MA398 Matrix Analysis and Algorithms: Exercise Sheet 7

1. Consider the matrix

$$A = \begin{pmatrix} 1 & 5 \\ 2 & 3 \\ 4 & 3 \end{pmatrix},$$

perform a manual QR factorisation using the Gram-Schmidt orthonormalisation process outlined in the notes. Compute the matrices Q and R.

Answer: Given the matrix A:

$$A = \begin{pmatrix} 1 & 5 \\ 2 & 3 \\ 4 & 3 \end{pmatrix},$$

We have the columns as vectors:

$$\mathbf{a}_1 = \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}$$

$$\mathbf{a}_2 = \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix}.$$

First, we will normalize $\mathbf{a}_1$ to find $\mathbf{q}_1$:

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|} = \frac{1}{\sqrt{1^2 + 2^2 + 4^2}} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} = \frac{1}{\sqrt{21}} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix}.$$

Next, we orthogonalize $\mathbf{a}_2$ with respect to $\mathbf{q}_1$ and normalize the result to get $\mathbf{q}_2$:

$$\mathbf{u}_2 = \mathbf{a}_2 - (\mathbf{q}_1 \cdot \mathbf{a}_2)\mathbf{q}_1 = \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix} - \left(\frac{1}{\sqrt{21}} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \right) \cdot \begin{pmatrix} 5 \\ 3 \\ 3 \end{pmatrix} \left(\frac{1}{\sqrt{21}} \begin{pmatrix} 1 \\ 2 \\ 4 \end{pmatrix} \right) = \begin{pmatrix} 3.9048 \\ 0.8095 \\ -1.3810 \end{pmatrix}.$$

We then normalize $\mathbf{u}_2$ to find $\mathbf{q}_2$:

$$\mathbf{q}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} = \frac{1}{4.2201} \begin{pmatrix} 3.9048 \\ 0.8095 \\ -1.3810 \end{pmatrix}.$$

Therefore, the QR factorization of A is:

$$Q = \frac{1}{19.3390} \begin{pmatrix} 1 & 3.9048 \\ 2 & 0.8095 \\ 4 & -1.3810 \end{pmatrix},$$

and

$$R = \begin{pmatrix} \mathbf{q}_1^T \cdot \mathbf{a}_1 & \mathbf{q}_1^T \cdot \mathbf{a}_2 \\ 0 & \mathbf{q}_2^T \cdot \mathbf{a}_2 \end{pmatrix}.$$

2. Using Python's numpy library, write a script to perform the following tasks:

   (a) Generate a random 3x2 matrix A.

   (b) Use numpy's built-in function to perform a QR factorisation of A.

   (c) Extract the matrices Q and R from the result of the QR factorisation.

(d) Check the correctness of the factorization by verifying that $QR = A$ within some small tolerance.

(e) Write a function to compute the Frobenius norm of a matrix (which is the square root of the sum of the absolute squares of its elements), and use this function to calculate the Frobenius norm of the difference between QR and A.

3. Consider the matrix

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 1 & 1 \\ 4 & 3 & 2 \end{pmatrix}$$

Perform a manual QR factorisation using the Householder reflections process. Compute the matrices Q and R. Verify the result by checking that $A = QR$.

Answer: Householder reflections on the given matrix A:

$$A = \begin{pmatrix} 2 & 3 & 4 \\ 1 & 1 & 1 \\ 4 & 3 & 2 \end{pmatrix} = R^{(0)}.$$

For $k = 1$:

First, we compute the vector $u^{(1)}$, which is simply the first column of A:

$$u^{(1)} = \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix}$$

Next, we calculate the Euclidean norm of $u^{(1)}$:

$$|u^{(1)}|_2 = \sqrt{2^2 + 1^2 + 4^2} = \sqrt{21}$$

Now, we calculate $\hat{v}^{(1)}$:

$$\begin{aligned} \hat{v}^{(1)} &= u^{(1)} + \text{sign}(u_1^{(1)}) \cdot |u^{(1)}|_2 \cdot e_1 \\ &= \begin{pmatrix} 2 \\ 1 \\ 4 \end{pmatrix} + 1 \cdot \sqrt{21} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 + \sqrt{21} \\ 1 \\ 4 \end{pmatrix} \end{aligned}$$

Next, we normalize $\hat{v}^{(1)}$ to get $v^{(1)}$:

$$v^{(1)} = \frac{1}{|\hat{v}^{(1)}|_2} \hat{v}^{(1)} = \frac{1}{\sqrt{(2 + \sqrt{21})^2 + 1^2 + 4^2}} \begin{pmatrix} 2 + \sqrt{21} \\ 1 \\ 4 \end{pmatrix}$$

Now, we compute the first Householder matrix $H_1 = I_3 - 2v^{(1)}(v^{(1)})^T$.

Lastly, we calculate the next R matrix:

$$R^{(1)} = H_1 R^{(0)}$$

You would then continue this process for $k = 2$. Finally, you would end up with matrices $R = R^{(2)}$ and $Q = Q_1 Q_2$, and you would verify that $A = QR$.

The Frobenius norm of the difference between A and QR would provide a measure of the accuracy of the QR factorization. You can calculate the Frobenius norm as the square root of the sum of the absolute squares of its elements. If the QR factorization is exact, the Frobenius norm of the difference would be zero.

4. Implement a Python function that performs the Jacobi method for a given system of linear equations, maximum number of iterations, and a specified tolerance for the norm of the residual vector. The function should return the solution vector and the number of iterations performed.

   Here is the function signature to get you started:

```
def jacobi_method(A, b, x0, max_iter, tol):
    # A is the input matrix
    # b is the right-hand side vector
    # x0 is the initial guess for the solution
    # max_iter is the maximum number of iterations
    # tol is the tolerance for the norm of the residual vector
    ...
```

   Test your function on a 3×3 system of equations. Experiment with different initial guesses, maximum number of iterations, and tolerance levels. Plot the norm of the residual vector as a function of the number of iterations. What do you observe about the convergence of the method?