Part A

In the following, we compute the number of floating point operations to solve

$$L\mathbf{c} = \mathbf{b}$$

where L is an $n \times n$ lower-triangular matrix with 1's on the diagonal and where $\mathbf{b}$ is a given n-dimensional vector.

1. Start with an easy case:

$$L = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \qquad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

a) How many floating point operations are needed to compute c_1?

b) How many more to compute c_2?

c) How many to solve the system?

2. Now the next easiest case:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ a_{21} & 1 & 0 \\ a_{31} & a_{32} & 1 \end{pmatrix}; \qquad c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

a) How many floating point operations are needed to compute c_1?

b) How many more to compute c_2? How many more to compute c_3?

c) How many to solve the system?

3. Now consider general case.

1. How many floating point operations are needed to determine c_1?
2. How many more floating point operations are needed to determine c_2?
3. If c_1, c_2, and up to c_{j-1} are all known, how many more floating point operations are needed to determine c_j?

4. Write the total number of floating point operations needed to solve $L\mathbf{c} = \mathbf{b}$ using summation notation. Then use the formula

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

to compute the sum.

5. If $\mathbf{x}$ is an n-dimensional vector, how many floating point operations are needed to compute the multiplication $L\mathbf{x}$? Which is more expensive, solving $L\mathbf{c} = \mathbf{b}$ or computing $L\mathbf{x}$?

Part B

6. Suppose instead you wish to solve $U\mathbf{x} = \mathbf{b}$ where U is an $n \times n$ upper triangular matrix with nonzero entries on the diagonal (but not necessarily 1s). How many floating point operations are required?

Part C

In this section we count the number of floating point operations needed to factor $A = LU$ (i.e. to perform Gaussian elimination).

7. Start with an easy case:

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

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!

8. Next easiest case:

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

1. How many operations are needed to clear a_{21}? What about a_{31}?

2. How many operations are needed to clear the first column?
3. How many operations are needed to clear a_{32}?
4. How many total operations are needed for Gaussian elimination of a 3×3 matrix?

9. What about a 4×4 matrix?

1. How many operations are needed to clear a_{21}? a_{31}? a_{41}?
2. How many operations are needed to clear the first column?
3. How many operations are needed to clear the second column?
4. How many operations are needed to clear the third column?

10. What about an $n \times n$ matrix?

1. How many operations are needed to clear a_{21}?
2. How many operations are needed to clear the first column?
3. How many operations are needed to clear the second column?
4. How many operations are needed to clear the j^{th} column?

11. Write the total number of operations needed to perform Gaussian elimination in summation notation. Then use the formula

$$\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$$

to compute the total number of floating point operations needed to factor $A = LU$ where A is $n \times n$.