Due Friday, March 10 in class

1. (22 points) The goal is to compare the effectiveness of five different techniques for orthogonalizing the columns of a matrix $A$:

   (A) classical Gram-Schmidt $A = \hat{Q}\hat{R}$,
   (B) modified Gram-Schmidt $A = \hat{Q}\hat{R}$,
   (C) doing classical Gram-Schmidt twice: doing a QR factorization of the matrix $\hat{Q}$ that results from a QR factorization of $A$ to get a better $Q$,
   (D) Householder orthogonalization $A = QR$, and
   (E) using $A G^{-T}$ where $GG^T$ is the Cholesky factorization of the matrix $A^T A$ from the normal equations. (It is not difficult to see that $(AG^{-T})G^T$ is a QR factorization of $A$.)

You may write your program in Matlab, C, or Python. Instructions will be given for a Matlab version with modifications in bracketed italics for a C version.

(a) Write a $GS$ function and an $MGS$ function, each having as its parameter a matrix $A$ whose row dimension $m$ is $\geq$ its column dimension $n$ and each returning an $m$ by $n$ matrix $Q$ obtained by orthonormalizing the columns of $A$. The $GS$ function should use classical Gram-Schmidt and the $MGS$ function modified Gram-Schmidt. [In C, also include $m$ and $n$ in the parameter list.] Do not overwrite the $A$ matrix. Other than for $A$ and $Q$, there should not be any array storage used by the algorithm.

(b) For each of algorithms (A)–(E) above, compute the orthogonalization $Q$ of the $n$ by $n$ Hilbert matrix for $n = 2, 3, \ldots, 12$. The $(i,j)$th element of a Hilbert matrix is $1/(i+j-1)$. Print a table with the first column containing the values of $n$ and a remaining five columns containing the following measure of orthogonality

$$-\log_{10} \|I - Q^T Q\|_F$$

for each of the five algorithms. Label each of the columns. For the Householder orthogonalization you may use the Matlab function $qr(\cdot, 0)$. [For a C program, install CLAPACK and use DGEQRF followed by DORGQR to compute the $m$ by $n$ matrix $Q$. For installation notes, see http://www.netlib.org/clapack/readme.install For additional help, consult the teaching assistant. It is not necessary to use optimized BLAS. A wrapper and makefile is available from the web page.] For computing $AG^{-T}$ where $GG^T$ is the Cholesky factorization of $A^T A$, use the Matlab function $\text{chol}$ to produce $G$ and use the transpose operator and the slash operator to do postdivision of $A$ by $G^T$. [For C, use DPOSV to obtain a Cholesky factorization. A wrapper is available from the web page. You probably have to program your own forward substitution to
compute \((G - A^T)^T\). \textit{Do worry} about avoiding division by zero if it causes premature termination of the program. Other Matlab functions used in our solution are \texttt{eye}, \texttt{sum}, and \texttt{diag}.

Turn in a printed copy of your output and your source code listings.

Here is the Matlab code:

\texttt{GS.m}

function res = GS(A)
[m,n] = size(A);
B = zeros(m,n);
if m < n
    fprintf(1, '\nIncorrect input!!!\n');
    res = B;
    return;
end
for k = 1:n
    B(:,k)=A(:,k);
    for j = 1:k-1
        r_jk = transpose(B(:,j))*A(:,k);
        B(:,k) = B(:,k) - B(:,j)*r_jk;
    end
    B(:,k) = B(:,k)/norm(B(:,k));
end
res = B;
return;
end

\texttt{MGS.m}

function res = MGS(A)
[m,n] = size(A);
B = zeros(m,n);
if m < n
    fprintf(1, '\nIncorrect input!!!\n');
    res = B;
    return;
end
B = A;
for k = 1:n
    B(:,k)=B(:,k)/norm(B(:,k));
    for j = k+1:n
        r_jk = transpose(B(:,k))*B(:,j);
        B(:,j) = B(:,j) - B(:,k) * r_jk;
    end
end
res = B;
return;
function measure()
fprintf(1, 'n\tAlgorithmA\tAlgorithmB\tAlgorithmC\tAlgorithmD\tAlgorithmE\n');
for n = 2:12
    A = zeros(n);
    for i = 1:n
        for j = 1:n
            A(i,j)=1/(i+j-1);
        end
    end
    flag = 0;
    Q1 = GS(A);
    Q2 = MGS(A);
    Q3 = GS(Q1);
    [Q4,R4]=qr(A);
    [G,d] = chol(transpose(A)*A);
    if rank(G) == n
        Q5 = A*pinv(G);
        flag = 1;
    end
    temp = eye(n) - transpose(Q1)*Q1;
    error1 = -log10(sqrt(sum(diag(temp' * temp))));
    temp = eye(n) - transpose(Q2)*Q2;
    error2 = -log10(sqrt(sum(diag(temp' * temp))));
    temp = eye(n) - transpose(Q3)*Q3;
    error3 = -log10(sqrt(sum(diag(temp' * temp))));
    temp = eye(n) - transpose(Q4)*Q4;
    error4 = -log10(sqrt(sum(diag(temp' * temp))));
    if flag == 1
        temp = eye(n) - transpose(Q5)*Q5;
        error5 = -log10(sqrt(sum(diag(temp' * temp))));
        fprintf(1, '%d\t%f\t%f\t%f\t%f\t%f\n', n, error1, error2, error3, error4, error5);
    else
        fprintf(1, '%d\t%f\t%f\t%f\t%f\tINF\n', n, error1, error2, error3, error4);
    end
end
end

The output is:

<table>
<thead>
<tr>
<th>n</th>
<th>AlgorithmA</th>
<th>AlgorithmB</th>
<th>AlgorithmC</th>
<th>AlgorithmD</th>
<th>AlgorithmE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>13.012834</td>
<td>13.923977</td>
<td>15.640395</td>
<td>15.561037</td>
<td>11.320727</td>
</tr>
<tr>
<td>4</td>
<td>10.311054</td>
<td>12.374646</td>
<td>15.533988</td>
<td>15.022464</td>
<td>8.620376</td>
</tr>
<tr>
<td>5</td>
<td>7.818435</td>
<td>10.958642</td>
<td>15.603752</td>
<td>15.139241</td>
<td>2.804851</td>
</tr>
<tr>
<td>6</td>
<td>3.619169</td>
<td>9.134468</td>
<td>15.603752</td>
<td>15.082841</td>
<td>INF</td>
</tr>
<tr>
<td>7</td>
<td>1.137963</td>
<td>8.170110</td>
<td>15.290512</td>
<td>15.082841</td>
<td>INF</td>
</tr>
</tbody>
</table>
2. (6 points) Determine the eigenvalues of $xy^T$ where $x$ and $y$ are column vectors.

Case I. $x^T y = 0$
Let $xy^Tv = v\lambda$, $v \neq 0$.
Hence $v\lambda^2 = xy^Tv\lambda = xy^Txy^Tv = 0$.
Therefore $\lambda = 0$; i.e., all eigenvalues are zero.

Case II. $x^T y \neq 0$
Then $(xy^T)x = x(y^Tx)$, so there is a nonzero eigenvalue $x^Ty$.
Also, there are $n-1$ linearly independent vectors $\perp y$.
These are eigenvectors corresponding to the eigenvalue 0.

In either case, the eigenvalues are $y^Tx, 0, \ldots, 0$.

3. (5 points) Apply an orthogonal similarity transformation to the following matrix so that the (3,1) and (4,1) elements of the transformed matrix are both zero:

$$
\begin{bmatrix}
1 & 1 & 3 & 2 \\
-4 & 0 & 1 & 0 \\
0 & 0 & 2 & 1 \\
3 & 5 & 1 & 0
\end{bmatrix}
$$

You may leave your answer as a product of 4 by 4 matrices.

$$
\begin{bmatrix}
-4 \\
0 \\
3
\end{bmatrix}
\text{reflect}
\begin{bmatrix}
5 \\
0 \\
0
\end{bmatrix}, \text{we obtain } v = \begin{bmatrix}
0 \\
-9 \\
0 \\
3
\end{bmatrix} \text{ and } v^Tv = 90.
$$

Then, $P = I - \frac{2vv^T}{v^Tv} = I - \frac{1}{45}
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 81 & 0 & -27 \\
0 & 0 & 0 & 0 \\
0 & -27 & 0 & 9
\end{bmatrix}$

$$
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \frac{3}{5} & 0 & \frac{2}{5} \\
0 & 3 & 0 & \frac{2}{5} \\
0 & -\frac{3}{5} & 0 & \frac{2}{5}
\end{bmatrix}.
$$

Then, $PA = \begin{bmatrix}
1 & 1 & 3 & 2 \\
5 & 3 & -\frac{1}{5} & 0 \\
0 & 0 & 2 & 1 \\
0 & 4 & \frac{2}{5} & 0
\end{bmatrix}$. Finally, $PAP = \begin{bmatrix}
1 & 2 & 3 & 11 \\
5 & -\frac{12}{5} & -\frac{1}{5} & \frac{9}{5} \\
0 & \frac{3}{5} & 2 & 3 \\
0 & -\frac{16}{5} & \frac{7}{5} & \frac{12}{5}
\end{bmatrix}$.

4. (6 points) Prove that the shifted QR iteration preserves an upper Hessenberg form for a matrix with nonzero subdiagonal elements. Hints:

(i) Do a $(1,n-1)$ by $(n-1,1)$ partitioning of $A_k - \sigma_k I$ and $Q_k I$ and an $(n-1,1)$ by $(n-1,1)$ partitioning of $R_{k+1}$, and show that $Q_{k+1}$ must be upper Hessenberg.
(ii) Do not use the fact that $Q_{k+1}$ is orthogonal.

\[ A_{k+1} = Q_{k+1}^T (A_k - \sigma_{k+1} I) Q_{k+1} + \sigma_{k+1} I \quad \text{and} \quad R_{k+1} = Q_{k+1}^T (A_k - \sigma_{k+1} I) \]

\[ A_k - \sigma_{k+1} I = \begin{bmatrix} b^T & \alpha \\ A & a \end{bmatrix}, \quad Q_{k+1} = \begin{bmatrix} p^T & \pi \\ Q & q \end{bmatrix}, \quad \text{and} \quad R_{k+1} = \begin{bmatrix} R & r \\ 0^T & \rho \end{bmatrix} \]

First we show $Q_{k+1}$ is upper Hessenberg. Enough to show $Q$ is triangular.

\[ A_k - \sigma_{k+1} I = Q_{k+1} R_{k+1} \]

$A$ is upper triangular with non-zero diagonal elements.

$R$ is nonsingular because $A$ is.

$Q = R^{-1} A$.

$R^{-1}$ is upper triangular because $R$ is.

$Q$ is upper triangular because $R^{-1}$ and $A$ are.

$A_{k+1}$ is upper Hessenberg because $R_{k+1}$ is upper triangular and $Q_{k+1}$ is upper Hessenberg.