Lab Session 8

MA-423 : Matrix Computations Lab 2017 R. Alam

---

1. The function `eigshow` is available in the MATLAB demos directory. The input to `eigshow` is a real, 2-by-2 matrix A, or you can choose an A from a pull-down list in the title. Initially, `eigshow` plots the unit vector $x = [1, 0]^T$, as well as the vector Ax, which starts out as the rst column of A. You can then use your mouse to move x, shown in green, around the unit circle. As you move x, the resulting Ax, shown in blue, also moves. What is the shape of the resulting orbit of Ax? SVD tells us that the blue curve is an ellipse. `eigshow` provides a proof by GUI.

2. The purpose of this example is to illustrate that Householder QR factorization (or for that matter QR factorization obtained by using rotations) is backward stable, that is, computed QR factors are the exact QR factors of a slightly perturbed matrix.

```
>>R = triu(randn(50)); % compute a 50-by-50 random upper triangular matrix.
>>[Q, X] = qr(randn(50)); % compute a 50-by-50 random unitary matrix.
>>A = Q*R; % A is a matrix with known QR factors.
>>[S, T] = qr(A); % Compute Householder QR factorization of A.
```

   How accurate are `S` and `T`? Show that `S` and `T` are very far from the known QR factors Q and R of A. This illustrates that the Householder QR factrization algorithm is not what is called **forward stable**, that is, the computed factors are not close to the exact factors. Compute the errors

```
>>[norm( Q-S ), norm( R-T )] % what do you observe?
```

   Now test the backward stability of the Householder QR factorization. Setting `E = S*T - A`, we have `A+E = S*T`. Hence if $\|E\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$ then clearly the algorithm `qr` is backward stable. Compute

```
>>norm(A-S*T) % what is your conclusion?
```

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.

*** End ***