Homework #2

Noa Shadmon 999765980

February 3, 2017

Files attached:
hw2test.m –> this produces the output for 1c, 6, 7b.
myr.m –> quadratic equation solver
number1c.m –> my cholesky decomposition

1 First Problem

A.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} * \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{bmatrix}$$

$l_{11}^2 = a_{11} \rightarrow l_{11} = \sqrt{a_{11}}$

$l_{11} l_{21} = a_{12} \rightarrow l_{21} = \frac{a_{12}}{l_{11}} = \frac{a_{12}}{\sqrt{a_{11}}}$

$l_{21}^2 + l_{22}^2 = a_{22} \rightarrow l_{22}^2 = \frac{a_{22}}{l_{21}^2} \rightarrow l_{22}^2 = \frac{a_{22} * a_{11}}{a_{21}^2} \rightarrow l_{22} = \frac{\sqrt{a_{22} * a_{11}}}{a_{21}}$

B. Pattern: when i != j, $l_{ij} = l_{ji} = \frac{1}{l_{ii}} * [a_{ij} - \sum_{k=1}^{i-1} l_{ji} * l_{ik}]$

Pattern: when i = j, $l_{ii} = (a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2)^{.5}$

C. My results are the same as matlab's built in chol function. Result in attached .m code.

2 Second Problem

$$\begin{bmatrix} 1000 & 999 \\ 999 & 998 \end{bmatrix} * x = \begin{bmatrix} 1999 \\ 1997 \end{bmatrix}$$

Through matrix multiplication we know

$$x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

However, we are given that

$$\hat{x} = \begin{bmatrix} 1.01 \\ 1.01 \end{bmatrix}$$

Thus, the error can be calculated as $\frac{||\hat{x}-x||}{||x||} \leq ||A|| * ||A^{-1}|| * \frac{||r||}{||b||}$
$\frac{||\hat{x}-x||}{||x||} = \frac{||[1.01-1;1.01-1]||}{||[1;1]||} = 0.0141$

When

$$\hat{x} = \begin{bmatrix} 20.97 \\ -18.99 \end{bmatrix}$$

$$x - \hat{x} = \begin{bmatrix} -19.97 \\ 19.99 \end{bmatrix}$$

then $\frac{||\hat{x}-x||}{||x||} = \frac{28.2560}{1} = 28.2560$

Clearly the second example has a much higher error. Therefore it is not fit to approximate x unlike the first $\hat{x}$. Also, since cond(A) is relatively large (3.9920e+06) there are going to be two solutions. One in which the difference of the norms of x are small and one when they are large. We see this happen in this problem.

3 Third Problem

A. The p value will equal to 6 so machine precision will be -5. Since we want the smallest number, Yet, we want to get the smallest number with only one extra bit after the value 1. The precision would then be 2^{-6}.
B. The smallest positive normalized number would be based on the 4 exponent bit 1111 with a negative sign bit. This would be $1.0 * 2^{-15}$.

4 Fourth Problem

We first begin with x=1 and a small delta value. We increment the x value by delta each time it goes through the loop. I hypothesized that we would continue to get a value of 1 regardless that x is constantly incremented. This was indeed the case. In matlab, there is a specific range until a new number is recognized. After placing a breakpoint after the

first iteration the value was still 1. Then after running the code for a while, the value of x still remained at one. Thus, the small increment in x was not being recognized as a new value. This can be seen with the following equation: 1+d*2^x =1

5 Fifth Problem

Here the end value the value is infinity even though the k only equals to 1024. This is because of the range of positive normalized numbers for double float precision in the N-max calculation.Thus k can only go up to a certain number. Because of this, no k value can exceed this nmax so the program stops but twox can still be reported to diverge.

6 Sixth Problem

The first function is a factored equation which results in an upside down parabola. However, this is expensive for operation storage. The second equation is an expanded polynomial so we have to use horners method to write a new function. When we get this function we get a riggedy/unsmooth upside parabola. This is because subraction constantly occurs with horners method. The reason is because floating point subtraction means that you need to constantly re normalize because the difference between the two side by side values can be very small. However, using horners method is good because it removes bits and give you more storage. The reason the graph is so rigid is because when the values are multiplied it always occurs on the error so there can be many inconsistencies.
The program is attached.

7 Seventh Problem

A. code attached B. My result is the same as matlab's built in roots function. Output comparing the two will be seen when you run myroots.m.