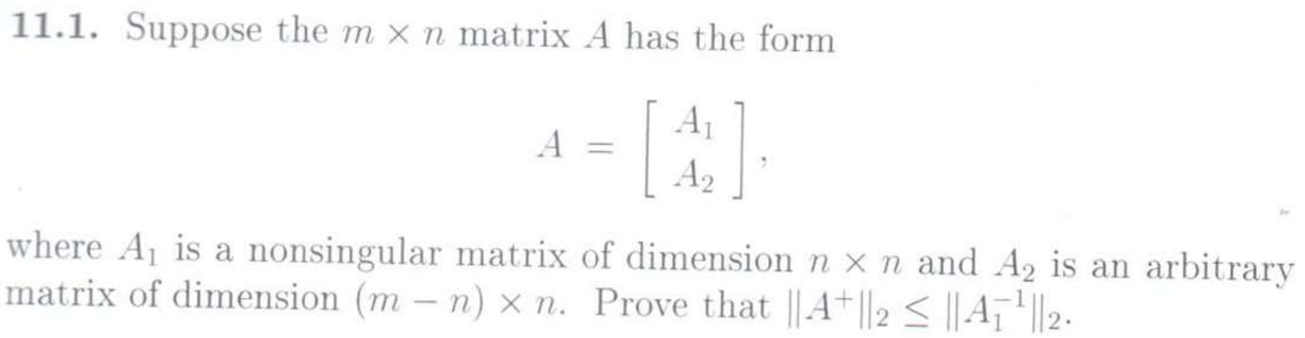
HW4

Monday, February 26, 2024 8:34 PM



11.1. Suppose the $m \times n$ matrix A has the form

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where A_1 is a nonsingular matrix of dimension $n \times n$ and A_2 is an arbitrary matrix of dimension $(m-n) \times n$. Prove that $\|A^+\|_2 \leq \|A_1^{-1}\|_2$.

$A^+ = (A^TA)^{-1}A^T \qquad A = QR \qquad Q^T = Q^{-1} \qquad \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix}$

$$A^+ = (R^T \overset{I}{\cancel{Q^TQ}} R)^{-1} R^T Q^T$$

$$= R^{-1} \overset{I}{\cancel{R^{T^{-1}} R^T}} Q^T$$

$$= R^{-1} Q^T$$

$$= R^{-1} Q_1^{-1} Q_1 Q^T$$

$$= A_1^{-1} Q_1 Q^T$$

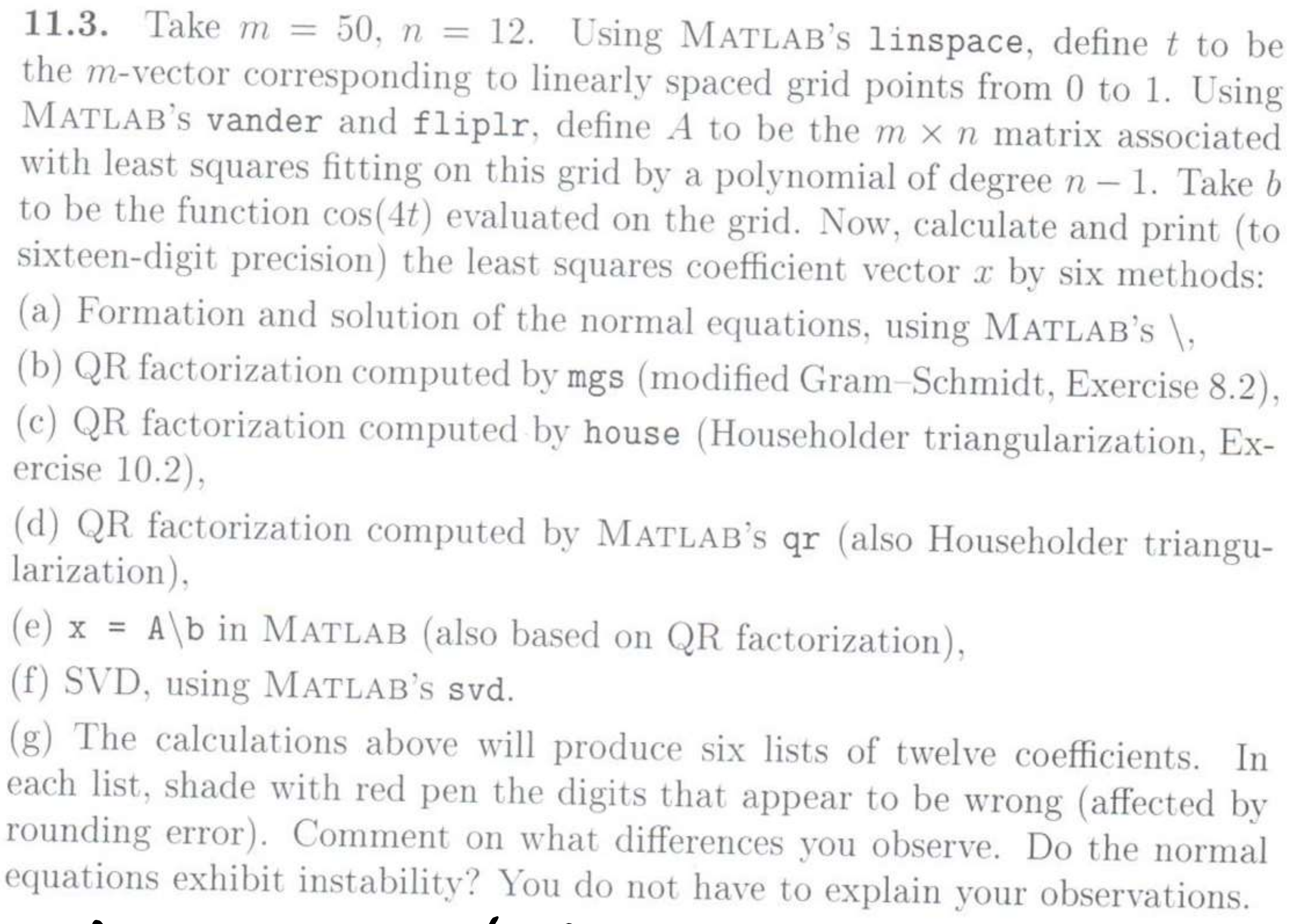
$A_1 = Q_1 R \Rightarrow$

$A_1^{-1} = R^{-1} Q_1^T$

$\|A^+\|_2 = \|A_1^{-1} Q_1 Q^T\|_2 \leq \|A_1^{-1}\|_2 \|Q_1 Q^T\|_2$

Note since Q_1 and Q^T are unitary, $Q_1 Q^T$ is unitary so the singular values are 1.

$\therefore \|Q_1 Q^T\|_2 = 1 \Rightarrow \|A^+\|_2 \leq \|A_1^{-1}\|_2$



11.3. Take $m = 50$, $n = 12$. Using MATLAB's `linspace`, define t to be the m-vector corresponding to linearly spaced grid points from 0 to 1. Using MATLAB's `vander` and `fliplr`, define A to be the $m \times n$ matrix associated with least squares fitting on this grid by a polynomial of degree $n - 1$. Take b to be the function $\cos(4t)$ evaluated on the grid. Now, calculate and print (to sixteen-digit precision) the least squares coefficient vector x by six methods:

(a) Formation and solution of the normal equations, using MATLAB's \,

(b) QR factorization computed by `mgs` (modified Gram–Schmidt, Exercise 8.2),

(c) QR factorization computed by `house` (Householder triangularization, Exercise 10.2),

(d) QR factorization computed by MATLAB's `qr` (also Householder triangularization),

(e) `x = A\b` in MATLAB (also based on QR factorization),

(f) SVD, using MATLAB's `svd`.

(g) The calculations above will produce six lists of twelve coefficients. In each list, shade with red pen the digits that appear to be wrong (affected by rounding error). Comment on what differences you observe. Do the normal equations exhibit instability? You do not have to explain your observations.

a) $A^TAx = A^Tb \Rightarrow x = (A^TA)\backslash A^Tb$

b) $QRx = b$ $\quad x = R\backslash Q^Tb$ w/MGS

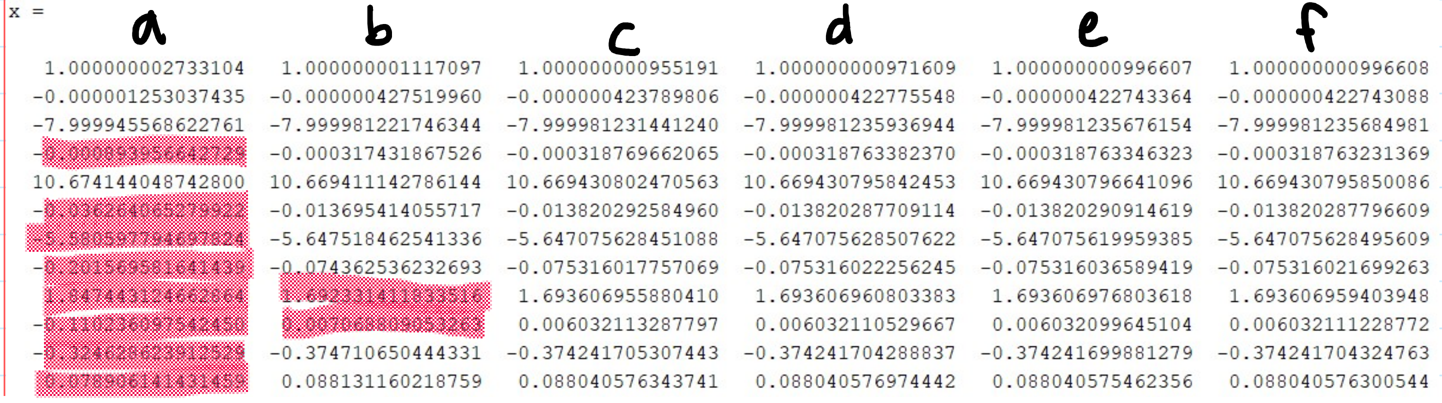
c) Same w/ Householders

d) Same w/ qr( )

e) $x = A\backslash b$

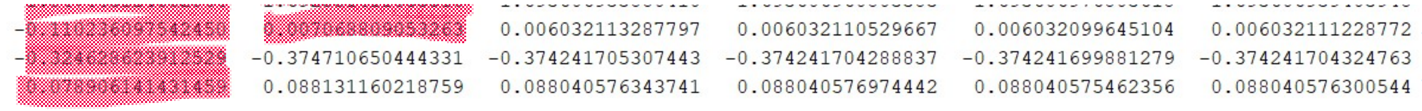
f) $U\Sigma V^Tx = b \Rightarrow x = V\Sigma\backslash U^Tb$

g)



```
x =
         a                   b                   c                   d                   e                   f

  1.000000002733104   1.000000001117097   1.000000000955191   1.000000000971609   1.000000000996607   1.000000000996608
 -0.000001253037435  -0.000000427519960  -0.000000423789806  -0.000000422775548  -0.000000422743364  -0.000000422743088
 -7.999945568622761  -7.999981221746344  -7.999981231441240  -7.999981235936944  -7.999981235676154  -7.999981235684981
 -0.000893956642729  -0.000317431867526  -0.000318769662065  -0.000318763382370  -0.000318763346323  -0.000318763231369
 10.674144048742800  10.669411142786144  10.669430802470563  10.669430795842453  10.669430796641096  10.669430795850086
 -0.036264065279922  -0.013695414055717  -0.013820292584960  -0.013820287709114  -0.013820290914619  -0.013820287796609
 -5.580597794697824  -5.647518462541336  -5.647075628451088  -5.647075628507622  -5.647075619959385  -5.647075628495609
 -0.201569581641439  -0.074362536232693  -0.075316017757069  -0.075316022256245  -0.075316036589419  -0.075316021699263
  1.647443124662864   1.692331411833516   1.693606955880410   1.693606960803383   1.693606976803618   1.693606959403948
 -0.110236097542450   0.007068809053263   0.006032113287797   0.006032110529667   0.006032099645104   0.006032111228772
 -0.324626623912529  -0.374710650444331  -0.374241705307443  -0.374241704288837  -0.374241699881279  -0.374241704324763
  0.078906141431459   0.088131160218759   0.088040576343741   0.088040576974442   0.088040575462356   0.088040576300544
```



```
-0.110236097542450   0.007068809053263   0.006032113287797   0.006032110529667   0.006032099645104   0.006032111228772
-0.324628623912529  -0.374710650444331  -0.374241705307443  -0.374241704288837  -0.374241699881279  -0.374241704324763
 0.078906141431459   0.088131160218759   0.088040576343741   0.088040576974442   0.088040575462356   0.088040576300544
```

The majority gave similar answers, a) did seem to be the most off since the values did not agree the majority of the time.

2. **[Q3] (15 points)** Implement Householder reduced QR factorization with *column pivoting*. At step k, consider columns k through n of the current A (has been updated in previous steps), find the column j ($k \le j \le n$) such that $\|A(k\!:\!m, j)\|_2 = \max_{k\le\ell\le n} \|A(k\!:\!m,\ell)\|_2$, and switch columns k and j. Similar to GEPP, we need a permutation matrix P to record column swapping, such that $AP = QR$ numerically.

Generate a new test matrix as follows.

```
U = randn(1024,10);
A4 = U*randn(10,15);
```

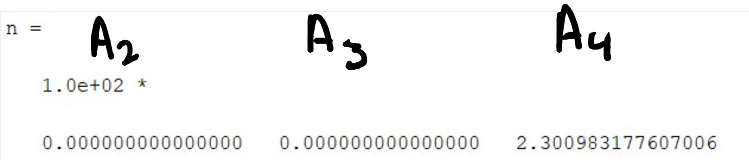
(a) Test your code on A_2, A_3 (see HW3 [Q2]) and A_4, compare your upper triangular matrices with those generated by MATLAB's command `[Q,R,P] = qr(A,0);`
(b) Show that the diagonal elements of R are monotonically decreasing in modulus.
(c) Comment on the use of this algorithm to extract a set of numerically linearly *independent* columns from a matrix with numerically linearly *dependent* columns.

a) Let R be from my code, let Rm be from Matlab

A_2 $\|R-R_m\|_2 = O(\Sigma_m)$

A_3 $\|R-R_m\|_2 = O(\Sigma_m)$

A_4 $\|R-R_m\|_2 \neq O(\Sigma_m)$



A_2 A_3 A_4

```
n =

   1.0e+02 *

   0.000000000000000   0.000000000000000   2.300983177607006
```

b) In Householder QR,

$$v_i = \frac{x - \alpha e_i}{\|x - \alpha e_i\|_2} \qquad \alpha = s_{x_i}\|x\|$$

$$r_{ii} = \tfrac{1}{2}\|v_i\|$$

Where x is the same vector used in Householders at each step.

Consider $s_{x_i} = 1$

Thus $4\|r_{ii}\|^2 = (x_i - \|x\|)^2 + \Sigma_{k=i+1}^m x_k^2$

$= \cancel{x_i^2} + 2x_i\|x\| - \cancel{\|x\|^2} + \cancel{\Sigma_{k=i+1}^m x_k^2}$

$= 2x_i\|x\|$

Consider $s_{x_i} = -1$

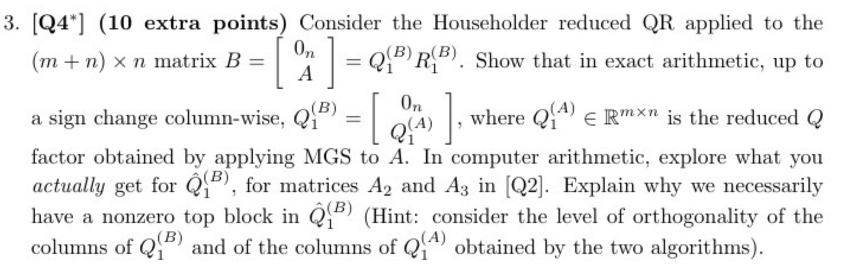
$4\|r_{ii}\|^2 = (-|x_i| + \|x\|)^2 + \Sigma_{k=i+1}^m x_k^2$

$= \|x\|^2 - 2|x_i|\|x\| + \|x\|^2 = 2(\|x\|^2 - |x_i|\|x\|)$

$4\|r_{ii}\|^2 = (-|x_i| + \|x\|)^2 + \sum_{k=i+1} x_k$
$= \|x\|^2 - 2|x_i|\|x\| + \|x\|^2 = 2(\|x\|^2 - |x_i|\|x\|)$

Since we maximized $\|x\|$ at each step and $\|r_{ii}\|^2 \propto \|x\|^2$, $\|r_{ii}\|$ will be montonically decreasing at each step.

c) By maximizing $\|x\|$ at each step, we reduce the largest column left at each point.
This avoids round off error that would make two vectors L.D. numerically.



3. **[Q4*]** **(10 extra points)** Consider the Householder reduced QR applied to the $(m+n) \times n$ matrix $B = \begin{bmatrix} 0_n \\ A \end{bmatrix} = Q_1^{(B)} R_1^{(B)}$. Show that in exact arithmetic, up to a sign change column-wise, $Q_1^{(B)} = \begin{bmatrix} 0_n \\ Q_1^{(A)} \end{bmatrix}$, where $Q_1^{(A)} \in \mathbb{R}^{m\times n}$ is the reduced Q factor obtained by applying MGS to A. In computer arithmetic, explore what you *actually* get for $\hat{Q}_1^{(B)}$, for matrices A_2 and A_3 in [Q2]. Explain why we necessarily have a nonzero top block in $\hat{Q}_1^{(B)}$ (Hint: consider the level of orthogonality of the columns of $Q_1^{(B)}$ and of the columns of $Q_1^{(A)}$ obtained by the two algorithms).

$$B = \begin{bmatrix} 0 \\ A \end{bmatrix} = \begin{bmatrix} 0 \\ Q^A R^A \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ Q_1^A R^A & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ Q_1^A & Q_2^A \end{bmatrix} \begin{bmatrix} R^A \\ 0 \end{bmatrix}$$

It follows $Q^B = \begin{bmatrix} 0 & 0 \\ Q_1^A & Q_2^B \end{bmatrix}$ & $R^B = \begin{bmatrix} R^A \\ 0 \end{bmatrix}$.

Thus $Q_1^B = \begin{bmatrix} 0 \\ Q_1^A \end{bmatrix}$.

The top block is non-zero because 0 is L.D. so there isn't any vector to be reflected over. The columns of Q^B are more numerically L.D. because of the 0 matrix. So the updating introduces error.

I'm not sure how to show the results of this code effectively, I promise I did it.

```
format long

u = (-1:2/40:1)';
A2 = u.^(0:23);
A3 = u.^(0:40);
U = randn(1024,10);
A4 = U*randn(10,15);

%Problem 11.3
t = linspace(0,1,50);
A1 = fliplr(vander(t));
A1 = A1(:,1:12);
b = cos(4*t)';

x1 = (A1'*A1)\A1'*b;

[Q,R] = MGS(A1);
x2 = R\Q'*b;

[Q,R] = HQR(A1);
x3 = R\Q'*b;

[Q,R] = qr(A1);
x4 = R\Q'*b;

x5 = A1\b;

[U,S,V] = svd(A1,"econ");
x6 = V*(S\U')*b;

x = [x1 x2 x3 x4 x5 x6];

%problem 3
n2 = prob3(A2);
n3 = prob3(A3);
n4 = prob3(A4);

n = [n2 n3 n4];

%problem 4
[m, n] = size(A2);
B2 = [zeros(n,n); A2];

[m, n] = size(A3);
B3 = [zeros(n,n); A3];

[Q2, R2, P2] = qr(B2, 0);
[Q3, R3, P3] = qr(B3, 0);

function n = prob3(A)
    [Q, R, P] = HQRwP(A);
    [Qm, Rm, Pm] = qr(A, 0);
    n = norm(R-Rm);
end

function [Q, R] = MGS(A)
    [n, m] = size(A);
    Q = A;
    R = zeros(m, m);

    for k = 1:m
        for i = 1:k-1
            R(i,k) = Q(:,i)'*Q(:,k);
            Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
        end
        R(k,k) = norm(Q(:,k));
        Q(:,k) = Q(:,k)/R(k,k);
    end
end

function [Q, R] = HQR(A)
    [m,n] = size(A);
    R = A;
    V = zeros(m,n);
    Q = [eye(n);zeros(m-n,n)];

    for k = 1:n
        x = R(k:m,k);
        e = zeros(length(x),1);
        e(1) = norm(x);
        if x(1) == 0
            beta = 1;
        else
            beta = sign(x(1));
        end
        u = beta*e + x;
        v = u / norm(u);
        R(k:m,k:n) = R(k:m,k:n) - 2*v*(v'*R(k:m,k:n));
        V(:,k) = [zeros(k-1,1);v];
    end

    R = R(1:n,1:n);
    V = fliplr(V);

    for i = 1:n
        Q = Q - 2*V(:,i)*(V(:,i)'*Q);
    end

end

function [Q, R, P] = HQRwP(A)
    [m,n] = size(A);
    R = A;
    P = eye(n);
    V = zeros(m,n);
    Q = [eye(n);zeros(m-n,n)];
    for k = 1:n
        PP = eye(n);
        y = zeros(m-k,1);
        for j = k:n
            y(j,1) = norm(R(k:m,j));
        end

        [B,I] = sort(y,'descend');

        z = R(:,k);
        R(:,k) = R(:,I(1));
        R(:,I(1)) = z;

        z = PP(:,k);
        PP(:,k) = PP(:,I(1));
        PP(:,I(1)) = z;
        P = P*PP;

        x = R(k:m,k);
        e = zeros(length(x),1);
        e(1) = norm(x);
        if x(1) == 0
            beta = 1;
        else
            beta = sign(x(1));
        end
        u = beta*e + x;
        v = u / norm(u);
        R(k:m,k:n) = R(k:m,k:n) - 2*v*(v'*R(k:m,k:n));
        V(:,k) = [zeros(k-1,1);v];
    end

    R = R(1:n,1:n);
    V = fliplr(V);

    for i = 1:n
        Q = Q - 2*V(:,i)*(V(:,i)'*Q);
    end

end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate.
RCOND =  3.180450e-17.
```

Published with MATLAB® R2023b