use the value in `d` or the last iterate $\lambda^{(k)}$. Also make a simple estimate of the convergence rates (the factor C in $e^{(k+1)} = Ce^{(k)}$), and compare with the expected convergence rate (which you need the true eigenvalues in `d` for). Choose shifts that are:

a) Slightly below a distinct eigenvalue (but not too close!)

b) As in a) but above the eigenvalue. The Rayleigh quotient should behave as in a), but why does the eigenfunction flip back and forth?

c) Almost right between two distinct eigenvalues (about 10% closer to one of them)

d) Close to an eigenvalue with multiplicity > 1. Does it seem to have trouble finding the eigenvalue? You do not have to explain why.

3. Change your code so it performs Rayleigh quotient iteration instead of inverse iteration (a very small change!). Run the four shifts again and plot the errors versus the iteration number. The expected convergence is *cubic* – see if you can observe that the number of digits are tripled at some iteration.