# Homework 5

This homework assignment is about computational complexity, LU factorization (8.1) and iterative methods (8.4).

For the written exercises, you should upload a scanned PDF to Gradescope and then follow the prompts given by Gradescope to assign certain pages of your PDF document to the correct problems.

For the coding exercises, you will be prompted to upload your python file directly to Gradescope. Please make sure you submit only one file and that your variables are assigned to the correct variable names.

The course syllabus found on Canvas has information on how homework is graded and how homework should be presented and submitted. Please let me or the TAs know if you have any questions or concerns.

This assignment is due on Thursday, February 8 at 11:59pm.

## Written Assignment

1. Consider a vector with $n$ entries:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

Consider the following python function that calculates that computes the average of n entries of a 1D numpy array x:

**Code 1:**

```python
def Average_val_1(x):
    n = x.shape[0]
    sum = x[0]
    for i in range(1,x.shape[0]):
        sum = sum + x[i]
    sum = sum/x.shape[0]

    return sum
```

a.) What is the computational complexity of this function, i.e. what is the total number of operations (addition, subtraction, multiplications, and divisions) in this function?

b.) What is the asymptotic computational complexity of this function?

Consider a slightly different function for calculating the average value of the entries of the 1D array for x:

**Code 2:**

```python
def Average_val_2(x):
    n = x.shape[0]
    sum = 0
    for i in range(n):
        sum = sum + x[i]
    sum = sum/n

    return sum
```

c.) What is the computational complexity of this function, i.e. what is the total number of operations (addition, subtraction, multiplications, and divisions) in this function?

d.) What is the asymptotic computational complexity of this function?

Finally, consider yet another python function for computing the average value of the entries of the 1D array x:

**Code 3:**

```python
def Average_val_3(x):
    n = x.shape[0]
    sum = x[0]/n
    for i in range(1,n):
        sum = sum + x[i]/n
    return sum
```

e.) What is the computational complexity of this function, i.e. what is the total number of operations (addition, subtraction, multiplications, and divisions) in this function?

f.) What is the asymptotic computational complexity of this function?

2. Recall in the coding portion of homework 3 that we created a 2D array of size (12,15) to represent the following matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{12} & \cdots & \frac{1}{15} \\ 2 & \frac{2}{2} & \frac{2}{3} & \cdots & \frac{2}{12} & \cdots & \frac{2}{15} \\ 3 & \frac{3}{2} & \frac{3}{3} & \cdots & \frac{3}{12} & \cdots & \frac{3}{15} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 12 & \frac{12}{2} & \frac{12}{3} & \cdots & \frac{12}{12} & \cdots & \frac{12}{15} \end{bmatrix}.$$

The following function is one way to accomplish this in python:

**Code 4:**

```python
def create_A(n,m):
    A = np.zeros((n,m))
    for i in range(n):
        for j in range(m):
            A[i,j] = (i+1)/(j+1)

    return A
```

a.) What is the total number of operations (addition, subtraction, multiplications, and divisions) in this function?

b.) If $m = n$, what is the asymptotic computational complexity of this function?

c.) If the function above is called, with $n = 12$ and $m = 15$ how many total operations will be made? (use your answer from part a)

3. Complete the following problems from the textbook:

- Section 8.1 problem 2 both parts a and b, 11 part a only.
- Section 8.4 Computer Exercises (pg 423) problem 1a and 1b: Disregard the books directions for these problems and instead compute the $\mathbf{x}^{(3)}$ using both Richardson iteration and Jacobi iteration for the matrix $\mathbf{A}$ and $\mathbf{b}$ listed in problems 1a and 1b.

## Coding Assignment

The coding assignment for this week will ask you to write a function in python that solves the linear system of equations $\mathbf{Ax} = \mathbf{b}$ using the LU factorization, forward substitution, and then backwards substitution.

There is a template on canvas called hw5_template.py in the week 5 module. Here you will find three functions LU_fac(), forward_sub(), and backward_sub(). The LU_fac() and backward_sub() functions are completed, but you will need to complete fill out the forward_sub() function. This function should be fairly similar to the backward_sub() function, and you may refer to the pseudocode on page 365 when filling out the forward_sub() function. Remember that the pseudocode is written using the mathematical indexing starting at 1 and python indexing starts at 0.

1. Consider our example problem from lecture:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 6 & 10 \\ 3 & 14 & 28 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ -8 \end{bmatrix}.$$

a.) Create the matrix $\mathbf{A}$ as a 2D numpy array and assign it to the variable A1. The autograder wants this as a 2D array of ints. You can use the variable explorer in Spyder to check.

b.) Create the matrix $\mathbf{b}$ and assign it to the variable b1. The autograder wants this as an 1D array of ints. You can the variable explorer in Spyder to check.

c.) Call your LU_fac function with argument A1 and assign its return values to the variables L1 and U1. The autograder wants these as a 2D array of floats. You can use the variable explorer in Spyder to check.

d.) Call your forward_sub function with arguments L1 and b1 and assign its return value to the variable z1. The autograder wants this as a 1D array of floats. You can use the variable explorer in Spyder to check.

e.) Call your backward_sub function with the arguments U1 and z1 and assign its return value to the variable x1. The autograder wants this as a 1D array of floats. You can use the variable explorer in Spyder to check.

2. Consider the following matrix:

$$\mathbf{A} = \begin{bmatrix} (1+2C) & -C & 0 & 0 \\ -C & (1+2C) & -C & 0 \\ 0 & -C & (1+2C) & -C \\ 0 & 0 & -C & (1+2C) \end{bmatrix}.$$

This matrix can be used to estimate the unsteady condition of heat in a rod when the temperatures at points $T_1, T_2, T_3, T_4$ on the rod change with time. The constant $C$ depends on the physical nature of the rod, the distance $\Delta x$ between the points $T_1, T_2, T_3, T_4$, and the length of time between temperature measurements. (See figure 1 below)

Let the vector $\mathbf{t}_k$ be a collection of the temperatures at the times $k\Delta t$ for $k = 0, 1, 2, \ldots$. If the two ends of the rod are maintained at 0 degrees, then the temperature vectors satisfy the equation $\mathbf{A}\mathbf{t}_{k+1} = \mathbf{t}_k$ (k = 0, 1, 2, ...).

a.) For $C = 1$ represent the matrix above as a 2D numpy array and assign it to the variable A2. The autograder wants this as a 2D array of ints.

b.) Call your LU_fac() function with the argument A2 and assign its return values to L2 and U2. The autograder wants these as a 2D array of floats. You can use the variable explorer in Spyder to check.

c.) Suppose that the temperatures at time zero are

$$\mathbf{t}_0 = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 15 \\ 15 \\ 10 \end{bmatrix}$$

Using a 1D array, assign the vector above to the variable t0. The autograder wants a 1D array of ints for t0.

d.) Call your forward_sub() function with arguments L2 and t0 and assign its return value to the variable z2. The autograder wants a 1D array of floats.

e.) Call your backward_sub() function with arguments U2 and z2 and assign its return value to the variable t1. The autograder wants a 1D array of floats.

f.) Call your forward_sub() function with arguments L2 and t1 and assign its return value to the variable z3. The autograder wants a 1D array of floats.

g.) Call your backward_sub() function with arguments U2 and z3 and assign its return value to the variable t2. The autograder wants a 1D array of floats.

h.) Call your forward_sub() function with arguments L2 and t2 and assign its return value to the variable z4. The autograder wants a 1D array of floats.

i.) Call your backward_sub() function with arguments U2 and z4 and assign its return value to the variable t3. The autograder wants a 1D array of floats.

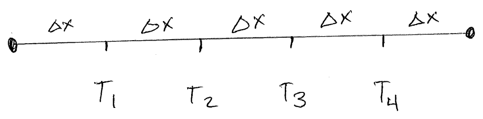

Figure 1: figure for problem 2 coding