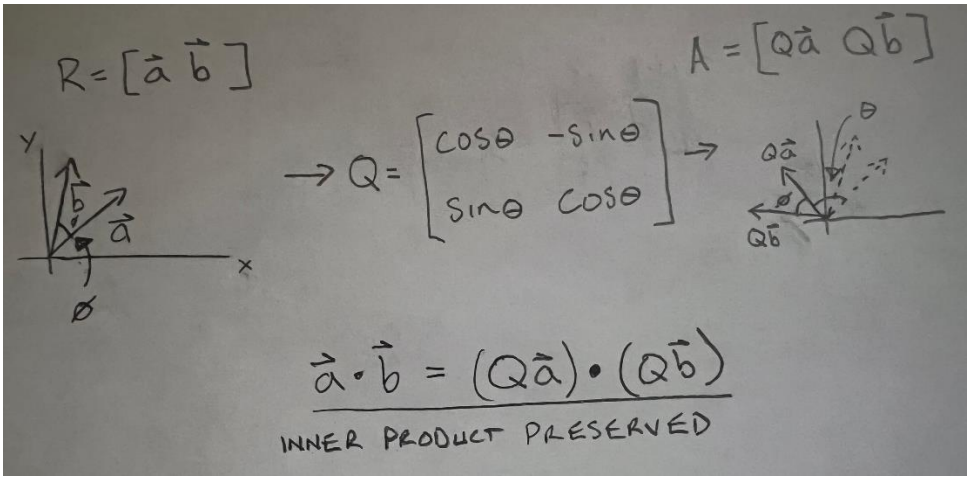
QR Decomposition

Definition

The QR decomposition of an *m*-by-*n* matrix A with *m>n,* is the matrix product A = QR, where Q is an *m*-by-*n* **unitary** matrix, and R is **upper triangular**.

Matrix Q

The matrix Q is a transformation which preserves inner products of column vectors of R. If the inner product space is real, the matrix Q is equivalently **orthogonal**. One possibility of such a transformation is a rotation.



Another possibility of such an orthogonal transformation is a reflection. The matrix Q in general is a combination of rotations and reflections.

Matrix R

The matrix R is **upper triangular**, which has the following properties:

- The determinant is equal to the product of the diagonal entries
- The eigenvalues are equal to the diagonal entries

Given an upper triangular matrix *R*, it is easy to solve the linear system represented by ***Rx*=*b*** by back substitution.

Computation

In order to compute the decomposition of *A*, the matrix is iteratively transformed by unitary matrices $\{U_i : 0 < i < k\}$ until the product is upper triangular. This upper triangular matrix is the matrix R in A=QR.

$$U_k U_{k-1} \ldots U_1 A = R$$

The matrix Q can then be computed as the matrix product of the inverse transforms U_i^T.

$$Q = U_1^T U_2^T \ldots U_k^T$$

The key to solving for *R* is to choose transformations U_i which produce zeros below the diagonal of the matrix product,

$$U_k U_{k-1} \ldots U_i \ldots U_1 A,$$

and can iteratively be applied to converge to *R* as quickly as possible. Two choices for U_i are Householder reflections, and Givens rotations.

Householder Reflections

The Householder reflection is a unitary transformation which reflects a vector ***x*** across a hyperplane. The hyperplane is defined by its unit normal vector ***v***.

The transformation matrix is given by,

$$P = I - 2vv^T$$

Where *I* is the identity matrix.

Givens rotations

A Givens rotation is a unitary transformation which rotates a vector ***x*** counter-clockwise in a chosen plane. For example, possible Givens rotation matrices in $\mathbb{R}^4$ are,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & c & -s \\ 0 & 0 & s & c \end{bmatrix},$$

where $c = \cos\theta$ and $s = sin\theta$. All of the examples are valid Givens transformations, but have the effect of rotating the vector in different planes. The Givens rotation is parallelizable because it only effects two dimensions of the input vector. For example the second and last transformations above could simultaneously be computed on the vector ***x***, then the results combined by selecting the first 2 dimensions of the first result, and the second two dimensions of the second result.

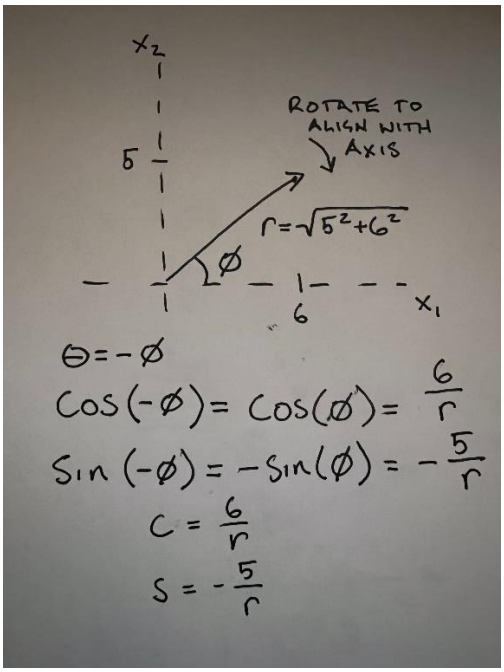
A Givens rotation can easily be computed to introduce zeros in the matrix. The scalars c and s can be computed directly from elements in the input in order to zero out targeted elements. For example, say we want to zero out element (2,1) in the following matrix,

$$A = \begin{bmatrix} 6 & 5 & 0 \\ 5 & 1 & 4 \\ 0 & 4 & 3 \end{bmatrix}.$$

We target the second dimension of the column vector, so we rotate on the plane spanned by the first two dimensions. We don't choose the plane spanned by the second and third dimensions, because we would end up losing the zero in the third row in the process. The Givens rotation to rotate on this plane is of the form,

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

which will leave the third row untouched. We are aligning the column vector with the axis of the first dimension, making the component of the vector along the second dimension zero. Below is a geometric illustration of the rotation.



The scalars ***c*** and ***s*** are computed directly as shown above. The angle does not need to be computed. The transformation to introduce the zero is then,

$$GA = \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 5 & 0 \\ 5 & 1 & 4 \\ 0 & 4 & 3 \end{bmatrix}$$

$$GA = \begin{bmatrix} 6/r & 5/r & 0 \\ -5/r & 6/r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 5 & 0 \\ 5 & 1 & 4 \\ 0 & 4 & 3 \end{bmatrix},$$

where $r = \sqrt{6^2 + 5^2}$,

$$GA = \begin{bmatrix} 6/r & 5/r & 0 \\ -5/r & 6/r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 & 5 & 0 \\ 5 & 1 & 4 \\ 0 & 4 & 3 \end{bmatrix} = \begin{bmatrix} 7.8 & 4.5 & 2.6 \\ 0 & -2.4 & 3.1 \\ 0 & 4 & 3 \end{bmatrix}.$$

Application to Linear Least Squares

A linear least squares solution is an approximate solution to an over-constrained system of equations, represented as,

Ax+b,

where A is an *m-by-n* matrix with *m>n*. Imagine you have 3 (x, y) data points,

X	Y
1	1

2	1
2	2

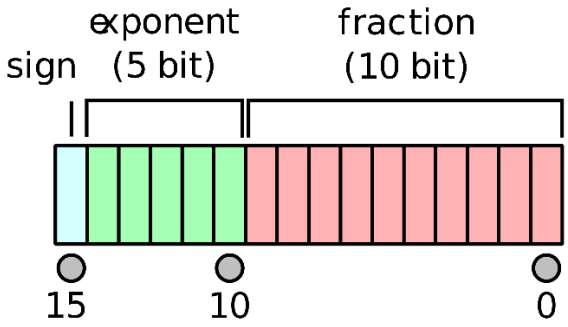
and you are to find the line which the points lie on. This defines an over-constrained system with no solution for the slope *m* and intercept *b*.

$$\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

A least squares solution is an approximate solution for a line which minimizes the squared error between the approximated solution, and the measured output.

IEEE FP16

The IEEE-754 standard defines a 16-bit floating point representation of a number with 1 bit representing the sign, 5 bits specifying the exponent, and 10 bits representing the fractional part of the number, the mantissa.



The equation for converting between the binary encoded decimal numbers representing individual bit fields to the encoded floating point value is

$$(-1)^s \times 2^{E-15} \times 1.M,$$

where s is the sign decimal value, E is the exponent value, and M is the mantissa value.

A special case where the exponent value is 0 means the equation is,

$$(-1)^s \times 2^{-14} \times 0.M,$$

and if the exponent field is 31, the encoding represents infinity.

The exponent uses an offset binary representation, meaning 15 is subtracted from the binary encoded decimal number of the exponent bits. The range of the exponent is -14 to 15.

Rounding Error

The rounding error depends on the value of the exponent. For the smallest representable numbers, when the exponent after offset is -14, the interval between representable numbers is 1 LSB of the mantissa multiplied by 2^{-14},

$$interval = (\tfrac{1}{2})^{10} \times 2^{-14} = 5.96E-8\,,$$

In general, the equation for the interval between numbers is,

$$interval = (\frac{1}{2})^{10} \times 2^{E-15} .$$

The maximum rounding error is ½ of the interval between numbers.

Condition Number

The condition number of a matrix represents the change in the output of the transformation relative to small changes in the input. A large condition number means the error at the output is relatively large for a small error in the input, and such a transformation is **ill-conditioned**. A small condition number means the inverse is true, and the transformation is **well-conditioned**.