In [37]:

```
import numpy as np
import scipy.linalg as la
from matplotlib import pyplot as plt
%matplotlib inline
```

The rls algorithm, according to the paper R is just taken to be the identity matrix

In [38]:

```
def rls(A,B):
    x = np.ones(A.shape[1])
    K = np.ones((A.shape[1],A.shape[1]))
    R = np.eye(B.size)
    for i in range(B.size):
        #import pdb;pdb.set_trace()
        a = A[:i+1]
        r = R[:i+1,:i+1]
        K = K - K.dot(a.T).dot(la.inv(r + a.dot(K).dot(a.T))).dot(a).dot(K)
        x = x - (K.dot(a.T).dot(la.inv(r))).dot(a.dot(x) - B[i])
    return x
```

For a size n, compare the error from RLS and OLS for 10 different random matricies, and take the average error for the two methods

In [39]:

```
def compare(n,repeat = 100):
    diffs = np.zeros((2,10))
    for i in xrange(repeat):
        A = np.random.rand(n,n-1)
        B = np.random.rand(n)

        xOls = la.lstsq(A,B)[0]
        diffOls = la.norm(A.dot(xOls) - B)

        xRls = rls(A,B)
        diffRls = la.norm(A.dot(xRls) - B)

        diffs[:,i] = (diffOls,diffRls)

    return np.mean(diffs,axis=1)
```

Generate the mean errors for several sizes of n

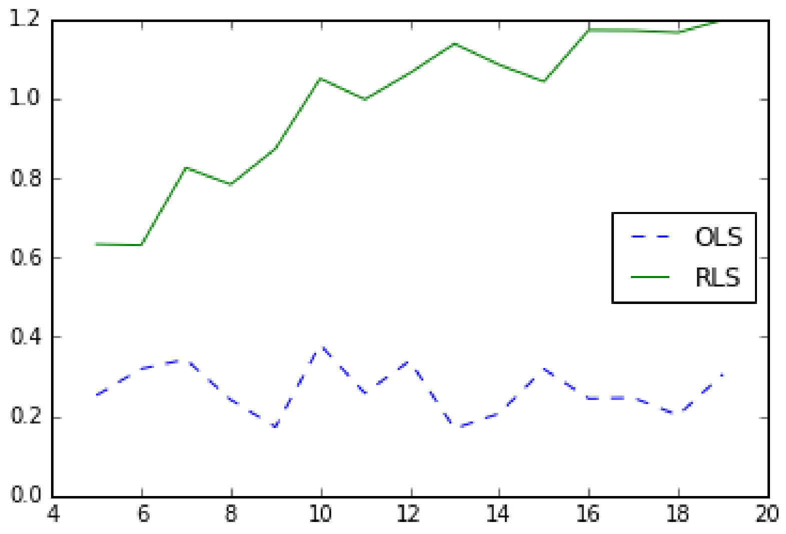
In [ ]:

```
diffs = np.array([compare(n) for n in np.arange(5,20)])
```

Plot the mean error for the two methods

```
In [49]:
```

```
plt.plot(np.arange(5,20),diffs[:,0],label="OLS",linestyle="--")
plt.plot(np.arange(5,20),diffs[:,1],label="RLS")
plt.legend(loc=7)
plt.show()
```



Apparently my RLS doesn't work as well as OLS, and gets worse with larger n, but at least its in the ballpark