# Gram-Schmidt Procedure and QR Factorization

In [1]:
```
using DrWatson;
@quickactivate "MATH361Lectures";
```

In [2]:
```
import MATH361Lectures
using LinearAlgebra
```

## Solving Linear Least Squares with Cholesky

The following Julia function uses the Cholesky factorization method to solve the problem of least squares, that is, given $A$ and $b$, we find $x$ that solves

$$\text{argmin}_{v\in\mathbb{R}^n}\|Av-b\|_2$$

In [3]:
```
function lsqcholesky(A,b)
   L,U = cholesky(A'*A);
   w = MATH361Lectures.forwardsub(L,A'*b);
   x = MATH361Lectures.backsub(U,w);
   return x
end
```

Out[3]:
```
lsqcholesky (generic function with 1 method)
```

In [4]:
```
A = [1 2 -4; 3 -1 1; 1 -2 1; 3 -2 -1; 4 2 -1];
b = [-1; 2; -2; 1; 3];
x = lsqcholesky(A,b)
```

Out[4]:
```
3-element Vector{Float64}:
 0.6296350152682856
 0.6441762396393778
 0.5746691871455578
```

Compare this with what we obtain using the Julia backslash operator:

In [5]:
```
x_bs = A \ b
```

Out[5]:
```
3-element Vector{Float64}:
 0.6296350152682856
 0.6441762396393776
 0.5746691871455579
```

Additionally, for the solution we have obtained, let's compute $\|Ax-b\|_2$.

In [6]:
```
norm(A*x-b,2)
```

Out[6]:
```
2.2560728637642335
```

In [7]:
```
norm(A*x_bs-b,2)
```

Out[7]:
```
2.2560728637642335
```

Observe what happens if we introduce a small perturbation to $x$:

In [8]:
```
x_pert = x + [0.00001,-0.00002,0.00005]
```

Out[8]:
```
3-element Vector{Float64}:
 0.6296450152682855
 0.6441562396393777
 0.5747191871455578
```

In [9]:
```
norm(A*x_pert - b,2)
```

Out[9]:
```
2.256072880563336
```

We see that there is a small increase in the two norm for $\|Ax_{\text{pert}}-b\|_2$. To confirm that it is indeed a small perturnation, let's compute $\|x-x_{\text{pert}}\|_2$.

In [10]:
```
norm(x - x_pert,2)
```

Out[10]:
```
5.4772255750510576e-5
```

## Background for QR Factorization

### Orthonormal Vectors

Recall that the dot product of two column vectors $\mathbf{u}=[u_1,u_2,\ldots,u_n]^T$ and $\mathbf{v}=[v_1,v_2,\ldots,v_n]^T$ is

$$\mathbf{u}\cdot\mathbf{v}=\mathbf{u}^T\mathbf{v}=u_1v_1+u_2v_2+\cdots+u_nv_n.$$

Observe that if $\mathbf{u}$ is a vector, then $\|\mathbf{u}\|_2^2=\mathbf{u}^T\mathbf{u}$, and also that $\mathbf{u}^T\mathbf{v}=\mathbf{v}^T\mathbf{u}$.

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be **orthogonal** if their dot product is zero, that is, if $\mathbf{u}\cdot\mathbf{v}=\mathbf{u}^T\mathbf{v}=0$. We say that a vector is normalized (in the 2-norm) if $\|\mathbf{u}\|_2=1$.

### Orthonormal Set of Vectors

A set of vectors $\{\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_n\}$ is an **orthogonal set** if $\mathbf{q}_i^T\mathbf{q}_j=0$ whenever $i\neq j$. Furthermore, an orthogonal set of vectors is an **orthonormal set** if, in addition $\|\mathbf{q}_i\|_2=1$ for all $i$.

### Orthogonal Matrices

A matrix $Q$ is **orthogonal** if it's columns form a orthogonal set of vectors.

A matrix $Q$ is **ONC** if it's columns form an orthonormal set. Equivalently, a matrix $Q$ is ONC if $Q^TQ=I$.

As an example, any permutation matrix $P$ is ONC.

### The Gram Schmidt Procedure

Given any set of linearly independent vectors, $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n$, there is a procedure known as the Gram Schmidt procedure that produces an orthonormal set $\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_n$ with the same span as the original independent set. The Gram Schmidt procedure works as follows:

Set

$$\begin{aligned}
\mathbf{q}_1 &= \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|_2},\\
\mathbf{q}_2 &= \frac{\mathbf{a}_2-(\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1}{\|\mathbf{a}_2-(\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1\|_2},\\
\mathbf{q}_3 &= \frac{\mathbf{a}_3-(\mathbf{q}_1^T\mathbf{a}_3)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_3)\mathbf{q}_2}{\|\mathbf{a}_3-(\mathbf{q}_1^T\mathbf{a}_3)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_3)\mathbf{q}_2\|_2},\\
&\vdots\\
\mathbf{q}_n &= \frac{\mathbf{a}_n-(\mathbf{q}_1^T\mathbf{a}_n)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_n)\mathbf{q}_2-\cdots-(\mathbf{q}_{n-1}^T\mathbf{a}_n)\mathbf{q}_{n-1}}{\|\mathbf{a}_n-(\mathbf{q}_1^T\mathbf{a}_n)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_n)\mathbf{q}_2-\cdots-(\mathbf{q}_{n-1}^T\mathbf{a}_n)\mathbf{q}_{n-1}\|_2},
\end{aligned}$$

Now, note that we can "reverse" the result of the Gram Schmidt procedure by solving for the $\mathbf{a}_i$ vectors as linear combinations of the $\mathbf{q}_i$ vectors:

$$\begin{aligned}
\mathbf{a}_1 &= \|\mathbf{a}_1\|_2\mathbf{q}_1,\\
\mathbf{a}_2 &= (\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1+\|\mathbf{a}_2-(\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1\|_2\mathbf{q}_2,\\
\mathbf{a}_3 &= (\mathbf{q}_1^T\mathbf{a}_3)\mathbf{q}_1+(\mathbf{q}_2^T\mathbf{a}_3)\mathbf{q}_2+\|\mathbf{a}_3-(\mathbf{q}_1^T\mathbf{a}_3)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_3)\mathbf{q}_2\|_2\mathbf{q}_3,\\
&\vdots\\
\mathbf{a}_n &= (\mathbf{q}_1^T\mathbf{a}_n)\mathbf{q}_1+(\mathbf{q}_2^T\mathbf{a}_n)\mathbf{q}_2+\cdots+(\mathbf{q}_{n-1}^T\mathbf{a}_n)\mathbf{q}_{n-1}+\|\mathbf{a}_n-(\mathbf{q}_1^T\mathbf{a}_n)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_n)\mathbf{q}_2-\cdots-(\mathbf{q}_{n-1}^T\mathbf{a}_n)\mathbf{q}_{n-1}\|_2\mathbf{q}_n,
\end{aligned}$$

We can simplify the expressions in the last cell by defining

$$\begin{aligned}
r_{kk} &= \|\mathbf{a}_k-(\mathbf{q}_1^T\mathbf{a}_k)\mathbf{q}_1-(\mathbf{q}_2^T\mathbf{a}_k)\mathbf{q}_k-\cdots-(\mathbf{q}_{k-1}^T\mathbf{a}_k)\mathbf{q}_{k-1}\|_2,\\
r_{kl} &= \mathbf{q}_k^T\mathbf{a}_l,\quad \text{for } l>k
\end{aligned}$$

Then we obtain:

$$\begin{aligned}
\mathbf{a}_1 &= r_{11}\mathbf{q}_1,\\
\mathbf{a}_2 &= r_{12}\mathbf{q}_1+r_{22}\mathbf{q}_2,\\
\mathbf{a}_3 &= r_{13}\mathbf{q}_1+r_{23}\mathbf{q}_2+r_{33}\mathbf{q}_3,\\
&\vdots\\
\mathbf{a}_n &= r_{1n}\mathbf{q}_1+r_{2n}\mathbf{q}_2+\cdots+r_{n-1n}\mathbf{q}_{n-1}+r_{nn}\mathbf{q}_n,
\end{aligned}$$

Which can be simplified even further to $A=QR$ if we think of the vectors $\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_n$ as forming the columns of the matrix $A$, the vectors $\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_n$ as forming the columns of a matrix $Q$, and the scalars $r_{ij}$ as forming the entries of an **upper triangular** matrix $R$.

The following Julia function implements the Gram Schmidt procedure on the columns of an $m\times n$ matrix to produce an ONC matrix $Q$:

In [11]:
```
function gsqr(A)
   m,n = size(A);
   Q = Matrix{Float64}(A);
   R = Matrix{Float64}(I,n,n);
   R[1,1] = norm(Q[:,1],2);
   Q[:,1] = (1/R[1,1]) *  Q[:,1]; # get first column of Q
   for j = 2:n # loop through columns
        for i = 1:j-1
            R[i,j] = dot(Q[:,i],Q[:,j]);
            Q[:,j] = Q[:,j] .- R[i,j]*Q[:,i];
        end
        R[j,j] = norm(Q[:,j],2);
        Q[:,j] = (1/R[j,j])*Q[:,j];
    end
    return Q, R
end
```

Out[11]:
```
gsqr (generic function with 1 method)
```

Let's look at an example:

In [12]:
```
A = [1 2 3;-1 1 -1; 0 1 0;2 -1 2]
```

Out[12]:
```
4×3 Matrix{Int64}:
  1   2   3
 -1   1  -1
  0   1   0
  2  -1   2
```

In [13]:
```
Q,R = gsqr(A);
```

In [14]:
```
Q
```

Out[14]:
```
4×3 Matrix{Float64}:
  0.408248   0.82885    0.382546
 -0.408248   0.318788  -0.255031
  0.0        0.382546  -0.82885
  0.816497  -0.255031  -0.318788
```

In [15]:
```
R
```

Out[15]:
```
3×3 Matrix{Float64}:
 2.44949  -0.408248  3.26599
 0.0       2.61406   1.6577
 0.0       0.0       0.765092
```

We will check manually that indeed $Q$ is ONC:

In [16]:
```
Q[:,1]'*Q[:,1]
```

Out[16]:
```
1.0000000000000002
```

In [17]:
```
Q[:,2]'*Q[:,2]
```

Out[17]:
```
0.9999999999999998
```

In [18]:
```
Q[:,3]'*Q[:,3]
```

Out[18]:
```
0.9999999999999997
```

In [19]:
```
Q[:,1]'*Q[:,2]
```

Out[19]:
```
5.551115123125783e-17
```

In [20]:
```
Q[:,1]'*Q[:,3]
```

Out[20]:
```
-1.3322676295501878e-15
```

In [21]:
```
Q[:,2]'*Q[:,3]
```

Out[21]:
```
3.3306690738754696e-16
```

Here are a few important points:

1) The Gram Schmidt procedure results in reduce QR factorization. Of course this is all that is needed for solving the linear least squares problem.

2) The Gram Schmidt method is not the most numerically stable method for QR factorization. In the next lecture, we will look at another approach that uses so-called Householder reflectors in order to obtain (full) QR factorization. In preparation for the next lecture, please watch the video on QR factorization.