Fast Methods for PDE and Integral Equations (Spring 2012)
P Set #2 - due April 19

1) Consider N source points y_j drawn uniformly at random in the unit square $[0,1]^2$, and N target points x_i drawn uniformly at random in some square to be specified. Consider the kernel $G(x,y) = \log(\|x-y\|)$. Compute the relative ϵ-rank of the interactions, i.e. the minimum rank of a decomposition $G \simeq UV^T$, so that $\|G - UV^T\|_2/\|G\|_2 \leq \epsilon$, with $\epsilon = 10^{-5}$ and for the following scenarios:

(a) $N = 100$ and the target square is $[2,3] \times [0,1]$. (Use the SVD.)

(b) $N = 1000$ and the target square is $[2,3] \times [0,1]$. How does your answer compare to that obtained in point (a), and why?

(c) $N = 100$ and the target square is $[1.25, 1.5] \times [0, 0.25]$. How does your answer compare to that obtained in point (a), and why?

(d) Again, $N = 100$ and the target square is $[2,3] \times [0,1]$, but use the explicit 2D multipole expansion formula as seen in the review paper by Beatson and Greengard (link from the class website). Find the equivalent of the ϵ-rank, i.e., p such that the resulting error $\|G - UV^T\|_2/\|G\|_2 \leq \epsilon$. How does your answer compare to that obtained in point (a), and why?

2) For this problem, we developed a set of matlab functions to help you understand aspects of fast methods applied to some of the integral equation methods we have discussed in class. Physically, the method will be used to answer the question: How much charge must we put on a unit plate to raise its voltage from zero volts to one volt? This quantity is also referred to as the capacitance of the unit plate.

We can solve for the charge density on the plate surface by solving the integral equation

$$\psi(r) = \int_{surface} \frac{\sigma(r')}{\|r - r'\|} dS'$$

where the potential $\psi(r) = 1$ for all points r on the conductor surface and σ is the unknown charge density. We have developed a set of matlab codes that breaks the unit plate into many small charged panels, like the methods we discussed in class, and then solves for the charges to determine the capacitance.

The files are: **calccap.m:** This is the main routine which calculates capacitance.
calcp.m: This file contains the routine which analytically computes

$$\int_{panel} \frac{1}{\|r - r'\|} dS'$$

plategen.m: This routine generates the panels that represent the unit plate.
gencolloc.m: This routine computes panels centroids.
collocation.m: This routine sets up the collocation matrix. That is, the matrix that relates panel centroid potentials to panel charges.

circulant.m: A routine useful for generating circulant matrices.

a) You can generate the matrix associated with representing the plate with a 5x5 array of panels, as well as associated approximation to the unit plate capacitance, by typing (in matlab)

[C, matrix] = calccap(5,5)

and then you can examine the matrix using, for example, the matlab mesh command. The capacitance of a unit plate is approximately 40.8×10^{-12}, please estimate how many panels you will need to get within 0.1 percent of this answer. Be forwarned, if you type calccap(100,100), you will generate a 10,000 panel discretization and a 10,000x10,000 dense matrix. This will take a LONG time.

b) The code we have given you generates dense matrices and solves them with direct factorization, but you have learned many techniques that can be used to dramatically accelerate our code. Although not really physical, first consider discretizations of the plate in to strips by calling calccap(5,1), calccap(10,1), ... calcap(100,1), etc. If you examine the resulting matrix, you will see it is a Toeplitz matrix. Modify calccap to take advantage of this property to accelerate forming the matrix (hint: consider reducing either the number of collocation points, or the number of panels, in the call to collocation() in calccap).

c) Try using gmres to solve for the charges in the strip discretization case instead of direct factorization, and try solving for the charges for both a uniform and a random set of collocation point potentials. Matlab's gmres function will return information about convergence, use it to determine how the number of iterations scales with the number of strips.

d) We know we can use the FFT to quickly invert circulant matrices, but not Toeplitz matrices. So, for the strip-discretized case, try using the circulant function we supplied to generate a preconditioner for the strip-discretized Toeplitz matrix. How does GMRES convergence scale with number of strips in your circulant-preconditioned case?

e) (Challenging, bonus) Now try to generalize (b)-(d) to the case of a 2-D array of panels (such as generated by calling calccap(10,10)). In this case, the generated matrices will be block-Toeplitz with Toeplitz blocks. If you take advantage of generalizations of all the above tricks, how large an $n \times n$ array of panels can your code handle? How does that compare to the time and memory required for our original code? Can you use a sufficient number of panels to compute the capacitance to 0.1 percent?