QR decomposition

In linear algebra, a **QR decomposition** (also called a **QR factorization**) of a matrix is a decomposition of a matrix A into a product $A = QR$ of an orthogonal matrix Q and an upper triangular matrix R. QR decomposition is often used to solve the linear least squares problem, and is the basis for a particular eigenvalue algorithm, the QR algorithm.

1 Cases and definitions

1.1 Square matrix

Any real square matrix A may be decomposed as

$$A = QR,$$

where Q is an orthogonal matrix (its columns are orthogonal unit vectors meaning $Q^{\mathrm{T}}Q = I$) and R is an upper triangular matrix (also called right triangular matrix). If A is invertible, then the factorization is unique if we require that the diagonal elements of R be positive.

If instead A is a complex square matrix, then there is a decomposition $A = QR$ where Q is a unitary matrix (so $Q^{*}Q = I$).

If A has n linearly independent columns, then the first n columns of Q form an orthonormal basis for the column space of A. More generally, the first k columns of Q form an orthonormal basis for the span of the first k columns of A for any $1 \le k \le n$.[1] The fact that any column k of A only depends on the first k columns of Q is responsible for the triangular form of R.[1]

1.2 Rectangular matrix

More generally, we can factor a complex $m \times n$ matrix A, with $m \ge n$, as the product of an $m \times m$ unitary matrix Q and an $m \times n$ upper triangular matrix R. As the bottom $(m-n)$ rows of an $m \times n$ upper triangular matrix consist entirely of zeroes, it is often useful to partition R, or both R and Q:

$$A = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1, Q_2 \end{bmatrix}\begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where R_1 is an $n \times n$ upper triangular matrix, 0 is an $(m-n) \times n$ zero matrix, Q_1 is $m \times n$, Q_2 is $m \times (m-n)$, and Q_1 and Q_2 both have orthogonal columns.

Golub & Van Loan (1996, §5.2) call $Q_1 R_1$ the *thin QR factorization* of A; Trefethen and Bau call this the *reduced QR factorization*.[1] If A is of full rank n and we require that the diagonal elements of R_1 are positive then R_1 and Q_1 are unique, but in general Q_2 is not. R_1 is then equal to the upper triangular factor of the Cholesky decomposition of $A^{*}A$ ($= A^{\mathrm{T}}A$ if A is real).

1.3 QL, RQ and LQ decompositions

Analogously, we can define QL, RQ, and LQ decompositions, with L being a *lower* triangular matrix.

2 Computing the QR decomposition

There are several methods for actually computing the QR decomposition, such as by means of the Gram–Schmidt process, Householder transformations, or Givens rotations. Each has a number of advantages and disadvantages.

2.1 Using the Gram–Schmidt process

For more details on this topic, see Gram–Schmidt § Numerical stability.

Consider the Gram–Schmidt process applied to the columns of the full column rank matrix $A = [\mathbf{a}_1, \cdots, \mathbf{a}_n]$, with inner product $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^{\top}\mathbf{w}$ (or $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^{*}\mathbf{w}$ for the complex case).

Define the projection:

$$\operatorname{proj}_{\mathbf{e}}\mathbf{a} = \frac{\langle \mathbf{e}, \mathbf{a} \rangle}{\langle \mathbf{e}, \mathbf{e} \rangle}\mathbf{e}$$

then:

$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{a}_2 - \operatorname{proj}_{\mathbf{u}_1} \mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{a}_3 - \operatorname{proj}_{\mathbf{u}_1} \mathbf{a}_3 - \operatorname{proj}_{\mathbf{u}_2} \mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}_j} \mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}
\end{aligned}
$$

We can now express the $\mathbf{a}_i$ s over our newly computed orthonormal basis:

$$
\begin{aligned}
\mathbf{a}_1 &= \langle \mathbf{e}_1, \mathbf{a}_1 \rangle \mathbf{e}_1 \\
\mathbf{a}_2 &= \langle \mathbf{e}_1, \mathbf{a}_2 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_2 \rangle \mathbf{e}_2 \\
\mathbf{a}_3 &= \langle \mathbf{e}_1, \mathbf{a}_3 \rangle \mathbf{e}_1 + \langle \mathbf{e}_2, \mathbf{a}_3 \rangle \mathbf{e}_2 + \langle \mathbf{e}_3, \mathbf{a}_3 \rangle \mathbf{e}_3 \\
&\vdots \\
\mathbf{a}_k &= \sum_{j=1}^{k} \langle \mathbf{e}_j, \mathbf{a}_k \rangle \mathbf{e}_j
\end{aligned}
$$

where $\langle \mathbf{e}_i, \mathbf{a}_i \rangle = \|\mathbf{u}_i\|$. This can be written in matrix form:

$$A = QR$$

where:

$$Q = [\mathbf{e}_1, \cdots, \mathbf{e}_n]$$

and

$$
R = \begin{pmatrix}
\langle \mathbf{e}_1, \mathbf{a}_1 \rangle & \langle \mathbf{e}_1, \mathbf{a}_2 \rangle & \langle \mathbf{e}_1, \mathbf{a}_3 \rangle & \cdots \\
0 & \langle \mathbf{e}_2, \mathbf{a}_2 \rangle & \langle \mathbf{e}_2, \mathbf{a}_3 \rangle & \cdots \\
0 & 0 & \langle \mathbf{e}_3, \mathbf{a}_3 \rangle & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{pmatrix}.
$$

2.1.1 Example

Consider the decomposition of

$$
A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.
$$

Recall that an orthonormal matrix Q has the property

$$Q^T Q = I.$$

Then, we can calculate Q by means of Gram–Schmidt as follows:

$$
U = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{pmatrix} = \begin{pmatrix} 12 & -69 & -58/5 \\ 6 & 158 & 6/5 \\ -4 & 30 & -33 \end{pmatrix};
$$

$$
Q = \begin{pmatrix} \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} & \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} & \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \end{pmatrix} = \begin{pmatrix} 6/7 & -69/175 & -58/175 \\ 3/7 & 158/175 & 6/175 \\ -2/7 & 6/35 & -33/35 \end{pmatrix}.
$$

Thus, we have

$$Q^T A = Q^T Q\, R = R;$$

$$
R = Q^T A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & 35 \end{pmatrix}.
$$

2.1.2 Relation to RQ decomposition

The RQ decomposition transforms a matrix *A* into the product of an upper triangular matrix *R* (also known as right-triangular) and an orthogonal matrix *Q*. The only difference from QR decomposition is the order of these matrices.

QR decomposition is Gram–Schmidt orthogonalization of columns of *A*, started from the first column.

RQ decomposition is Gram–Schmidt orthogonalization of rows of *A*, started from the last row.

2.1.3 Advantages and disadvantages

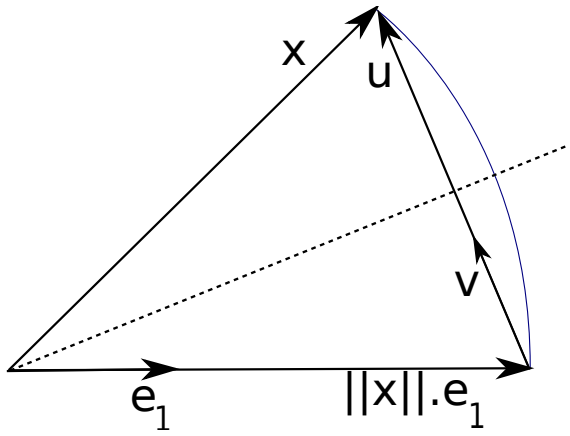
The Gram-Schmidt process is inherently numerically unstable. While the application of the projections has an appealing geometric analogy to orthogonalisation, the orthogonalisation itself is prone to numerical error. A significant advantage however is the ease of implementation, which makes this a useful algorithm to use for prototyping if a pre-built linear algebra library is unavailable.

2.2 Using Householder reflections

A Householder reflection (or *Householder transformation*) is a transformation that takes a vector and reflects it about some plane or hyperplane. We can use this operation to calculate the *QR* factorization of an *m*-by-*n* matrix *A* with $m \geq n$.

Q can be used to reflect a vector in such a way that all coordinates but one disappear.

Let $\mathbf{x}$ be an arbitrary real *m*-dimensional column vector of *A* such that $\|\mathbf{x}\| = |\alpha|$ for a scalar α. If the algorithm is implemented using floating-point arithmetic,



Householder reflection for QR-decomposition: The goal is to find a linear transformation that changes the vector x into a vector of same length which is collinear to e_1 . We could use an orthogonal projection (Gram-Schmidt) but this will be numerically unstable if the vectors x and e_1 are close to orthogonal. Instead, the Householder reflection reflects through the dotted line (chosen to bisect the angle between x and e_1). The maximum angle with this transform is 45 degrees.

then α should get the opposite sign as the *k*-th coordinate of $\mathbf{x}$, where x_k is to be the pivot coordinate after which all entries are 0 in matrix *A*'s final upper triangular form, to avoid loss of significance. In the complex case, set

$$\alpha = -\mathrm{e}^{\mathrm{i}\arg x_k}\|\mathbf{x}\|$$

(Stoer & Bulirsch 2002, p. 225) and substitute transposition by conjugate transposition in the construction of *Q* below.

Then, where $\mathbf{e}_1$ is the vector (1,0,...,0)$^{\mathrm{T}}$, $\|\cdot\|$ is the Euclidean norm and I is an *m*-by-*m* identity matrix, set

$$\mathbf{u} = \mathbf{x} - \alpha\mathbf{e}_1,$$

$$\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|},$$

$$Q = I - 2\mathbf{v}\mathbf{v}^T.$$

Or, if A is complex

$$Q = I - (1+w)\mathbf{v}\mathbf{v}^H \text{ , where } w = \mathbf{x}^H\mathbf{v}/\mathbf{v}^H\mathbf{x}$$

where $\mathbf{x}^H$ is the conjugate transpose (also Hermitian transpose or transjugate) of $\mathbf{x}$

Q is an *m*-by-*m* Householder matrix and

$$Q\mathbf{x} = (\alpha, 0, \cdots, 0)^T.$$

This can be used to gradually transform an *m*-by-*n* matrix *A* to upper triangular form. First, we multiply *A* with the Householder matrix Q_1 we obtain when we choose the first matrix column for $\mathbf{x}$. This results in a matrix Q_1A with zeros in the left column (except for the first row).

$$Q_1A = \begin{bmatrix} \alpha_1 & \star & \dots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{bmatrix}$$

This can be repeated for A' (obtained from Q_1A by deleting the first row and first column), resulting in a Householder matrix Q'_2. Note that Q'_2 is smaller than Q_1. Since we want it really to operate on Q_1A instead of A' we need to expand it to the upper left, filling in a 1, or in general:

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & Q'_k \end{pmatrix}.$$

After t iterations of this process, $t = \min(m-1, n)$,

$$R = Q_t \cdots Q_2 Q_1 A$$

is an upper triangular matrix. So, with

$$Q = Q_1^T Q_2^T \cdots Q_t^T,$$

$A = QR$ is a QR decomposition of A .

This method has greater numerical stability than the Gram–Schmidt method above.

The following table gives the number of operations in the *k*-th step of the QR-decomposition by the Householder transformation, assuming a square matrix with size *n*.

Summing these numbers over the *n* – 1 steps (for a square matrix of size *n*), the complexity of the algorithm (in terms of floating point multiplications) is given by

$$\frac{2}{3}n^3 + n^2 + \frac{1}{3}n - 2 = O(n^3).$$

2.2.1 Example

Let us calculate the decomposition of

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.$$

First, we need to find a reflection that transforms the first column of matrix *A*, vector $\mathbf{a}_1 = (12, 6, -4)^T$, into $\|\mathbf{a}_1\|\,\mathrm{e}_1 = (14, 0, 0)^T$.

Now,

$$\mathbf{u} = \mathbf{x} - \alpha \mathbf{e}_1,$$

and

$$\mathbf{v} = \frac{\mathbf{u}}{\|\mathbf{u}\|}.$$

Here,

$$\alpha = 14 \text{ and } \mathbf{x} = \mathbf{a}_1 = (12, 6, -4)^T$$

Therefore

$$\mathbf{u} = (-2, 6, -4)^T = (2)(-1, 3, -2)^T \text{ and}$$
$$\mathbf{v} = \tfrac{1}{\sqrt{14}}(-1, 3, -2)^T \text{, and then}$$
$$Q_1 = I - \tfrac{2}{\sqrt{14}\sqrt{14}} \begin{pmatrix} -1 \\ 3 \\ -2 \end{pmatrix} \begin{pmatrix} -1 & 3 & -2 \end{pmatrix}$$
$$= I - \tfrac{1}{7} \begin{pmatrix} 1 & -3 & 2 \\ -3 & 9 & -6 \\ 2 & -6 & 4 \end{pmatrix}$$
$$= \begin{pmatrix} 6/7 & 3/7 & -2/7 \\ 3/7 & -2/7 & 6/7 \\ -2/7 & 6/7 & 3/7 \end{pmatrix}.$$

Now observe:

$$Q_1 A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & -49 & -14 \\ 0 & 168 & -77 \end{pmatrix},$$

so we already have almost a triangular matrix. We only need to zero the (3, 2) entry.

Take the (1, 1) minor, and then apply the process again to

$$A' = M_{11} = \begin{pmatrix} -49 & -14 \\ 168 & -77 \end{pmatrix}.$$

By the same method as above, we obtain the matrix of the Householder transformation

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -7/25 & 24/25 \\ 0 & 24/25 & 7/25 \end{pmatrix}$$

after performing a direct sum with 1 to make sure the next step in the process works properly.

Now, we find

$$Q = Q_1^T Q_2^T = \begin{pmatrix} 6/7 & -69/175 & 58/175 \\ 3/7 & 158/175 & -6/175 \\ -2/7 & 6/35 & 33/35 \end{pmatrix}$$

Then

$$Q = Q_1^T Q_2^T = \begin{pmatrix} 0.8571 & -0.3943 & 0.3314 \\ 0.4286 & 0.9029 & -0.0343 \\ -0.2857 & 0.1714 & 0.9429 \end{pmatrix}$$

$$R = Q_2 Q_1 A = Q^T A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & -35 \end{pmatrix}.$$

The matrix *Q* is orthogonal and *R* is upper triangular, so *A* = *QR* is the required QR-decomposition.

2.2.2 Advantages and disadvantages

The use of Householder transformations is inherently the most simple of the numerically stable QR decomposition algorithms due to the use of reflections as the mechanism for producing zeroes in the *R* matrix. However, the Householder reflection algorithm is bandwidth heavy and not parallelisable, as every reflection that produces a new zero element changes the entirety of both *Q* and *R* matrices.

2.3 Using Givens rotations

QR decompositions can also be computed with a series of Givens rotations. Each rotation zeroes an element in the subdiagonal of the matrix, forming the *R* matrix. The concatenation of all the Givens rotations forms the orthogonal *Q* matrix.

In practice, Givens rotations are not actually performed by building a whole matrix and doing a matrix multiplication. A Givens rotation procedure is used instead which does the equivalent of the sparse Givens matrix multiplication, without the extra work of handling the sparse elements. The Givens rotation procedure is useful in situations where only a relatively few off diagonal elements need to be zeroed, and is more easily parallelized than Householder transformations.

2.3.1 Example

Let us calculate the decomposition of

$$A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.$$

First, we need to form a rotation matrix that will zero the lowermost left element, $\mathbf{a}_{31} = -4$. We form this matrix using the Givens rotation method, and call the matrix G_1 . We will first rotate the vector $(12, -4)$, to point along the *X* axis. This vector has an angle $\theta = \arctan\left(\frac{-(-4)}{12}\right)$. We create the orthogonal Givens rotation matrix, G_1 :

$$G_1 = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

$$\approx \begin{pmatrix} 0.94868 & 0 & -0.31622 \\ 0 & 1 & 0 \\ 0.31622 & 0 & 0.94868 \end{pmatrix}$$

And the result of $G_1 A$ now has a zero in the $\mathbf{a}_{31}$ element.

$$G_1 A \approx \begin{pmatrix} 12.64911 & -55.97231 & 16.76007 \\ 6 & 167 & -68 \\ 0 & 6.64078 & -37.6311 \end{pmatrix}$$

We can similarly form Givens matrices G_2 and G_3 , which will zero the sub-diagonal elements a_{21} and a_{32} , forming a triangular matrix R . The orthogonal matrix Q^T is formed from the product of all the Givens matrices $Q^T = G_3 G_2 G_1$. Thus, we have $G_3 G_2 G_1 A = Q^T A = R$, and the *QR* decomposition is $A = QR$.

2.3.2 Advantages and disadvantages

The QR decomposition via Givens rotations is the most involved to implement, as the ordering of the rows required to fully exploit the algorithm are not trivial to determine. However, it has a significant advantage in that each new zero element affects only the row with the element to be zeroed and the row above. This makes the Givens rotation algorithm more bandwidth efficient and parallelisable, in contrast with the Householder reflection technique.

3 Connection to a determinant or a product of eigenvalues

We can use QR decomposition to find the absolute value of the determinant of a square matrix. Suppose a matrix is decomposed as $A = QR$. Then we have

$$\det(A) = \det(Q) \cdot \det(R).$$

Since *Q* is unitary, $|\det(Q)| = 1$. Thus,

$$|\det(A)| = |\det(R)| = \left| \prod_i r_{ii} \right|,$$

where r_{ii} are the entries on the diagonal of *R*.

Furthermore, because the determinant equals the product of the eigenvalues, we have

$$\left| \prod_i r_{ii} \right| = \left| \prod_i \lambda_i \right|,$$

where λ_i are eigenvalues of A .

We can extend the above properties to non-square complex matrix A by introducing the definition of QR-decomposition for non-square complex matrix and replacing eigenvalues with singular values.

Suppose a QR decomposition for a non-square matrix *A*:

$$A = Q \begin{pmatrix} R \\ O \end{pmatrix}, \qquad Q^* Q = I,$$

where O is a zero matrix and Q is a unitary matrix.

From the properties of SVD and determinant of matrix, we have

$$\left| \prod_i r_{ii} \right| = \prod_i \sigma_i,$$

where σ_i are singular values of A .

Note that the singular values of A and R are identical, although their complex eigenvalues may be different. However, if *A* is square, the following is true:

$$\prod_i \sigma_i = \left| \prod_i \lambda_i \right|.$$

In conclusion, QR decomposition can be used efficiently to calculate the product of the eigenvalues or singular values of a matrix.

4 Column pivoting

QR decomposition with column pivoting introduces a permutation matrix *P*:

$$AP = QR \quad \iff \quad A = QRP^T$$

Column pivoting is useful when *A* is (nearly) rank deficient, or is suspected of being so. It can also improve numerical accuracy. *P* is usually chosen so that the diagonal elements of *R* are non-increasing: $|r_{11}| \geq |r_{22}| \geq \ldots \geq |r_{nn}|$. This can be used to find the (numerical) rank of *A* at lower computational cost than a singular value decomposition, forming the basis of so-called rank-revealing QR algorithms.

5 Using for solution to linear inverse problems

Compared to the direct matrix inverse, inverse solutions using QR decomposition are more numerically stable as

evidenced by their reduced condition numbers [Parker, Geophysical Inverse Theory, Ch1.13].

To solve the underdetermined ($m < n$) linear problem $Ax = b$ where the matrix A has dimensions $m \times n$ and rank m , first find the QR factorization of the transpose of A: $A^T = QR$, where Q is an orthogonal matrix (i.e. $Q^T = Q^{-1}$), and R has a special form: $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$. Here R_1 is a square $m \times m$ right triangular matrix, and the zero matrix has dimension $(n - m) \times m$. After some algebra, it can be shown that a solution to the inverse problem can be expressed as: $x = Q \begin{bmatrix} (R_1^T)^{-1} b \\ 0 \end{bmatrix}$ where one may either find R_1^{-1} by Gaussian elimination or compute $(R_1^T)^{-1} b$ directly by forward substitution. The latter technique enjoys greater numerical accuracy and lower computations.

To find a solution, $\hat{x}$, to the overdetermined ($m \geq n$) problem $Ax = b$ which minimizes the norm $\|A\hat{x} - b\|$, first find the QR factorization of A: $A = QR$. The solution can then be expressed as $\hat{x} = R_1^{-1}(Q_1^T b)$, where Q_1 is an $m \times n$ matrix containing the first n columns of the full orthonormal basis Q and where R_1 is as before. Equivalent to the underdetermined case, back substitution can be used to quickly and accurately find this $\hat{x}$ without explicitly inverting R_1 . (Q_1 and R_1 are often provided by numerical libraries as an "economic" QR decomposition.)

6 Generalizations

Iwasawa decomposition generalizes QR decomposition to semisimple Lie groups.

7 See also

- Polar decomposition
- Eigenvalue decomposition
- Spectral decomposition
- LU decomposition
- Singular value decomposition

8 References

[1] L. N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, 1997).

- Golub, Gene H.; Van Loan, Charles F. (1996), *Matrix Computations* (3rd ed.), Johns Hopkins, ISBN 978-0-8018-5414-9.
- Horn, Roger A.; Johnson, Charles R. (1985), *Matrix Analysis*, Cambridge University Press, ISBN 0-521-38632-2. Section 2.8.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 2.10. QR Decomposition", *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8
- Stoer, Josef; Bulirsch, Roland (2002), *Introduction to Numerical Analysis* (3rd ed.), Springer, ISBN 0-387-95452-X.

9 External links

- Online Matrix Calculator Performs QR decomposition of matrices.
- LAPACK users manual gives details of subroutines to calculate the QR decomposition
- Mathematica users manual gives details and examples of routines to calculate QR decomposition
- ALGLIB includes a partial port of the LAPACK to C++, C#, Delphi, etc.
- Eigen::QR Includes C++ implementation of QR decomposition.

10 Text and image sources, contributors, and licenses

10.1 Text

- **QR decomposition** *Source:* https://en.wikipedia.org/wiki/QR_decomposition?oldid=759236182 *Contributors:* Michael Hardy, Wshun, Stevenj, Drz~enwiki, Dysprosia, Jitse Niesen, Shizhao, MathMartin, Yacht, Smb1001, Giftlite, BenFrantzDale, CyborgTosser, Chad.netzer, Phe, Eliosh, Jacob grace, Ary29, Simoneau, Chadernook, Mecanismo, Gauge, Pt, Oyz, 3mta3, Musiphil, Squizzz~enwiki, Oleg Alexandrov, Jftsang, Btyner, AySz88, Marozols, Lebha, Mathbot, Kri, Vonkje, YurikBot, Bruguiea, Ligand, SmackBot, Bluebot, Silly rabbit, Giganut, Trifon Triantafillidis, John, Rogério Brito, SMWatt, Thijs!bot, AlexAlex, BigJohnHenry, GromXXVII, Drizzd~enwiki, Sanchom, MwGamera, K.menin, JonMcLoone, Vasilaky, JohnBlackburne, TXiKiBoT, Daviddoria, Rhanekom, EnJx, JackSchmidt, OKBot, Bfx0, Marcus.bishop, Shai mach, Realwhz, BOTarate, Qwfp, Addbot, Some jerk on the Internet, Crillion x, Yobot, AnomieBOT, StevePny, Citation bot, Cerniagigante, Aldebrn, ApocryphalAuthor, FrescoBot, BenzolBot, DrilBot, RedBot, Ambrevar, Dropsciencenotbombs, Marie Poise, Jfmantis, John of Reading, Chaohuang, Dcirovic, Redav, Bugmenot10, ClueBot NG, Wcherowi, Joel B. Lewis, Danfrankj, BG19bot, ServiceAT, JMineroff, Patnaik.awhan, Dexbot, Citizentoad, Wqwt, Ghjung89, Oisguad, AHusain3141, Oseledets1983, Mikeyrichardson, Suetus, InternetArchiveBot, GreenC bot, Laxatives11, Td1961, Huoyuanzi and Anonymous: 152

10.2 Images

- **File:Householder.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/7/7b/Householder.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Bruguiea
- **File:Wiki_letter_w.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/6/6c/Wiki_letter_w.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?
- **File:Wiki_letter_w_cropped.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/1c/Wiki_letter_w_cropped.svg *License:* CC-BY-SA-3.0 *Contributors:* This file was derived from Wiki letter w.svg:
Original artist: Derivative work by Thumperward

10.3 Content license

- Creative Commons Attribution-Share Alike 3.0