In [1]:
```
# Alexander Hebert
# ECE 6390
# Computer Project #3
```

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]:
```
import itertools
```

In [8]:
```
from MatrixSignFunction import msf
```

In [9]:
```
from IPython.display import display
```

In [10]:
```
from sympy.interactive import printing
```

In [11]:
```
np.set_printoptions(linewidth=200,
                    formatter={'all':lambda x: format(x,'10.5f')})
```

In [12]:
```
#np.set_printoptions(suppress=True)
```

In [13]:
```
# Original system:
```

In [14]:
```
A = np.loadtxt('A.txt')
print(A)
```

```
[[    0.00000     1.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000
     0.00000]
 [    0.00000    -0.11323    -0.98109   -11.84700   -11.84700   -63.08000   -34.33900   -34.33900   -27.64500
     0.00000]
 [  324.12100    -1.17550   -29.10100     0.12722     2.83448  -967.73000  -678.14000  -678.14000     0.00000
  -129.29000]
 [ -127.30000     0.46167    11.42940    -1.03790    13.12370   380.07900   266.34100   266.34100     0.00000
  1054.85000]
 [ -186.05000     0.67475    16.70450     0.86092   -17.06800   555.50200   389.26800   389.26800     0.00000
  -874.92000]
 [  341.91700     1.09173  1052.75000   756.46500   756.46500   -29.77400     0.16507     3.27626     0.00000
     0.00000]
 [  -30.74800    -0.09817   -94.67400   -68.02900   -68.02900     2.67753    -2.65580     4.88497     0.00000
     0.00000]
 [ -302.36000    -0.96543  -930.96000  -668.95000  -668.95000    26.32920     2.42028    -9.56030     0.00000
     0.00000]
 [    0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000    -1.66670
     0.00000]
 [    0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000
   -10.00000]]
```

In [15]:
```
n,nc = A.shape
```

In [16]:
```
B = np.loadtxt('B.txt')
print(B)
```

```
[[    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    0.00000     0.00000]
 [    1.66667     0.00000]
 [    0.00000    10.00000]]
```

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

In [18]:
```
C = np.loadtxt('C.txt')
print(C)
```

```
[[   1.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000]
 [  -0.49134    0.00000   -0.63203    0.00000    0.00000   -0.20743    0.00000    0.00000    0.00000
    0.00000]]
```

In [19]:
```
D = np.zeros((2,2))
print(D)
```

```
[[   0.00000    0.00000]
 [   0.00000    0.00000]]
```

In [20]:
```
# Compute eigenvalues/poles of A to determine system stability:
```

In [21]:
```
A_eigvals, M = np.linalg.eig(A)
```

In [22]:
```
# Sort eigenvalues in descending order
idx = A_eigvals.argsort()[::-1]
A_eigvals = A_eigvals[idx]
print(A_eigvals)
```

```
[-0.23448+0.00000j -0.34915+6.34401j -0.34915-6.34401j -1.04205+0.00000j -1.66670+0.00000j -10.00000
+0.00000j -10.74614+0.00000j -17.66419+0.00000j -29.46253+313.93671j -29.46253-313.93671j]
```

In [23]:
```
# All poles are stable.
```

In [24]:
```
# Sort eigenvectors
M = M[:,idx]
```

In [25]:
```
# Mean of eigenvalues
gamma = sum(np.real(A_eigvals)) / n
print(gamma)
```

```
-10.097693
```

In [26]:
```
A_hat = A - gamma * np.eye(n)
print(A_hat)
```

```
[[  10.09769    1.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000]
 [   0.00000    9.98446   -0.98109  -11.84700  -11.84700  -63.08000  -34.33900  -34.33900  -27.64500
    0.00000]
 [ 324.12100   -1.17550  -19.00331    0.12722    2.83448 -967.73000 -678.14000 -678.14000    0.00000
 -129.29000]
 [-127.30000    0.46167   11.42940    9.05979   13.12370  380.07900  266.34100  266.34100    0.00000
 1054.85000]
 [-186.05000    0.67475   16.70450    0.86092   -6.97031  555.50200  389.26800  389.26800    0.00000
 -874.92000]
 [ 341.91700    1.09173 1052.75000  756.46500  756.46500  -19.67631    0.16507    3.27626    0.00000
    0.00000]
 [ -30.74800   -0.09817  -94.67400  -68.02900  -68.02900    2.67753    7.44189    4.88497    0.00000
    0.00000]
 [-302.36000   -0.96543 -930.96000 -668.95000 -668.95000   26.32920    2.42028    0.53739    0.00000
    0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    8.43099
    0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    0.09769]]
```

In [27]:
```
A_hat_eigvals, M_hat = np.linalg.eig(A_hat)
```

In [28]:
```
idx2 = A_hat_eigvals.argsort()[::-1]
A_hat_eigvals = A_hat_eigvals[idx2]
print(A_hat_eigvals)
```

```
[9.86321+0.00000j 9.74855+6.34401j 9.74855-6.34401j 9.05564+0.00000j 8.43099+0.00000j 0.09769+0.0000
0j -0.64845+0.00000j -7.56650+0.00000j -19.36484+313.93671j -19.36484-313.93671j]
```

In [29]:
```
# There are six dominant (unstable) eigenvalues of A_hat.
# Therefore, the top six eigenvalues are selected
# for the reduced order model.
```

In [30]:
```
# Compute sign(A_hat)
eps = 1.0e-7
maxiter = 100
sign_A_hat, jp1, flag = msf(A_hat,eps,maxiter)
print('j = %d' %jp1)
print('flag = %d' %flag)
```

```
j = 17
flag = 1
```

In [31]:
```
print(sign_A_hat)
```

```
[[   1.03870   -0.00336    0.02278    0.01414    0.03112   -0.11007   -0.06049   -0.11721   -0.00972
     2.85405]
 [  -0.54195    1.03876   -1.01065   -0.68724   -0.98331    0.85785    0.40622    1.01189    0.10912
   -43.80039]
 [  -1.84661   -0.00187   -5.01676   -4.33066   -4.03548   -0.13851   -0.24001   -0.13494    0.00303
    32.96137]
 [   1.52237   -0.04730    5.23084    4.63410    5.25086    0.17573    0.29052    0.17014   -0.09360
   189.82583]
 [   0.20288    0.05275    0.36461    0.40344   -0.63261   -0.05607   -0.07666   -0.05317    0.09759
  -237.34365]
 [   1.95411    0.03022    0.40569    0.43978    0.41876   -4.12085   -3.74582   -3.14521    0.09592
   -11.81757]
 [  -0.96414    0.07184    0.00472    0.01879   -0.00164    3.21693    2.86791    3.24639    0.21814
   -23.72170]
 [  -0.77019   -0.11709   -0.39984   -0.45331   -0.40735    1.18904    1.45797    0.19074   -0.35967
    40.52374]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    1.00000
     0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
     1.00000]]
```

In [32]:
```
sign_pos_A_hat = 0.5*(sign_A_hat + np.identity(n))
print(sign_pos_A_hat)
```

```
[[   1.01935   -0.00168    0.01139    0.00707    0.01556   -0.05504   -0.03024   -0.05861   -0.00486
     1.42703]
 [  -0.27098    1.01938   -0.50533   -0.34362   -0.49165    0.42893    0.20311    0.50595    0.05456
   -21.90020]
 [  -0.92330   -0.00094   -2.00838   -2.16533   -2.01774   -0.06925   -0.12000   -0.06747    0.00152
    16.48068]
 [   0.76119   -0.02365    2.61542    2.81705    2.62543    0.08787    0.14526    0.08507   -0.04680
    94.91292]
 [   0.10144    0.02637    0.18230    0.20172    0.18369   -0.02803   -0.03833   -0.02658    0.04879
  -118.67183]
 [   0.97705    0.01511    0.20284    0.21989    0.20938   -1.56042   -1.87291   -1.57261    0.04796
    -5.90878]
 [  -0.48207    0.03592    0.00236    0.00940   -0.00082    1.60846    1.93396    1.62320    0.10907
   -11.86085]
 [  -0.38509   -0.05854   -0.19992   -0.22665   -0.20368    0.59452    0.72898    0.59537   -0.17983
    20.26187]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    1.00000
     0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
     1.00000]]
```

In [33]:
```
np.trace(sign_pos_A_hat)
```

Out[33]: 6.0

In [34]:
```
# Select eigenvectors from sign_pos_A_hat
m_hat_1 = np.concatenate((sign_pos_A_hat[:,0:4],
                          sign_pos_A_hat[:,8:]), 1)
```

In [35]:
```
sign_neg_A_hat = 0.5*(np.identity(n) - sign_A_hat)
print(sign_neg_A_hat)
```

```
[[  -0.01935    0.00168   -0.01139   -0.00707   -0.01556    0.05504    0.03024    0.05861    0.00486
   -1.42703]
 [   0.27098   -0.01938    0.50533    0.34362    0.49165   -0.42893   -0.20311   -0.50595   -0.05456
   21.90020]
 [   0.92330    0.00094    3.00838    2.16533    2.01774    0.06925    0.12000    0.06747   -0.00152
  -16.48068]
 [  -0.76119    0.02365   -2.61542   -1.81705   -2.62543   -0.08787   -0.14526   -0.08507    0.04680
  -94.91292]
 [  -0.10144   -0.02637   -0.18230   -0.20172    0.81631    0.02803    0.03833    0.02658   -0.04879
  118.67183]
 [  -0.97705   -0.01511   -0.20284   -0.21989   -0.20938    2.56042    1.87291    1.57261   -0.04796
    5.90878]
 [   0.48207   -0.03592   -0.00236   -0.00940    0.00082   -1.60846   -0.93396   -1.62320   -0.10907
   11.86085]
 [   0.38509    0.05854    0.19992    0.22665    0.20368   -0.59452   -0.72898    0.40463    0.17983
  -20.26187]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000    0.00000
    0.00000]]
```

In [36]:
```
np.trace(sign_neg_A_hat)
```

Out[36]: 4.0

In [37]:
```
# Select eigenvectors from sign_neg_A_hat
m_hat_2 = sign_neg_A_hat[:,0:4]
```

In [38]:
```python
M_msf = np.concatenate((m_hat_1, m_hat_2), 1)
print(M_msf)
```

```
[[   1.01935   -0.00168    0.01139    0.00707   -0.00486    1.42703   -0.01935    0.00168   -0.01139
    -0.00707]
 [  -0.27098    1.01938   -0.50533   -0.34362    0.05456  -21.90020    0.27098   -0.01938    0.50533
     0.34362]
 [  -0.92330   -0.00094   -2.00838   -2.16533    0.00152   16.48068    0.92330    0.00094    3.00838
     2.16533]
 [   0.76119   -0.02365    2.61542    2.81705   -0.04680   94.91292   -0.76119    0.02365   -2.61542
    -1.81705]
 [   0.10144    0.02637    0.18230    0.20172    0.04879 -118.67183   -0.10144   -0.02637   -0.18230
    -0.20172]
 [   0.97705    0.01511    0.20284    0.21989    0.04796   -5.90878   -0.97705   -0.01511   -0.20284
    -0.21989]
 [  -0.48207    0.03592    0.00236    0.00940    0.10907  -11.86085    0.48207   -0.03592   -0.00236
    -0.00940]
 [  -0.38509   -0.05854   -0.19992   -0.22665   -0.17983   20.26187    0.38509    0.05854    0.19992
     0.22665]
 [   0.00000    0.00000    0.00000    0.00000    1.00000    0.00000    0.00000    0.00000    0.00000
     0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000    1.00000    0.00000    0.00000    0.00000
     0.00000]]
```

In [39]:
```python
# Check rank of M_hat
np.linalg.matrix_rank(M_msf)
```

Out[39]: 10

In [40]:
```
M_msf_inv = np.linalg.inv(M_msf)
print(M_msf_inv)
```

```
[[    1.00000    -0.00000     0.00000     0.00000     0.00745    -3.00473    -3.62961    -3.02771    -0.00000
      0.00000]
 [   -0.00000     1.00000    -0.00000    -0.00000    -0.12208   121.36333   147.73967   122.24146     0.00000
    -0.00000]
 [   -0.00000     0.00000     1.00000    -0.00000     0.24684   662.69532   808.38183   667.09694     0.00000
    -0.00000]
 [    0.00000    -0.00000     0.00000     1.00000     0.69977  -613.40105  -748.24942  -617.47503    -0.00000
      0.00000]
 [    0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000     1.00000
      0.00000]
 [   -0.00000    -0.00000    -0.00000    -0.00000     0.00000    -0.00000    -0.00000     0.00000     0.00000
      1.00000]
 [    1.00000    -0.00000     0.00000     0.00000     1.60527    -2.81461    -2.16547    -1.36052     0.05304
   174.32474]
 [    0.00000     1.00000    -0.00000    -0.00000    26.72206     7.23488    -3.55224    29.13826     3.92205
  2603.27644]
 [    0.00000    -0.00000     1.00000    -0.00000    17.67588     1.85031    -0.78878     7.94046     0.56126
  1921.83776]
 [   -0.00000     0.00000     0.00000     1.00000   -24.32204    -1.34169     2.07621   -10.43331    -0.80479
 -2753.15778]]
```

In [41]:
```
# Diagonalize A using modal matrix from above
Ad = M_msf_inv.dot(A).dot(M_msf)
print(Ad)
```

```
[[   -1.69221     1.00404    -0.79535    -0.65912    -0.00000    -6.52054    -0.00000     0.00000    -0.00000
    -0.00000]
 [   15.27267    -0.21879   -26.78675   -28.84512   -27.64500   106.81115     0.00000    -0.00000     0.00000
     0.00000]
 [  305.43843    -0.91883    54.21425    58.25123     0.00000  -345.25166     0.00000    -0.00000     0.00000
     0.00000]
 [ -282.81502     0.85008   -50.51501   -54.27808     0.00000   442.60752    -0.00000    -0.00000    -0.00000
    -0.00000]
 [    0.00000     0.00000     0.00000     0.00000    -1.66670     0.00000     0.00000     0.00000     0.00000
     0.00000]
 [    0.00000     0.00000     0.00000     0.00000     0.00000   -10.00000    -0.00000     0.00000    -0.00000
    -0.00000]
 [   -0.00000     0.00000     0.00000     0.00000     0.00000    -0.00000  -783.07297     0.53643 -1464.6621
9 -1070.33745]
 [   -0.00000     0.00000     0.00000     0.00000     0.00000    -0.00000 -11198.93019    -1.96622 -18728.3
3307 -13766.29216]
 [    0.00000     0.00000    -0.00000    -0.00000     0.00000    -0.00000 -4708.44985     5.18282 -5103.496
54 -3843.07312]
 [   -0.00000    -0.00000     0.00000     0.00000    -0.00000     0.00000  7029.84495    -7.54558  7709.10727
  5801.20032]]
```

In [42]:
```
Bd = M_msf_inv.dot(B)
print(Bd)
```

```
[[  -0.00000    0.00000]
 [   0.00000   -0.00000]
 [   0.00000   -0.00000]
 [  -0.00000    0.00000]
 [   1.66667    0.00000]
 [   0.00000   10.00000]
 [   0.08841 1743.24737]
 [   6.53676 26032.76440]
 [   0.93543 19218.37757]
 [  -1.34132 -27531.57780]]
```

In [43]:
```
Cd = C.dot(M_msf)
print(Cd)
```

```
[[   1.01935   -0.00168    0.01139    0.00707   -0.00486    1.42703   -0.01935    0.00168   -0.01139
   -0.00707]
 [  -0.11996   -0.00172    1.22169    1.31947   -0.00852   -9.89178   -0.37138    0.00172   -1.85372
   -1.31947]]
```

In [44]:
```
# Block decoupling
```

In [45]:
```
Lambda1 = Ad[0:6,0:6]
Lambda2 = Ad[6:,6:]

print(Lambda1)
```

```
[[  -1.69221    1.00404   -0.79535   -0.65912   -0.00000   -6.52054]
 [  15.27267   -0.21879  -26.78675  -28.84512  -27.64500  106.81115]
 [ 305.43843   -0.91883   54.21425   58.25123    0.00000 -345.25166]
 [-282.81502    0.85008  -50.51501  -54.27808    0.00000  442.60752]
 [   0.00000    0.00000    0.00000    0.00000   -1.66670    0.00000]
 [   0.00000    0.00000    0.00000    0.00000    0.00000  -10.00000]]
```

In [46]:
```
Bd1 = Bd[0:6,:]
print(Bd1)
```

```
[[  -0.00000    0.00000]
 [   0.00000   -0.00000]
 [   0.00000   -0.00000]
 [  -0.00000    0.00000]
 [   1.66667    0.00000]
 [   0.00000   10.00000]]
```

In [47]:
```
Bd2 = Bd[6:,:]
```

In [48]:
```
Cd1 = Cd[:,0:6]
print(Cd1)
```

```
[[  1.01935  -0.00168   0.01139   0.00707  -0.00486   1.42703]
 [ -0.11996  -0.00172   1.22169   1.31947  -0.00852  -9.89178]]
```

In [49]:
```
Cd2 = Cd[:,6:]
```

In [50]:
```
Dstar = (C.dot(np.linalg.inv(-1*A)).dot(B) + D
              - Cd1.dot(np.linalg.inv(-1*Lambda1)).dot(Bd1))
```

In [51]:
```
print(Dstar)
```

```
[[  0.00073  -0.98193]
 [  0.00155   5.11005]]
```

In [52]:
```
np.savetxt("Lambda1.txt",Lambda1,fmt=list(itertools.repeat('%.18e',6)))
np.savetxt("Bd1.txt",Bd1,fmt=list(itertools.repeat('%.18e',2)))
np.savetxt("Cd1.txt",Cd1,fmt=list(itertools.repeat('%.18e',6)))
np.savetxt("Dstar.txt",Dstar,fmt=list(itertools.repeat('%.18e',2)))
```

In [53]:
```
L1_eigvals, M_L1 = np.linalg.eig(Lambda1)
np.savetxt("Lambda1_eigvals.txt",L1_eigvals.reshape((6,1)),
           fmt=list(itertools.repeat('%.18e%+.18ej',1)))
print(L1_eigvals)
```

```
[-0.34915+6.34401j -0.34915-6.34401j -1.04205+0.00000j -0.23448+0.00000j -10.00000+0.00000j -1.66670
+0.00000j]
```

In [53]:
```
```