Computational Linear Algebra – Programming Assignment 1

(points: 30, due on November 7)

1 Important points

- All the commented codes have to be crisp and submitted online with a report. Submit the codes and report in a zip folder with folder name '*STUDENTNAME_pga1*.zip'.
- If the question is just to implement a program and the matrix is not specified, show an example in the report. The code will be checked for correctness and plagiarism.
- Inbuilt commands should not be used unless specified.
- All the programs should take the matrix and size of the matrix as input (n if matrix $A \in \mathbb{R}^{n\times n}$).
- The programs have to be coded in MATLAB or python.
- Error between two vectors x and $y \in \mathbb{R}^n$ is the euclidean norm of their difference, i.e. square root of $\sum_{i=1}^{n} (x_i - y_i)^2$.
- Error between two matrices A and $B \in \mathbb{R}^{n\times n}$ is the frobenius norm of their difference.

$$\text{Error}(A, B) = \sqrt{\sum_{i,j=1}^{n} (A_{ij} - B_{ij})^2}.$$

2 Questions

2.1 LU decomposition (4+6)

a) 1) Write a program (name it *tridiag.m* or *tridiag.py*) to solve tridiagonal system of equations using LU without pivoting.

$$\sum_{k=\max\{1,i-1\}}^{\min\{n,i+1\}} a_{ik}x_k = y_i. \quad (i \in \{1, 2, \ldots n\}).$$

For all $i \in \{1, \ldots, n\}$, the coefficients are defined as:

$$\begin{aligned} a_{ik} &\neq 0 \text{ for all } k \in \{\max\{1, i-1\}, \ldots, \min\{n, i+1\}\} \\ &= 0 \text{ elsewhere} \end{aligned}$$

2) Comment on the flop counts for LU decomposition in the above case as compared to the flop counts for LU decomposition of an arbitrary matrix.

b) 1) For an invertible square matrix, write a program (name it *LUpartial.m* or *LUpartial.py*) to implement LU with partial pivoting and return the determinant and inverse of the matrix. Note that the program should not contain any inversion subroutine applied for L or U matrix. Refer the textbook 'Numerical Linear Algebra' for understanding partial pivoting.
2) Note the error between your inverse and the inverse calculated by MATLAB/numpy. Also, check the property $AA^{-1} = I$ for your inverse.

2.2 QR decomposition (6+3+6)

a) Implement QR decomposition using Gram-Schmidt (GS) procedure. We saw in the class that GS has stability issues, Q matrix produced is far from orthogonal. Modify GS by first normalizing the j-th vector and then removing the components in direction of j-th vector from the vectors numbered $j+1:n$. Repeat this for $j=1:n$ in a loop. Name the programs as *gs.m* or *gs.py* and *mgs.m* or *mgs.py*.

b) For the matrix $A = 0.00001 * eye(n) + hilb(n)$, where $eye(n)$ returns an identity matrix and $hilb(n)$ return a hilbert matrix of size n.
1) Apply QR decomposition for the above matrix using both GS and its modified version.
2) Note the error between $Q^\top Q$ and identity matrix.
4) Which algorithm gives a better decomposition (lesser error between A and QR)?

c) For $n = 3$, follow the below procedure and use both GS and its modified version to find the estimate of x. Which algorithm is better (lesser error between x and estimate of x)? Explain the reason by debugging the respective codes of the algorithms.

Generate two orthonormal vectors $v1$ and $v2$.
Construct matrix $A = 50000 * v1 * v1^\top + 2 * v2 * v2^\top$.
Generate $x = randn(n, 1)$ and $b = Ax$.
Take this b as input and find the estimate of x.

2.3 Connecting decompositions (5)

1) Given LU decomposition of a square matrix, write a program (name it *lutoqr.m* or *lutoqr.py*) to output QR decomposition of the corresponding matrix without explicitly constructing the matrix and vice versa (name it *qrtolu.m* or *qrtolu.py*).
2) Explain what is internally happening in this kind of transformation.