Chapter 7: QR and Cholesky Factorizations

FuSen F. Lin

Department of Computer Science and Engineering
National Taiwan Ocean University

Scientific Computing, Fall 2010

The QR and Cholesky Factorizations

- 7.0 Introduction
- 7.1 Least Squares Fitting
- 7.2 The QR Factorization
- 7.3 (*) The Cholesky Factorization
- 7.4 (*) High-Performance Cholesky

Introduction (1)

- The overdetermined linear systems $A\mathbf{x} = \mathbf{b}$, which have more equations than unknowns, arises in many applications of science and engineering. The solution of overdetermined systems of linear equations is central to computational science.
- Solving this problem, we must lower our aim and be content to make $A\mathbf{x}$ close to $\mathbf{b}$. Least squares fitting results when 2-norm of $A\mathbf{x}$ is used to quantify success.

Introduction (2)

- In Sec.7.1, we introduce the least squares problem and solve a simple fitting problem using built-in MATLAB features.
- In Sec.7.2, we present the QR factorization and show how it can be used to solve the least squares problem.
- The solution of linear systems with symmetric positive definite coefficient matrix is discussed in Sec.7.3 and a special version of the LU factorization called Cholesky factorization is introduced.
- In the last section we look at two Cholesky implementations that have appeal in advinced computing environments.

Least Squares Fitting (1)

- A square nonsingular system $A\mathbf{x} = \mathbf{b}$ has a unique solution, since it has the same number of unknowns as equations. However, if we have more equations than unknowns, then it may not have $\mathbf{x}$ to satisfy $A\mathbf{x} = \mathbf{b}$.
- For the overdetermined $A\mathbf{x} = \mathbf{b}$ problem to have a solution, it is necessary for $\mathbf{b}$ to be in the span (space) of A's columns.
- For example, the 3-by-2 case,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

  has no solution.
- So with more equations than unknowns, We need to adjust our aims. instead of trying to "reach" $\mathbf{b}$ with $A\mathbf{x}$, we try to get as close as possible. Vector norms can be used to quantify the degree of success.

Least Squares Fitting (2)

- The least squares (LS) problem becomes

  Given $A \in R^{m \times n}$ and $\mathbf{b} \in R^m$, find $\mathbf{x} \in R^n$ **to minimize** $||A\mathbf{x} - \mathbf{b}||_2$.

- The apt terminology coming from that the 2-norm involves a sum of squares:

$$||A\mathbf{x} - \mathbf{b}||_2 = \sqrt{\sum_{i=1}^{m} (A(i,:)x - b(i))^2}$$

- The goal is to minimize the discrepancies in each equation:

$$(A(i,:)x - b(i))^2 = (a_{i1}x_1 + \cdots + a_{in}x_n - b_i)^2$$

- For the column point of view, the goal is to find a linear combination of A's columns that gets as close as possible in the 2-norm sense.

Setting Up Least Squares Problems (1)

- Least squares fitting problems often arise when a scientist attempts to fit a model experimentally obtained data.
- For example, suppose a biologist conjectures that plant height h is a function of 4 soil nutrient concentrations a_1, a_2, a_3, and a_4:

$$h = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4.$$

  This is a *linear model*, and x_1, x_2, x_3, and x_4 are *model parameters* whose value must be determined.
- To this end, the biologist performs m (a large number) experiments. The i experiment consists of establishing the 4 nutrient values a_{i1}, a_{i2}, a_{i3}, and a_{i4} in the soil, planting the seed, and observing the resulting height h_i. If the model is perfect, then for $i = 1 : m$ we have

$$h_i = a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + a_{i4}x_4.$$

Setting Up Least Squares Problems (2)

- That is, (see page 242)

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & a_{m4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ \vdots \\ h_m \end{bmatrix}$$

- Of course, the model will not be perfect, making it impossible to find such an **x**. The aims of the biologist are lowered and the minimizer (some **x**) of $||A\mathbf{x} - h||_2$ is sought.
- LS fitting also arises in the approximation of known functions. Suppose the designer of a built-in squre root function needs to develop an approximation to the function $f(x) = \sqrt{x}$ on the interval $[0.25, 1]$. A linear approximation of the form $g(x) = \alpha + \beta x$ is sought. (Think of $g(x)$ as a two-parameter model with parameters α and β.)

Setting Up Least Squares Problems (3)

- We could set this function to be just the linear interpolant of f at two well-chosen points. Alternatively, if a partition

$$0.25 = x_1 < x_2 < \cdots < x_m = 1$$

is given, then the parameters α and β can be chosen so that he quantity

$$\phi_m(\alpha, \beta) = \sum_{i=1}^{m} [(\alpha + \beta x_i) - \sqrt{x_i}]^2$$

is minimized.

- Note that if we set $f_i = \sqrt{x_i}$, then in the language of matrices, vectors, and norms we have (see page 242) the m-by-2 least squares problem needs to be solved in order to resolve α and β.

MATLAB's Least Squares Tools (1)

- The *backslash operator* can be use to solve the least squares problem in MATLAB, once it is cast in the matrix (vector) terms, i.e., $\min ||Ax - b||_2$. See the m-file **ShowLSFit**.
- In general, if we try to fit data points $(x_1, f_1), (x_2 f_2), \ldots, (x_n, f_n)$ in the least squares sense with a polynomial of degree d, then an m-by-$(d+1)$ least squares problem arises.
- The MATLAB function **polyfit** can be use to solve this problem and **polyval** can be used to evaluate the approximant.
- The script

$$\begin{aligned} c &= \text{polyfit}(x, y, d); \\ xvals &= \text{linspace}(\min(x), \max(x)); \\ yvals &= \text{polyval}(c, xvals); \\ & \quad \text{plot}(xvals, yvals, x, y, 'o'); \end{aligned}$$

  plots the approximating polynomial and the data. The generation of the polynomial coefficients in c involves $O(md^2)$ flops.

QR Factorization (1)

- In Linear Algebra, we learned that the how to producing a factorization $A = QR$, where Q is unitary and R is upper triangular, by Gram-Schmidt process.
- However, numerically, the algorithm of QR-factorization is an iterative procedure designed to find the eigenvalues of A effectively. the QR-algorithm based on the rotation matrix and relatived to the least squares problem.
- Suppose that the given linear system $A\mathbf{x} = \mathbf{b}$ is equivalent to an easy-to-solve system $(MA)\mathbf{x} = M\mathbf{b}$. This problem is similar to that seeking a comparable strategy in the least squares setting.

QR Factorization (2)

- Our goal is to produce an m-by-m matrix Q so the given least squares problem
$$\min_{x \in R^n} ||A\mathbf{x} - \mathbf{b}||_2$$
is equivalent to a transformed problem
$$\min_{x \in R^n} ||(Q^T A)\mathbf{x} - (Q^T \mathbf{b})||_2$$
where, by design, $Q^T A$ is "simple".
- A family of matrices known as orthogonal matrices can be used for this purpose. A matrix $Q \in R^{m \times m}$ is orthogonal if $Q^T = Q^{-1}$, or equivalently, if $QQ^T = Q^T Q = I$. Here is a 2-by-2 example:
$$Q = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$
For this particular Q, it is easy to show that $Q\mathbf{x}$ is obtained by rotating clockwise by θ radians.

QR Factorization (3)

- The key property of orthogonal matrices that make them useful in the least squares context is that they preserve 2-norm. Indeed, if $Q \in R^{m\times m}$ is orthogonal and $r \in R^m$, then

$$||Q^T r||_2^2 = (Q^T r)^T (Q^T r) = (r^T Q)(Q^T r) = r^T (QQ^T) r = r^T I_m r = r^T r = ||r||_2^2$$

- If Q is orthogonal, then any $\mathbf{x}$ minimizes $||A\mathbf{x} - \mathbf{b}||_2$ also minimizes $||(Q^T A)\mathbf{x} - (Q^T\mathbf{b})||_2$ since

$$||(Q^T A)\mathbf{x} - (Q^T\mathbf{b})||_2 = ||Q^T (A\mathbf{x} - \mathbf{b})||_2 = ||A\mathbf{x} - \mathbf{b}||_2$$

- Our plan is to apply a sequence of orthogonal transformations to A that reduce it to upper triangular form, so that the linear system becomes an easy-to-solve problem. For example,... (see page 248).

QR Factorization (4)

- The columns of an orthogonal matrix define an *orthonormal basis*. This means that the column of Q are mutually orthogonal and $Q^TQ = I$. That is,

$$\begin{cases} Q(:,i)^T Q(:,j) = 1, & \forall i = j; \\ Q(:,i)^T Q(:,j) = 0, & \forall i \neq j. \end{cases}$$

- Finding an orthogonal $Q \in R^{m\times m}$ and an upper triangular $R \in R^{m\times n}$ so that $A = QR$ is the factorization problem. It amounts to finding an orthogonal basis for subspace defined by the columns of A. The j-th column of the equation $A = QR$ says that

$$A(:,j) = Q(:,1) * R(1,j) + Q(:,2) * R(2,j) + \cdots + Q(:,j) * R(j,j).$$

That maens that the j-th column of A is the linear combination of vectors $\{Q(:,1), Q(:,2), \ldots, Q(:,j)\}$ with coefficients $R(1,j), R(2,j), \ldots, R(j,j)$. For example,... (see page 248).

Rotations (1)

- The Q in the QR factorization can be computed by using a special family of orthogonal matrices that are called *rotations*. We have seen that the 2-by-2 rotation matrix

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} := G$$

- If $\mathbf{x} = [x_1, x_2]^T$, it is possible to choose (c, s) so that if $y = G\mathbf{x}$, then $y_2 = 0$. Indeed, from $y_2 = -sx_1 + cx_2 = 0$, it is easy to obtain that

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}} \quad \text{and} \quad s = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}$$

  However, a preferred algorithm for computing the (c, s) pairs is **Rotate** MATLAB function (see page 249).

Rotations (2)

- The rotation matrix G can be extended to higher dimensions. Suppose $m = 4$ and define

$$G(1,3,\theta) = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  This is a rotation in the $(1,3)$ plane. It is easy to check that $G(1,3,\theta)$ is orthogonal.

- For general m, rotations in the (i,k) plane is defined as $G(i,k,\theta)$ (see page 250). This may be organized as

$$A([i,\ k],:) = [c\, s; -s\, c] * A([i,\ k],:)$$

  The integer vector $[i,\ k]$ is used to extract rows i and k from matrix A.

Rotations (3)

- A sequence of row rotations $G_1, G_2, \ldots, G_t$ can make a retangular matrix become an upper trianglar R (see page 250). That means that
$$G_t G_{t-1} \cdots G_1 = R$$
- If we define
$$Q^T = G_t G_{t-1} \cdots G_1$$
then $Q^T A = R$ and mutiplying Q on both sides, we obtain
$$A = QR$$
The MATLAB function is **QRRot** (see page 251).
- Once we have the QR factorization of A, then the given least squares problem of minimizing $||A\mathbf{x} - \mathbf{b}||_2$ transforms as follows:
$$||A\mathbf{x} - \mathbf{b}||_2 = ||Q^T(A\mathbf{x} - \mathbf{b})||_2 = ||R\mathbf{x} - c||_2$$
- The algorithm for computing the least squares problem is the **LSq** function (see page 252).