1 Systems of linear equations

A system of linear equations of dimension n is a set of n linear algebraic equations in n unknown variables $x_1, \ldots, x_n$,

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

The system can be represented in matrix form as

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

where A is the $n \times n$ matrix of coefficients, $\mathbf{x}$ is the size-n column-vector of variables, and and $\mathbf{b}$ is a size-n column-vector of right-hand-sides.

An efficient method to solve a system of linear equations is to transform the original system into an equivalent triangular system, $T\mathbf{y} = \mathbf{c}$, where T is an (upper) triangular matrix. A triangular system can be readily solved by *back-substitution*:

$$y_i = \frac{1}{T_{ii}} \left(c_i - \sum_{k=i+1}^{n} T_{ik} y_k \right) , \ i = n, \ldots, 1 \ . \tag{3}$$

One of the popular methods to transform a system of linear equations into a triangular form is QR *decomposition*. QR decomposition can be also used for for linear least-squares fits.

1.1 QR decomposition

A rectangular $n \times m$ matrix A can be represented as a product, $A = QR$, of an orthogonal $n \times m$ matrix Q, $Q^T Q = 1$, and a right-triangular $m \times m$ matrix R.

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

$$R\mathbf{x} = Q^T \mathbf{b}, \tag{4}$$

which can be solved by back-substitution.

QR decomposition of a matrix can be computed by means of *modified Gram-Schmidt orthogonalization*.

1.1.1 Gram-Schmidt orthogonalization

Modified Gram-Schmidt orthogonalization is a method to perform QR decomposition of a matrix A by orthogonalization of its column-vectors $\mathbf{a}_i$:

`for` $i = 1, \ldots, m$:

$R_{ii} = (\mathbf{a}_i^T \mathbf{a}_i)^{1/2}$

$\mathbf{q}_i = \mathbf{a}_i / R_{ii}$

`for` $j = i+1, \ldots, m$:

$R_{ij} = \mathbf{q}_i^T \mathbf{a}_j$

$\mathbf{a}_j = \mathbf{a}_j - \mathbf{q}_i R_{ij}$

The matrix Q, made of columns $\mathbf{q}_i$, is orthogonal, matrix R is right-triangular, and $A = QR$. The factorization is unique under requirement that the diagonal elements of R are positive. For a square matrix of size n the complexity of the algorithm is $O(n^3)$.

1.2 Determinant of a matrix

QR-decomposition allows an $O(n^3)$ calculation of (the absolute value of) the determinant of a matrix. Indeed,

$$\det(A) = \det(QR) = \det(Q) \det(R) \ . \tag{5}$$

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

$$|\det(A)| = |\det(R)| \ , \tag{6}$$

where the determinant $\det(R)$ of a triangular matrix R is simply a product of its diagonal elements.

1.3 Matrix inverse

The inverse A^{-1} of a square $n \times n$ matrix A can be calculated by solving n linear equations $A\mathbf{x}_i = \mathbf{z}_i$, $i = 1, \ldots, n$, 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.

1.4 C♯ implementation

The following class implements QR-decomposition and backsubstitution. The classes `matrix` and `vector` are supposed to overload the following operators: "*" (multiplication; dot-product), "-" (subtraction), "^" (transposition and multiplication), "[]" (referencing a column vector of a matrix; referencing an element of a vector), "[,]" (referencing an element of a matrix). //

```
using matrix=vmatrix; using vector=vector;
public class QRdecomposition{
public matrix Q, R;
public QRdecomposition(matrix A){
   int m = A.ncols;
   R = new matrix(m,m);
   Q = A.copy();
   for(int i=0;i<m;i++){
      R[i,i]=System.Math.Sqrt(Q[i]^Q[i]);
      Q[i]*=(1/R[i,i]);
      for(int j=i+1;j<m;j++){
         R[i,j]=Q[i]^Q[j];
         Q[j]-=Q[i]*R[i,j]; } }
}
public vector qrback(vector b){
   int m=R.ncols;
```

```
   vector c = Q^b;
   vector x = new vector(m);
   for(int i=m-1;i>=0;i--){
      double s=0;
      for(int k=i+1;k<m;k++)
         s+=R[i,k]*x[k];
      x[i]=(c[i]-s)/R[i,i]; }
   return x;
   }
}
//
```