Math 231 Assignment 0

Due online: Tuesday, February 4

General Instructions: Submit online a PDF copy of your written solution with your name and student number. This may be hand written or typesetted in LaTeX. Be sure scanned pages are in the correct orientation, with written answers (in numerically increasing order) for each questions clearly labelled. For questions involving programming, submit a single Jupyter notebook with all relevant program code, outputs, comments and explanations of your results. You may work together but you must write your own solution.
Online Submission Files: A PDF file named A0-StudentNumber-LastName-FirstName.pdf and a Jupyter Notebook named A0-StudentNumber-LastName-FirstName.ipynb. For example, my submission files would be named A0-123456789-Wan-Andy.pdf and A0-123456789-Wan-Andy.ipynb.

Q1. Solving an upper triangular linear system
An upper triangular matrix has the form

$$U := \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & u_{22} & u_{23} & \ddots & u_{2n} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & u_{n-1n} \\ 0 & \dots & 0 & 0 & u_{nn} \end{pmatrix}.$$

(a) Devise an $\mathcal{O}(n^2)$ algorithm (i.e. backsubstitution) to solve the system $U\boldsymbol{x} = \boldsymbol{b}$ in $\mathbb{R}^n$ and write a pseudocode of your algorithm. You may assume all the diagonal terms U_{ii} are all nonzero.
(b) Implement your pseudocode to solve $U\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{b}$ is a randomly generated vector in $\mathbb{R}^{10}$ and U is a randomly generated unit[1] upper triangular matrix of size 10×10 . Validate your algorithm works by computing the ℓ_2 norm of the residual vector $\|U\boldsymbol{x} - \boldsymbol{b}\|_2$. *Hint: See `tutotial.ipynb` where it contains an implementation of forward substitution.*
(c) Now vary the size n in appropriate powers of 10 and record their respective computation times. Generate a log-log plot to show that the time complexity of your implementation is indeed $\mathcal{O}(n^2)$. If you are not able to show the correct time complexity, explain why this might the case.

Q2. Solving a tridiagonal linear system
A tridiagonal matrix has the form

$$T := \begin{pmatrix} d_1 & u_1 & 0 & \dots & 0 \\ l_1 & d_2 & u_2 & \ddots & \vdots \\ 0 & l_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & u_{n-1} \\ 0 & \dots & 0 & l_{n-1} & d_n \end{pmatrix}.$$

(a) Devise an $\mathcal{O}(n)$ algorithm to solve the system $T\boldsymbol{x} = \boldsymbol{b}$ in $\mathbb{R}^n$ and write a pseudocode of your algorithm. You may assume either l_i are all nonzero or u_i are all nonzero.
(b) Implement your pseudocode to solve $T\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{b}$ is a randomly generated vector in $\mathbb{R}^{10}$ and U is a randomly generated upper triangular matrix of size 10×10. Validate your algorithm works by computing the ℓ_2 norm of the residual vector $\|T\boldsymbol{x} - \boldsymbol{b}\|_2$. *Hint: you may want to use `np.diag(l,-1)+np.diag(d,0)+np.diag(u,1)` to create T where l, d, u are randomly generated vectors.*
(c) Now vary the size n in appropriate powers of 10 and record their respective computation times. Generate a log-log plot to show that the time complexity of your implementation is indeed $\mathcal{O}(n)$. If you are not able to show the correct time complexity, explain why this might the case.

Q3. Naive Gaussian Elimination
Recall from your numerical linear algebra course that solving $A\boldsymbol{x} = \boldsymbol{b}$ using naive Gaussian Elimination (i.e. without partial pivoting) requires two mains steps:

1. Forward Eliminate to transform $A\boldsymbol{x} = \boldsymbol{b}$ into $U\boldsymbol{x} = \boldsymbol{y}$, where U is upper triangular.
2. Back Substitute to solve $U\boldsymbol{x} = \boldsymbol{y}$.

The Forward Eliminate algorithm has the following pseudocode:

[1] I.e. the diagonal elements of U are all equal to one.

Pseudocode: Forward Eliminate

```
1 function ForwardEliminate(A, b)
2     for k = 1 to n − 1 do
          // for each k-th row
3         for i = k + 1 to n do
              // for rows below k-th row
4             r ← A_ik / A_kk
5             for j = k + 1 to n do
                  // for columns after (k − 1)-th column
6                 A_ij ← A_ij − r × A_kj
7             b_i ← b_i − r × b_k
8     return U ← A, y ← b
```

(a) Show that Forward Eliminate algorithm has a time complexity of $\frac{2}{3}n^3 + \mathcal{O}(n^2)$ and conclude that naive Gaussian Elimination has a time complexity of $\mathcal{O}(n^3)$. *Hint: For a fixed $k = 1, \ldots, n-1$, how many divisions (in terms of n, k) are needed for line 4? How many multiplications/subtractions are needed for lines 6, 7? Sum up over k for the total time to arrive at the time complexity. You will need the following exact sums:*

$$\sum_{i=1}^{n} i = \frac{n(n-1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n-1)(2n-1)}{6}$$

(b) Implement the pseudocode of Forward Eliminate and use your Backsubstitution from **Q1** to solve $A\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{b}$ is a randomly generated vector in $\mathbb{R}^{10}$ and A is a randomly generated real matrix of size 10×10. Validate your algorithm works by computing the ℓ_2 norm of the residual vector $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$.
(c) As you increase the size n to 10^2 or more, what happens to the quality of the solution when checking the ℓ_2 norm of the residual vector $\|A\boldsymbol{x} - \boldsymbol{b}\|_2$? Explain why this might be the case.