UC Berkeley Math 221, Spring 2018: Problem Set 7

Due April 26

This last problem set is different from the previous ones, in the sense that you can pick one of several proposed problems. They are slightly more extensive problems, and also more open-ended, but the total work should not be more than a typical problem set. In addition, you may suggest a different topic that interests you if it is relevant to what we have covered in class. If you are interested in this, please send a short description of your proposed problem by *Thursday April 12* to persson@berkeley.edu, and we will let you know if it is appropriate.

Do only one of the problems below:

1. *The QR algorithm:* (Trefethen/Bau 29.1) The following details are required for full credit:
   - In (a), make sure you make advantage of the sparsity so the resulting algorithm has the optimal operation count (26.2).
   - In (b), make sure you take advantage of the sparsity using Givens rotations or 2×2 Householder reflections (see exercise 28.2).
   - Also do a new part (f): Run and time your codes for a sequence of increasing matrix sizes, and confirm the expected computational complexities for both the tridiagonal reduction and for finding the eigenvalues.

2. *Bisection and divide-and-conquer:* Implement two codes for computing eigenvalues of a symmetric tridiagonal matrix A:
   (a) Method of bisection: Find all the eigenvalues in a given interval $[a, b]$, using a standard bisection method and Sturm sequences.
   (b) Divide-and-conquer: Find all the eigenvalues of A, by solving the secular equation using Newton's method.

   Run tests and timings that verify the expected computational complexities of your codes, and compare the accuracy with MATLAB's eig function.

   Reference: Trefethen/Bau, chapter 30.

3. *The Multigrid Method:* Implement a code that solves the Poisson model problem on a uniform 2D grid, for arbitrary right-hand sides b, using the multigrid method. Choose appropriate parameters, and try to verify the linear complexity in $N = n^2$.

   Reference: Demmel 6.9, MIT lecture notes

4. *The Fast Multipole Method (FMM):* Implement a code that takes a list of n particles in 2D, with given x_i, y_i-coordinates and charges q_i, and computes the all-to-all potentials $V_i = -\sum_{j \neq i} q_j \log(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|)$ using the Fast Multipole Method. Use complex arithmetic and assume the points are evenly distributed in space. Test your code on random input data and confirm the linear complexity in n.

   If you prefer an easier assignment, implement the code in 1D only.

   Reference: Greengard/Rokhlin, JCP 1987.

Turn page $\longrightarrow$

5. *Communication Avoiding Algorithm:* (This question will require a low-level programming language, such as C, C++, or Fortran). Implement a communication-avoiding algorithm for matrix-matrix multiplication. Compare performance with BLAS/LAPACK and your own non-communication avoiding algorithms. Perform experiments to find the optimal block size for the blocked versions of the algorithm and to study how performance varies for different size systems.

   Reference: To be posted on the course web page.

Please hand in a write-up which includes a description of your problem, which decisions you made in terms of algorithms, parameters, test problems, etc, and some results. Also include the actual code, possibly in an appendix.