Lab Session 4

MA-423 : Matrix Computations Lab 2018 R. Alam

1. The purpose of this exercise is to compare classical and modified Gram-Schmidt schemes. Let $x_1, \ldots, x_n$ be linearly independent vectors.Then CGS (classical Gram-Schmidt process) and MGS (modified Gram-Schmidt process) that orthonormalize $x_1, \ldots, x_n$ are as follows:

```
for $i = 1\colon n$
    $q_i \colon= x_i$
    for $j = 1\colon i-1$
        $r_{ji} \colon= q_j^* x_i$ /*(for CGS)/
        $r_{ji} \colon= q_j^* q_i$ /*( for MGS )/
    $q_i \colon= q_i - q_j * r_{ji}$
    end
$r_{ii} \colon= \|q_i\|_2$
if $r_{ii} = 0$ then quit else
$q_i \colon= q_i / r_{ii}$
end
```

Your task is to write matlab functions implementing classical and modified Gram-Schmidt schemes:

```
function [Q, R] = cgs(A)
% [Q, R] = cgs(A) employs classical Gram-Schmidt scheme to compute
% an isometry Q, an upper triangular matrix R such that A=QR.
```

```
function [Q, R] = mgs(A)
% [Q, R] = mgs(A) employs modified Gram-Schmidt scheme to compute
% an isometry Q, an upper triangular matrix R such that A=QR.
```

Generate the test matrix A as follows.

```
[U, X] = qr(randn(80));
[V, X] = qr(randn(80));
S = diag( 2 .^ (-1:-1:-80));
A = U*S*V;
```

Now compute QR factorization of A using `cgs`, `mgs` and the matlab function `qr`:

```
[QC, RC] = cgs(A);
[QM, RM] = mgs(A);
[Q, R] = qr(A);
```

To test how close these matrices are to being unitary, compute `norm( QC'*QC-eye(80))`, `norm( QM'*QM-eye(80))`, `norm( Q'*Q-eye(80))`. Which method is worse? Which method gives better result?

To explain your results, plot the absolute values of the diagonal entries of `RC, RM, R`. Use commands

```
x= (1:80)';
hold off
semilogy(x, abs(diag( RC ) ), 'bo')
hold on
semilogy(x, abs(diag( RM ) ), 'rx')
semilogy(x, abs(diag( R ) ), 'k+')
title('abs(diag(R)) for cgs, mgs and qr')
gtext('cgs=o, mgs = x, qr=+')
```

What is your conclusion from this plot? Next, re-orthonormalise `QC` as follows:

```
[QQC, RRC] = cgs(QC);
RR= RRC*RC;
```

Again compute `norm( QQC'*QQC-eye(80))`, plot the absolute values of diagonal entries of `RR`, `RM`, `R` and comment on the results.

Comment: Due to rounding errors we cannot expect our algorithms to produce vectors which are exactly orthogonal. At best we can hope that $|\langle Q(:,i), Q(:,j)\rangle| = \mathcal{O}(\mathbf{u})$ for $i \neq j$ where $\mathbf{u}$ is the unit roundoff. Since the matrix $Q \in \mathbb{C}^{m\times n}$ produced by each of the above algorithms is isometric, in exact arithmetic it must satisfy $\mathtt{Q}' * \mathtt{Q} - \mathtt{eye(n,n)} = 0$. However, if they deviate slightly from orthonormality then $\mathtt{Q}' * \mathtt{Q} - \mathtt{eye(n,n)}$ will be close to zero. Therefore we take $\|\mathtt{Q}' * \mathtt{Q} - \mathtt{eye(n,n)}\|_2$ as a measure of deviation from orthonormality. This acts as a criterion for analyzing the performance of any algorithm for producing orthonormal vectors. The Q produced by the Modified Gram-Schmidt (MGS) algorithm satisfies

$$\|\mathtt{Q}' * \mathtt{Q} - \mathtt{eye(n,n)}\|_2 \approx \mathbf{u} * \mathtt{cond(V)}.$$

This indicates that although MGS is usually better than CGS in the presence of rounding error, it is unlikely to perform well when $\mathtt{cond(V)}$ is large, that is, when the columns of V are nearly linearly dependent. However if $\mathtt{cond(V)}$ satisfies $1 \ll \mathtt{cond(V)} \ll 1/\mathbf{u}$, then $\mathbf{u} * \mathtt{cond(V)} \ll 1$ and in such a case MGS will return a Q which is quite close to being an isometry. If we are not satisfied with the value of $\|Q' * Q - \mathtt{eye(n,n)}\|_2$ after running MGS once then we may run it once again with the Q produced from the first run as the input matrix V. This process is call reorthogonalization. So, if $\hat{Q}$ is the matrix produced from the second run of the MGS then

$$\|\hat{Q}' * \hat{Q} - \mathtt{eye(n,n)}\|_2 \approx \mathbf{u} * \mathtt{cond(Q)} \approx \mathbf{u}$$

as $\mathtt{cond(Q)}$ will be quite small. On the other hand, if we decide to do a reorthogonalization in the beginning itself, without waiting to check the deviation from orthonormality in the first run of the MGS algorithm, then the MGS algorithm may be modified to orthogonalize the vectors twice at each step. This leads to a new algorithm called Modified Gram Schmidt with Reorthogonalization. You may take a look at the code for this process on page 233 of *Fundamentals of Matrix Computations* by *D. S. Watkins.*

> **Take Home Problem.** Write a function program $[\mathtt{Q}, \mathtt{R}] = \mathtt{mgsrep(V)}$ that performs Modified Gram Schmidt with Reorthogonalization by making appropriate changes to your function program `mgs`.

2. Consider the n-by-n Hilbert matrix H (use MATLAB command `H = hilb (n)` to generate H). Your task is to use different methods listed below to orthonormalize the columns of H for $n = 7$ and $n = 12$.

   (a) Classical Gram-Schmidt method (CGS).

   (b) Modified Gram-Schmidt method (MGS).

   (c) Modified Gram-Schmidt method applied twice.

   (d) (**Take Home Problem**) Modified Gram-Schmidt with reorthogonalization.

   (e) QR decomposition with reflectors. Use MATLAB command `[Q,R] = qr(H, 0)`, which produces an 'economy size' QR decomposition of H with Q being an isometry.

   Examine the deviation from orthonormality by computing $\|Q' * Q - \mathtt{eye(n)}\|_2$ in each case (MATLAB command `norm(eye(n)-Q'*Q)`). Also check the residual `norm(H - Q*R)`.

   Find the condition number of H and check whether or not the matrix Q obtained from the MGS program satisfies $\|Q' * Q - \mathtt{eye(n)}\|_2 \approx u * \mathtt{cond(H)}$.

   Did you get what you would expect in light of the values of unit roundoff $\mathbf{u}$ and cond(H)? Which among all the above methods produces the smallest deviation from orthonormality?

3. **Analysis of "Filip" data set from NIST:** The filip data set consists of several dozen observations of a variable y at different x. Your task is to model y by a polynomial $p(x)$ of degree 10. You will find the filip dataset at the following URL:

   `http://www.itl.nist.gov/div898/strd/lls/data/Filip.shtml`

   This dataset is controversial because the NIST certified polynomial cannot be reproduced by many algorithms. You task is to report what MATLAB does with it.

   (a) Your first task is to download the data from the above website. Next, extract the value of x and y and load the data into MATLAB. Plot y versus x (plot it with '.') and then invoke Basic Fitting tool available under the Tools menu on the figure window. Select the 10th degree polynomial fit. (Ignore the warning that MATLAB may give.) From the Tools menu compute the coefficients of the polynomial fit. How do the coefficients compare with the certified values on NIST web page? How does the plotted fit compare with the graphic on the NIST Web page? The basic fit tools also displays the norm of the residuals $\|r\|$. Compare this with the NIST quantity "Residual Standard Deviation", which is $\frac{\|r\|}{\sqrt{n-p}}$. Here p is the degree of the polynomial and n is the number of data.

   (b) Next, examine the dataset by using the following methods to compute the polynomial fit. Explain all the warning messages you received during these computations.

   - MATLAB Backslash command.
   - Pseudoinverse (`pinv`).
   - Normal equation (theoretically the LSP is of full rank).
   - Certified coefficients: Obtain the coefficients from the NIST Web page.

   Prepare a table giving coefficients of the polynomial fit and the norm of the residuals obtained by each method. Plot the polynomial fits. Use dots, '.' at the data values and plot the curves $y = p(x)$ by evaluating $p(x)$ at a few hundred points over the range of the x's. Some plots may not be visibly distinct. Which methods produce which plots? Generate similar plots as given in the NIST web page.

4. **Planetary Orbit:** Consider the quadratic form $q(x, y) := ax^2 + bxy + cy^2 + dx + ey + f$. The set $\mathcal{C} := \{(x, y) \in \mathbb{R}^2 : q(x, y) = 0\}$ is a *conic section.* Cutting the surface $z = q(x, y)$ by the place $z = 0$, one obtains the conic $\mathcal{C}$. Write a MATLAB script using commands `meshgrid` and `countor` to generate $\mathcal{C}$.

   Suppose that a planet follows an elliptical orbit. Here are ten observations of its position in (x, y) plane:

$$\begin{aligned} x &= \begin{bmatrix}1.02 & 0.95 & 0.87 & 0.77 & 0.67 & 0.56 & 0.44 & 0.30 & 0.16 & 0.01\end{bmatrix} \\ y &= \begin{bmatrix}0.39 & 0.32 & 0.27 & 0.22 & 0.18 & 0.15 & 0.13 & 0.12 & 0.13 & 0.15\end{bmatrix}\end{aligned}$$

   (a) Determine the conic section that fits this data (in the sense of least squares). This can be done by setting one of the coefficients, say, $f = 1$ (WHY?) and solving the resulting 10-by-5 over determined LSP using backslash command. Plot the orbit (that is the conic) with x on the x-axis and y on the y-axis. Superimpose the ten data points on the plot. Write a single MATLAB script that implements the above task.

   (b) Conclude that the LSP is nearly rank deficient. Your next task is to illustrate the effect of small perturbation in the data to the orbit of the planet. Perturb the data x and y by adding to each component a random number uniformly distributed in the interval $[-0.005, 0.005]$. Compute the new coefficients resulting from the perturbed data. Plot the new orbit on the same plot of the old orbit. Write a MATLAB script that implements the job. Run the script a couple of times and comment on your results.

*** End ***