Gram-Schmidt and Modified Gram-Schmidt

In [1]:

```
#keep
import numpy as np
import numpy.linalg as la
```

In [2]:

```
#keep
A = np.random.randn(3, 3)
```

In [3]:

```
#keep
def test_orthogonality(Q):
    print("Q:")
    print(Q)

    print("Q^T Q:")
    QtQ = np.dot(Q.T, Q)
    QtQ[np.abs(QtQ) < 1e-15] = 0
    print(QtQ)
```

In [4]:

```
#keep
Q = np.zeros(A.shape)
```

Now let us generalize the process we used for three vectors earlier:

In [5]:

```
for k in range(A.shape[1]):
    avec = A[:, k]
    q = avec
    for j in range(k):
        q = q - np.dot(avec, Q[:,j])*Q[:,j]

    Q[:, k] = q/la.norm(q)
```

This procedure is called Gram-Schmidt Orthonormalization (https://en.wikipedia.org/wiki/Gram–Schmidt_process).

In [6]:

```
#keep
test_orthogonality(Q)
```

```
Q:
[[-0.65609385 -0.75361761 -0.04001703]
 [ 0.65802183 -0.59722164  0.4586214 ]
 [ 0.36952419 -0.2745666  -0.88773028]]
Q^T Q:
[[  1.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   1.00000000e+00  -1.05471187e-15]
 [  0.00000000e+00  -1.05471187e-15   1.00000000e+00]]
```

Now let us try a different example (Source (http://fgiesen.wordpress.com/2013/06/02/modified-gram-schmidt-orthogonalization/)):

In [7]:

```
#keep

np.set_printoptions(precision=13)

eps = 1e-8

A = np.array([
    [1,  1,  1],
    [eps,eps,0],
    [eps,0,  eps]
    ])

A
```

Out[7]:

```
array([[  1.0000000000000e+00,   1.0000000000000e+00,   1.0000000000
000e+00],
       [  1.0000000000000e-08,   1.0000000000000e-08,   0.0000000000
000e+00],
       [  1.0000000000000e-08,   0.0000000000000e+00,   1.0000000000
000e-08]])
```

In [8]:

```
#keep
Q = np.zeros(A.shape)
```

In [9]:

```
#keep
for k in range(A.shape[1]):
    avec = A[:, k]
    q = avec
    for j in range(k):
        print(q)
        q = q - np.dot(avec, Q[:,j])*Q[:,j]

    print(q)
    q = q/la.norm(q)
    Q[:, k] = q
    print("norm -->", q)
    print("-------")
```

```
[  1.0000000000000e+00   1.0000000000000e-08   1.0000000000000e-08]
norm --> [  1.0000000000000e+00   1.0000000000000e-08   1.0000000000
000e-08]
-------
[  1.0000000000000e+00   1.0000000000000e-08   0.0000000000000e+00]
[  0.0000000000000e+00   0.0000000000000e+00  -1.0000000000000e-08]
norm --> [ 0.  0. -1.]
-------
[  1.0000000000000e+00   0.0000000000000e+00   1.0000000000000e-08]
[  0.0000000000000e+00  -1.0000000000000e-08   0.0000000000000e+00]
[  0.0000000000000e+00  -1.0000000000000e-08  -1.0000000000000e-08]
norm --> [ 0.                -0.7071067811865 -0.7071067811865]
-------
```

In [10]:

```
#keep
test_orthogonality(Q)
```

```
Q:
[[  1.0000000000000e+00   0.0000000000000e+00   0.0000000000000e+00]
 [  1.0000000000000e-08   0.0000000000000e+00  -7.0710678118655e-01]
 [  1.0000000000000e-08  -1.0000000000000e+00  -7.0710678118655e-01]
]
Q^T Q:
[[  1.0000000000000e+00  -1.0000000000000e-08  -1.4142135623731e-08]
 [ -1.0000000000000e-08   1.0000000000000e+00   7.0710678118655e-01]
 [ -1.4142135623731e-08   7.0710678118655e-01   1.0000000000000e+00]
]
```

Questions:

- What happened?
- How do we fix it?

In [11]:

```
#keep
Q = np.zeros(A.shape)
```

In [12]:

```
for k in range(A.shape[1]):
    q = A[:, k]
    for j in range(k):
        q = q - np.dot(q, Q[:,j])*Q[:,j]

    Q[:, k] = q/la.norm(q)
```

In [13]:

```
#keep
test_orthogonality(Q)
```

```
Q:
[[  1.0000000000000e+00   0.0000000000000e+00   0.0000000000000e+00]
 [  1.0000000000000e-08   0.0000000000000e+00  -1.0000000000000e+00]
 [  1.0000000000000e-08  -1.0000000000000e+00   0.0000000000000e+00]
]
Q^T Q:
[[  1.0000000000000e+00  -1.0000000000000e-08  -1.0000000000000e-08]
 [ -1.0000000000000e-08   1.0000000000000e+00   0.0000000000000e+00]
 [ -1.0000000000000e-08   0.0000000000000e+00   1.0000000000000e+00]
]
```

This procedure is called *Modified* Gram-Schmidt Orthogonalization.

Questions:

- Is there a difference mathematically between modified and unmodified?
- Why are there 10^{-8} values left in $Q^T Q$?

In [16]:

```
A = np.random.rand(10,10)
print(np.linalg.cond(A))
print(np.linalg.cond(A)**2)
print(np.linalg.cond(A.T.dot(A)))
```

```
53.4739304307
2859.4612357
2859.4612357
```

In [ ]:

```

```

In [ ]: