In [1]:
```
# Alexander Hebert
# ECE 6390
# Computer Project #1
```

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

In []:
```
# Import libraries
```

In [3]:
```
import numpy as np
```

In [4]:
```
import scipy
```

In [5]:
```
import sympy
```

In [6]:
```
from IPython.display import display
```

In [7]:
```
from sympy.interactive import printing
```

In [8]:
```
printing.init_printing(use_latex='mathjax')
```

In [9]:
```
from __future__ import division
```

In [10]:
```
np.set_printoptions(precision=4)
```

In [11]:
```
np.set_printoptions(suppress=True)
```

In []:
```
# Original system:
```

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

In [13]:
```
A
```

Out[13]:
```
matrix([[   0.   ,    1.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [  -0.202,   -1.15 ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    1.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    1.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,   -2.36 ,  -13.6  ,  -12.8  ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            1.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    1.   ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,   -1.62 ,
           -9.4  ,   -9.15 ,    0.   ,    0.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    1.   ,    0.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    1.   ,    0.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    1.   ],
        [   0.   ,    0.   ,    0.   ,    0.   ,    0.   ,    0.   ,
            0.   ,    0.   , -188.   , -111.6  , -116.4  ,  -20.8  ]])
```

In [14]:
```
B = np.transpose(np.matrix(np.loadtxt('B.txt')))
```

In [15]:
```
B
```

Out[15]:
```
matrix([[ 0.    ,  0.    ],
        [ 1.0439,  4.1486],
        [ 0.    ,  0.    ],
        [ 0.    ,  0.    ],
        [-1.794 ,  2.6775],
        [ 0.    ,  0.    ],
        [ 0.    ,  0.    ],
        [ 1.0439,  4.1486],
        [ 0.    ,  0.    ],
        [ 0.    ,  0.    ],
        [ 0.    ,  0.    ],
        [-1.704 ,  2.6775]])
```

In [16]:
```
C = np.matrix(np.loadtxt('C.txt'))
```

In [17]:
```
C
```

Out[17]:
```
matrix([[  0.264,   0.806,  -1.42 , -15.   ,   0.   ,   0.   ,   0.   ,
           0.   ,   0.   ,   0.   ,   0.   ,   0.   ,   0.   ],
        [  0.   ,   0.   ,   0.   ,   0.   ,   0.   ,   4.9  ,   2.12 ,
           1.95 ,   9.35 ,  25.8  ,   7.14 ,   0.   ]])
```

In [18]:
```
D = np.zeros((2,2))
```

In [19]:
```
D
```

Out[19]:
```
array([[ 0.,  0.],
       [ 0.,  0.]])
```

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

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

In [21]:
```
A_eigvals
```

Out[21]:
```
array([ -0.2164+0.j    ,  -0.9336+0.j    , -11.6500+0.j    ,
        -0.2172+0.j    ,  -0.9328+0.j    ,  -8.0004+0.j    ,
        -0.2172+0.j    ,  -0.9325+0.j    , -10.9955+0.j    ,
        -9.0644+0.j    ,  -0.3700+1.3226j,  -0.3700-1.3226j])
```

In [22]:
```
# All eigenvalues/poles of A are in the LHP. Therefore the system is stabl
e.
```

In [23]:
```
# Sort eigenvalues and eigenvectors in descending order:
```

In [24]:
```
idx = A_eigvals.argsort()[::-1]
```

In [25]:
```
A_eigvals = A_eigvals[idx]
```

In [26]:
```
A_eigvals
```

Out[26]:
```
array([ -0.2164+0.j    ,  -0.2172+0.j    ,  -0.2172+0.j    ,
        -0.3700+1.3226j,  -0.3700-1.3226j,  -0.9325+0.j    ,
        -0.9328+0.j    ,  -0.9336+0.j    ,  -8.0004+0.j    ,
        -9.0644+0.j    , -10.9955+0.j    , -11.6500+0.j    ])
```

In [27]:
```
M = M[:,idx]
```

In []:
```
# Compute linear transformation Tc to get controller-type block companion
form.
```

In [28]:
```
gamma = 12/2
```

In [29]:
```
gamma
```

Out[29]: $$6.0$$

In [30]:
```
# gamma is an integer.
```

In [31]:
```
Bc = np.matrix([[0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [0,0],
                [1,0],
                [0,1]])
```

In [32]:
```
Bc
```

Out[32]:
```
matrix([[0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [1, 0],
        [0, 1]])
```

In [33]:
```
Tc1 = (Bc.T)*np.linalg.inv(np.concatenate(
(B,A*B,(A**2)*B,(A**3)*B,(A**4)*B,(A**5)*B),1))
```

In [34]:
```
Tc1
```

Out[34]:
```
matrix([[  5.6689,   1.448 ,   0.0041,  -0.0025,  -0.0001, -45.3482,
         -17.2531,  -1.448 ,  -0.0337,   0.0162,   0.003 ,   0.0001],
        [  3.6078,   0.9215,   0.0274,   0.0006,  -0.    , -28.8598,
         -10.9799,  -0.9215,  -0.2282,  -0.0331,  -0.0003,   0.    ]])
```

In [35]:
```
Tc = np.concatenate(
(Tc1,Tc1*A,Tc1*(A**2),Tc1*(A**3),Tc1*(A**4),Tc1*(A**5)),0)
```

In [36]:
```
Tc
```

Out[36]:
```
matrix([[   5.6689,    1.448 ,    0.0041,   -0.0025,   -0.0001,  -45.3482,
          -17.2531,   -1.448 ,   -0.0337,    0.0162,    0.003 ,    0.0001]
        ,
         [   3.6078,    0.9215,    0.0274,    0.0006,   -0.    ,  -28.8598,
          -10.9799,   -0.9215,   -0.2282,   -0.0331,   -0.0003,    0.    ]
        ,
         [  -0.2925,    4.0037,    0.0001,    0.0049,   -0.0017,    2.3458,
          -31.7368,   -4.0037,   -0.0115,   -0.0405,    0.0091,    0.0018]
        ,
         [  -0.1861,    2.548 ,    0.0001,    0.0279,    0.001 ,    1.4929,
          -20.1975,   -2.548 ,   -0.0073,   -0.2326,   -0.0376,   -0.0011]
        ,
         [  -0.8087,   -4.8968,    0.004 ,    0.0234,    0.0268,    6.4861,
           39.981 ,    4.8974,   -0.3335,   -0.2095,   -0.247 ,   -0.0278]
        ,
         [  -0.5147,   -3.1164,   -0.0025,   -0.0141,    0.0146,    4.1278,
           25.4441,    3.1167,    0.2031,    0.1133,   -0.1068,   -0.0152]
        ,
         [   0.9891,    4.8225,   -0.0633,   -0.3608,   -0.32  ,   -7.9338,
          -39.5496,   -4.8303,    5.2337,    2.7733,    3.031 ,    0.3321]
        ,
         [   0.6295,    3.0691,   -0.0345,   -0.2013,   -0.2012,   -5.0491,
          -25.1696,   -3.0741,    2.8532,    1.8968,    1.8798,    0.2089]
        ,
         [  -0.9741,   -4.5568,    0.7551,    4.2881,    3.7347,    7.8251,
           37.4714,    4.648 ,  -62.4292,  -31.8253,  -35.8796,   -3.8761]
        ,
         [  -0.62  ,   -2.9   ,    0.4749,    2.7024,    2.3746,    4.98  ,
           23.847 ,    2.958 ,  -39.2657,  -20.4556,  -22.4145,   -2.4645]
        ,
         [   0.9205,    4.2661,   -8.8139,  -50.037 ,  -43.5162,   -7.5297,
          -35.866 ,   -5.0577,  728.7022,  370.1408,  419.3498,   44.7427]
        ,
         [   0.5858,    2.715 ,   -5.604 ,  -31.8193,  -27.6922,   -4.792 ,
          -22.8253,   -3.2188,  463.3185,  235.768 ,  266.4076,   28.8462]
        ])
```

In [37]:
```
Tc_inv = np.linalg.inv(Tc)
```

In [38]:
```
Tc_inv
```

Out[38]:
```
matrix([[  76.9462,  -11.6521,  448.169 ,  -60.8902,  444.4161,  -26.8159,
           57.0493,   30.767 ,    1.0439,    4.1486,    0.    ,   -0.    ]
        ,
        [  -0.    ,    0.    ,   76.9462,  -11.6521,  448.169 ,  -60.8902,
          444.4161,  -26.8159,   57.0493,   30.767 ,    1.0439,    4.1486]
        ,
        [ -28.6997,   44.8675,  -15.4174,   22.8961,  -17.2783,   25.8555,
           -1.794 ,    2.6775,   -0.    ,    0.    ,   -0.    ,   -0.    ]
        ,
        [  -0.    ,   -0.    ,  -28.6997,   44.8675,  -15.4174,   22.8961,
          -17.2783,   25.8555,   -1.794 ,    2.6775,   -0.    ,    0.    ]
        ,
        [  -0.    ,   -0.    ,    0.    ,   -0.    ,  -28.6997,   44.8675,
          -15.4174,   22.8961,  -17.2783,   25.8555,   -1.794 ,    2.6775]
        ,
        [   9.5945,   -1.4529,   54.8332,   -7.4335,   48.6981,   -2.4218,
            1.0439,    4.1486,    0.    ,   -0.    ,    0.    ,   -0.    ]
        ,
        [  -0.    ,    0.    ,    9.5945,   -1.4529,   54.8332,   -7.4335,
           48.6981,   -2.4218,    1.0439,    4.1486,    0.    ,    0.    ]
        ,
        [  -0.    ,    0.    ,   -0.    ,    0.    ,    9.5945,   -1.4529,
           54.8332,   -7.4335,   48.6981,   -2.4218,    1.0439,    4.1486]
        ,
        [  -0.3452,    0.5424,   -1.9596,    3.0791,   -1.704 ,    2.6775,
           -0.    ,    0.    ,   -0.    ,    0.    ,   -0.    ,   -0.    ]
        ,
        [  -0.    ,    0.    ,   -0.3452,    0.5424,   -1.9596,    3.0791,
           -1.704 ,    2.6775,   -0.    ,   -0.    ,   -0.    ,   -0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,   -0.    ,   -0.3452,    0.5424,
           -1.9596,    3.0791,   -1.704 ,    2.6775,    0.    ,    0.    ]
        ,
        [  -0.    ,    0.    ,   -0.    ,    0.    ,    0.    ,   -0.    ,
           -0.3452,    0.5424,   -1.9596,    3.0791,   -1.704 ,    2.6775]
        ])
```

In [39]:
```
Ac = Tc*A*Tc_inv
```

In [40]:
```
Ac
```

Out[40]:
```
matrix([[  -0.    ,   -0.    ,    1.    ,   -0.    ,    0.    ,   -0.    ,
            0.    ,   -0.    ,    0.    ,    0.    ,   -0.    ,   -0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,    1.    ,   -0.    ,   -0.    ,
           -0.    ,    0.    ,   -0.    ,    0.    ,    0.    ,   -0.    ]
        ,
        [   0.    ,   -0.    ,   -0.    ,    0.    ,    1.    ,    0.    ,
            0.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,    0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,   -0.    ,   -0.    ,    1.    ,
            0.    ,    0.    ,    0.    ,   -0.    ,    0.    ,    0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,   -0.    ,
            1.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,    0.    ]
        ,
        [  -0.    ,    0.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,
           -0.    ,    1.    ,   -0.    ,   -0.    ,   -0.    ,   -0.    ]
        ,
        [   0.    ,   -0.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,
           -0.    ,    0.    ,    1.    ,   -0.    ,   -0.    ,   -0.    ]
        ,
        [  -0.    ,    0.    ,   -0.    ,    0.    ,   -0.    ,   -0.    ,
            0.    ,   -0.    ,    0.    ,    1.    ,    0.    ,    0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,   -0.    ,   -0.    ,    0.    ,
           -0.    ,   -0.    ,    0.    ,   -0.    ,    1.    ,    0.    ]
        ,
        [   0.    ,   -0.    ,    0.    ,   -0.    ,    0.    ,   -0.    ,
            0.    ,   -0.    ,    0.    ,    0.    ,    0.    ,    1.    ]
        ,
        [ -31.5119,   43.5244, -219.7319,  275.8889, -428.7516,  406.4136,
         -442.4287,  304.3612, -239.2004,  155.6859,  -31.5415,   15.0711]
        ,
        [   4.1826,  -10.3846,   12.139 ,  -63.2263,  -56.5344,  -81.2714,
         -122.6727,  -55.9729,  -62.8001,  -41.4416,   -6.1042,  -12.3585]
])
```

In [41]:
```
Bc_verify = Tc*B
```

In [42]:
```
Bc_verify
```

Out[42]:
```
matrix([[-0., -0.],
        [-0., -0.],
        [ 0.,  0.],
        [ 0.,  0.],
        [-0.,  0.],
        [-0.,  0.],
        [-0.,  0.],
        [ 0.,  0.],
        [-0.,  0.],
        [-0., -0.],
        [ 1.,  0.],
        [ 0.,  1.]])
```

In [43]:
```
Cc = C*Tc_inv
```

In [44]:
```
Cc
```

Out[44]:
```
matrix([[  61.0674,  -66.788 ,  632.7239, -730.992 ,  734.3456, -436.3131,
          634.9827, -405.1258,   73.1673,  -14.269 ,    0.8414,    3.3438]
        ,
        [  43.7857,   -2.0479,  261.7949,    3.2789,  304.622 ,   77.8893,
          157.3251,   91.7629,   85.0079,   23.1899,    2.0356,    8.0898]
        ])
```

In []:
```
# With the controller-type block companion form, the right MFD is obtained
.
```

In [45]:
```
A11 = -1*Ac[10:12,0:2]
```

In [46]:
```
A12 = -1*Ac[10:12,2:4]
```

In [47]:
```
A13 = -1*Ac[10:12,4:6]
```

In [48]:
```
A14 = -1*Ac[10:12,6:8]
```

In [49]:
```
A15 = -1*Ac[10:12,8:10]
```

In [50]:
```
A16 = -1*Ac[10:12,10:12]
```

In [51]:
```
A17 = np.identity(2)
```

In [52]:
```
A21 = Cc[0:2,0:2]
```

In [53]:
```
A22 = Cc[0:2,2:4]
```

In [54]:
```
A23 = Cc[0:2,4:6]
```

In [55]:
```
A24 = Cc[0:2,6:8]
```

In [56]:
```
A25 = Cc[0:2,8:10]
```

In [57]:
```
A26 = Cc[0:2,10:12]
```

In []:
```
# Model reduction using second Cauer method:
```

In [58]:
```
# order of 2nd Cauer reduced model
```

In [59]:
```
order = 2
```

In [60]:
```
n_end = gamma + 1
```

In [61]:
```
q = 1
```

In [62]:
```
for m in np.linspace(1,order*2,order*2):
    exec("H%d = (A%d%d)*np.linalg.inv(A%d%d)" %(m,m,1,m+1,1))
    exec("A%d%d = np.zeros((2,2))" %(m+1,n_end))

    for n in np.linspace(1,n_end-1,n_end-1):
        exec("A%d%d = (A%d%d) - (H%d)*(A%d%d)" %(m+2,n,m,n+1,m,m+1,n+1))


    if (np.equal(q,1)):
        q = 0
        n_end = n_end - 1

    q = q+1
```

In [63]:
```
H1
```

Out[63]:
```
matrix([[ 0.6577, -0.1977],
        [-0.1594,  0.1267]])
```

In [64]:
```
H2
```

Out[64]:
```
matrix([[-0.1178,  0.793 ],
        [ 0.6139,  2.3887]])
```

In [65]:
```
H3
```

Out[65]:
```
matrix([[-0.2786, -0.0307],
        [-0.0022, -0.8821]])
```

In [66]:
```
H4
```

Out[66]:
```
matrix([[ 3.0498, -0.6611],
        [-0.5713, -0.5007]])
```

In [67]:
```
A_hat = -1*np.concatenate((np.concatenate((H1*H2,H1*H4),1),
                           np.concatenate((H1*H2,(H1+H3)*H4),1)),0)
```

In [68]:
```
A_hat
```

Out[68]:
```
matrix([[ 0.1988, -0.0494, -2.1189,  0.3359],
        [-0.0966, -0.1764,  0.5585, -0.0419],
        [ 0.1988, -0.0494, -1.2867,  0.1363],
        [-0.0966, -0.1764,  0.0611, -0.4851]])
```

In [69]:
```
A_hat_eigvals, M_hat = np.linalg.eig(A_hat)
```

In [70]:
```
A_hat_eigvals
```

Out[70]:
```
array([-0.8617+0.j    , -0.4482+0.j    , -0.2197+0.0014j, -0.2197-0.0014j]
)
```

In [71]:
```
B_hat = np.concatenate((np.identity(2),np.identity(2)),0)
```

In [72]:
```
B_hat
```

Out[72]:
```
array([[ 1.,  0.],
       [ 0.,  1.],
       [ 1.,  0.],
       [ 0.,  1.]])
```

In [73]:
```
C_hat = np.concatenate((H2,H4),1)
```

In [74]:
```
C_hat
```

Out[74]:
```
matrix([[-0.1178,  0.793 ,  3.0498, -0.6611],
        [ 0.6139,  2.3887, -0.5713, -0.5007]])
```

In [75]:
```
np.savetxt('A_hat_cp1.csv', A_hat, delimiter=',')
```

In [76]:
```
np.savetxt('B_hat_cp1.csv', B_hat, delimiter=',')
```

In [77]:
```
np.savetxt('C_hat_cp1.csv', C_hat, delimiter=',')
```

In []:
```
# Model reduction using residualization:
```

In [81]:
```
# M computed previously (see above).
```

In [78]:
```
M
```

```
Out[78]: matrix([[ 0.9774+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.7309+0.j
                 ,
                   0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ],
                [-0.2115+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.6824+0.j
                 ,
                   0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    ,  0.9762+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.0000+0.j    , -0.6169+0.j    ,  0.0000+0.j
                 ,
                   0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.0073+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    , -0.2120+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.0000+0.j    ,  0.5755+0.j    ,  0.0000+0.j
                 ,
                   0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0855+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0460+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.0000+0.j    , -0.5368+0.j    ,  0.0000+0.j
                 ,
                   0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.9963+0.j
                 ],
                [ 0.0000+0.j    ,  0.9762+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    , -0.6171+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                  -0.0155+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    , -0.2120+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    ,  0.5755+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.1240+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    ,  0.0460+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000-0.j    , -0.5366+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                  -0.9922+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.2013+0.1884
               j,
                   0.2013-0.1884j,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000+0.j    ,  0.0013+0.j    , -0.0007+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.3237+0.1965
               j,
                  -0.3237-0.1965j,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
                 ,
                   0.0000+0.j    , -0.0121+0.j    ,  0.0082+0.j    ,  0.0000+0.j
                 ],
                [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.1401-0.5008
               j,
                  -0.1401+0.5008j,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
```

In [79]:
```
M_inv = np.linalg.inv(M)
```

In [80]:
```
M_inv
```

Out[80]:
```
matrix([[   1.3317+0.j    ,    1.4264+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ],
        [   0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    1.3726+0.j    ,
            1.6436+0.j    ,    0.1840+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ],
        [   0.0000+0.j    ,    0.0000+0.j    ,    1.3606+0.j    ,
            1.5754+0.j    ,    0.1252+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ],
        [   0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.5332-1.9206j,
           -1.1348-1.1372j,   -0.2446-0.1703j,   -0.0125-0.0075j],
        [   0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.5332+1.9206j,
           -1.1348+1.1372j,   -0.2446+0.1703j,   -0.0125+0.0075j],
        [   0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.5568+0.j    ,
            2.6337+0.j    ,    0.3205+0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ],
        [   0.0000-0.j    ,    0.0000-0.j    ,    0.5347+0.j    ,
            2.5079+0.j    ,    0.2113+0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ],
        [   0.4127+0.j    ,    1.9073+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ],
        [   0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,   -0.2375-0.j    ,
           -1.3482-0.j    ,   -1.1727-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ],
        [  -0.0000+0.j    ,   -0.0000+0.j    ,   -0.0000+0.j    ,
           -0.0000+0.j    ,   -0.0000+0.j    ,   -0.0000+0.j    ,
           -0.0000+0.j    ,   -0.0000+0.j    ,  104.0605+0.j    ,
           50.2920+0.j    ,   58.8806+0.j    ,    5.0173+0.j    ],
        [   0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,    0.0000+0.j    ,
            0.0000+0.j    ,    0.0000+0.j    ,  103.0909+0.j    ,
           51.8208+0.j    ,   59.1157+0.j    ,    6.0294+0.j    ],
        [   0.0000-0.j    ,    0.0000-0.j    ,   -0.2252-0.j    ,
           -1.2786-0.j    ,   -1.1118-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ,
            0.0000-0.j    ,    0.0000-0.j    ,    0.0000-0.j    ]])
```

In [82]:
```
Ad = M_inv*A*M
```

In [83]:
```
Ad
```

Out[83]:
```
matrix([[ -0.2164+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,  -0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,  -0.2172+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,  -0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,  -0.2172+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
          -0.3700+1.3226j,  -0.0000-0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
          -0.0000-0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000-0.j    ,  -0.3700-1.3226j,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
          -0.0000+0.j    ,   0.0000-0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,  -0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,  -0.9325+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,  -0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,  -0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
          -0.9328+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,  -0.9336+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,  -0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,  -8.0004+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000-0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
          -9.0644-0.j    ,  -0.0000+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000-0.j    ,  -0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000-0.j    , -10.9955+0.j    ,   0.0000+0.j    ],
         [  0.0000+0.j    ,   0.0000+0.j    ,  -0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    ,   0.0000+0.j    ,
           0.0000+0.j    ,   0.0000+0.j    , -11.6500+0.j    ]])
```

In [84]:
```
Bd = M_inv*B
```

In [85]:
```
Bd
```

Out[85]:
```
matrix([[  1.4890+0.j    ,   5.9176+0.j    ],
        [  0.1921+0.j    ,   0.7633+0.j    ],
        [ -0.2246+0.j    ,   0.3352+0.j    ],
        [  0.0212+0.0128j,  -0.0334-0.0202j],
        [  0.0212-0.0128j,  -0.0334+0.0202j],
        [  0.3346+0.j    ,   1.3296+0.j    ],
        [ -0.3791+0.j    ,   0.5658+0.j    ],
        [  1.9911+0.j    ,   7.9127+0.j    ],
        [ -1.2242+0.j    ,  -4.8651+0.j    ],
        [ -8.5494-0.j    ,  13.4338+0.j    ],
        [-10.2742-0.j    ,  16.1438+0.j    ],
        [  1.9946+0.j    ,  -2.9768+0.j    ]])
```

In [86]:
```
Cd = C*M
```

In [87]:
```
Cd
```

Out[87]:
```
matrix([[ 0.0876+0.j    ,  0.0000+0.j    ,  1.7937+0.j    ,  0.0000+0.j
          ,
          0.0000+0.j    ,  0.0000+0.j    , -7.7564+0.j    ,  0.3571+0.j
          ,
          0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -1.2724+0.j
          ],
        [ 0.0000+0.j    ,  4.4237+0.j    ,  0.0000+0.j    , -7.4695+3.2554
         j,
         -7.4695-3.2554j, -2.8504+0.j    ,  0.0000+0.j    ,  0.0000+0.j
          ,
         -1.7478+0.j    ,  0.4833+0.j    , -0.4412+0.j    ,  0.0000+0.j
          ]])
```

In []:
```
# Keep 6 eigenvalues/poles because 3rd and 4th are complex conjugate pair
```

In []:
```
# and need 1 more for gamma (ratio of states to inputs) to be an integer.
```

In [91]:
```
A11 = Ad[0:6,0:6]
```

In [92]:
```
A11
```

Out[92]:
```
matrix([[-0.2164+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
          ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    , -0.2172+0.j    ,  0.0000+0.j    ,  0.0000+0.j
          ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    , -0.2172+0.j    ,  0.0000+0.j
          ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.3700+1.3226
         j,
         -0.0000-0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000-0.j
          ,
         -0.3700-1.3226j,  0.0000+0.j    ],
        [ 0.0000+0.j    , -0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
          ,
          0.0000+0.j    , -0.9325+0.j    ]])
```

In [93]:
```
A12 = Ad[0:6,6:12]
```

In [94]:
```
A12
```

Out[94]:
```
matrix([[ 0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j, -0.-0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j, -0.+0.j,  0.-0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j]])
```

In [95]:
```
A21 = Ad[6:12,0:6]
```

In [96]:
```
A21
```

Out[96]:
```
matrix([[ 0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j, -0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j,  0.-0.j,  0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j,  0.+0.j,  0.-0.j, -0.+0.j,  0.+0.j],
        [ 0.+0.j,  0.+0.j, -0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j]])
```

In [97]:
```
A22 = Ad[6:12,6:12]
```

In [98]:
```
A22
```

Out[98]:
```
matrix([[ -0.9328+0.j,   0.0000+0.j,   0.0000+0.j,   0.0000+0.j,
           0.0000+0.j,   0.0000+0.j],
        [  0.0000+0.j,  -0.9336+0.j,   0.0000+0.j,   0.0000+0.j,
           0.0000+0.j,   0.0000+0.j],
        [  0.0000+0.j,   0.0000+0.j,  -8.0004+0.j,   0.0000+0.j,
           0.0000+0.j,   0.0000+0.j],
        [  0.0000+0.j,   0.0000+0.j,   0.0000+0.j,  -9.0644-0.j,
          -0.0000+0.j,   0.0000+0.j],
        [  0.0000+0.j,   0.0000+0.j,   0.0000+0.j,   0.0000-0.j,
         -10.9955+0.j,   0.0000+0.j],
        [  0.0000+0.j,   0.0000+0.j,   0.0000+0.j,   0.0000+0.j,
           0.0000+0.j, -11.6500+0.j]])
```

In [99]:
```
B1 = Bd[0:6,0:2]
```

In [100]:
```
B1
```

Out[100]:
```
matrix([[ 1.4890+0.j    ,  5.9176+0.j    ],
        [ 0.1921+0.j    ,  0.7633+0.j    ],
        [-0.2246+0.j    ,  0.3352+0.j    ],
        [ 0.0212+0.0128j, -0.0334-0.0202j],
        [ 0.0212-0.0128j, -0.0334+0.0202j],
        [ 0.3346+0.j    ,  1.3296+0.j    ]])
```

In [101]:
```
B2 = Bd[6:12,0:2]
```

In [102]:
```
B2
```

Out[102]:
```
matrix([[ -0.3791+0.j,   0.5658+0.j],
        [  1.9911+0.j,   7.9127+0.j],
        [ -1.2242+0.j,  -4.8651+0.j],
        [ -8.5494-0.j,  13.4338+0.j],
        [-10.2742-0.j,  16.1438+0.j],
        [  1.9946+0.j,  -2.9768+0.j]])
```

In [103]:
```
C1 = Cd[0:2,0:6]
```

In [104]:
```
C1
```

Out[104]:
```
matrix([[ 0.0876+0.j    ,  0.0000+0.j    ,  1.7937+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  4.4237+0.j    ,  0.0000+0.j    , -7.4695+3.255
4j,
         -7.4695-3.2554j, -2.8504+0.j    ]])
```

In [105]:
```
C2 = Cd[0:2,6:12]
```

In [106]:
```
C2
```

Out[106]:
```
matrix([[-7.7564+0.j,  0.3571+0.j,  0.0000+0.j,  0.0000+0.j,  0.0000+0.j,
         -1.2724+0.j],
        [ 0.0000+0.j,  0.0000+0.j, -1.7478+0.j,  0.4833+0.j, -0.4412+0.j,
          0.0000+0.j]])
```

In [107]:
```
Ar = A11 - A12*np.linalg.inv(A22)*A21
```

In [108]:
```
Ar
```

Out[108]:
```
matrix([[-0.2164+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    , -0.2172+0.j    ,  0.0000+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    , -0.2172+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    , -0.3700+1.322
6j,
         -0.0000-0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j    ,  0.0000-0.j
  ,
         -0.3700-1.3226j,  0.0000+0.j    ],
        [ 0.0000+0.j    , -0.0000+0.j    ,  0.0000+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    , -0.9325+0.j    ]])
```

In [109]:
```
Br = B1 - A12*np.linalg.inv(A22)*B2
```

In [110]:
```
Br
```

Out[110]:
```
matrix([[ 1.4890+0.j    ,  5.9176+0.j    ],
        [ 0.1921+0.j    ,  0.7633+0.j    ],
        [-0.2246+0.j    ,  0.3352+0.j    ],
        [ 0.0212+0.0128j, -0.0334-0.0202j],
        [ 0.0212-0.0128j, -0.0334+0.0202j],
        [ 0.3346+0.j    ,  1.3296+0.j    ]])
```

In [111]:
```
Cr = C1 - C2*np.linalg.inv(A22)*A21
```

In [112]:
```
Cr
```

Out[112]:
```
matrix([[ 0.0876+0.j    ,  0.0000+0.j    ,  1.7937+0.j    ,  0.0000+0.j
  ,
          0.0000+0.j    ,  0.0000+0.j    ],
        [ 0.0000+0.j    ,  4.4237+0.j    ,  0.0000+0.j    , -7.4695+3.255
4j,
         -7.4695-3.2554j, -2.8504+0.j    ]])
```

In [113]:
```
Dr = D - C2*np.linalg.inv(A22)*B2
```

In [114]:
```
Dr
```

Out[114]:
```
matrix([[ 3.6961+0.j, -1.3536+0.j],
        [ 0.2238-0.j,  1.1313+0.j]])
```

In [115]:
```
np.savetxt('Ar_cp1.csv', Ar, delimiter=',')
```

In [116]:
```
np.savetxt('Br_cp1.csv', Br, delimiter=',')
```

In [117]:
```
np.savetxt('Cr_cp1.csv', Cr, delimiter=',')
```

In [118]:
```
np.savetxt('Dr_cp1.csv', Dr, delimiter=',')
```

In [119]:
```
# The reduced model from residualization is 6x6.
```

In [120]:
```
# Therefore, the second Cauer method will be applied again.
```

In []:
```
# Apply second Cauer method to reduced model from residualization:
```

In [121]:
```
gamma = 6/2
```

In [122]:
```
gamma
```

Out[122]: $$3.0$$

In [123]:
```
# gamma is an integer.
```

In [124]:
```
Bc = np.matrix([[0,0],
                [0,0],
                [0,0],
                [0,0],
                [1,0],
                [0,1]])
```

In [125]:
```
Bc
```

Out[125]:
```
matrix([[0, 0],
        [0, 0],
        [0, 0],
        [0, 0],
        [1, 0],
        [0, 1]])
```

In [132]:
```
Tc1 = (Bc.T)*np.linalg.inv(np.concatenate(
(Br,Ar*Br,(Ar**2)*Br),1))
```

In [133]:
```
Tc1
```

Out[133]:
```
matrix([[  334.1623+0.j    , -2593.5049+0.j    ,    -1.8259+0.j    ,
            -6.4853+5.0194j,    -6.4853-5.0194j,     1.6533+0.j    ],
        [  212.6657+0.j    , -1650.5170-0.j    ,     0.4594-0.j    ,
             1.6319-1.263j ,     1.6319+1.263j ,     1.0522-0.j    ]])
```

In [135]:
```
Tc = np.concatenate((Tc1,Tc1*Ar,Tc1*(Ar**2)),0)
```

In [136]:
```
Tc
```

Out[136]:
```
matrix([[  334.1623 +0.j    , -2593.5049 +0.j    ,    -1.8259 +0.j    ,
            -6.4853 +5.0194j,    -6.4853 -5.0194j,     1.6533 +0.j    ],
        [  212.6657 +0.j    , -1650.5170 -0.j    ,     0.4594 -0.j    ,
             1.6319 -1.263j ,     1.6319 +1.263j ,     1.0522 -0.j    ],
        [  -72.2983 +0.j    ,   563.1846 -0.j    ,     0.3965 -0.j    ,
            -4.2390-10.435j ,    -4.2390+10.435j ,    -1.5417 -0.j    ],
        [  -46.0117 -0.j    ,   358.4129 +0.j    ,    -0.0998 +0.j    ,
             1.0667 +2.6257j,     1.0667 -2.6257j,    -0.9811 +0.j    ],
        [   15.6422 +0.j    ,  -122.2966 +0.j    ,    -0.0861 +0.j    ,
            15.3701 -1.7453j,    15.3701 +1.7453j,     1.4376 +0.j    ],
        [    9.9549 +0.j    ,   -77.8301 -0.j    ,     0.0217 -0.j    ,
            -3.8675 +0.4392j,    -3.8675 -0.4392j,     0.9149 -0.j    ]])
```

In [137]:
```
Tc_inv = np.linalg.inv(Tc)
```

In [138]:
```
Tc_inv
```

Out[138]:
```
matrix([[ 0.3015-0.j    ,  1.1983+0.j    ,  1.7118-0.j    ,  6.8030+0.j
  ,
          1.4890-0.j    ,  5.9176+0.j    ],
        [ 0.0387-0.j    ,  0.1540+0.j    ,  0.2207-0.j    ,  0.8769+0.j
  ,
          0.1921-0.j    ,  0.7633+0.j    ],
        [-0.4183-0.j    ,  0.6537+0.j    , -0.1675+0.j    ,  0.2429-0.j
  ,
         -0.2246+0.j    ,  0.3352+0.j    ],
        [-0.0020+0.0071j,  0.0031-0.0112j, -0.0045+0.0356j,  0.0071-0.056
j ,
          0.0212+0.0128j, -0.0334-0.0202j],
        [-0.0020-0.0071j,  0.0031+0.0112j, -0.0045-0.0356j,  0.0071+0.056
j ,
          0.0212-0.0128j, -0.0334+0.0202j],
        [ 0.0157-0.j    ,  0.0625+0.j    ,  0.1450-0.j    ,  0.5764+0.j
  ,
          0.3346-0.j    ,  1.3296+0.j    ]])
```

In [139]:
```
Arc = Tc*Ar*Tc_inv
```

In [140]:
```
Arc
```

Out[140]:
```
matrix([[ 0.0000+0.j, -0.0000-0.j,  1.0000-0.j,  0.0000+0.j, -0.0000-0.j,
          -0.0000+0.j],
        [ 0.0000-0.j, -0.0000+0.j,  0.0000-0.j,  1.0000+0.j, -0.0000-0.j,
          -0.0000+0.j],
        [-0.0000+0.j,  0.0000-0.j,  0.0000+0.j,  0.0000-0.j,  1.0000+0.j,
           0.0000-0.j],
        [ 0.0000+0.j,  0.0000+0.j,  0.0000-0.j,  0.0000+0.j,  0.0000-0.j,
           1.0000-0.j],
        [-0.3060-0.j,  0.4119+0.j, -1.5948-0.j,  1.7969+0.j, -1.0731+0.j,
          -0.4603-0.j],
        [ 0.0660+0.j, -0.1475-0.j,  0.2878+0.j, -0.9034-0.j, -0.0737-0.j,
          -1.2502+0.j]])
```

In [141]:
```
Brc_verify = Tc*Br
```

In [143]:
```
Brc_verify
```

Out[143]:
```
matrix([[ 0.+0.j,  0.+0.j],
        [-0.-0.j, -0.+0.j],
        [-0.-0.j, -0.-0.j],
        [-0.+0.j, -0.+0.j],
        [ 1.+0.j, -0.-0.j],
        [-0.-0.j,  1.-0.j]])
```

In [144]:
```
Crc = Cr*Tc_inv
```

In [145]:
```
Crc
```

Out[145]:
```
matrix([[-0.7239-0.j,  1.2775+0.j, -0.1506+0.j,  1.0316-0.j, -0.2724+0.j,
           1.1196+0.j],
        [ 0.1098-0.j,  0.5297+0.j,  0.3981-0.j,  2.4950+0.j, -0.5048+0.j,
           0.2165+0.j]])
```

In []:
```
# With the controller-type block companion form, the right MFD is obtained
.
```

In [154]:
```
Ar11 = Arc[4:6,0:2]
```

In [155]:
```
Ar11
```

Out[155]:
```
matrix([[-0.3060-0.j,  0.4119+0.j],
        [ 0.0660+0.j, -0.1475-0.j]])
```

In [156]:
```
Ar12 = Arc[4:6,2:4]
```

In [157]:
```
Ar12
```

Out[157]:
```
matrix([[-1.5948-0.j,  1.7969+0.j],
        [ 0.2878+0.j, -0.9034-0.j]])
```

In [158]:
```
Ar13 = Arc[4:6,4:6]
```

In [159]:
```
Ar13
```

Out[159]:
```
matrix([[-1.0731+0.j, -0.4603-0.j],
        [-0.0737-0.j, -1.2502+0.j]])
```

In [160]:
```
Ar14 = np.identity(2)
```

In [161]:
```
Ar14
```

Out[161]:
```
array([[ 1.,  0.],
       [ 0.,  1.]])
```

In [162]:
```
Ar21 = Crc[0:2,0:2]
```

In [163]:
```
Ar21
```

Out[163]:
```
matrix([[-0.7239-0.j,  1.2775+0.j],
        [ 0.1098-0.j,  0.5297+0.j]])
```

In [164]:
```
Ar22 = Crc[0:2,2:4]
```

In [165]:
```
Ar22
```

Out[165]:
```
matrix([[-0.1506+0.j,  1.0316-0.j],
        [ 0.3981-0.j,  2.4950+0.j]])
```

In [166]:
```
Ar23 = Crc[0:2,4:6]
```

In [167]:
```
Ar23
```

Out[167]:
```
matrix([[-0.2724+0.j,  1.1196+0.j],
        [-0.5048+0.j,  0.2165+0.j]])
```

In []:
```
# Model reduction using second Cauer method:
```

In [168]:
```
# order of 2nd Cauer reduced model
```

In [169]:
```
order = 2
```

In [170]:
```
order
```

Out[170]: $$2$$

In [171]:
```
n_end = gamma + 1
```

In [172]:
```
n_end
```

Out[172]: $$4.0$$

In [173]:
```
q = 1
```

In [174]:
```
q
```

Out[174]: $$1$$

In [175]:
```
for m in np.linspace(1,order*2,order*2):
    exec("Hr%d = (Ar%d%d)*np.linalg.inv(Ar%d%d)" %(m,m,1,m+1,1))
    exec("Ar%d%d = np.zeros((2,2))" %(m+1,n_end))

    for n in np.linspace(1,n_end-1,n_end-1):
        exec("Ar%d%d = (Ar%d%d) - (Hr%d)*(Ar%d%d)" %(m+2,n,m,n+1,m,m+1,n+
1))



    if (np.equal(q,1)):
        q = 0
        n_end = n_end - 1

    q = q+1
```

In [176]:
```
Hr1
```

Out[176]:
```
matrix([[ 0.3959+0.j, -0.1770-0.j],
        [-0.0976-0.j, -0.0429+0.j]])
```

In [177]:
```
Hr2
```

Out[177]:
```
matrix([[ 0.2723-0.j, -1.1200+0.j],
        [-0.4715-0.j, -2.0022-0.j]])
```

In [178]:
```
Hr3
```

Out[178]:
```
matrix([[-15926.4016-0.j,      3.2082-0.j],
        [  5201.2473+0.j,     -0.4997+0.j]])
```

In [179]:
```
Hr4
```

Out[179]:
```
matrix([[-0.0000-0.j,  0.0002-0.j],
        [ 0.3372+0.j, -0.1000-0.j]])
```

In [180]:
```
Ar_hat = -1*np.concatenate((np.concatenate((Hr1*Hr2,Hr1*Hr4),1),
                            np.concatenate((Hr1*Hr2,(Hr1+Hr3)*Hr4),1)),0)
```

In [181]:
```
Ar_hat
```

Out[181]:
```
matrix([[-0.1913-0.j,  0.0890-0.j,  0.0597+0.j, -0.0178-0.j],
        [ 0.0064+0.j, -0.1953+0.j,  0.0145+0.j, -0.0043-0.j],
        [-0.1913-0.j,  0.0890-0.j, -1.6693-0.j,  2.9216+0.j],
        [ 0.0064+0.j, -0.1953+0.j,  0.3943+0.j, -0.9094-0.j]])
```

In [182]:
```
Ar_hat_eigvals, Mr_hat = np.linalg.eig(Ar_hat)
```

In [183]:
```
Ar_hat_eigvals
```

Out[183]:
```
array([-2.4262-0.j    , -0.1609-0.0736j, -0.1609+0.0736j, -0.2172+0.j
])
```

In [184]:
```
Br_hat = np.concatenate((np.identity(2),np.identity(2)),0)
```

In [185]:
```
Br_hat
```

Out[185]:
```
array([[ 1.,  0.],
       [ 0.,  1.],
       [ 1.,  0.],
       [ 0.,  1.]])
```

In [186]:
```
Cr_hat = np.concatenate((Hr2,Hr4),1)
```

In [187]:
```
Cr_hat
```

Out[187]:
```
matrix([[ 0.2723-0.j, -1.1200+0.j, -0.0000-0.j,  0.0002-0.j],
        [-0.4715-0.j, -2.0022-0.j,  0.3372+0.j, -0.1000-0.j]])
```

In [188]:
```
np.savetxt('Ar_hat_cp1.csv', Ar_hat, delimiter=',')
```

In [189]:
```
np.savetxt('Br_hat_cp1.csv', Br_hat, delimiter=',')
```

In [190]:
```
np.savetxt('Cr_hat_cp1.csv', Cr_hat, delimiter=',')
```

In []:
```

```