```
In [1]:  # Alexander Hebert
         # ECE 6390
         # Computer Project #2
         # Pole assignment using controller type block companion form
         #  and state feedback

In [2]:  # Tested using Python v3.4 and IPython v2

In [3]:  # Import libraries

In [4]:  import numpy as np

In [5]:  import scipy

In [6]:  import sympy

In [7]:  from IPython.display import display

In [8]:  from sympy.interactive import printing

In [9]:  np.set_printoptions(precision=6)

In [10]: #np.set_printoptions(suppress=True)

In [11]: # Original system:

In [12]: A = np.loadtxt('A_ex1.txt')

In [13]: A

Out[13]: array([[ 1.38  , -0.2077,  6.715 , -5.676 ],
                [-0.5814, -4.29  ,  0.    ,  0.675 ],
                [ 1.067 ,  4.273 , -6.654 ,  5.893 ],
                [ 0.048 ,  4.273 ,  1.343 , -2.104 ]])

In [14]: n,nc = A.shape

In [15]: B = np.loadtxt('B_ex1.txt')

In [16]: B

Out[16]: array([[ 0.   ,  0.   ],
                [ 5.679,  0.   ],
                [ 1.136, -3.146],
                [ 1.136,  0.   ]])

In [17]: nr,m = B.shape

In [18]: # Compute eigenvalues/poles of A to determine system stability:

In [19]: A_eigvals, M = np.linalg.eig(A)
In [20]: A_eigvals

Out[20]: array([ 1.99096 ,  0.063508, -5.056574, -8.665894])

In [21]: # Two poles lie in the RHP and are unstable.

In [22]: A_eigvals_desired = np.array([-0.2,-0.5,A_eigvals[2],A_eigvals[3]])

In [23]: A_eigvals_desired

Out[23]: array([-0.2     , -0.5     , -5.056574, -8.665894])

In [24]: Lambda = np.diag(A_eigvals_desired)

In [25]: Lambda

Out[25]: array([[-0.2     ,  0.      ,  0.      ,  0.      ],
                [ 0.      , -0.5     ,  0.      ,  0.      ],
                [ 0.      ,  0.      , -5.056574,  0.      ],
                [ 0.      ,  0.      ,  0.      , -8.665894]])

In [26]: # Pole assignment using controller type block companion form
         #  and state feedback

In [27]: Bc = np.array([[0,0],[0,0],[1.,0],[0,1.]])
         Bc

Out[27]: array([[ 0.,  0.],
                [ 0.,  0.],
                [ 1.,  0.],
                [ 0.,  1.]])

In [28]: Phi_c = np.concatenate((B,A.dot(B)),1)
         Tc1 = np.dot(Bc.T,np.linalg.inv(Phi_c))
         Tc1

Out[28]: array([[ -7.111928e-03,  -7.113180e-03,   0.000000e+00,   3.555964e-02],
                [ -4.733667e-02,  -2.611878e-07,   0.000000e+00,   1.305709e-06]])

In [29]: Tc = np.concatenate((Tc1,Tc1.dot(A)),0)
         Tc

Out[29]: array([[ -7.111928e-03,  -7.113180e-03,   0.000000e+00,   3.555964e-02],
                [ -4.733667e-02,  -2.611878e-07,   0.000000e+00,   1.305709e-06],
                [ -3.971995e-03,   1.839390e-01,   0.000000e+00,  -3.925158e-02],
                [ -6.532438e-02,   9.838525e-03,  -3.178640e-01,   2.686800e-01]])

In [30]: Tc_inv = np.linalg.inv(Tc)
         Tc_inv

Out[30]: array([[  7.757000e-04,  -2.112539e+01,  -8.497769e-16,  -1.140029e-17],
                [  6.268667e+00,  -1.418335e+00,   5.679000e+00,  -7.569394e-17],
                [  2.502434e+01,   4.864526e-01,   1.136000e+00,  -3.146000e+00],
                [  2.937588e+01,  -4.508795e+00,   1.136000e+00,  -4.076722e-16]])
```

```
In [31]: Ac = Tc.dot(A).dot(Tc_inv)
         Ac
```

```
Out[31]: array([[  0.000000e+00,   2.775558e-17,   1.000000e+00,   2.123989e-18],
               [ -1.776357e-15,   2.220446e-16,  -5.551115e-17,   1.000000e+00],
               [ -1.243937e+00,   2.698279e+00,  -5.258809e+00,   2.497509e-01],
               [ -1.106169e+01,   1.954029e+01,  -1.383198e+00,  -6.409191e+00]])
```

```
In [32]: Ac2 = -1*Ac[2:4,0:2]
         Ac2
```

```
Out[32]: array([[  1.243937,  -2.698279],
               [ 11.061689, -19.540295]])
```

```
In [33]: Ac1 = -1*Ac[2:4,2:4]
         Ac1
```

```
Out[33]: array([[ 5.258809, -0.249751],
               [ 1.383198,  6.409191]])
```

```
In [34]: # Check Bc
         Tc.dot(B)
```

```
Out[34]: array([[  1.387779e-17,   0.000000e+00],
               [  2.117582e-22,   0.000000e+00],
               [  1.000000e+00,   0.000000e+00],
               [  5.551115e-17,   1.000000e+00]])
```

```
In [35]: # Calculations for Kc

         # (s+0.2)*(s+0.5) = s^2 + 0.7s + 0.1
         d1 = 0.7
         d2 = 0.1

         # (s+5.0566)*(s+8.6659) = s^2 + 13.7225s + 43.82
         d3 = 13.7225
         d4 = 43.82
```

```
In [36]: D1 = np.array([[d1,0],[0,d3]])
         D1
```

```
Out[36]: array([[  0.7    ,   0.    ],
               [  0.     ,  13.7225]])
```

```
In [37]: D2 = np.array([[d2,0],[0,d4]])
         D2
```

```
Out[37]: array([[  0.1 ,   0.  ],
               [  0.  ,  43.82]])
```

```
In [38]: Kc1 = D1 - Ac1
         Kc1
```

```
Out[38]: array([[-4.558809,  0.249751],
               [-1.383198,  7.313309]])
```

```
In [39]: Kc2 = D2 - Ac2
         Kc2
```

```
Out[39]: array([[ -1.143937,   2.698279],
               [-11.061689,  63.360295]])
```

```
In [40]: Kc = np.concatenate((Kc2,Kc1),1)
         Kc

Out[40]: array([[ -1.143937,   2.698279,  -4.558809,   0.249751],
                [-11.061689,  63.360295,  -1.383198,   7.313309]])

In [41]: K = Kc.dot(Tc)
         K

Out[41]: array([[-0.117799, -0.827949, -0.079387,  0.205369],
                [-3.392838, -0.103805, -2.324637,  1.625966]])

In [42]: np.linalg.norm(K)

Out[42]: 4.5075145258660738

In [43]: A_hat = A - B.dot(K)
         A_hat

Out[43]: array([[  1.38    ,  -0.2077  ,   6.715   ,  -5.676   ],
                [  0.087582,   0.411925,   0.450838,  -0.491291],
                [ -9.47305 ,   4.886981, -13.877126,  10.774988],
                [  0.18182 ,   5.213551,   1.433183,  -2.337299]])

In [44]: A_hat_eigvals, M_hat = np.linalg.eig(A_hat)
         A_hat_eigvals

Out[44]: array([-8.665897, -5.056603, -0.2     , -0.5     ])

In [45]: idx = A_hat_eigvals.argsort()[::-1]
         A_hat_eigvals = A_hat_eigvals[idx]
         A_hat_eigvals

Out[45]: array([-0.2     , -0.5     , -5.056603, -8.665897])

In [46]: M_hat = M_hat[:,idx]

In [47]: np.linalg.cond(M_hat)

Out[47]: 53.907192905412508

  In []:
```