CES 703 Assignment #2 Due date: Monday, February 4, 4:00pm

Dr. Imre Pólik (40+10 points)

Electronic submission is preferred. Do not print anything that's generated electronically. Whatever can be submitted electronically, please do it that way. Please submit the source code and the output along with the solutions.

For the programming problems you can use your favourite language (C, Fortran, Matlab, Python, Java, etc.) provided that it can handle the problem. If you choose a compiled language please provide instructions for compilation (possibly a Makefile).

---

1. (20 points) Consider $Ax = b$ where

$$A = \begin{pmatrix} a_0 & a_1 & \dots & a_n \\ a_1 & a_0 & & \\ \vdots & & \ddots & \\ a_n & & & a_0 \end{pmatrix} \tag{1}$$

is an arrowhead matrix. When is this system solvable? What is the best way to solve this linear system? Investigate the solution methods (direct and iterative) discussed in class and evaluate them based on their performance and memory requirement. What is the cost of the solution?

Now consider the system $A^k x = b$ for $k = 2, 4, 100, n/2$. How does the answer change? Your input consists of A, k (not A^k) and b.

How does the answer change if you get A^k instead of A but you know about the structure? What would you do if you had both A and A^k available? How does the preferred method change as k and n grows?

2. (20 points) Find the best way to solve a linear system with the following coefficient matrix:

$$\begin{pmatrix} a_1 & b_1 & 1 & \dots & \dots & 1 \\ b_1 & a_2 & b_2 & 1 & \dots & 1 \\ 1 & b_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & 1 & \ddots & \ddots & \ddots & 1 \\ \vdots & \vdots & \ddots & \ddots & a_{n-1} & b_{n-1} \\ 1 & 1 & \dots & 1 & b_{n-1} & a_n \end{pmatrix} \tag{2}$$

Typically, $|b_i| \le n$ and $3n \le |a_i| \le n^2$. The dimension of the problem ranges from 10 to 10000 (or as large as you can handle on your computer). Consider both direct and iterative methods on one and more CPUs. Support your claims with test runs. Use existing implementations if possible.

Bonus: (10 points) Implement the multifrontal Cholesky algorithm in Matlab. For simplicity, assume that the size of the matrix is a power of 2. Compare the performance and scaling of your code to the built-in Cholesky factorization function `chol`. (Hint: Use a recursion.)