(1) Start with an easy case:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

In this case, all we need to do is clear a_{21}. How many floating point operations are needed? Be efficient! When you have an answer, please discuss it with me!

Since we are storing the values needed to reduce $A \to U$ in L there are no additional floating point operations needed to construct L. To construct L we need to eliminate a_{21}, to do this we subtract $k = a_{21}/a_{11}$. We need only compute the value going in the 22 spot since the value in the 21 spot will be 0. We will then subtract $k * a_{12}$ from a_{22} to find the value at that spot. There are 3 operations, one to compute k and two to subtract $k * a_{12}$.

(2) Next easiest case:

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

We first need to eliminate the first row terms, calculate $k_2 = a_{21}/a_{11}$ and $k_3 = a_{31}/a_{11}$, two floating point ops. Now set a_{21} and a_{31} to zero. Subtract k_2 copies of row 1 from row two, except for the fist column, four floating point operations, and the same with k_3 and the third row another four operations. Now we have a two by two to reduce, witch we know takes 3 operations, for a total of 13 operations.

(3) What about a 4×4 matrix?

In this case we will compute 3 k's in a similar fashion to the 3×3, 3 floating point operations, then we will set the first column except the top entry to zero's. now subtract the first row times each multiplier k from the row beneath, ignoring the first row, we take $2*3*3$ operations. Now we need to reduce a 3×3 witch will take 13 operations, bringing our total to 34.

(4) What about a $n \times n$ matrix?

We will need $n - 1$ operations to compute the k's, then $2(n - 1)^2$ operations to eliminate the first entries. After that we are left with a $n - 1 \times n - 1$. So assuming $f(n)$ is a function giving the number of operations to reduce a $n \times n$, we know that $f(n) = f(n - 1) + n - 1 + 2(n - 1)^2$ and $f(1) = 0$ since a 1×1 is reduced.

$$f(n) = \sum_{i=1}^{n-1} i + 2i^2 = (n-1)(n)/2 + (n-1)n(2n-1)/3$$

.

Exercise : Problem 7.1

$$A = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

$$U_1 = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix} L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$U_2 = \begin{bmatrix} 4 & -1 & -1 \\ 0 & 3.75 & -1.25 \\ 0 & -1.25 & 3.75 \end{bmatrix} L_2 = \begin{bmatrix} 1 & 0 & 0 \\ -.25 & 1 & 0 \\ -.25 & 0 & 1 \end{bmatrix}$$

$$U_3 = \begin{bmatrix} 4 & -1 & -1 \\ 0 & 3.75 & -1.25 \\ 0 & 0 & 10/3 \end{bmatrix} L_3 = \begin{bmatrix} 1 & 0 & 0 \\ -.25 & 1 & 0 \\ -.25 & -1/3 & 1 \end{bmatrix}$$

Starting over:

$$A = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

Assume there exists a L lower triangular where $A = LL^T$.

$$L = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix}$$

$$LL^T = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix} \cdot \begin{bmatrix} a & b & d \\ 0 & c & e \\ 0 & 0 & f \end{bmatrix} = \begin{bmatrix} a^2 & ba & da \\ ba & b^2 + c^2 & bd + ce \\ da & bd + ce & d^2 + e^2 + f^2 \end{bmatrix}$$

Lets solve for what we can, $a = 2$

$$\begin{bmatrix} 4 & 2b & 2d \\ 2b & b^2 + c^2 & bd + ce \\ 2d & bd + ce & d^2 + e^2 + f^2 \end{bmatrix}$$

$b = -1/2, d = -1/2$

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 1/4 + c^2 & 1/4 + ce \\ -1 & 1/4 + ce & 1/4 + e^2 + f^2 \end{bmatrix}$$

$1/4 + c^2 = 4 \rightarrow c = 1.9365$

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & 1/4 + ce \\ -1 & 1/4 + ce & 1/4 + e^2 + f^2 \end{bmatrix}$$

$1/4 + ce = -1 \rightarrow e = (-1 - 1/4)/c = -0.64550$

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 1/4 + e^2 + f^2 \end{bmatrix}$$

$1/4 + e^2 + f^2 = 4 \rightarrow f = \sqrt{4 - 1/4 - e^2} = 1.8257$

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

We now know that $LL^T = A$.

$$L = \begin{bmatrix} 2 & 0 & 0 \\ -1/2 & 1.9365 & 0 \\ -1/2 & -0.64550 & 1.8257 \end{bmatrix}$$

Exercise : Problem 7.2

```
function x=usolve(U,y)
    n=length(y);
    x=zeros(n,1);
    for(i=n:-1:1)
        #calc x
        x(i)=y(i)/U(i,i);
        #eliminate entries
        #note that we will not acctualy change the values of U only
        #remember that all terms furthur out than i are zero's
        for(j=1:(i-1))
            y(j)=y(j)-x(i)*U(j,i);
        end
    end
endfunction
```

Just to test this code let's use a random U and y and see if the x produced works.

```
>> u
u =

     1.00000     3.00000     4.00000
     0.00000     0.32400    43.34500
     0.00000     0.00000   235.00000

>> y
y =

   342.00000
     0.23000
   523.00000

>> x=usolve(u,y)
x =

   1224.1690
   -297.0237
      2.2255

>> u*x
ans =

   342.00000
     0.23000
   523.00000

>> diary off
```

It definitely worked.

Exercise : Problem 7.4

$$Ax = b$$

$$PAx = Pb$$

$$LUx = Pb$$

$$Ux = c$$

$$Lc = Pb$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/3 & 1/4 & 1 \end{bmatrix} c = \begin{bmatrix} -12 \\ 2 \\ 10 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/4 & 1 \end{bmatrix} c = \begin{bmatrix} -12 \\ 8 \\ 14 \end{bmatrix}$$

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} c = \begin{bmatrix}-12\\8\\12\end{bmatrix}$$

$$c = \begin{bmatrix}-12\\8\\12\end{bmatrix}$$

$$Ux = \begin{bmatrix}-12\\8\\12\end{bmatrix}$$

$$\begin{bmatrix}2 & 3 & 1\\0 & 1 & 2\\0 & 0 & 2\end{bmatrix} x = \begin{bmatrix}-12\\8\\12\end{bmatrix}$$

$$\begin{bmatrix}2 & 3 & 0\\0 & 1 & 0\\0 & 0 & 2\end{bmatrix} x = \begin{bmatrix}-18\\-4\\12\end{bmatrix}$$

$$\begin{bmatrix}2 & 0 & 0\\0 & 1 & 0\\0 & 0 & 2\end{bmatrix} x = \begin{bmatrix}-6\\-4\\12\end{bmatrix}$$

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} x = \begin{bmatrix}-3\\-4\\6\end{bmatrix}$$

$$x = \begin{bmatrix}-3\\-4\\6\end{bmatrix}$$

Exercise : Problem 7.6

(a) We simply add each entry to the corresponding entry in the other vector, n additions.

(b) Well the product of two n vectors is n multiplications and $n-1$ additions. To multiply a $m \times n$ by a n vector we need to do m products of two n vectors, $m * n$ multiplications and $m * (n-1)$ additions.

(c)

$$Ux = y$$

We need to do substitution, first calculate the lowest x value,one division, then substitute it back in, $n-1$ multiplications and $n-1$ subtractions. Do this for each row:
number of divisions = $\sum_{i=1}^{n} 1 = n$
number of multiplications = $\sum_{i=1}^{n} i - 1 = n(n+1)/2 - n = n^2/2 - n/2$
number of subtractions = $\sum_{i=1}^{n} i - 1 = n(n+1)/2 - n = n^2/2 - n/2$

Exercise : Problem 6.2

(a) Absolute condition number $C(x) = |f'(x)| = |1/3x^{-2/3}|$.
Relative condition number $K(x) = |C(x)x/f(x)| = 1/3$.

(b) In a absolute sense f is well conditioned away from zero. In a relative sense it is well conditioned everywhere.

(c) $|f(\hat{x}) - f(x)| \approx C(x)|\hat{x} - x| = 1/3 * 10^{34/3} * 9 * 10^{-16} = 3 * 10^{-14/3}$

Exercise : Problem 6.4

Here I show the error and values for $\tan(x)$ for the x_j in the text:

```
>> err=abs(tan(pi/4+2*pi*10.^[1:20])'-1)
err =

   1.03250741290140e-14
   5.44009282066327e-14
   2.03004280052710e-12
   4.50606219004612e-12
   1.96613614278363e-10
   2.22133422767001e-11
   1.25612853452139e-09
   2.31829585306542e-07
   5.06952314571762e-07
   4.50298580600972e-06
   5.46009865889374e-05
   1.56145765890858e-03
   3.45386108511325e-03
   1.09919740601338e-01
   4.23348269339045e-01
   1.96821974865261e+00
   3.05184464888953e+00
   4.57476533358056e+00
   9.71188030245360e+00
   2.05069410342145e+00

>> diary off
>> val=tan(pi/4+2*pi*10.^[1:20])'
val =

   9.99999999999990e-01
   9.99999999999946e-01
   9.99999999997970e-01
   9.99999999995494e-01
   9.99999999803386e-01
   9.99999999977787e-01
   1.00000000125613e+00
   9.99999768170415e-01
   1.00000050695231e+00
   9.99995497014194e-01
   9.99945399013411e-01
   9.98438542341091e-01
   9.96546138914887e-01
   8.90080259398662e-01
   5.76651730660955e-01
  -9.68219748652608e-01
  -2.05184464888953e+00
   5.57476533358056e+00
  -8.71188030245360e+00
  -1.05069410342145e+00

>> diary off
```

$K(x) = |f'(x)x/f(x)| = |\frac{xcos(x)}{\cos^2(x)sin(x)}| = |\frac{x}{\cos(x)sin(x)}|$ If $x = x_j$ we then get $K(x) = 2|x_j|$. If we Examine $x_j = \pi/4+2\pi*10^j = \pi(1/4+2*10^j)$, it is clear if we have errors in π on the order of 10^{-16} we will get relative errors in x_j on the order of $\frac{10^{-16}(1/4+2*10^j)}{\pi(1/4+2*10^j)} = \frac{10^{-16}}{\pi}$. We would expect our error in $f(x_j)$ to be $|f(x_j) - f(\hat{x}_j)| = f(x_j) * 2|x_j| * \frac{10^{-16}}{\pi} = 2|x_j| * \frac{10^{-16}}{\pi}$. Now we can ask if this accurately models the error we see:

```
>> [(pi/4+2*pi*10.^[1:20]')*2*10^-16/pi,err]
ans =

   4.05000000000000e-15   1.03250741290140e-14
   4.00500000000000e-14   5.44009282066327e-14
   4.00050000000000e-13   2.03004280052710e-12
   4.00005000000000e-12   4.50606219004612e-12
   4.00000500000000e-11   1.96613614278363e-10
   4.00000050000000e-10   2.22133422767001e-11
   4.00000005000000e-09   1.25612853452139e-09
   4.00000000500000e-08   2.31829585306542e-07
   4.00000000050000e-07   5.06952314571762e-07
   4.00000000005000e-06   4.50298580600972e-06
   4.00000000000500e-05   5.46009865889374e-05
   4.00000000000050e-04   1.56145765890858e-03
   4.00000000000005e-03   3.45386108511325e-03
   4.00000000000000e-02   1.09919740601338e-01
   4.00000000000000e-01   4.23348269339045e-01
   4.00000000000000e+00   1.96821974865261e+00
   4.00000000000000e+01   3.05184464888953e+00
   4.00000000000000e+02   4.57476533358056e+00
   4.00000000000000e+03   9.71188030245360e+00
   4.00000000000000e+04   2.05069410342145e+00

>> diary off
```

It looks like this model for error works well until we get to high values. This difference at high values can be explained by the nature of tan. Basically once your error in input is bigger than π, the repeat factor for tan, your outputs are no longer based on your input at all, they are essentially the same as the output of tan evaluated at a random input, witch is far more likely to fall close to zero than far away from zero. This explains why we didn't see our error match the predicted error for large *i* values.