Linear equations

A system of n linear equations with m unknowns is generally written in the form

$$\sum_{j=1}^{m} A_{ij}x_j = b_i \ , \ i = 1, \dots, n \ , \tag{1}$$

where $x_1, x_2, \dots, x_m$ are the unknown variables, $A_{11}, A_{12}, \dots, A_{nm}$ are the (constant) coefficients of the system, and $b_1, b_2, \dots, b_n$ are the (constant) right-hand side terms.

The system can be written in matrix form as

$$A\mathbf{x} = \mathbf{b} \ . \tag{2}$$

where $A \doteq \{A_{ij}\}$ is the $n \times m$ matrix of the coefficients, $\mathbf{x} \doteq \{x_j\}$ is the size-n column-vector of the unknown variables, and $\mathbf{b} \doteq \{b_i\}$ is the size-m column-vector of right-hand side terms.

Systems of linear equations occur regularly in applied mathematics. The computational algorithms for finding solutions of linear systems are therefore an important part of numerical methods.

A system of non-linear equations can often be approximated by a linear system, a helpful technique—called *linearization*—in creating a mathematical model of an otherwise a more complex system.

If $m = n$, the matrix A is called *square*. A square system has a unique solution if A is invertible.

Triangular systems and back-substitution

An efficient algorithm to solve numerically a square system of linear equations is to transform the original system into an equivalent *triangular system*,

$$T\mathbf{y} = \mathbf{c} \ , \tag{3}$$

where T is a *triangular matrix*: a special kind of square matrix where the matrix elements either below or above the main diagonal are zero.

An upper triangular system can be readily solved by *back substitution*:

$$y_i = \frac{1}{T_{ii}} \left(c_i - \sum_{k=i+1}^{n} T_{ik} y_k \right) , \ i = n, n-1, \dots, 1 \ . \tag{4}$$

For the lower triangular system the equivalent procedure is called *forward substitution*.

Note that a diagonal matrix—that is, a square matrix in which the elements outside the main diagonal are all zero—is also a triangular matrix.

Reduction to triangular form

Popular algorithms for transforming a square system to triangular form are *LU decomposition* and *QR decomposition*.

LU decomposition

LU decomposition is a factorization of a square matrix into a product of a lower triangular matrix L and an upper triangular matrix U,

$$A = LU \ . \tag{5}$$

The linear system $A\mathbf{x} = \mathbf{b}$ after LU-decomposition of the matrix A becomes $LU\mathbf{x} = \mathbf{b}$ and can be solved by first solving $L\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$ and then $U\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$ with two runs of forward and backward substitutions.

If A is a $n \times n$ matrix, the condition (5) is a set of n^2 equations,

$$\sum_{k=1}^{n} L_{ik} U_{kj} = A_{ij} \ , \tag{6}$$

for n^2+n unknown elements of the triangular matrices L and U. The decomposition is thus not unique.

Usually the decomposition is made unique by providing extra n conditions e.g. by the requirement that the elements of the main diagonal of the matrix L are equal one,

$$L_{ii} = 1\,,\ i = 1\dots n\,. \tag{7}$$

The system (6) can then be easily solved row after row using e.g. the *Doolittle algorithm*,

for $i = 1$ **to** n **:**
  $L_{ii} = 1$
  for $j = 1$ **to** $i-1$ **:**
    $L_{ij} = \left(A_{ij} - \sum_{k<j} L_{ik}U_{kj}\right)/U_{jj}$
  for $j = i$ **to** n **:**
    $U_{ij} = A_{ij} - \sum_{k<i} L_{ik}U_{kj}$.

QR decomposition

QR decomposition is a factorization of a matrix into a product of an orthogonal matrix Q, such that $Q^TQ = 1$ (where ^{T} denotes transposition), and a right triangular matrix R,

$$A = QR\,. \tag{8}$$

QR-decomposition can be used to convert the linear system $A\mathbf{x} = \mathbf{b}$ into the triangular form

$$R\mathbf{x} = Q^T\mathbf{b}, \tag{9}$$

which can be solved directly by back-substitution.

QR-decomposition can also be performed on non-square matrices with few long columns. Generally speaking a rectangular $n \times m$ matrix A can be represented as a product, $A = QR$, of an orthogonal $n \times m$ matrix Q, $Q^TQ = 1$, and a right-triangular $m \times m$ matrix R.

QR decomposition of a matrix can be computed using several methods, such as Gram-Schmidt orthogonalization, Householder transformations, or Givens rotations.

Gram-Schmidt orthogonalization *Gram-Schmidt orthogonalization* is an algorithm for orthogonalization of a set of vectors in a given inner product space. It takes a linearly independent set of (column-)vectors $A = \{\mathbf{a}_1, \dots, \mathbf{a}_m\}$ and generates an orthogonal set $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_m\}$ which spans the same subspace as A. The algorithm is given as

for $i = 1$ **to** m **:**
  $\mathbf{q}_i \leftarrow \mathbf{a}_i/\|\mathbf{a}_i\|$ **//normalization**
  for $j = i+1$ **to** m **:**
    $\mathbf{a}_j \leftarrow \mathbf{a}_j - \langle \mathbf{a}_j, \mathbf{q}_i\rangle \mathbf{q}_i$ **//orthogonalization .**

where $\langle \mathbf{a}, \mathbf{b}\rangle$ is the inner product of two vectors, and $\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a}\rangle}$ is the vector's norm. This variant of the algorithm, where all remaining vectors $\mathbf{a}_j$ are made orthogonal to $\mathbf{q}_i$ as soon as the latter is calculated, is considered to be numerically stable and is referred to as *stabilized* or *modified*.

Stabilized Gram-Schmidt orthogonalization can be used to compute QR decomposition of a matrix A by orthogonalization of its column-vectors $\mathbf{a}_i$ with the inner product

$$\langle \mathbf{a}, \mathbf{b}\rangle = \mathbf{a}^T\mathbf{b} \equiv \sum_{k=1}^{n} (\mathbf{a})_k(\mathbf{b})_k\ , \tag{10}$$

where n is the length of column-vectors $\mathbf{a}$ and $\mathbf{b}$, and $(\mathbf{a})_k$ is the kth element of the column-vector,

for $i = 1\dots m$ **:**
  $R_{ii} = (\mathbf{a}_i^T\mathbf{a}_i)^{1/2}$ **;** $\mathbf{q}_i = \mathbf{a}_i/R_{ii}$
  for $j = i+1\dots m$ **:**
    $R_{ij} = \mathbf{q}_i^T\mathbf{a}_j$ **;** $\mathbf{a}_j = \mathbf{a}_j - \mathbf{q}_iR_{ij}$.

The factorization is unique under requirement that the diagonal elements of R are positive. For a $n \times m$ matrix the complexity of the algorithm is $O(m^2n)$.

Table 1: QR decomposition in C++ using Armadillo matrices

```
#include<armadillo>
using namespace arma;
void qrdec(mat& A, mat& R)// QR-decomposition of matrix A (A is replaced with Q)
{
for(size_t i = 0; i < A.n_cols; i++){
   double r = dot( A.col(i), A.col(i) );
   R(i,i) = sqrt(r);
   A.col(i) /= sqrt(r); //normalization
   for(size_t j=i+1; j < A.n_cols; j++){
      double s = dot( A.col(i), A.col(j) );
      A.col(j) -= s*A.col(i); //orthogonalization
      R(i,j) = s;
      }
   }
}
```

Householder transformation An $n \times n$ matrix H of the form

$$H = 1 - \frac{2}{u^T u} u u^T \tag{11}$$

is called *Householder matrix* where the vector u is called a *Householder vector*. Householder matrices are symmetric and orthogonal,

$$H^T = H \;,\; H^T H = 1 \;. \tag{12}$$

The transformation induced by the Householder matrix on a given vector a,

$$a \to Ha \;, \tag{13}$$

is called a Householder reflection. The transformation changes the sign of the affected vector's component in the u direction, or, in other words, makes a reflection of the vector about the hyperplane perpendicular to u, hence the name.

Householder transformation can be used to zero selected components of a given vector a. For example one can zero all components but the first one, such that

$$Ha = \gamma e^1 \;, \tag{14}$$

where γ is a number and e^1 is the unit vector in the first direction. The factor γ can be easily calculated,

$$||a||^2 \doteq a^T a = a^T H^T H a = (\gamma e^1)^T (\gamma e^1) = \gamma^2 \;, \tag{15}$$

$$\Rightarrow \gamma = \pm ||a|| \;. \tag{16}$$

To find the Householder vector, we notice that

$$a = H^T H a = H^T \gamma e^1 = \gamma e^1 - \frac{2u_1}{u^T u} u \;, \tag{17}$$

$$\Rightarrow \frac{2u_1}{u^T u} u = \gamma e^1 - a \;, \tag{18}$$

where u_1 is the first component of the vector u. One usually chooses $u_1 = 1$ for the sake of the possibility to store the other components of the Householder vector in the zeroed elements of the vector a; and stores the factor

$$\frac{2}{u^T u} \equiv \tau \tag{19}$$

separately. With this convention one readily finds τ from the first component of equation (18),

$$\tau = \gamma - a_1 \;. \tag{20}$$

where a_1 is the first element of the vector a. For the sake of numerical stability the sign of γ has to be chosen opposite to the sign of a_1,

$$\gamma = -\text{sign}(a_1) ||a|| \;. \tag{21}$$

Finally, the Householder reflection which zeroes all component of a vector a but the first,

$$H = 1 - \tau u u^T \ , \ \tau = -\mathrm{sign}(a_1)||a|| - a_1 \ , \ u_1 = 1 \ , \ u_{i>1} = -\frac{1}{\tau} a_i \ . \tag{22}$$

A typical strategy to perform a QR-decomposition of a matrix A by Hoseholder transformations is as folowing:

1. Build the Householder vector u from equation (22) (the reflection with which zeroes the subdiagonal components of the first column of matrix A);
2. Apply this Householder reflection to all columns of matrix A;
3. Store the elements of u in the zeroed elements of matrix A and store τ in a separate array for keeping taus;
4. Apply the algorithm recursively to the matrix $A_{2\dots n,2\dots m}$ (that is the matrix A wihout the first column and the first row).

One typically does not explicitely builds the Q matrix but rather applies (in an effective way avoiding matrix-matrix operations) the successive Householder reflections stored during the decomposition.

Determinant of a matrix

LU- and QR-decompositions allow $O(n^3)$ calculation of the determinant of a square matrix. Indeed, for the LU-decomposition,

$$\det A = \det LU = \det L \det U = \det U = \prod_{i=1}^{n} U_{ii} \ . \tag{23}$$

For the QR-decomposition

$$\det A = \det QR = \det Q \det R \ . \tag{24}$$

Since Q is an orthogonal matrix $(\det Q)^2 = 1$ and therefore

$$|\det A| = |\det R| = \left| \prod_{i=1}^{n} R_{ii} \right| \ . \tag{25}$$

Matrix inverse

The inverse A^{-1} of a square $n \times n$ matrix A can be calculated by solving n linear equations $A\mathbf{x}_i = \mathbf{z}_i$, $i = 1 \dots n$, where $\mathbf{z}_i$ is a column where all elements are equal zero except for the element number i, which is equal one. The matrix made of columns $\mathbf{x}_i$ is apparently the inverse of A.