Math 116 : Homework 1

due Thurs Jan 19; I suggest you spread out the work

Some of your homework time is devoted to getting started with either Matlab (a versatile and powerful package used by a large fraction of the applied math and science community) or an alternative environment if you are already used to one (Maple, Mathematica, python, C, java, etc). For Matlab help, always start with our course website `http://math.dartmouth.edu/~m116w06`, then ask friends, or myself. Install Matlab (or equivalent) on your personal machine, *e.g.* from `http://hydra.dartmouth.edu/matlab/downloads`. Susan Schwarz can help. Instead you could work at computer labs where Matlab is already installed. Possibly attend the 2 Matlab workshops on Jan 5 (intermediate), Jan 11 (beginner), see
`http://diglib.dartmouth.edu/classes/calendar.php?MAIN_CALENDAR_ID=41`

1. Let's investigate a finite-dimensional linear operator $A : \mathbb{R}^N \to \mathbb{R}^N$. Given N, fill the $N \times N$ matrix A with entries $a_{ij} = (1/N)e^{-[5(i-j)/N]^2}$. This is a symmetric Toeplitz matrix. [Hint: `toeplitz` may help, and you can visualize it with `imagesc(A);colorbar;`]
    (a) Choose $N = 20$. Explore and describe A's action on vectors $\mathbf{b}$ of the form $[0, 0, \ldots, 0, 1, 0, \ldots, 0]^T$. You may want to `plot(A*b, '+');`
    (b) What do the eigenvalues $\{\lambda_j\}$ of A do as N grows from *e.g.* 5 to 50?
    (c) Solve the linear system $A\mathbf{x} = \mathbf{b}$. You can do this via `x = A\b;` Use $\mathbf{b}$ of the above form. Describe how $||\mathbf{x}||_2/||\mathbf{b}||_2$, which can be thought of as the 'amplification factor' of A^{-1}, changes with N. How does this compare to the ratio λ_N/λ_1 at the same N? Plot an example $\mathbf{x}$.
    (d) In an ideal world the residual $||A\mathbf{x} - \mathbf{b}||_2$ should vanish. How does it in fact change with N? Why?
    (e) Now replace A by $A - I$, where I is the identity. Do the increasing-N effects in the last two parts change, if so how? Why?

2. Let's build (and test) some low-level *modules* (subroutines) you'll need for boundary integral methods; this will help you stay sane later.
    (a) Write subroutine which returns the value at $x \in \mathbb{R}^2$ of a dipole source $\partial\Phi(x, y)/\partial n_y$ at located at $y \in \mathbb{R}^2$, given the two components of the unit normal vector (dipole direction) $\hat{n}_y$. A suggested interface is `[u] = dipole(x1, x2, y1, y2, ny1, ny2)`.
    (b) Modify it so that when a *list* of x values is input, a similar list of output values are returned (*e.g.* in C you'll need to add extra arguments such as the length of the arrays).
    (c) Set up a square grid of n by n points (hint: if `g` is a list of numbers use, `[x1, x2] = meshgrid(g, g);`). Feed this to `dipole`, and plot the result for *e.g.* $n = 50$ (hint: `imagesc(g, g, u); set(gca, 'ydir', 'normal');` may help). Adjust the color scale (or use contours) to make a nice labelled plot.

3. Finally here we test our first layer potentials.
    (a) Modify `dipole` so that it produces the field due to the sum of *many* dipoles of given strengths. This means the input y may now be a list of locations, and you'll need *e.g.* `tau` giving the corresponding list of dipole strengths.

(b) Make a y list which is the coordinates of N boundary points of the unit disk, equally spaced in angle θ. Feed this to `dipole`, with all dipole strengths unity, and plot the resulting field over a region including the disk. How does the value inside scale with N? (hint: keep n small otherwise this'll be very slow!).

(c) The above is (close to) an example of

$$u(x) = \int_{\partial\Omega} \frac{\partial\Phi(x,y)}{\partial n_y} ds_y = \begin{cases} -1, & x \in \Omega \\ 0, & x \in \mathbb{R}^2 \setminus \overline{\Omega} \end{cases}$$

which is the field due to a unit density double layer. You are approximating the integral by a sum, as in $\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^N f(x_i)$. Notice $\frac{b-a}{N}$ is the interval length associated with equally-spaced quadrature points x_i. Scale your source strength accordingly: you should now find the field inside is close to `-1`.

(d) Now for the fun: change your list of N boundary points to describe the C^∞ radial function $f(\theta) = 1 + 0.3\cos 3\theta$. Keep the same angles as before, so your N quadrature points are no longer equally spaced in arclength. Estimate to $O(1/N^3)$ the correct arclength associated with each point; you may want to call this list `ds`. Finally put this together to plot the field due to a unit double layer density on this curve, checking it converges to `-1` inside, 0 outside. How do errors converge with N? Where in Ω is it least accurate?

You may find the computations quite slow in Matlab; I encourage you to '*vectorize*' if you have time (more later. . .)