HW3

Saturday, February 17, 2024 7:46 PM

1. **[Q1] (10 pts)** (a) Determine the eigenvalues, determinant, and singular values of a Householder reflector $H = I - 2\frac{vv^T}{v^Tv}$. For the eigenvalues, give a geometric argument as well as an algebraic proof.

   (b) Consider the *Givens rotation* $G = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$. Give a geometric interpretation of the action of G on a vector in $\mathbb{R}^2$. Do the same analysis as part (a) for G, but no geometric interpretation is needed for the eigenvalues.

a) $\left(I - 2\frac{vv^T}{v^Tv}\right)v = \lambda v$

$v - 2v = \lambda v$

$-v = \lambda v \Rightarrow \lambda = -1$

$\left(I - 2\frac{vv^T}{v^Tv}\right)w = \lambda v \qquad w \perp v$

$w = \lambda v \Rightarrow \lambda = 1$

Eigenvalues are 1 & -1

Determinant is product of eigenvalues. Thus, $-1 \cdot 1 = 1$.

Singular values are $\sqrt{\lambda_{AA^T}}$ where $AA^T = \left(I - 2\frac{vv^T}{v^Tv}\right)^2$. The eigenvalues are 1 with multiplicity of 2 thus $\sigma_A = 1$.

The eigenvalues show that vectors are either reflected, or projected onto itself.

b) $\begin{vmatrix} \cos\theta - \lambda & -\sin\theta \\ \sin\theta & \cos\theta - \lambda \end{vmatrix} = \cos^2\theta - 2\lambda\cos\theta + \lambda^2 + \sin^2\theta = 0$

$\lambda^2 - 2\lambda\cos\theta + 1 = 0$

eigenvalues $\lambda = \frac{2\cos\theta \pm \sqrt{4\cos^2\theta - 4}}{2}$

$\lambda = \cos\theta \pm \sqrt{-\sin^2\theta}$

$\lambda = \cos\theta \pm i\sin\theta = e^{\pm i\theta}$

Determinant $e^{i\theta}e^{-i\theta} = 1$

$\lambda = \cos\theta \pm i\sin\theta = e^{\pm i\theta}$

Determinant $e^{i\theta} e^{-i\theta} = 1$

Singular values

$$GG^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} \cos^2\theta + \sin^2\theta & \cos\theta\sin\theta - \sin\theta\cos\theta \\ \cos\theta\sin\theta - \sin\theta\cos\theta & \sin^2\theta + \cos^2\theta \end{bmatrix} = I_2$$

Thus $\sigma_A = \sqrt{1} = 1$

G rotates a vector by angle θ in $\mathbb{R}^2$

2. **[Q2] (10 pts)** Implement QR factorizations in MATLAB based on
(i) classical Gram-Schmidt (CGS)
(ii) modified Gram-Schmidt (MGS)
(iii) MGS with double orthogonalization, and
(iv) Householder reflectors (for Householder $H = I - \frac{2vv^T}{v^Tv}$, let $v = x + \text{sign}(x_1)\|x\|_2 e_1$, with $\text{sign}(z) = 1$ for $z = 0$ and $e^{i\theta}$ for $z = \rho e^{i\theta} \neq 0$).
Then we construct three matrices as follows.

```
A1 = randn(2^20,15);    %(large but well-conditioned)
u = (-1:2/40:1)';
A2 = u.^(0:23);  %(partial Vandermonde)
A3 = u.^(0:40);  %(full Vandermonde)
```

For each matrix, run the algorithms, then compute $\frac{\|A-\hat{Q}\hat{R}\|_F}{\|A\|_F}$ and $\|\hat{Q}^T\hat{Q} - I_n\|_2$. Draw conclusions about the backward stability of these algorithms, and the orthogonality of the computed Q factors, probably related to the condition numbers of the matrices.

(see code)

```
l1 =

  5×3 string array

    ""     "A1"            ""
    "1"    "1.4167e-16"    "3.5528e-15"
    "2"    "1.4167e-16"    "3.5528e-15"
    "3"    "2.5823e-16"    "0.0068969"
    "4"    "2.4349e-15"    "3.5533e-15"

l2 =

  5×3 string array

    ""     "A2"            ""
    "1"    "1.105e-16"     "1.9466"
    "2"    "1.105e-16"     "1.9466"
    "3"    "5.1303e-16"    "9.9008"
```

All of the algo's seem Backward stable. The Orthogonality of Q for Householders is consistant. The G-S methods don't keep great orthogonality.

great orthogonality.

""	"A2"	""
"1"	"1.105e-16"	"1.9466"
"2"	"1.105e-16"	"1.9466"
"3"	"5.1303e-16"	"9.9008"
"4"	"3.3127e-16"	"9.8635e-16"

l3 =

5×3 string array

""	"A3"	""
"1"	"1.4014e-16"	"10.2844"
"2"	"1.4014e-16"	"10.2844"
"3"	"1.5949e-15"	"17.2975"
"4"	"3.74e-16"	"1.5109e-15"

3. **[Q3] (10 pts)** Evaluate the arithmetic work needed to retrieve the reduced factor $Q_L \in \mathbb{R}^{m\times n}$ from the Householder and Givens reduction of A to R, respectively (second phase of QR). Compare the cost with that for the first phase.

Q_L for Householder.

work
Apply H_i^T at step k $\quad 2(n-k)(m-k+1)$

$$Q_L = H_1^T H_2^T \dots H_n^T \begin{bmatrix} I \\ 0 \end{bmatrix}$$

$$2\sum_{k=1}^{n}(n-k)(m-k+1) = \sum_{k=1}^{n}(m+n)n - k(m+1-n) + k^2$$
$$= (m+1)n^2 - n(m+1+n)(n+1) + \tfrac{1}{3}n(n+1)(2n+1)$$
$$= 2mn^2 - mn^2 - n^3 + \tfrac{2}{3}n^3 + O(mn) + O(n^2)$$
$$= mn^2 - \frac{n^3}{3} + O(mn) + O(n^2)$$

Q_L For Givens

$$Q_L = G_1^T \cdots G_n^T \begin{bmatrix} I \\ 0 \end{bmatrix}$$

Work
Apply G_i^T at step k $\quad 6(n+1-k)(m-k)$

$$6\sum_{k=1}^{n} mn + m - mk - kn - k + k^2$$
$$= 6\sum_{k=1}^{n}(mn+m) - (m+n+1)k + k^2$$
$$= 6mn^2 - 3(m+n+1)n(n+1) + n(n+1)(2n+1)$$
$$= 6mn^2 - 3mn^2 - 3n^3 + 2n^3 + O(mn) + O(n^2)$$
$$= 3mn^2 - n^3 + O(mn) + O(n^2)$$

4. **[Q4] (10 pts)** Implement the algorithm for solving linear system $Ax = b$ or linear least squares problem $\min \|b - Ax\|_2$ based on Householder QR. Make sure that the reduced Q factor is NOT formed explicitly to save the cost of the second phase.

   Then solve the linear least squares problem $\min \|b - Ax\|_2$ where $A = A_2$, and the linear system $Ax = b$, where $A = A_3$ in [Q2], and $b = [1, -1, 1, -1, \ldots]^T$. Report your $\frac{\|b - A\hat{x}\|_2}{\|A\|_2 \|\hat{x}\|_2}$ for both solves, and compare with this quantity associated with the solutions obtained by MATLAB's backslash.

```
l4 =

  3×3 string array

    ""        "Calculated"     "MatLab"
    "A2"      "3.1026e-08"     "3.1026e-08"
    "A3"      "3.3642e-17"     "5.4432e-17"
```

(See code)

Here is a table

```
A1 = randn(2^20,15);
u = (-1:2/40:1)';
A2 = u.^(0:23);
A3 = u.^(0:40);

%problem 2
l1 = list(A1);

l1 = ["","A1",""; l1];

l2 = list(A2);

l2 = ["","A2",""; l2];

l3 = list(A3);

l3 = ["","A3",""; l3];

%problem 4
[nor1, sol1] = LS(A2);
[nor2, sol2] = LS(A3);

l4=["","Calculated", "MatLab"; "A2", nor1, sol1; "A3", nor2, sol2];

l1
l2
l3
l4

function l = list(A)
    l = zeros(4,3);

    [Q, R] = CGS(A);
    [n1, n2] = normp2(A, Q, R);
    l(1,:) = [1, n1, n2];

    [Q, R] = MGS(A);
    [n1, n2] = normp2(A, Q, R);
    l(2,:) = [2, n1, n2];

    [Q, R] = MGSwRO(A);
    [n1, n2] = normp2(A, Q, R);
    l(3,:) = [3, n1, n2];

    [Q, R, V] = HQR(A);
    [n1, n2] = normp2(A, Q, R);
    l(4,:) = [4, n1, n2];
end

function [n1, n2] = normp2(A, Q, R)
    [m, n] = size(Q);
    n1 = norm(A-Q*R,"fro")/norm(A,"fro");
```

```
    n2 = norm(Q'*Q - eye(n));
end

function [nor, sol] = LS(A)
    [Q, R, V] = HQR(A);
    [m, n] = size(A);
    b = -cumprod(-ones(1,m))';
    bp = -cumprod(-ones(1,m))';
    V = fliplr(V);

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

    b = b(1:n,:);

    xh = R\b;
    xt = A\bp;

    nor = norm(bp - A*xh)/(norm(A)*norm(xh));
    sol = norm(bp - A*xt)/(norm(A)*norm(xt));
end

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

    R(1, 1) = norm(A(:, 1));
    Q(:, 1) = A(:, 1) / R(1, 1);

    for j = 2:m
        Q(:, j) = A(:, j);
        for i = 1:j-1
            Q(:, j) = Q(:, j) - Q(:, i) * (Q(:, i)' * A(:, j));
            R(i, j) = Q(:, i)' * A(:, j);
        end
        R(j, j) = norm(Q(:, j));
        Q(:, j) = Q(:, j) / R(j, j);
    end
end

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

    R(1, 1) = norm(A(:, 1));
    Q(:, 1) = A(:, 1) / R(1, 1);

    for j = 2:m
        Q(:, j) = A(:, j);
        for i = 1:j-1
            Q(:, j) = Q(:, j) - Q(:, i) * (Q(:, i)' * A(:, j));
```

```
            R(i, j) = Q(:, i)' * A(:, j);
        end
        R(j, j) = norm(Q(:, j));
        Q(:, j) = Q(:, j) / R(j, j);
    end
end

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

    R(1, 1) = norm(A(:, 1));
    Q(:, 1) = A(:, 1) / R(1, 1);

    for j = 2:m
        Q(:, j) = A(:, j);
        for i = 1:j-1
            Q(:, j) = Q(:, j) - Q(:, i) * (Q(:, i)' * A(:, j));
            R(i, j) = Q(:, i)' * A(:, j);
        end
        for i = 1:j-1
            Q(:, j) = Q(:, j) - Q(:, i) * (Q(:, i)' * A(:, j));
            R(i, j) = R(i, j) + Q(:, i)' * A(:, j);
        end
        R(j, j) = norm(Q(:, j));
        Q(:, j) = Q(:, j) / R(j, j);
    end
end

function [Q, R, V] = 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
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate.
RCOND =  2.550021e-19.
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate.
RCOND =  2.491242e-20.

l1 =

  5×3 string array

    ""      "A1"              ""
    "1"     "1.4167e-16"      "3.5528e-15"
    "2"     "1.4167e-16"      "3.5528e-15"
    "3"     "2.5823e-16"      "0.0068969"
    "4"     "2.4349e-15"      "3.5533e-15"


l2 =

  5×3 string array

    ""      "A2"              ""
    "1"     "1.105e-16"       "1.9466"
    "2"     "1.105e-16"       "1.9466"
    "3"     "5.1303e-16"      "9.9008"
    "4"     "3.3127e-16"      "9.8635e-16"


l3 =

  5×3 string array

    ""      "A3"              ""
    "1"     "1.4014e-16"      "10.2844"
    "2"     "1.4014e-16"      "10.2844"
    "3"     "1.5949e-15"      "17.2975"
    "4"     "3.74e-16"        "1.5109e-15"


l4 =

  3×3 string array

    ""       "Calculated"     "MatLab"
    "A2"     "3.1026e-08"     "3.1026e-08"
    "A3"     "3.3642e-17"     "5.4432e-17"
```

Published with MATLAB® R2023b