MA/CSC 580, Homework #3
Lindsay Eddy

1. (a): Let $x_e$ be the solution of $Ax = b$ assuming that $det(A) \neq 0$, $\bar{x}$ be the solution of $Ax = b + \delta b$, show that

$$\frac{\|x_e - \bar{x}\|}{\|x_e\|} \leq cond(A)\frac{\|\delta b\|}{\|b\|}.$$

**Hint:** $\|b\| = \|AA^{-1}b\| \leq \|A\|\|A^{-1}b\|$.

$$\frac{\|x_e - \bar{x}\|}{\|x_e\|} = \frac{\|A^{-1}b - A^{-1}b - A^{-1}\delta b\|}{\|A^{-1}b\|} = \frac{\|A^{-1}\delta b\|}{\|A^{-1}b\|} \leq \frac{\|A^{-1}\|\|\delta b\|}{\|A^{-1}b\|}$$
$$= \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|A\|\|A^{-1}b\|} \leq \text{cond}(A)\frac{\|\delta b\|}{\|b\|}$$

(b): Given a matrix $A$ that is invertible. Let $E$ is be such a matrix that $\|A^{-1}E\| < \alpha < 1$. Show that $A + E$ is invertible and $\|(A + E)^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \alpha}$.

Since $\|A^{-1}E\| < 1$, $I + A^{-1}E$ is invertible. Note that $A + E = A(I + A^{-1}E)$, and

$$(I + A^{-1}E)^{-1}A^{-1}A(I + A^{-1}E) = I.$$

Thus $(A + E)^{-1}$ exists and is equal to $(I + A^{-1}E)^{-1}A^{-1}$.

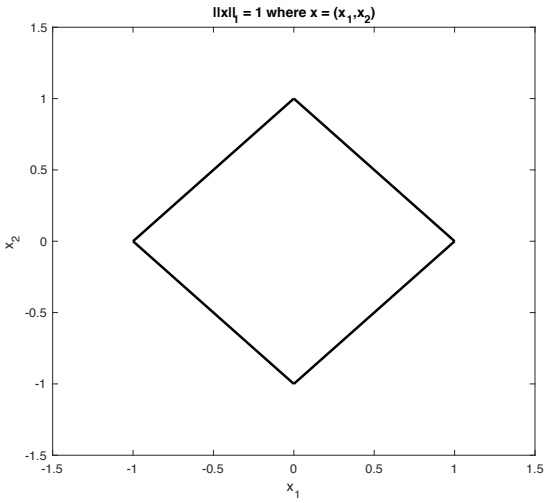
Now, note that since $\|A^{-1}E\| < 1$,

$$\|(I + A^{-1}E)^{-1}\| \leq \frac{1}{1 - \|A^{-1}E\|} < \frac{1}{1 - \alpha}.$$

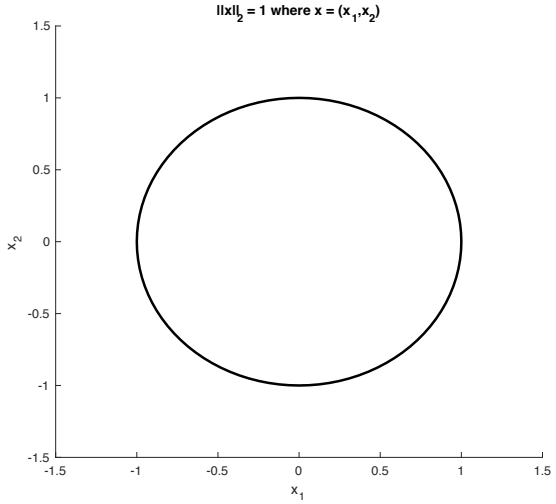
Since $(A + E)^{-1} = (I + A^{-1}E)^{-1}A^{-1}$,

$$\|(A + E)^{-1}\| = \|(I + A^{-1}E)^{-1}A^{-1}\| \leq \|(I + A^{-1}E)^{-1}\|\|A^{-1}\| < \frac{\|A^{-1}\|}{1 - \alpha}.$$

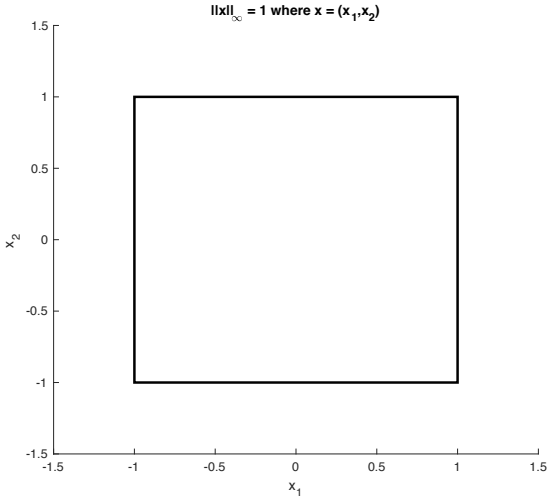
2. Plot or sketch the set of $\|\mathbf{x}\|_p = 1$ in two-dimensions, $p = 1, 2, \infty$. This example helps us to understand geometric meanings of vector norms.




(a) $p = 1$




(b) $p = 2$




(c) $p = \infty$

3. Given an $n$-by-$n$ non-singular matrix $A$, how do you efficiently solve the following problems using Gaussian elimination with partial column pivoting?

(a) Solve $A^k x = b$, where $k$ is a positive integer.

(1) Describe algorithm.

To solve $A^k x = b$, we notice that the solution $x_1$ to $Ax_1 = b$ is equal to $A^{k-1}x$. This yields $A^{k-1}x = x_1$. Now, the solution $x_2$ to $Ax_2 = x_1$ is equal to $A^{k-2}x$. We can continue this iterative process until we reach $Ax = x_{k-1}$. The solution to this system is $x$, which is the solution to the overall problem, $A^k x = b$. Notice that we only need to perform Gaussian elimination with partial pivoting *once* to get the $PA = LU$ decomposition.

Then for *each of the k steps* of our algorithm, we solve $Ax_i = x_{i-1}$ for $x_i$ using the $PA = LU$ decomposition we've already obtained. Note that we have $LUx_i = Px_{i-1}$. Thus we use forward subsitution to solve $Ly_i = Px_{i-1}$ and backward substitution to solve $Ux_i = y_i$.

(2) Present pseudo-code.

First use Gaussian elimination with partial pivoting to get $PA = LU$. (This only needs to be done once.) Then,

```
xold = b
for I = 1:k
        %Forward substitution: Solve L*y = P*xold for y
        x = P*xold;
        y = zeros(n,1);
        for i = 1:n
                y(i) = x(i);
                for j = 1:i-1
                        y(i) = y(i) - L(i,j)*y(j);
                end
                y(i) = y(i) / L(i,i);
        end
        %Backward substitution: Solve U*xnew = y for xnew
        for i = n:-1:1
                xnew(i) = y(i);
                for j = i+1:n
                        xnew(i) = xnew(i) - U(i,j)*xnew(j);
                end
                xnew(i) = xnew(i) / U(i,i);
        end
        xold = xnew;
end
```

When the for-loop completes, `xold` will be the solution to $A^k x = b$.

(3) Find the required operation counts.

The required operation counts for this algorithm are

$$1 \times \begin{bmatrix} \text{operations needed for} \\ \text{GE with partial pivoting} \end{bmatrix} + k \times \left( \begin{bmatrix} \text{operations needed for} \\ \text{forward substitution} \end{bmatrix} + \begin{bmatrix} \text{operations needed for} \\ \text{backward substitution} \end{bmatrix} \right)$$

$$= O\left(\frac{2n^3}{3}\right) + k\Big(O(n^2) + O(n^2)\Big) = O\left(\frac{2n^3}{3}\right) + k\Big(O(2n^2)\Big).$$

This is the total number of additions/subtractions/multiplications/divisions.

(b) Computer $\alpha = c^T A^{-1} b$, where $c$ and $b$ are two vectors.

(1) Describe algorithm.

Use Gaussian elimination with partial pivoting to solve $Ax = b$. Then $x = A^{-1}b$. Now compute $\alpha = c^T x$ using standard vector multiplication.

(2) Present pseudo-code.

First use Gaussian elimination with partial pivoting to solve $Ax = b$. Then compute $\alpha = c^T x$ in the following way:

```
alpha = 0
for i = 1:n
        alpha = alpha + cT(i)*x(i);
end
```

(3) Find the required operation counts.

The required number of operation counts is equal to the operation counts used in Gaussian elimination with partial pivoting plus the operation counts used in computing $c^T x$. There are $n$ multiplications and $n-1$ additions used to compute $c^T x$, which is a total of $O(2n)$ operations. Thus the total number of operations used to compute $\alpha = c^T A^{-1} b$ is $O\left(\frac{2n^3}{3}\right) + O(2n) = O\left(\frac{2n^3}{3}\right)$. Thus the cost of calculating $\alpha$ is basically just the cost of performing Gaussian elimination with partial pivoting.

You should (1) describe your algorithm; (2) present a pseudo-code; (3) find out the required operation counts.

4. Often a small determinant is a sign of an ill-conditioned matrix. But this exercise is an exception. Given the following matrix $A$. (a): Find $\|A\|_\infty$ and $det(A)$. (b): Using Gaussian elimination to find the $A = LU$ decomposition. (c): Using your result to find $cond_\infty(A)$. Do you think this matrix is ill-conditioned? **Hint:** Find the inverse of $U^T$ to get $U^{-1}$.

$$A = \begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ \vdots & \ddots & \ddots & & 1 \\ -1 & -1 & \cdots & -1 & 1 \end{bmatrix}.$$

(a)

$$\|A\|_\infty = n,\ \det(A) = 2^{n-1}$$

(b) After the first step of GE we have,

$$\begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & -1 & 1 & & 2 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & -1 & \cdots & -1 & 2 \end{bmatrix}$$

with

$$L_1 = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 1 & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Then after the second step we have

$$\begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & 0 & 1 & & 4 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & \cdots & -1 & 4 \end{bmatrix}$$

with

$$L_2 = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & 1 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 1 & \cdots & 0 & 1 \end{bmatrix}.$$

And after the final $((n-1)$st) step, we have

$$U = \begin{bmatrix} 1 & & & & 1 \\ 0 & 1 & & & 2 \\ 0 & 0 & 1 & & 4 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & 2^{n-1} \end{bmatrix}$$

with

$$L' = L_{n-1}L_{n-2}\cdots L_2L_1 = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Note that $L'A = U$. Thus, $A = LU$ where

$$L = L'^{-1} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

(c) $\text{cond}_\infty(A) = \|A\|\|A^{-1}\|$

First we will find $A^{-1}$. Since $A = LU$, $A^{-1} = U^{-1}L^{-1}$. To find $U^{-1}$, we note that $(U^{-1})^T = (U^T)^{-1}$. It is easy to find $(U^T)^{-1}$ using the GE method for finding matrix inverses. We can do this by multiplying each row $i = 1, \ldots, n-1$ by $-2^{i-1}$ and adding that result to the $n$th row, and then dividing the $n$th row by 8. Applying these same operations in the same order to the identity matrix, we find that

$$(U^T)^{-1} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ -2^{-(n-1)} & -2^{-(n-2)} & \cdots & -2^{-1} & 2^{-(n-1)} \end{bmatrix}.$$

Thus

$$U^{-1} = \begin{bmatrix} 1 & & & -2^{-(n-1)} \\ & 1 & & -2^{-(n-2)} \\ & & \ddots & \vdots \\ & & 1 & -2^{-1} \\ & & & 2^{-(n-1)} \end{bmatrix}.$$

Since $L$ is a unit lower triangular matrix, to find $L^{-1}$ we simply switch the signs of the below diagonal entries. Thus

$$L^{-1} = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -1 & -1 & -1 & \cdots & 1 \end{bmatrix}.$$

Then

$$A^{-1} = \begin{bmatrix} 1 & & & -2^{-(n-1)} \\ & 1 & & -2^{-(n-2)} \\ & & \ddots & \vdots \\ & & 1 & -2^{-1} \\ & & & 2^{-(n-1)} \end{bmatrix} \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -1 & -1 & -1 & \cdots & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2^{-1} & -2^{-2} & -2^{-3} & \cdots & -2^{-(n-1)} & -2^{-n} \\ & 2^{-1} & -2^{-2} & \cdots & -2^{-(n-2)} & -2^{-(n-1)} \\ & & \ddots & & & \vdots \\ & & & & 2^{-1} & -2^{-1} \\ 2^{-1} & 2^{-2} & \cdots & 2^{-(n-1)} & 2^{-n} & 2^{-n} \end{bmatrix}$$

Therefore $\|A^{-1}\|_\infty = 1$. Note that $\|A\|_\infty = n$, due to the magnitudes of the $n$th row. Thus $\text{cond}_\infty(A) = n$.

5. Check whether the following matrices are:

   - Strictly column diagonally dominant.
   - Symmetric positive definite.

   Justify your conclusion. What is the significance of knowing these special matrices to the Gaussian related algorithms? Answer this question by considering issues of accuracy, speed, and the storage.

$$\begin{bmatrix} -5 & 2 & 1 & 0 \\ 2 & 7 & -1 & -1 \\ 1 & -1 & 5 & 1 \\ 0 & -1 & 1 & 4 \end{bmatrix}, \quad \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad \begin{bmatrix} \alpha & \beta \\ -1 & 2 \end{bmatrix}$$

   Find the Cholesky decomposition $A = LL^T$ or $A = LDL^T$ for the middle matrix. Note that, you need to determine the range of $\alpha$ and $\beta$.

   The first matrix is strictly column diagonal dominant , because in the first column, $5 > 2+1+0 = 3$, in the second, $7 > 2+1+1 = 4$, in the third, $5 > 1+1+1 = 3$, and in the fourth, $4 > 0+1+1 = 2$. The first matrix is not SPD because $A_1 = -5 \leq 0$.

   The second matrix is not strictly column diagonal dominant because in the second column, $|a_{22}| = 2 \not> |-1| + |-1| = |a_{12}| + |a_{32}|$. The second matrix is SPD because $A = A^T$ and

$$\det(A_1) = 2 > 0,$$

$$\det(A_2) = \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} = 4 - 1 = 3 > 0, \text{ and}$$

$$\det(A_3) = \det(A) = \begin{vmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & 01 & 2 \end{vmatrix} = 2 \begin{vmatrix} 2 & -1 \\ -1 & 2 \end{vmatrix} - (-1) \begin{vmatrix} -1 & -1 \\ 0 & 2 \end{vmatrix} = 2(3) - 2 = 4 > 0.$$

The third matrix is strictly column diagonal dominant when $|\alpha| > 1$, and $|\beta| < 2$. The third matrix is SPD when $\alpha = 2$ and $\beta = -1$. This is because for the third matrix to be symmetric, it must be the case that $\alpha = 2$ and $\beta = -1$. Now,

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

is SPD because $\det(A_1) = 2 > 0$ and $\det(A_2) = \det(A) = 4 - 1 = 3 > 0$.

For strictly column diagonal dominant matrices, we can do direct Gaussian elimination without having to use partial pivoting because at every step, the absolute value of the on-diagonal element is greater than that of every other element in the column. This means that even without using partial pivoting, which adds to the computational time, we do not experience the potentially very large round off error that can occur with non-strictly column diagonal dominant matrices. In SPD matrices, instead of just having an $A = LU$ decomposition, we also have an $A = LL^T$ decomposition. That is, we can use Cholesky decomposition instead of Gaussian elimination. Cholesky decomposition uses only half of the computational time and storage as Gaussian elimination. We also do not need to use partial pivoting on SPD matrices due to the low growth factor associated with this algorithm. This allows us to more quickly decompose the matrix without compromising accuracy.

The Cholesky decomposition of the middle matrix follows:

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}$$

$$\begin{aligned} l_{11} &= \sqrt{2} \\ l_{21}l_{11} = -1 &\Rightarrow l_{21} = -1/\sqrt{2} \\ l_{31}l_{11} = 0 &\Rightarrow l_{31} = 0 \text{ (since } l_{11} \neq 0) \\ l_{21}^2 + l_{22}^2 = 2 &\Rightarrow l_{22} = \sqrt{3/2} \\ l_{31}l_{21} + l_{32}l_{22} = -1 &\Rightarrow l_{32} = -\sqrt{2/3} \\ l_{31}^2 + 1_{32}^2 + l_{33}^2 = 2 &\Rightarrow l_{33} = \sqrt{4/3} \end{aligned}$$

Thus

$$L = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & \sqrt{3/2} & 0 \\ 0 & -\sqrt{2/3} & \sqrt{4/3}. \end{bmatrix}$$

6. Given a matrix $A$ and a vector $b$

$$A = \begin{bmatrix} \frac{1}{100} & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \qquad b = \begin{bmatrix} \frac{1}{100} \\ 1 \\ 1 \end{bmatrix}$$

(a) Find the condition number of $A$ in 2-norm.

$\|A\|_2 = \max_i\{|\lambda_i(A)|\}$

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 1/100 - \lambda & 0 & 0 \\ 0 & 2-\lambda & -1 \\ 0 & -1 & 2-\lambda \end{vmatrix} = (1/100 - \lambda)\begin{vmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} \\ &= (1/100 - \lambda)(\lambda - 3)(\lambda - 1) = 0 \\ &\Rightarrow \lambda_1 = 1/100,\ \lambda_2 = 3,\ \lambda_3 = 1 \\ &\Rightarrow \|A\|_2 = \max_i\{|\lambda_i(A)|\} = 3. \end{aligned}$$

Using the Gaussian elimination method for finding inverses, I found that

$$A^{-1} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 2/3 & 1/3 \\ 0 & 1/3 & 2/3 \end{bmatrix}$$

$\|A^{-1}\|_2 = max_i\{|\lambda_i(A^{-1})|\}$

$$\begin{aligned} \det(A^{-1} - \lambda I) &= \begin{vmatrix} 100 & 0 & 0 \\ 0 & 2/3 & 1/3 \\ 0 & 1/3 & 2/3 \end{vmatrix} = (100 - \lambda)\begin{vmatrix} 2/3-\lambda & 1/3 \\ 1/3 & 2/3-\lambda \end{vmatrix} \\ &= \frac{1}{9}(100 - \lambda)(\lambda - 7)(\lambda - 5) = 0 \\ &\Rightarrow \lambda_1 = 100,\ \lambda_2 = 7,\ \lambda_3 = 7 \\ &\Rightarrow \|A^{-1}\|_2 = \max_i\{|\lambda_i(A^{-1})|\} = 100. \end{aligned}$$

Therefore $\text{cond}_2(A) = \|A\|\|A^{-1}\| = 3(100) = 300$.

(b) Given $x_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$ and $x_2 = \begin{bmatrix} 1.1 & 1 & 0.9 \end{bmatrix}^T$, find the residual, and the norm of the residual, for both $x_1$ and $x_2$ in 2-norm.

$$\begin{aligned} r(x_1) &= b - Ax_1 \\ &= \begin{bmatrix} 1/100 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/100 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 1/100 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/100 \\ 1 \\ 1 \end{bmatrix} = 0 \end{aligned}$$

$$\|r(x_1)\|_2 = \|0\|_2 = 0$$

$$
\begin{aligned}
r(x_2) &= b - Ax_2 \\
&= \begin{bmatrix} 1/100 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/100 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1.1 \\ 1 \\ 0.9 \end{bmatrix} \\
&= \begin{bmatrix} 1/100 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1.1/100 & 1.1 & 0.8 \end{bmatrix} \\
&= \begin{bmatrix} -1/1000 \\ -1/10 \\ 4/5 \end{bmatrix}
\end{aligned}
$$

$$\|r(x_2)\|_2 = \sqrt{(-1/100)^2 + (-1/10)^2 + (4/5)^2} = \sqrt{0.650001} \approx 0.8062.$$

(c) Is one of $x_1$ and $x_2$ the exact solution of the system $Ax = b$?

Yes, $x_1$ is the exact solution to the system $Ax = b$ since $r(x_1) = 0$.

(d) If $x_1$ or $x_2$ is not the solution, use the error estimate discussed in class (relation between the residual and the relative error) to give an estimate error bound for the relative error.

$$\frac{\|r(x_2)\|}{\|A\|\|x_2\|} \leq \frac{\|x_e - x_2\|}{\|x_2\|} \leq \|A^{-1}\| \frac{\|r(x_2)\|}{\|x_2\|}$$

$$\|x_2\|_2 = \sqrt{(1.1)^2 + 1^2 + (0.9)^2} = \sqrt{3.02}$$

The values of $\|A\|$, $\|A^{-1}\|$, and $\|r(x_2)\|$ were found earlier in the problem. Now we have

$$0.15464 \approx \frac{\sqrt{0.650001}}{3\sqrt{3.02}} \leq \frac{\|x_e - x_2\|}{\|x_2\|} \leq 100 \frac{\sqrt{0.650001}}{\sqrt{3.02}} \approx 46.39312$$

(e) Find the actual relative error for $x_1$ and $x_2$ as approximations to $Ax = b$ and compare with the error bound that you just have got. How much is the difference?

Relative error of $x_2$ is $\frac{\|x_e - x_2\|}{\|x_2\|}$.

$$x_e - x_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1.1 \\ 1 \\ 0.9 \end{bmatrix} = \begin{bmatrix} -1/10 \\ 0 \\ 1/10 \end{bmatrix}$$

$$\|x_e - x_2\| = \sqrt{(-1/10)^2 + (1/10)^2} = \sqrt{2/100}$$

$$\Rightarrow \frac{\|x_e - x_2\|}{\|x_2\|} = \frac{\sqrt{2/100}}{\sqrt{3.02}} \approx 0.81379$$

The relative error is between the error bounds that I calculated in part (d). It is 45.5793 lower than the upper bound and 0.6592 higher than the lower bound.

7. (Programming Part) Let $A$ be a symmetric positive definite matrix.

   (a) Derive the algorithms for $A = LDL^T$ decomposition, where $L$ is a unit lower triangular matrix, and $D$ is a diagonal matrix.

   Pseudo code:

$$
\begin{aligned}
&\text{for } k = 1, \ldots, n \\
&\qquad d_k = a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 d_j \\
&\qquad \text{for } i = k+1, \ldots, n \\
&\qquad\qquad l_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} l_{ij} d_j l_{kj}}{d_k}
\end{aligned}
$$

(b) Write a Matlab code (or other language if you prefer) to do the factorization and solve the linear system of equations $Ax = b$ using the factorization. **Hint:** the process is the following:

$$\begin{aligned} Ly &= b, && y \text{ is the unknown,} \\ Dz &= y, && z \text{ is the unknown,} \\ L^T x &= z, && x \text{ is the unknown, which is the solution.} \end{aligned}$$

Construct at least one example that you know the exact solution to validate your code.

The matlab file `Q7.m` solves $Ax = b$ using the $A = LDL^T$ factorization and is submitted to Wolfware. The code was tested using $n$ by $n$ matrices ($n$ = 10, 20, 40, 80, 160, 320, 640). First an $n$ by $n$ matrix $B$ was randomly generated using Matlab's `rand` function, then an SPD matrix $A$ was chosen as the product $B^T B$. Each of these matrices was tested for two systems of equations: one with exact solution $x_{e,1} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$ and one with $x_{e,2} = \begin{bmatrix} 4 & 0 & 0 & 4 & 0 & 0 & \cdots \end{bmatrix}^T$. The residuals and relative errors were found for each matrix for each system.

$10^{-11}\times$

| $n =$ | 10 | 20 | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|---|---|
| $r(x_{c,1})$ | 0.0011 | 0.0028 | 0.0171 | 0.1592 | 0.8185 | 4.7294 | 20.3727 |
| $r(x_{c,2})$ | 0.0007 | 0.0057 | 0.0341 | 0.1364 | 0.7276 | 5.0932 | 37.8350 |

Figure 2: Infinity norm of residuals ($r(x_c) = \|Ax_c - b\|_\infty$) for each linear system.

$10^{-11}\times$

| $n =$ | 10 | 20 | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|---|---|
| Rel error of $x_{c,1}$ | 0.0099 | 0.0018 | 0.0273 | 0.2595 | 4.9262 | 43.8424 | 212.9530 |
| Rel error of $x_{c,2}$ | 0.0028 | 0.0110 | 0.0156 | 0.9554 | 0.4291 | 50.3237 | 19.6489 |

Figure 3: Relative errors (rel error = $\|x_c - x_e\|_\infty / \|x_e\|_\infty$) for each linear system.

For each matrix and linear system, the relative error and residuals were both less than $10^{-8}$, and many were as small as $10^{-11}$. This shows that the code sucessfully finds solutions to systems of the type $Ax = b$.