Math 4610: Homework 5

Chandler Justice - A02313187

Question 1: Write code to solve an upper triangular system of equations. This is the backsubstitution algorithm for class. Put this in an object file or Python file if programming in Python. Test on at least one upper triangular system of equations.

Solution: Here is my back-substitution Algorithm:

```
import numpy as np

def backSub(A: np.array, y: np.array):
    x = np.zeros(len(y))
    n = len(y)


    x[n - 1] = y[n-1] / A[n-1][n - 1]
    for i in range(n-2, -1, -1):
        sum = 0
        for j in range(i + 1, n):
            sum += A[i][j] * x[j]
        x[i] = (y[i] - sum) / A[i][i]

    return x
```

Question 2: What are the minimum conditions on an upper triangular system for your algorithm to produce a unique solution?

Solution: In order for an upper triangular matrix to produce a unique solution:

- The matrix must have a unique LU-decomposition
- each column must exhibit linear independence
- no columns in the first (n-1) columns are zero vectors

Question 3: Repeat your work on Question 1 with modifications to the algoritihm for a code that will produce the solution of a lower triangular linear system of equation. We worked on this in class in the context of the LU factorization of a matrix and solving linear systems via LU factorization.

Solution: This is called forward-elimination. I implemented this algorithm using the code below:

```
import numpy as np

def forward_elim(A, b):
    n = len(A)
    y = np.zeros(n)

    y[0] = b[0]
```

```
    for i in range(1,n):
        total = 0.0
        for j in range(i):
            total += A[i][j] * y[j]
        y[i] = b[i] - total
    return y
```

Question 4: Write a code that will solve a linear system of equations using Gaussian Eliminaton (GE). The input to the code will be a square matrix, , a right hand side vector, , and the number of rows and column in the matrix. You should have separate routines for the elimination step and the backsubstitution step. In a software manual you would need to explain the relationship between the two codes and their usage.

Solution: I already showed my back substituition algorithm in a previous question, so here is the code to perform gaussian elimination

```
import numpy as np

def Gaus_elim(A: np.matrix, b: np.array):
    n = len(b)

    for k in range(n):
        for i in range(k + 1, n):
            if(i != k):
                factor = A[i][k] / A[k][k]
                for j in range(k + 1, n):
                    # print(A[i][k])
                    A[i][j] = A[i][j] - factor * A[k][j]
                b[i] = b[i] - factor * b[k]
    return A, b
```

Question 5: Write a module/object file that will take as input a matrix, A, and vector, y, and possibly the size of the matrix n, and produce a vector, b, that is the product of the matrix multiplied into the vector from the left.

Solution: Here is the routine I created to produce the product of a matrix and vector

```
def multiplyMatrixByVector(A, b):
    row_size = len(A[0])

    n = len(b)
    total = np.zeros(n)

    for i in range(n): # for every element in the vector b

        for j in range(row_size): # for every component in row i of the
matrix A
            print(f"{b[i] * A[i][j]}")
            total[i] += b[i] * A[i][j]
    return total
```

Question 6/7: Test the result of your work from Question 4 on diagonally dominant matrices as discussed in class. That is, use the following steps to define a single test of your code. You can define a loop to test your code on any number of problems you need.

Repeat the work from the previous Questions, but aim the work at using the LU factoriation of matrix. Recall that this approach will require only a small modification to your code in the elimination phase and the additional work of solving a lower triangular matrix before solvingthe associated upper triangular system of equations.

Solution: Here is the test routine I wrote to test my software routines

```
import gausElim, backsub, matrixVector,LUDefactor, forwardElim, diagMatrix
import numpy as np
def main():
    #Testing Gaus elim
    [A, y0, b] = diagMatrix.createDiagDominantMatrix(10)

    [guas_elim , y1] = gausElim.Gaus_elim(A, b)
    final_result = backsub.backSub(guas_elim, y1)

    error = np.zeros(10)
    for i in range(10):
        error[i] = abs(final_result[i] - y0[i])

    print(f'the error of the GE function is represented by this
vector:\n{error}')

    # testing LU Defactorization
    [A_lu, y0_lu, b_lu] = diagMatrix.createDiagDominantMatrix(10)

    [lower_m , upper_m] = LUDefactor.LU_Defactor(A_lu)

    z_lu = matrixVector.multiplyMatrixByVector(lower_m, b_lu)

    z1_lu = forwardElim.forward_elim(lower_m, z_lu)

    x_lu = backsub.backSub(upper_m, z1_lu)

    error_lu = np.zeros(10)
    for i in range(10):
        error_lu[i] = abs(x_lu[i] - y0_lu[i])
    print(f'the error of the LU-factorization function is represented by
this vector:\n {error_lu}')




main()
```

Question 8: Since GE and LU require about the same amount of work, say `O(n^3)` and forward elination and backsubstitution require about the same amount of work, say `O(n^2)`, write out the following work loads for your solvers. Assume that the matrix does not change and the right hand side vector may vary.

Solution:

- my `GuasElim` algorithm has runtime complexity of `O(n^5)` since the `GuasElim` routine is `O(n^3)` and the backsubstituion routine is `O(n^2)`
- my `LUDecomp` algorithm has runtime complexity of `O(n^3)`, the back substition is `O(n^2)` and the forward elimination is `O(n^2)`. This means LU-decomposition requires a staggaring `O(n^7)` computations.

Question 9: Build a "package" or shared library of codes that contain the work from this set of Questions. In the package, only include what you need to solve linear systems. Your code should be completely self-contained with a test code as described in Question 6.

Solution: I have collected all of my routines into a `src` folder in my git repository. Alongside that folder, I have a `doc` folder that contains the software manual. Below is a representation of the package structure

```
hw05
    - doc
        - softwareManual.md
    - src
        - backsub.py
        - diagMatrix.py
        - forwardElim.py
        - guasElim.py
        - LUDefactor.py
        - matrixVector.py
        - test.py
```

Question 10: Write a software manual for the package

Solution: I am not going to include my whole manual in this document as it would more than double the length of the submittable. Below I have included a link to my GitHub repository where the software manaual is hosted:

https://github.com/chandlerj/math4610/blob/main/assignments/hw05/doc/softwareManual.md