Numerical Analysis

Computer Exercise 1

1. **Computing the QR factorization.**
The QR factorization of a matrix

$$A = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^{m,n}, \quad m \geq n$$

can be computed by a sequence of Householder transformations. Here $a_i \in R^m$ denotes the i-th column of the matrix A.

Given a vector v the Householder transformation matrix H is defined as

$$H = I - 2\frac{vv^T}{v^T v}.$$

Recall that if x is a vector, by defining $v = x + \|x\|_2 sign(x_1)e_1$ we get
$Hx = -sign(x_1)\|x\|_2 e_1$.
The QR factorization can be computed using Algorthms 1 and 2 given below.

Algorithm1: Housholder QR factorization (producing R)

for $k = 1$ to n
$x = A_{k:m,k}$
$v_k = sign(x_1)\,\|x|_2\, e_1 + x$
$v_k = v_k/\|v_k\|_2$
$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$

Algorithm2: Housholder QR factorization (producing the product Qx)

for $k = n$ down to 1
$x_{k:m} = x_{k:m} - 2v_k(v_k^T x_{k:m})$

An alternative way to compute the QR factorization is to use the classical or the modified Gram-Schmidth algorithms.

Algorithm3: Classical Gram-Schmidth

for $j = 1$ to n
$v_j = a_j$
for $i = 1$ to $j - 1$
$r_{ij} = q_i^T a_j$
$v_j = v_j - r_{ij} q_i$
$r_{jj} = \|v_j\|_2$
$q_j = v_j / r_{jj}$

Algorithm4: Modified Gram-Schmidth

$$
\begin{array}{ll}
\text{for} & i = 1 \text{ to } n \\
& v_i = a_i \\
\text{for} & i = 1 \text{ to } n \\
& r_{ii} = \|v_i\| \\
& q_i = v_i / r_{ii} \\
& \text{for } j = i+1 \text{ to } n \\
& \quad r_{ij} = q_i^T v_j \\
& \quad v_j = v_j - r_{ij} q_i
\end{array}
$$

Q1 Using Algorithm 1 write a matlab function [V,R] = myhouseholder(A) that computes the QR factorization of A using Housholder transfomrations. The output variables are $V \in \mathbb{R}^{m,n}$ whose columns are the vectors defining the succesive Housholder transformations and $R \in \mathbb{R}^{n,n}$ an upper triangular matrix.

Q2 Using Algorithm 2 write a matlab function Q = findQ(V) that takes the matrix V produced by myhouseholder as input and computes the corresponding $m \times m$ orthogonal matrix Q.

Q3 Combine algorithms of Q1 and Q2 to a matlab function that returns both Q and R, i.e., [Q,R]=myqr(A).

Q4 Write a matlab function myclgs which implements the classical Gram-Schmidt algorithm Algorithm3. The input is A and the output is Q and R.

Q5 Write a matlab function mymgs which implements the modified Gram-Schmidt algorithm Algorithm4. The input is A and the output is Q and R.

Carefully debug your programs by testing them on small examples first.

2. **Discrete Legendre polynomials** Write a Matlab function to compute the first n Legendre polynomials evaluated on m equally spaced points on the interval $[-1, 1]$.

To do so you will use the Vandermonde matrix A defined as

$$
A = \begin{bmatrix}
1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\
1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\
\vdots & \vdots & \vdots & & \vdots \\
1 & x_m & x_m^2 & \dots & x_m^{n-1}
\end{bmatrix}
$$

whose columns are the monomials x^0, x^2, ..., x^{n-1} evaluated at points $x_1, x_2, \dots, x_m$ discretizing the interval $[-1, 1]$.

HELP: Using the following lines in matlab

```
> x = (-100,100)'/100;          set x to a discretization of[-1,1]
> A = [x.^0  x.^1  x.^2  x.^3]; construct Vandermonde matrix
> [Q,R] = qr(A,0);              Find its reduced QR factorization
```

produces the QR factorization of A (using the build-in qr matlab function which is based on the Householder transformation).

The columns of the matrix Q are essentially the first four Legendre polynomials $P_0(x) = 1, P_1(x) = x, P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}, P_3(x) = \frac{5}{2}x^3 - \frac{3}{2}x$ but they should be normalized so as to satisfy $P_k(1) = 1$. To do so we can normalize the columns of Q in the following way,

$> scaleQ = Q(201, :);$	select the value of Q at 1
$> Q = Q * diag(1./scaleQ);$	Rescale the columns of Q by the value of Q at 1

Following this model write a matlab function mylegendrepoly that computes the first n Legendre polynomials evaluated on m equally spaced points. The function should have as input the integers n, m and a third input argument which determines whether to call the classical Gram Schmidt Algorithm (myclgs), the modified one (mymgs) or your QR algorithm using the Householder transformation (myqr).

Q6 Call mylegendrepoly with various choices of input parameters, plotting the resulting Legendre polynomials. For larger n you may not want to plot all of them. How large does n have to get before classical Gram-Schmidt is so inaccurate that the plot is visibly different from the one you get with myqr? Show both sets of polynomials on the same graph, using different plot marks so the difference is clear.

Q7 Write a Matlab script which plots R_{jj} as a function of j for a matrix A with exponentially decreasing singular values. Use semilogy for the plot and show the results for all 3 QR routines myclgs, mymgs and myqr plotting the results with 3 different marks. HELP: To produce a matrix with exponentially decreasing singular values you can use the following lines in matlab

$> [U, X] = qr(randn(100));$	set U to a random orthogonal matrix
$> [V, X] = qr(randn(100));$	set U to a random orthogonal matrix
$> S = diag(2\hat{}(-1 : -1 : -100));$	set S to a diagonal matrix with exponentially decreasing entries
$> A = U * S * V;$	set A to a matrix with these entries as singular values

Comment on the results. How is this related with rounding errors and the instability of the classical Gram-Schmidt algorithm.

3. **Least squares curve fitting.**

Formulate the least squares problem you get when you are fitting a polynomial to a series of measurement data. If you have a series of m points x_i, $i = 1, \ldots, m$ equally spaced in $[-1, 1]$ and use polynomials of degree $n-1$ you will get the $m \times n$ matrix A. The measurements y_i will make up a column vector with m elements.

Q8 Study the singular value decomposition of the matrix A. Use $m = 101$ and $n = 20$. The singular values σ_k decrease rather fast in size for large k. Use semilogy to plot them. If the data are given to moderate accuracy, say 4 decimals, there is no great benefit to use any singular values σ_k smaller than 10^{-4}. How large numerical rank r should we choose then?

Q9 Plot the columns of A as functions of x. See that they start to look very similar for higher degrees, this means that the matrix columns are nearly linearly dependent.

Q10 Plot the first columns of U, the matrix of (left) singular vectors. They are orthogonal to each other. How does that show up? Look at the number of sign changes!

Q11 As a comparison, do a QR factorization of A (using myqr). Now each column q_k of the orthogonal factor Q is a polynomial of degree $k-1$. Plot them and compare to the columns of U!

Remark: The kth column q_k of Q is a polynomial of degree $k-1$. All columns u_k of U are polynomials of the full degree $m-1$ but they should oscillate like the trigonometric functions, $\sin(k\pi t)$.

Compute polynomials to fit the following data series:

(a) $y(x) = e^{-x}$. This is an entire function, which can be approximated by polynomials of any order. The approximation is better the higher degree you choose.

(b) $y(x) = |x|$. This function has a discontinuous derivative at $x = 0$ and you always get a large error close to that point.

BONUS $y(x) = exp(x) + \epsilon * randn(m, 1)$, where the random perturbation has size given by the parameter ϵ. Choose a sequence of values of $\epsilon = 10^{-16}, 10^{-12}, 10^{-8}, \ldots$ until you do not longer recognize the exponential function.

Q12 Plot the function y as well as the approximating polynomial. Look at the transformed right hand side $b = U^T y$. Its elements b_k should get smaller and smaller for larger k. Compare to the singular values of the matrix A. When an element b_k is significantly larger in absolute value than the singular value σ_k, we get a large component in the solution which does not decrease the residual by a significant amount. For which k does this happen?

BONUS The case with a random perturbation is of special interest. Here it is not useful to include any singular values σ_k smaller than the perturbation ϵ. Fix ϵ to a well-chosen value. Plot the residual $r_k(x) = y(x) - p_k(x) = y - A * x_k$ in some interesting cases k. Here x_k is the solution one gets by using the first k elements in b and a rank k approximation to the matrix A.

BONUS Plot $\|r_k\|_2$ as a function of $\|x_k\|_2$. There is a break even point when increasing the rank k only gives a marginally smaller residual norm at the cost of a very large solution norm. This last plot is a simple example of an L-curve.

BONUS The least square solution can be also computed using either QR factorization or the normal equations. Compare the least square solution obtained by the three methods (using SVD, QR and normal equations). What do you observe?

You should submit all your files and your report by email by 20/11/2014 at 9pm. Late submissions will not be considered. All your codes and the report should be compressed in one file named EA1xxx.tgz (or zip) where xxx is your AM. Include enough comments in the code to convince the reader that you understand the algorithms. Also comment and explain the results obtained in your report.