Homework 6

Please answer the following questions in complete sentences in a typed, clearly prepared manuscript and submit the solution by the due date on Blackboard (Monday, October 23rd, 2017, around 4-5 am)

Problem 0: Homework checklist

- Please identify anyone, whether or not they are in the class, with whom you discussed your homework. This problem is worth 1 point, but on a multiplicative scale.
- Make sure you have included your source-code and prepared your solution according to the most recent Piazza note on homework submissions.

Problem 1: Direct Methods for Tridiagonal systems

In class, we said the definition of a sparse matrix was one with enough zeros to take advantage of it. In this problem, we’ll show how to take advantage of tridiagonal structure in a matrix.

Let \( A \) be a symmetric, tridiagonal, positive definite matrix:

\[
A = \begin{bmatrix}
\alpha_1 & \beta_1 \\
\beta_1 & \alpha_2 & \ddots \\
& \ddots & \ddots & \ddots \\
& & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\
& & & \beta_{n-1} & \alpha_n
\end{bmatrix}
\]

1. Let \( A_{n-1} \) is the \( n-1 \times n-1 \) leading principal minor of \( A \). Suppose you are given the Cholesky factorization of \( A_{n-1} = L_{n-1}L_{n-1}^T \). Determine the structure of \( L_{n-1} \) (hint, \( L_{n-1}L_{n-1}^T \) must be tridiagonal!)

2. Now, show how to compute the Cholesky factorization of \( A = LL^T \) from \( L_{n-1} \) in a constant number of operations.

3. Use the result of the first two parts to write down an algorithm that computes the Cholesky factorization of \( A \) in a linear amount of work (starting from scratch!)

Problem 2: Sparse matrices in Julia

If you wish to use another language, please see me

In this problem, we’ll return to our favorite linear system, Poisson’s equation. (I told you that we’d see a lot of this system.)

We’ll assume that the boundary conditions are zero everywhere. Consequently, we’ll use the form of the system where we have removed all rows that are set
to zero. Some of you did this already, so you may have an easier time on this question.

For the $N = 4$ case, we have:

$$
\begin{bmatrix}
-4 & 1 & 1 \\
1 & -4 & 1 \\
1 & -4 & 1 \\
1 & 1 & -4 \\
1 & -4 & 1 \\
1 & 1 & -4
\end{bmatrix}
\begin{bmatrix}
U(1,1) \\
U(1,2) \\
U(1,3) \\
U(2,1) \\
U(2,2) \\
U(2,3)
\end{bmatrix}
= 
\begin{bmatrix}
F(1,1) \\
F(1,2) \\
F(1,3) \\
F(2,1) \\
F(2,2) \\
F(2,3)
\end{bmatrix}
$$

where $U(i,j) = u(x_i, y_j)$ and $F(i,j) = h^2 f(x_i, y_j)$.

1. You can change an entry of a sparse matrix in Julia with $A[i,j] = v$. Write a Julia routine to create the matrix above for a general $N$ by changing each element one at a time.

2. As shown in class, another way to create a sparse matrix is to use the coordinate form. Create the arrays $I, J, \text{and } V$ for the matrix above. Hint: there are $(N - 1)^2 + 4(N - 1)(N - 2)$ non-zeros. Use this information to initialize the arrays $I, J, \text{and } V$. Write a routine to create the matrix above for a general $N$ using the coordinate form.

3. Which way is faster? Try creating matrices with $N = 10, 50, 100, 500$.

4. Let $f(x, y) = 1$. Is solving the linear system using backslash faster or slower with a sparse matrix? Does it depend on the value of $N$? Investigate and report.

5. Again let $f(x, y) = 1$. Implement the Jacobi method for this problem. You should start from the iterate $x^{(0)} = b$, so start with the guess of the right hand side. How many iterations does it take to converge to a relative residual of $10^{-4}$? Use the same values of $N$ from part 3.

6. Does the number of iterations of your Jacobi solver depend on the the right hand side? Try comparing $f(x, y) = 1, f(x, y) = -1, f(x, y) = -(x - 0.5)^2 - (y - 0.5)^2, f(x, y) = \sin(100x) \cos(100y)$. Use the same values of $N$ from part 3.

**Problem 3: Fun with sparse matrices**

In Julia, calling $y = A*x$ on a sparse matrix $A$, calls a special function to perform this multiplication faster. To generate a sparse random matrix in Julia, you can call $A = \text{sprand}(M, N, d)$ where $M$ and $N$ are the dimensions of $A$, and $d$ is the density of $A$. 1. Generate a random sparse matrix $A$ and transform it to dense matrix $B$. Use a dense vector $x$ of suitable dimensions and compute $A*x$ and $B*x$. Vary $M, N, \text{and } d$ and report performance changes.

2. Generate a random square sparse matrix $A$, and a dense vector $x$ of suitable dimension and compute $A*x$ and $A'*x$. Is one faster than the other? Vary the dimensions and the density and report your results. Provide an analysis of your results.
3. Write a sparse matrix vector routine where both, the matrix and the vector are sparse. You can generate a sparse vector in Julia using \( \mathbf{x} = \text{sprand}(N, d) \), where \( N \) is the dimension and \( d \) is the density. Test your method and report performance results.

```
function spmatvec{T,F}(A::SparseMatrixCSC{T,Int64},x::SparseVector{F,Int64})
    y = <fill in>
    for j = <fill in>
        for k = A.colptr[j]:<fill in>:A.colptr[j+1]-1
            <fill in>
        end
    end
    return y
end
```