Lab Session 6

MA-423 : Matrix Computations Lab 2017 R. Alam

The Least Squares Problem (LSP) $Ax = b$ has a solution where the fit is good if b is nearly in the range $R(A)$ of A or in other words the angle θ between b and Ax is very small. The purpose of the following exercise is to show that in such cases, the QR method of solving the LSP $Ax = b$ is better than Normal Equations method.

1. Use the linspace command to generate a *column vector* X consisting of 50 equally spaced points between 0 and 1. Generate the Vandermonde matrix which has columns $X.^{i-1}$ for $i = 1 : 7$. Choose $\mathtt{w} = \mathtt{randn}(7, 1)$ and $\mathtt{b} = \mathtt{A} * \mathtt{w}$. Then $b \approx p(X)$ where p is the polynomial $p(t) = w(1) + w(2)t + \cdots + w(7)t^6$. This ensures that $\theta \approx 0$ for the LSP $Ax = b$. Solve this problem via Normal Equations method and QR method via reflectors (this is the default procedure so that you just have to type $\mathtt{A}\backslash\mathtt{b}$ for this!) and denote the solutions as `xhat` and `xtilde`, respectively. Examine the relative errors $\frac{\|\mathtt{xhat} - \mathtt{w}\|_2}{\|\mathtt{w}\|_2}$, and $\frac{\|\mathtt{xtilde} - \mathtt{w}\|_2}{\|\mathtt{w}\|_2}$ in the solutions as well as those in the fits $\frac{\|\mathtt{rhat}\|_2}{\|\mathtt{b}\|_2}$ and $\frac{\|\mathtt{rtilde}\|_2}{\|\mathtt{b}\|_2}$ where $\mathtt{rhat} := \mathtt{b} - \mathtt{A} * \mathtt{xhat}$ and $\mathtt{rtilde} := \mathtt{b} - \mathtt{A} * \mathtt{xtilde}$. Which method fares better? Also find the condition number of A.

   Repeat the above process for 10 Vandermonde matrices corresponding to polynomials of degrees 6 to 15. Store the relative errors in the solutions corresponding to each method in separate arrays and plot them on the same graph in $\log 10$ scale on the $y - axis$ for a comparative analysis. Do the same also for the relative errors in the fits (measured relative to b as above). Also store the condition numbers of the Vandermonde matrices at each step in a single array.

   What do you observe about the performance of the two methods as the polynomials increase in degree? Which is more sensitive to ill conditioning, the solutions or the fits?

The numerical rank of $A \in \mathbb{R}^{n\times m}$ is obtained in MATLAB by typing $\mathtt{rank}(\mathtt{A})$. MATLAB uses the `SVD` of A to obtain this value. More specifically, it is obtained by calculating a tolerance level $tol = \epsilon \max\{n, m\}\|A\|_2$ where ϵ stands for machine epsilon and then setting $\mathtt{rank}(\mathtt{A})$ to be the number of singular values of A which are greater than this value of tol. It is possible for the user to change this tol value to something else. Type `help rank` for details. The purpose of the next exercise is to illustrate that the above method computes the rank of a matrix quite efficiently in the presence of rounding.

2. Type $\mathtt{A} = \mathtt{randn}(7, 4)$ and examine its rank by typing $\mathtt{rank}(\mathtt{A})$. Since random matrices are usually full rank, its rank will be 4 in all probability. Add two more columns to A by typing $\mathtt{A}(:, 5, 6) = \mathtt{A}(: 2, 3) + \mathtt{A}(:, 3, 4)$; This will create a fifth column in A which is the sum of the 2nd and 3rd columns and a sixth column which is the sum of the 3rd and 4th columns. Theoretically, the rank of A should remain unchanged. What happens to the numerical rank? Type $\mathtt{rank}(\mathtt{A})$ to find out.

The purpose of the next exercise is to illustrate that in the presence of rounding, the `SVD` is generally more efficient in determining the rank of a matrix than the rank revealing `QR` factorization.

3. The *Kahan matrix* $R_n(\theta)$ is an $n \times n$ upper triangular matrix depending on a parameter θ. Let $c = \cos(\theta)$ and $s = \sin(\theta)$. Then

$$R_n(\theta) := \begin{pmatrix} 1 & & & & \\ & s & & & \\ & & s^2 & & \\ & & & \ddots & \\ & & & & s^{n-1} \end{pmatrix} \begin{pmatrix} 1 & -c & -c & \dots & -c \\ & 1 & -c & \dots & -c \\ & & 1 & & -c \\ & & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}.$$

If θ and n are chosen so that s is close to 1 and n is modestly large, then none of the main diagonal entries are extremely small. It appears that the matrix is far from rank deficient, which is actually not the case. Consider $R_n(\theta)$ when $n = 90$ and $\theta = 1.2$ radians. Verify for yourself that the largest main diagonal entry of $R_n(\theta)$ is 1 and the smallest is .001.

(a) To generate $R_n(\theta)$ in MATLAB and find its singular values type

$$\texttt{A} = \texttt{gallery}(\texttt{'kahan'}, 90, 1.2, 0); \\ \texttt{sig} = \texttt{svd}(\texttt{A})$$

Type `format short e` and examine σ_1, σ_{89} and σ_{90}. Type `rank(A)` to get MATLAB's opinion of the numerical rank of A.

(b) Type `A = gallery('kahan', 90, 1.2, 25)` to get a slightly perturbed version of the Kahan matrix. (This produces the Kahan matrix with very small perturbations to the diagonal entries. Type `help private/kahan` for more details.) Repeat part (a) for the perturbed matrix. Perform a QR decomposition by column pivoting on A by typing `[Q, R, E] = qr(A)`. Verify that no pivoting was done in this case by examining the value of `dif = norm(eye(90) - E)`. Examine $R(90, 90)$ and infer that the rank revealing QR decomposition failed to detect the numerical rank deficiency of A.

*** End ***