Lecture 6, Sci. Comp. for DPhil Students

Nick Trefethen, Thursday 31.10.19

Today

- II.3 QR factorization
- II.4 Computation of the QR factorization
- II.5 Linear least-squares

Handouts

- Quiz 4
- Householder's 4-page paper from 1958 on QR factorization
- m09_colorquiz.m, m10_QRfact.m, m11_leastsquares3.m

Announcements

- Assignment 2 is due on Tuesday.
- Read: Trefethen & Bau, chapters 7 & 11
- Please do Quiz 4

Just for fun, let's look at a game I had fun writing in MATLAB: `m09_colorquiz.m`

In MATLAB, colour images are processed beautifully via linear algebra. There's an Image Processing Toolbox.

II.3 QR factorisation

Recall from Tuesday: we often work with a matrix Q with ON columns. In the rectangular case there is no name for such matrices. Note that if Q is $m \times n$ with $m > n$ then

$$Q^T Q = I_n \quad \text{but} \quad QQ^T \neq I_m$$

(illustrate with figures).

Let A be an matrix with n columns. A (reduced) **QR factorization** of A is a factorisation

```
|   |   |    |   |       |   |   |    |   |    |r    r     .. r  |
|   |   |    |   |       |   |   |    |   |    | 11   12       1n|
|   |   |    |   |       |   |   |    |   |    |                 |
|   |   |    |   |       |   |   |    |   |    |     r           |
|   |   |    |   |       |   |   |    |   |    |      22       : |
|   |   |    |   |       |   |   |    |   |    |          .      |
|   |   |    |   |       |   |   |    |   |    |            .    |
|a  |a  |...|a  |   =   |q  |q  |...|q  |    |              r  |
| 1| 2|    | n|       | 1| 2|    | n|    |               nn|
```

```
    |   |   |    |   |         |   |   |    |   |
    |   |   |    |   |         |   |   |    |   |
    |   |   |    |   |         |   |   |    |   |
    |   |   |    |   |         |   |   |    |   |
    |   |   |    |   |         |   |   |    |   |
    |   |   |    |   |         |   |   |    |   |


              A           =           Q                         R
```

where the columns of Q are ON. We say that R is **upper-triangular**. In MATLAB, `[Q,R] = qr(A,0)`. (We'll talk about algorithms shortly.)

Note that we have

$$a_1 = r_{11} q_1,$$

$$a_2 = r_{12} q_1 + r_{22} q_2,$$

$$a_3 = r_{13} q_1 + r_{23} q_2 + r_{33} q_3,$$

$$\vdots$$

Thus the QR factorisation exhibits an expansion of the successive columns of A in the ON columns of Q. In notation of Lecture 2,

$$\text{span}\{a_1, \ldots, a_k\} = \text{span}\{q_1, \ldots, q_k\}, \quad 1 \le k \le n$$

in the case where A has full rank. (In the rank-deficient case you get $\subseteq$, and some zero entries on the main diagonal of R.)

Alternatively we could multiply by Q^T on the left to obtain

```
-----------------------------  |  |  |   |  |       |r   r     .. r  |
              T                |  |  |   |  |       | 11  12      1n|
             q                 |  |  |   |  |       |               |
              1                |  |  |   |  |       |    r          |
-----------------------------  |  |  |   |  |       |     22      : |
             :                 |  |  |   |  |       |         .     |
-----------------------------  |  |  |   |  |       |           .   |
              T                |a |a |...|a |   =   |             r |
             q                 | 1| 2|   | n|       |              nn|
              n                |  |  |   |  |
-----------------------------  |  |  |   |  |
                               |  |  |   |  |
                               |  |  |   |  |
                               |  |  |   |  |
                               |  |  |   |  |
              T
             Q                      A         =               R
```

Thus $r_{ij} = q_i^T a_j$.

Suppose x is an n-vector. Then

$$b = Ax$$

is a linear combination of the columns of A. If $A = QR$, then

$$b = QRx = Q(Rx)$$

or by multiplying on the left by Q^T, $Rx = Q^T b$.

Thus multiplying on the left by R takes us from a vector of coefficients for an expansion in columns of A to a vector of coefficients for an expansion in columns of Q. We talked about such things last lecture in the square case.

II.4 Computation of the QR factorisation

Given A, there are two approaches to computing Q and R such that $A = QR$.

(1) Gram-Schmidt orthogonalisation: triangular orthogonalisation

Construct vectors $q_1, q_2, \ldots$ in succession.

This amounts to multiplying A on the right by successive upper-triangular matrices until it has orthonormal columns:

$$AR_1R_2R_3 \cdots = Q$$

"triangular orthogonalisation": aim for Q and get R as a by-product.

Gram-Schmidt is a classical idea of great importance. On the other hand, as a linear algebra algorithm it is often not as good as the more accurate alternative:

(2) Householder triangularisation: orthogonal triangularisation

`[Householder 1958 paper]`

Construct columns $r_1, r_2, \ldots$ of R in succession.

This amounts to multiplying A on the left by successive orthogonal matrices until it has upper-triangular structure:

$$\cdots Q_3Q_2Q_1A = R \qquad \text{expand on this a bit}$$

"orthogonal triangularisation": aim for R and get Q as a by-product.

With rounding errors, this algorithm produces matrices Q whose columns are much more nearly orthogonal than with Gram-Schmidt.

For a one-off application like least-squares, this often does not matter.

It sometimes matters a good deal, however, for applications involving repeated calculation of QR factorisations, such as computing eigenvalues or singular values.

`[m10_QRfact.m]`

II.5 Linear least-squares

Let's return to the least-squares problem These ideas were invented independently by Gauss and Legendre before 1800, leading to a big fight.

An expert in this subject is *Åke Björck* – see his book in our list.

Suppose A is $m \times n$ with $m > n$. A **least-squares solution** to $Ax = b$ is a vector x with minimal residual norm, i.e.,

$$\|Ax - b\| = \text{minimum} \qquad (*)$$

```
          b
          *
              .             /
                  .        /
                      .   /
                         * Ax
                        /
                       /
                      /
                     /                  m
                    /                  R
-------------------/----------------------
                  /
                 /  0
                /
               /
              /  range(A)
```

It's clear geometrically, and easily proved, that $(*)$ is equivalent to the condition that the **residual** $b - Ax$ is orthogonal to range(A).

That is, for each column a_j of A, $a_j^T(b - Ax) = 0$. Equivalently,

$$A^T(b - Ax) = 0, \qquad \text{i.e.,} \qquad A^TAx = A^Tb.$$

These are the **normal equations**.

Theorem. rank$(A) = n$ $\Leftrightarrow$ *A^TA is SPD* $\Leftrightarrow$ *solution to $(*)$ is unique*

The normal equations are invaluable theoretically. They are also useful computationally. However, there is a more accurate "standard" method for least-squares problems of moderate size:

Least-squares via QR factorisation

Suppose $A = QR$ and we want to solve $Ax \approx b$.

```
          b
          *
              .             /
```

```
                  .        /
                      .   /
                         * Ax
                        /
                       /
                      /
                     /                   m
                    /                   R
-------------------/----------------------
                  /
                 /  0
                /
               /
              /  range(A)
```

Now Ax is the orthogonal projection of b onto range(A). Thus it can be written

```
           T
   Ax = Q Q b .
       /   \
orthogonal  expansion
  basis       coeffs
```

(QQ^T is called a **projection matrix**.)

Multiplying on the left by Q^T gives $Q^T QRx = Q^T b$, i.e.,

$$Rx = Q^T b.$$

You can solve this system to get the least-squares solution vector x.

It's a triangular system – easily solved by back-substitution successively for $x_n, x_{n-1}, \ldots, x_1$.

[m11_leastsquares3.m]