In [1]:
```
%matplotlib notebook

from matplotlib.pylab import *
from numpy import *
```

Homework #6

Problem #1

Figure 10.1 in Trefethan and Bau illustrates a Householder reflection that reflects the vector $\mathbf{x}$ about the space H to obtain the reflected vector $F\mathbf{x}$ on the x-axis. From properties of vector addition, we might be tempted to write $\mathbf{x} + \mathbf{v} = F\mathbf{x}$, where $\mathbf{v}$ is the vector in the space orthogonal to H. However, Algorithm 10.1 defines $\mathbf{v}$ as $\mathbf{v} = F\mathbf{x} + \mathbf{x}$. The source of the confusion may lie in the fact that a "reflection" is more naturally defined in terms of a light ray "reflecting" off of a shiny surface.

To see that we can derive the Household reflector more naturally, consider an *incident ray* $\mathbf{v}_i$ that hits a reflective surface and reflects back *reflected ray* $\mathbf{v}_s$. This reflected ray satisfies the expression

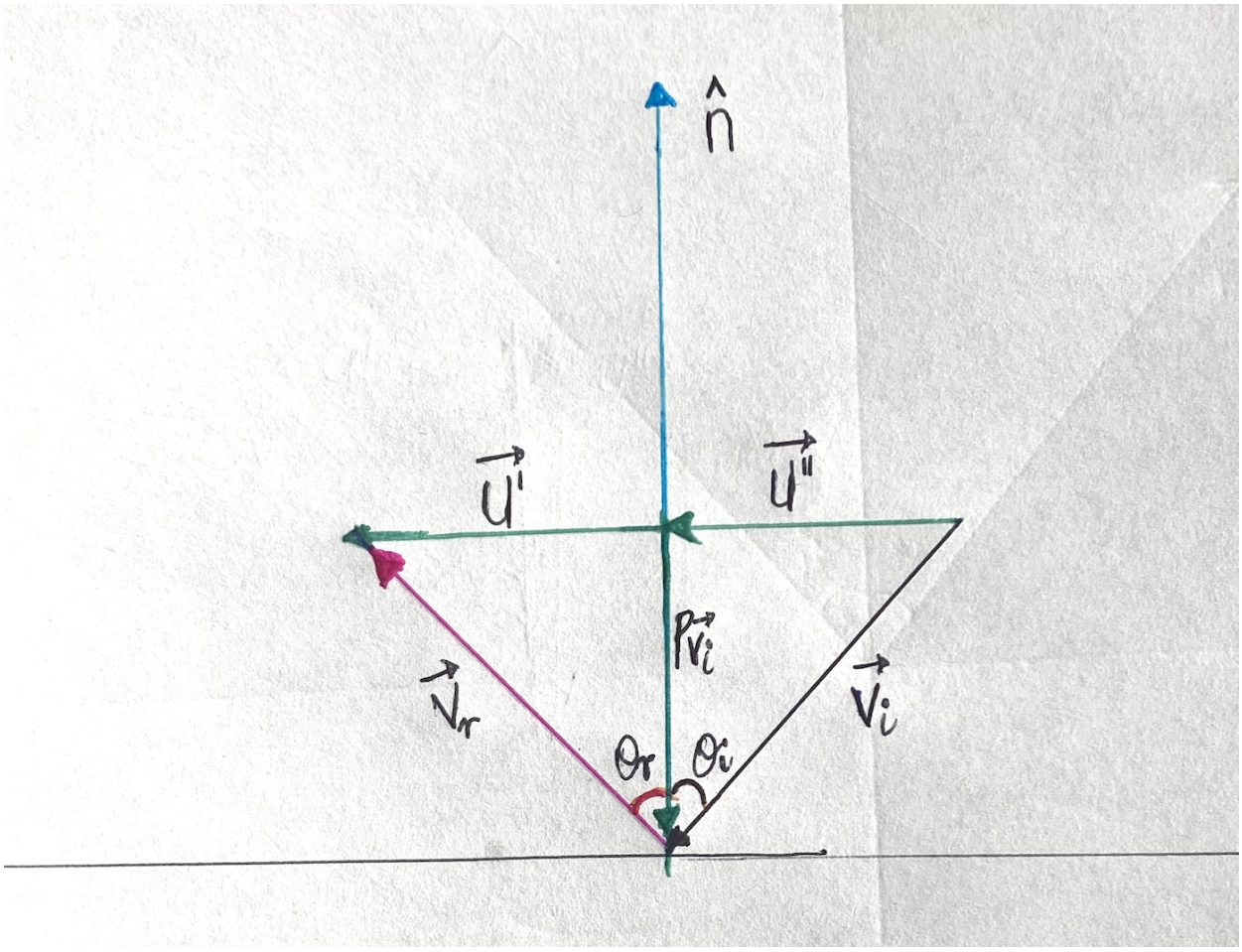
$$\mathbf{v}_s = R\mathbf{v}_i \tag{1}$$

where R is a unitary Householder projection of the form $I - 2P$ for some orthogonal projection P. The plot below illustates the these rays.

Problem

Derive the projector P needed to compute the reflector $R = I - 2P$ for incident ray $\mathbf{v}$. Show that the geometry using the "reflected light" analogy is consistent with the vector addition properties we know and with Algorithm 10.1. Provide a sample incident ray and show that your reflection R produces the expected reflected ray.

Hint: For this problem, you don't need to know anything about how light reflects, but should use the fact that the angles that the incident ray and reflected ray make with the surface normal are equal.



To derive the Household reflector R more naturally, we consider an incident ray $\mathbf{v}_i$ that hits a reflective surface and is reflected as ray $\mathbf{v}_r$. This reflected ray satisfies the expression;

$$\mathbf{v}_r = R\mathbf{v}_i \qquad (1) \qquad (2)$$

From the diagram above, if $\mathbf{v}_r$ is the reflected ray from the reflection of an incident ray $\mathbf{v}_i$ with respect to the unit surface normal $\hat{\mathbf{n}}$, then $\|\mathbf{v}_r\| = \|\mathbf{v}_i\|$ and $P\mathbf{v}_i$ is the projection of $\mathbf{v}_i$ onto $\hat{\mathbf{n}}$.

From the law of reflection i.e. $\theta_r = \theta_i = \theta$ and the fact that $\|\mathbf{v}_r\| = \|\mathbf{v}_i\|$, we get that $\mathbf{u}' = \mathbf{u}'' = (I - P)\mathbf{v}_i$.

And therefore from the diagram above, the incident and reflected rays are given by;

$$\mathbf{v}_i = P\mathbf{v}_i + (I - P)\mathbf{v}_i \qquad (3)$$
$$\mathbf{v}_r = -P\mathbf{v}_i + (I - P)\mathbf{v}_i \qquad (4)$$
$$= (I - 2P)\mathbf{v}_i \qquad (2) \qquad (5)$$

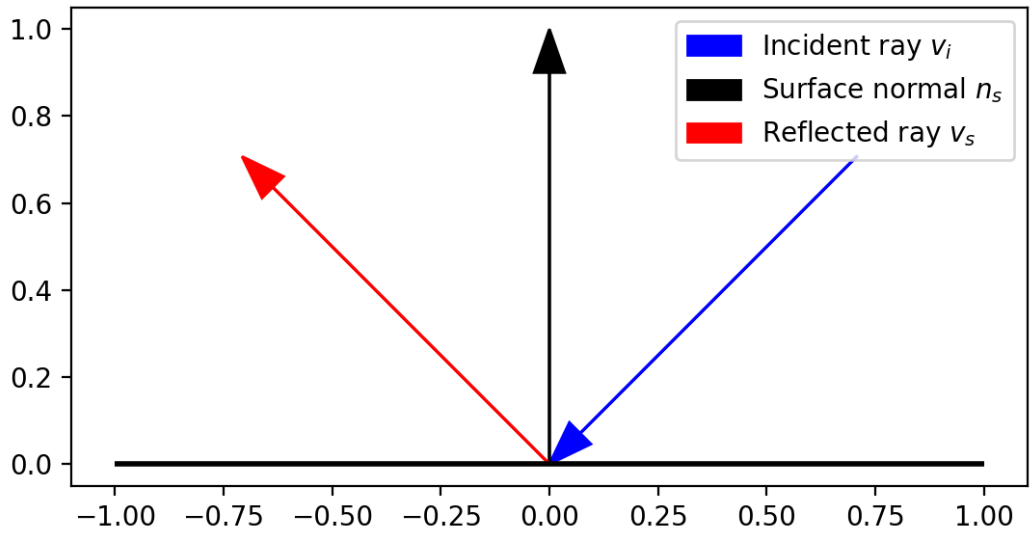
Thus comparing equations (1) and (2), we obtain that unitary Householder $R = I - 2P$.

Since $P\mathbf{v}_i$ is the projection of $\mathbf{v}_i$ onto the unit surface normal $\hat{\mathbf{n}}$, then P is given by;

$$P\mathbf{v}_i = (\hat{\mathbf{n}} \cdot \mathbf{v}_i)\hat{\mathbf{n}} \qquad (6)$$
$$\therefore P = \hat{\mathbf{n}}\hat{\mathbf{n}}^T \qquad (7)$$

In [8]:

```
# Problem #1 - Incident and reflected rays
figure(2)
clf()

def plot_vec(v,color='k',scale = 1,dir=1,label=''):
    v0 = float(v[0])
    v1 = float(v[1])
    L = scale

    if dir == 1:
        o1,o2,e1,e2 = (0,0,v0,v1)
    else:
         o1,o2,e1,e2 = (v0,v1,-v0,-v1)

    arrow(o1,o2,e1,e2,
          head_width=L*0.07, \
          head_length=L*0.1, \
          fc=color, \
          ec=color, \
          length_includes_head = True, \
          label=label)



hlines(0,xmin=-1,xmax=1,linewidth=2,color='k')

# Rays are all of unit length
a = 1/sqrt(2)
v = array([a,a])

plot_vec(v,color='b',dir=0,label=r'Incident ray $v_i$')



#P = array([0,-0.5])
#plot_vec(P,color='k',dir=1, label=r'Surface normal $n_s$')


n = array([0,1])
plot_vec(n,color='k',dir=1, label=r'Surface normal $n_s$')

#P = 2*np.dot(n,v)*n - v
#v = [-a,a]
R = -v + 2*np.dot(n,v)*n
#R = np.eye(2) - 2*(n@n.T)
#vs = R@v



plot_vec(R,color='r',dir=1,label=r'Reflected ray $v_s$')


legend()

#ylim([-0.05,1.25])

gca().set_aspect('equal')
```



Problem #2

Problem 6.5 (page 47) in TB asked you to show that for a non-zero projector P, we have $\|P\|_2 \geq 1$, with equality if and only if P is an orthogonal projector. One solution to this problem is to use the SVD decomposition of the projector P.

However, the SVD approach really doesn't provide much geometrical insight into the different between general projectors and orthogonal projectors. For this problem, show that you can demonstrate the following using only a geometrical argument.

- Show that for a non-zero projector, we have $\|P\|_2 \geq 1$.

Then show the two parts of the "if and only if".

- $(\Longleftarrow)$ Show that if P is an orthogonal projector, we have $\|P\|_2 = 1$
- $(\Longrightarrow)$Show that if $\|P\|_2 = 1$, P must be orthogonal.

To demonstrate the above geometrically, start with a decomposition of a vector $\mathbf{x} \in R^m$ into unique components in subspaces $S_1 = \text{range}(P)$ and $S_2 = \text{range}(I - P)$ given by

$$\mathbf{x} = P\mathbf{x} + (I - P)\mathbf{x} \tag{8}$$

You can define angles between subspaces using this idea

$$(P\mathbf{x})^T(I-P)\mathbf{x} = \|P\mathbf{x}\|_2\|(I-P)\mathbf{x}\|_2\cos\theta \tag{9}$$

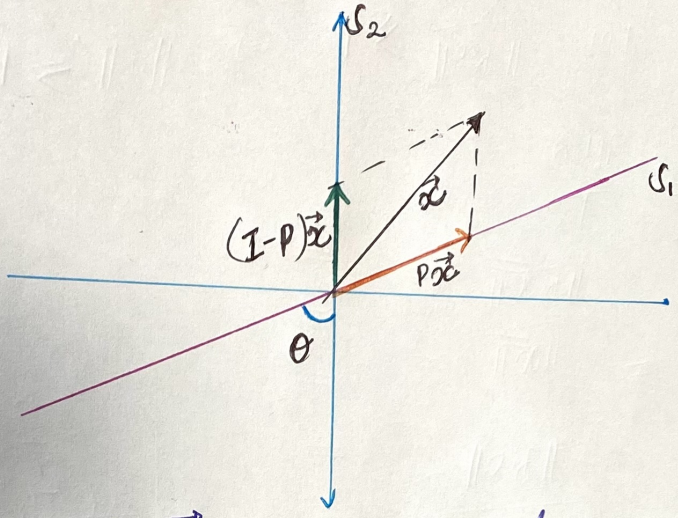
where θ is the angle between subspaces S_1 and S_2. Then use the definition of the matrix 2-norm as

$$\|P\|_2 = \max_{\mathbf{x}} \frac{\|P\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \tag{10}$$

and show that the maximum will be 1 if and only if $\theta = \pi/2$.

Solution

Let $\vec{x} \in \mathbb{R}^m$, x can be decomposed into unique components in subspaces $S_1 = \text{range}(P)$, and $S_2 = \text{range}(I-P)$ as shown in the diagrams below with an angle θ between the subspaces.

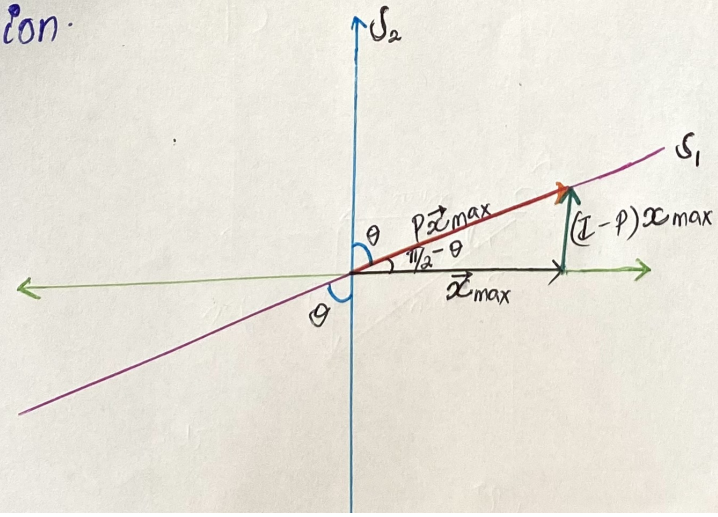


for this case $\vec{x}$ can be expressed as

$$\vec{x} = P\vec{x} + (I-P)\vec{x}$$

$\Rightarrow$ ~~when~~ for this case

$\|\vec{x}\| > \|P\vec{x}\|$ and therefore $\vec{x}$ is not maximum in that region.



from the diagram, we obtain

$$\|P\vec{x}_{max}\|_2^2 = \|\vec{x}_{max}\|_2^2 + \|(I-P)\vec{x}_{max}\|_2^2$$

where $\vec{x}_{max}$ is the maximum $\vec{x}$ ~~vector~~

$$\Rightarrow \frac{\|P\vec{x}_{max}\|_2^2}{\|\vec{x}_{max}\|_2^2} = 1 + \frac{\|(I-P)\vec{x}_{max}\|_2^2}{\|x_{max}\|_2^2}$$

But $$\max_{x} \frac{\|P\vec{x}\|_2}{\|\vec{x}\|_2} = \|P\|_2$$

$$\Rightarrow \|P\|_2^2 = 1 + \frac{\|(I-P)\vec{x}_{max}\|_2^2}{\|\vec{x}_{ma}\|_2^2}$$

$$\Rightarrow \|P\|_2 \geq 1$$

∴ For a non zero projector P, $\|P\|_2 \geq 1$

This can also be shown if we consider;

$$|\cos(\pi/2 - \theta)| = \frac{\|\vec{x}_{max}\|_2}{\|P\vec{x}_{max}\|_2}$$

But $\cos(\pi/2 - \theta) = \cos\pi/2 \cos\theta + \sin\pi/2 \sin\theta$

$\Rightarrow \cos(\pi/2 - \theta) = \sin\theta$

$$\Rightarrow \frac{\|Px_{max}\|_2}{\|x_{max}\|_2} = \frac{1}{|\sin\theta|}$$

$$\Rightarrow \|P\|_2 = \frac{1}{|\sin\theta|}$$

But $0 \leq |\sin\theta| \leq 1$

But taking $\theta \neq 0$

$\Rightarrow 0 < |\sin\theta \leq 1$

∴ $\|P\|_2 \geq 1$

An if $\theta = \pi/2$, we shall the have that

$\|P\|_2 = 1$

$\because \|P\|_2 = 1$ if and only if P is an orthogonal projection.

This can also be shown by:-

$$\vec{x} = P\vec{x} + (I-P)\vec{x}$$

$$\|\vec{x}\|_2^2 = \|(P\vec{x} + (I-P)\vec{x})^T (P\vec{x} + (I-P)\vec{x})$$

$$= (P\vec{x})^T P\vec{x} + ((I-P)\vec{x})^T (I-P)\vec{x} + 2(P\vec{x})^T (I-P)\vec{x}$$

$$= \|P\vec{x}\|_2^2 + \|(I-P)\vec{x}\|_2^2 + 2\|P\vec{x}\|_2^2 \|(I-P)x\|_2 \cos\theta$$

$$\Rightarrow \|\vec{x}\|_2^2 = \|P\vec{x}\|_2^2 + \|(I-P)\vec{x}\|_2^2 + 2\|P\vec{x}\|_2 \|(I-P)\vec{x}\|_2 \cos\theta$$

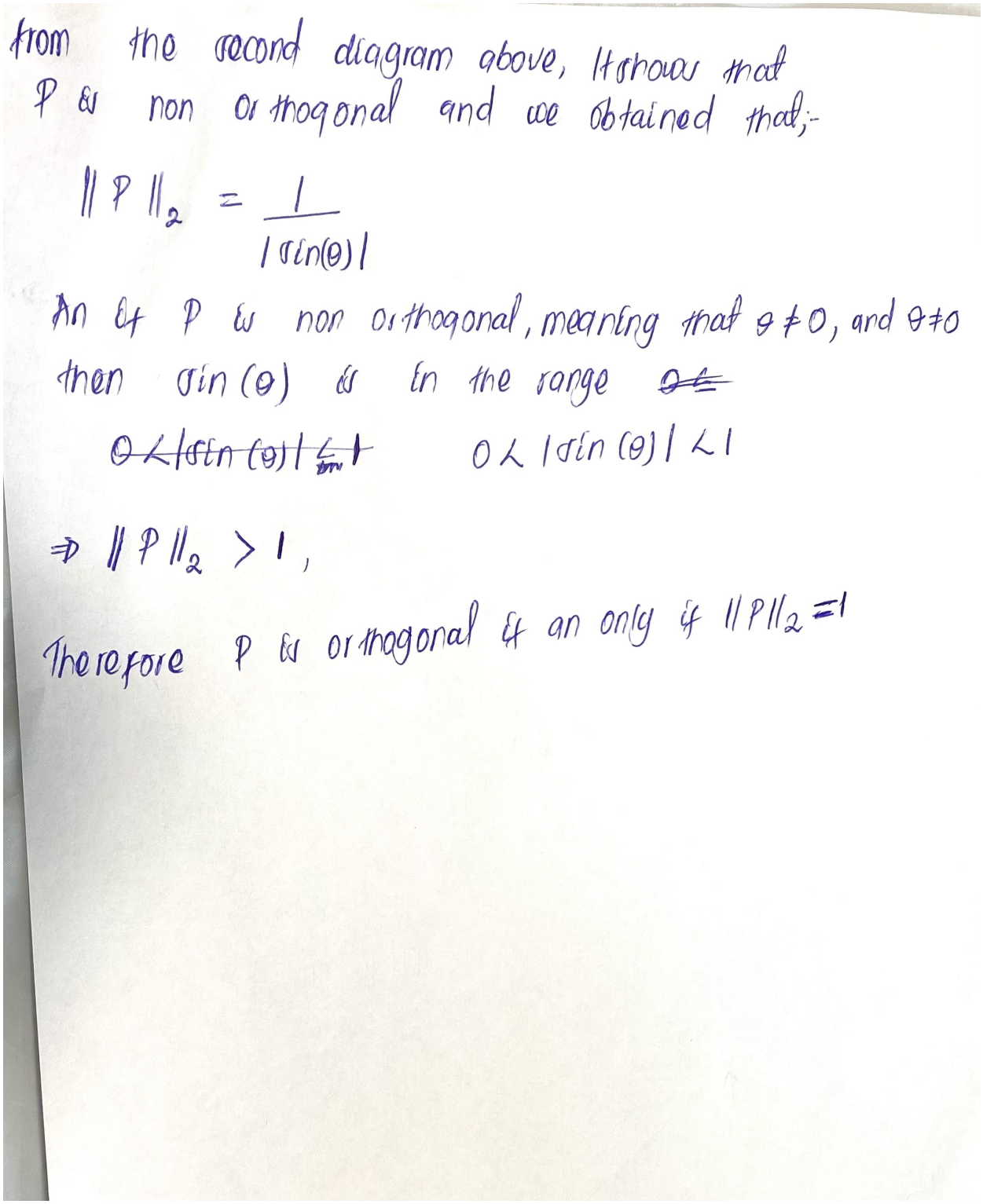
If $\theta = \pi/2$, then we have that

$$\|\vec{x}\|_2^2 = \|P\vec{x}\|_2^2 + \|(I-P)\vec{x}\|_2^2$$

$$\Rightarrow \frac{\|P\vec{x}\|_2^2}{\|\vec{x}\|_2^2} = 1 - \frac{\|(I-P)\vec{x}\|_2^2}{\|\vec{x}\|_2^2}$$

$\Rightarrow \|P\|_2 \leq 1$ But we showed that $\|Px\|_2 \geq 1$

$\Rightarrow \|P\|_2 = 1$ if and only if $\theta = \pi/2$



Problem #3

Modify the Householder code demonstrated in class to return the matrix Q as well as R. You may use the code below.

Problem

- Modify the code below to compute Q as well as R.
- Use your Householder reflections to obtain the QR decomposition for the matrix provided.
- Show that $QR = A$.
- Compare your R and Q to the decomposition returned from NumPy. You should get the same result.

Note: Do not try to use the approach described in TB. The problem is really much easier than that approach. Bascially, you will have to add one line in the loop below.

In [4]:
```
def display_mat(msg,A):
    print(msg)
    fstr = {'float' : "{:>10.4f}".format}
    with printoptions(formatter=fstr):
        display(A)
    print("")
```

In [5]:
```
def QR_House(A):
    m,n = A.shape

    R = A.copy()
    Q = eye(m)   # Update Q
    for j in range(n):
        x = R[j:m,j:j+1]
        I = eye(m-j)
        s = 1 if x[0] >= 0 else -1     # sign function, with sign(0)  = 1
        v = s*linalg.norm(x,2)*I[:,0:1] + x
        v = v/linalg.norm(v,2)
        F = I - 2*(v@v.T)
        R[j:m,j:n] = F@R[j:m,j:n]
        Q[j:] = F @ Q[j:]
    return Q.T,R
```

In [6]:
```
A = np.array([[1,1,2],[3,1,3],[1,1,1],[-1,0,4],[3,2,3]],dtype=float)


Q,R = QR_House(A)
q,r = qr(A,mode='complete')

display_mat("R = ",R)
display_mat("r (from NumPy) = ",r)

display_mat("Q = ",Q)
display_mat("q (from NumPy) = ",q)

display_mat("A = ",A)
display_mat("QR = ",Q@R)
```

```
R =
array([[   -4.5826,    -2.4004,    -3.7097],
       [    0.0000,     1.1127,     2.7817],
       [    0.0000,     0.0000,     4.1833],
```

```
       [   -0.0000,     0.0000,     0.0000],
       [    0.0000,     0.0000,    -0.0000]])
r (from NumPy) =
array([[   -4.5826,    -2.4004,    -3.7097],
       [    0.0000,     1.1127,     2.7817],
       [    0.0000,     0.0000,     4.1833],
       [    0.0000,     0.0000,     0.0000],
       [    0.0000,     0.0000,     0.0000]])
Q =
array([[   -0.2182,     0.4280,    -0.0000,    -0.4985,    -0.7216],
       [   -0.6547,    -0.5136,     0.4781,     0.1700,    -0.2241],
       [   -0.2182,     0.4280,    -0.2390,     0.8092,    -0.2392],
       [    0.2182,     0.4708,     0.8367,     0.0997,     0.1443],
       [   -0.6547,     0.3852,    -0.1195,    -0.2404,     0.5925]])
q (from NumPy) =
array([[   -0.2182,     0.4280,     0.0000,    -0.4985,    -0.7216],
       [   -0.6547,    -0.5136,     0.4781,     0.1700,    -0.2241],
       [   -0.2182,     0.4280,    -0.2390,     0.8092,    -0.2392],
       [    0.2182,     0.4708,     0.8367,     0.0997,     0.1443],
       [   -0.6547,     0.3852,    -0.1195,    -0.2404,     0.5925]])
A =
array([[    1.0000,     1.0000,     2.0000],
       [    3.0000,     1.0000,     3.0000],
       [    1.0000,     1.0000,     1.0000],
       [   -1.0000,     0.0000,     4.0000],
       [    3.0000,     2.0000,     3.0000]])
QR =
array([[    1.0000,     1.0000,     2.0000],
       [    3.0000,     1.0000,     3.0000],
       [    1.0000,     1.0000,     1.0000],
       [   -1.0000,    -0.0000,     4.0000],
       [    3.0000,     2.0000,     3.0000]])
```