COL 726

Homework III

Due on March 12, 2018

1. Let R be an upper triangular matrix $n \times n$, and u, v be two column vectors of length n. Show how to compute the QR factorisation of the matrix $R + uv^T$ in $O(n^2)$ time.

2. In this exercise, you will prove stability of QR factorisation. You can hide factors depending on m and n in the stability calculations. Let ε denote the machine precision.

   - Let $x = (x_1, \ldots, x_m)$ be a vector and consider the Householder transformation corresponding to it, i.e., let v be the unit vector along $x + \texttt{sign}(x_1)||x||e_1$. Suppose we compute a unit vector $\tilde{v}$ instead. Show that $||\tilde{v} - v|| = O(\varepsilon)$.
   - Let $x, v\tilde{v}$ be as above, and let y be a vector of length m. We would like to compute $b = y - 2 < v, y > v$, but we use $\tilde{v}$ instead of v here, and end up computing a vector $\tilde{b}$ instead. Prove that $\tilde{b} = \tilde{y} - 2 < v, \tilde{y} >$, for some $\tilde{y}$ such that $||\tilde{y} - y|| = O(\varepsilon) \cdot ||y||$.
   - Suppose our algorithm computes QR factorization of an $m \times n$ matrix A as $\tilde{Q}, \tilde{R}$, where $\tilde{Q}$ is implicitly represented by the vectors corresponding to the Householder transformation. Then $\tilde{Q} \cdot \tilde{R} = A + \Delta A$, where $||\Delta A||/||A||$ is $O(\varepsilon)$.

3. (**Heath**) To demonstrate how results from the normal equations method and QR factorization can differ numerically, we need a least squares problem that is ill-conditioned and also has a small residual. We can generate such a problem as follows. We will fit a polynomial of degree $n - 1$,

   $$p_{n-1}(t) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1},$$

   to m data points (t_i, y_i), $m > n$. We choose $t_i = \frac{i-1}{m-1}$, $i = 1, \ldots, m$, so that the data points are evenly spaced on the interval $[0, 1]$. We will generate the corresponding values y_i by first choosing values for the x_j, say $x_j = 1$, $j = 1, \ldots, n$, and evaluating the polynomial to obtain $y_i = p_{n-1}(t), i = 1, \ldots, m$. We could now see whether we can recover the x_j that we used to generate the y_i, but to make it more interesting, we first randomly perturb the y_i values to simulate the data error. Specifically, we take $y_i = y_i + (2u_i - 1) * \epsilon$, $i = 1, \ldots, m$, where each u_i is a random number uniformly generated from $[0, 1]$, and ϵ is a small positive number. For double precision, use $m = 21, n = 12, \epsilon = 10^{-10}$.

   Having generated the data set, we will now compare the two methods for least squares. First, form the system of normal equations, and solve it using Cholesky factorization (you can use MATLAB built-in function). Next solve the least squares using QR factorization implemented in Question 2. Compare the two resulting solutions. For which method is the solution more sensitive to the perturbation we introduced in the data ? Which method comes closer to recovering the x that we used to generate the data ?

4. You are given a set of lines, $y = a_i x + b_i, i = 1, \ldots 5$ in the (x, y) plane as below :

a_i	2.3	4.5	-1.3	-2.1	10.5
b_i	-0.6	-5.7	7	10	-24

Find the point whose sum of squares of distances to these lines is smallest. Plot these lines, and the point.

5. (**Heath**) The concentration of a drug in the bloodstream is expected to diminish exponentially with time. We will fit the model function

$$y = f(t, x_1, x_2) = x_1 e^{x_2 t},$$

to the following data

t	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
y	6.80	3.00	1.50	0.75	0.48	0.25	0.20	0.15

   (a) Perform the exponential fit using non-linear least squares. You should use the Gauss-Newton method.

   (b) Taking the logarithm of f gives $\log(x_1) + x_2 t$, which is now linear in x_2. Thus, an exponential fit can also be done using linear least squares, assuming that we also take logarithms of the data points y_i. Use linear least squares to compute x_1 and x_2 in this manner. Do the values obtained agree with those determined in the previous part ? Why ?

6. Experiment with solving 60×60 systems of equations $Ax = b$ by Gaussian elimination with partial pivoting, with A having the form

$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Do you observe that the results have high error because of the growth factor of the order of 2^{60} ? At first attempt, you may not observe this, because the integer entries of A may prevent any rounding errors from occuring. If so, find a way to modify the problem slightly so that the growth factor is the same or nearly so and catastrophic rounding errors really do take place.