Homework 5

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 2nd, 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: The Cholesky Factorization

Through this problem, assume that the matrices are all symmetric. You may assume this in your code as well.

1. Implement a Julia function for the Cholesky factorization.

2. Research the Cholesky factorization and answer the following question, either providing a proof or a reference. Is the Cholesky factorization unique?

3. Compare your Cholesky factorization code to Julia’s \texttt{chol} function on the following matrix.

\[
\begin{align*}
n & = 10 \\
Q,R & = \text{qr}(\text{randn}(n,n)) \\
D & = \text{abs.}(\text{randn}(n)) \\
A & = \text{triu}(Q*\text{diagm}(D)*Q)' \\
A & = A + \text{triu}(A,1)'
\end{align*}
\]

4. Your professor boldly asserted that using the Cholesky factorization was faster than using the LU factorization. Make sure he’s still right. Show the performance of Julia’s LU and Cholesky factorization for matrices with size that varies from 10 to 500.

5. Your professor also boldly asserted that we preserve positive definiteness after one step of the Cholesky factorization. Recall that this was:

\[
A = \begin{bmatrix} \alpha & b^T \\ b & C \end{bmatrix} = FF^T = \begin{bmatrix} \gamma & 0 \\ f & F_1 \end{bmatrix} \begin{bmatrix} \gamma & f^T \\ 0 & F_1^T \end{bmatrix}.
\]

So we set \( \gamma = \sqrt{\alpha} \) and \( f = b/\gamma \). Then, we recursively compute the Cholesky factorization of

\[
C - bb^T/\alpha = F_1F_1^T.
\]

This only works if \( C - bb^T/\alpha \) is positive definite. Prove that this is true.

(Hint: you can do it from the definition of positive definiteness: \( x^T A x > 0 \) for all \( x \).)
6. Briefly explain why step 5 justifies that a Cholesky factorization exists for any positive definite matrix. (This is really like a one or two sentence answer.)

7. One of the most useful aspects of the Cholesky factorization is as a way to check if a matrix is positive definite or negative definite. A matrix is negative definite if
\[ x^T Ax < 0 \]
for all \( x \). Change your Julia implementation to report if a matrix is not positive definite. (Hint: this relates to why a Cholesky factorization always exists for a positive definite matrix in the previous problem.) Use this to test if the matrix for Poisson’s equation from Homework 4, Problem 7 is positive or negative definite.

Problem 2: Stability analysis

1. Suppose that \( x_1, \ldots, x_n \) are a sequence of samples from a random variable. The sample variance of \( x_1, \ldots, x_n \) is:
\[ V = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \frac{1}{n} \sum_{i=1}^{n} x_i)^2. \]

Compute the condition number of the sample variance.

2. This is a real problem that I ran into! I was using a program written by someone else to compute a particular statistical estimate. The details don’t really matter, but at one step, the code needed to compute the Hurwitz zeta function:
\[ H(s, q) = \sum_{n=0}^{\infty} \frac{1}{(q + n)^s} \]
where \( s \) ranged from 1 to 7 and where \( q \) ranged from 1 to 500. They were using Matlab, which doesn’t provide a function for the Hurwitz zeta function, but does provide a function for the Riemann zeta function:
\[ R(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}. \]
We will use Julia (which also has the zeta function) to continue this analysis. The final result doesn’t differ much from Matlab’s. Note that the only difference between these two functions is that the first term in the Riemann zeta function is \( 1/(1^s) \) and the first term in the Hurwitz zeta function is \( 1/(q^s) \). To account for this difference, the code I was using did the following:

```julia
function h=hzeta(s,q)
    z = zeta(s)
    h = z - sum((1:(q-1)).^(-s));
end
```
In Julia,

```julia
using SpecialFunctions
function hzeta(s,q)
    z = zeta(s)
    h = z - sum(Float64.((1:(q-1))).^(-s))
    return h
end
```
This requires loading the SpecialFunctions package.
Problem 3: Backwards stability

Only 1 part! Use the properties of floating point arithmetic to show that computing $\|x\|_2$ is backwards stable.