# Homework 12

Ch En 263 – Numerical Tools

Due date: 21 May 2020

## Instructions

- For the handwritten problems, submitted a single pdf file on Learning Suite with the name "LastName_FirstName_HW12.pdf".
- For the problems in Excel, submit a workbook named "LastName_FirstName_HW12.xlsx" where each worksheet tab is named "Problem_1", "Problem_2", etc.
- For the problems in Python, submit a separate file for each problem named "LastName_FirstName_HW12_ProblemXX.py" where XX is the problem number.
- **Please report how long it took you to complete the assignment (in hours) in the "Notes" section on Learning Suite.**

## Problems

1. Answer the following questions using the function below which performs the back substitution algorithm. Turn your answers in via a pdf on Learning Suite.

```
def back_sub(A, b):
    n = len(b)
    x = np.zeros(n)
    x[n-1] = b[n-1]/A[n-1,n-1]
    for i in range(n-2, -1, -1):
        xi_sum = 0
        for j in range(i+1, n):
            xi_sum += A[i,j]*x[j]
        x[i] = (b[i] - xi_sum)/A[i,i]
    return x
```

   (a) If an integer is 4 bytes, a float is 8 bytes and $n = 100$, how many bytes of memory does the function need? *Hint: Include $A$ and $b$ in your calculation.*

   (b) If $c_1$ is the time it takes to execute the first line of code, $c_2$ is the time it takes to execute the second line, etc., write an expression $T(n)$ that describes the time it takes to execute the entire function. *Hint: The inner-most loop runs $(n-1)(n)/2$ times.*

   (c) What is the asymptotic behavior of $T(n)$ at large $n$? Using this asymptotic limit, estimate how long it will take for the function to execute when $n = 10^4$ if it takes 2 minutes using $n = 10^3$.

2. Use the tools in the Numpy linear algebra library to do the following.

   (a) Find the norm of the matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & -2 & 1 & 0 & 0 \\ 2 & 0 & -2 & 1 & 0 \\ -1 & 2 & 0 & -2 & -1 \\ 0 & -1 & 2 & 0 & -2 \\ 0 & 0 & -1 & 2 & 0 \end{bmatrix}$$

and the vector

$$\boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 4 \\ 1 \end{bmatrix}.$$

Print the value of both to the console.

(b) Solve the equation $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ using $\boldsymbol{A}$ and $\boldsymbol{b}$ from Part (a) with Numpy's linear algebra solver. Print $\boldsymbol{x}$ to the console and verify your solution by evaluating the residual $||\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}||$.

(c) Solve the equation $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ using $\boldsymbol{A}$ and $\boldsymbol{b}$ from Part (a) by finding $\boldsymbol{A}^{-1}$ using Numpy. Print $\boldsymbol{x}$ to the console and verify your solution by evaluating the residual $||\boldsymbol{A} \cdot \boldsymbol{x} - \boldsymbol{b}||$.

3. In this problem, we are going to compare the computational cost of using Numpy's methods to solve linear systems.

(a) Download the 3 pairs of files: `A_XXXX.dat` and `b_XXXX.dat` from the course website where `XXXX` is 1000, 2000, 5000.

(b) Using the `time` module (see example below), write a code that finds the average time it takes to solve $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ for all three pairs of $\boldsymbol{A}$ and $\boldsymbol{b}$ using Numpy's linear algebra solver. For example,

```
import time
t_start = time.time()
for i in range(5):
  # solve for x here
t_end = time.time()
t_avg = (t_end - t_start)/5
```

Save the times in an array called `t_Numpy`.

(c) In addition, write a piece of code that finds the average time it takes to solve $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ for all three pairs of $\boldsymbol{A}$ and $\boldsymbol{b}$ using Numpy's inverse function. Save the times in an array called `t_Inv`.

(d) Use your arrays from parts (b) and (c) to make a log-log plot using the matplotlib `plt.loglog(X, Y)` function that plots the average solution time versus the matrix size ($n = 1000, 2000$, or $5000$). Also plot the curve $T(n) = 10^{-11} \times n^3$ and add a legend labeling the different curves. What does your plot show about the asymptotic behavior of the method used by the Numpy solvers? How does that compare to the Gauss elimination method we learned last time? (Type your answers in the comments in your code).