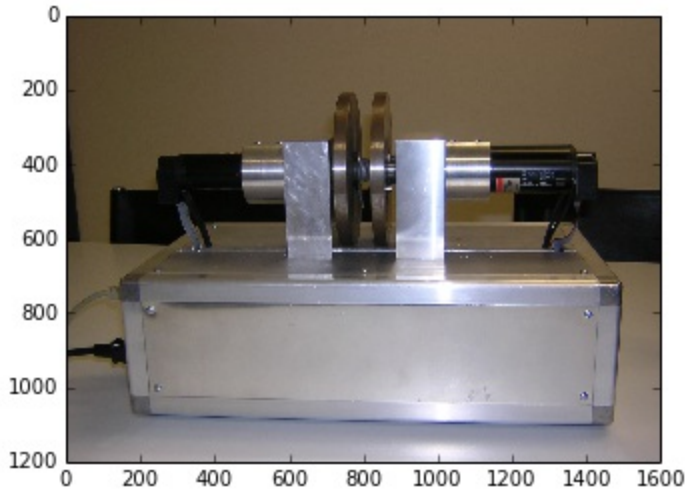
```
Python 2.7.6 (default, Jan 11 2014, 14:34:26)
Type "copyright", "credits" or "license" for more information.

IPython 1.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.
```

```
In [1]: from supsictrl.yottalab import *
   ...: from supsictrl.RCPblk import *
   ...: from scipy import mat, shape, size, array, zeros
   ...: from numpy import reshape, hstack
   ...: import numpy as np
   ...: import scipy as sp
   ...: import matplotlib.pyplot as plt
   ...:
```

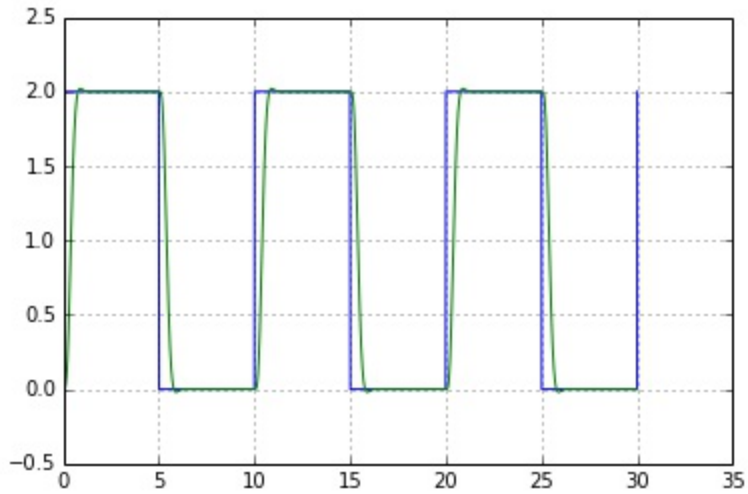
```
In [2]: # Plant
   ...: import matplotlib.image as mpimg
   ...: img=mpimg.imread('disks.jpg')
   ...: plt.imshow(img)
   ...:
Out[2]: <matplotlib.image.AxesImage at 0x3d7d610>
```



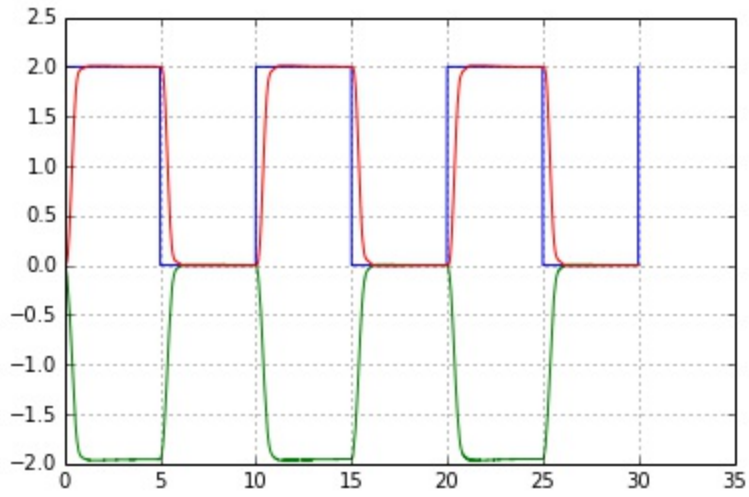

```
In [3]: # Motore 1
   ...: jm1 = 0.0000085; # inerzia [kg*m2]
   ...: kt1 = 0.0000382; # constante di coppia [Nm/mA] per la centralina al
posto di [Nm/A]
   ...:
   ...: # Volano motore 1
   ...: rho_ac = 7900; # densita alluminio [g/m3]
   ...: rv1 = 0.065;   # raggio [m]
   ...: hv1 = 0.01;    # spessore [m]
   ...: mv1 = ((rho_ac*(rv1**2))*np.pi)*hv1;  # massa [kg]
   ...: jv1 = (mv1*(rv1**2))/2;  # inerzia [kg*m2]
   ...: theta1 = jv1+jm1;  # inerzia totale [kg*m2]
   ...:
```

```
In [4]: # Motore 2
   ...: jm2 = 0.000003;  # inerzia [kg*m2]
   ...: kt2 = 0.0000205; # constante di coppia [Nm/mA] per la centralina al
posto di [Nm/A]
   ...:
   ...: # Volano motore 2
   ...: rho_ac = 7900;  # densita acciao [kg/m3]
   ...: rv2 = 0.065;    # raggio [m]
   ...: hv2 = 0.01;     # spessore [m]
   ...: mv2 = ((rho_ac*(rv2**2))*np.pi)*hv2;  # massa [kg]
   ...: jv2 = (mv2*(rv2**2))/2;             # inerzia [kg*m2]
   ...: theta2 = jv2+jm2;                  # inerzia totale [km*m2]
   ...:

In [5]: # Molla                      (frequenza d''oscillazione di 2 Hz)
   ...: d = 0.0027836;  # attrito molla (smorzamento)
   ...: c = 0.4797954;
   ...: d1 =0.0002953
   ...: d2 =0.0004001
   ...:

In [6]: # Plant in SS form
   ...: A=[ [0, 1, 0, 0], \
   ...:     [-c/theta1, -(d1+d)/theta1, -c/theta1, -d/theta1], \
   ...:     [0, 0, 0, 1], \
   ...:     [-c/theta2, -d/theta2, -c/theta2, -(d2+d)/theta2] ];
   ...:
   ...: B=[ [0], \
   ...:     [kt1/theta1], \
   ...:     [0], \
   ...:     [0] ];
   ...:
   ...: C=[ [1,0,0,0],[0,0, 1, 0] ];
   ...: C2=[0,0,1,0]
   ...: D=[[0],[0]];
   ...: D2=[0]
   ...:
   ...: ts = 5e-3
   ...:
   ...: gss = ss(A,B,C2,D2)
   ...: gz = bb_c2d(gss,ts)
   ...:

In [7]: # Control design
   ...: wn=10;
   ...: xi1=np.sqrt(2)/2;
   ...: xi2=0.85;
   ...:
   ...: cl_p1 = [1,2*xi1*wn,wn**2]
   ...: cl_p2 = [1,2*xi2*wn,wn**2]
   ...: cl_p3 = [1,wn]
   ...: cl_poly1=sp.polymul(cl_p1,cl_p2)
   ...: cl_poly=sp.polymul(cl_poly1,cl_p3)
   ...: cl_poles=sp.roots(cl_poly);  # Desired continous poles
   ...: cl_polesd=sp.exp(cl_poles*ts)  # Desired discrete poles
   ...:

In [8]: # Add discrete integrator for steady state zero error
```

```
   ...: Phi_f=np.vstack((gz.A,-gz.C*ts))
   ...: Phi_f=np.hstack((Phi_f,[[0],[0],[0],[0],[1]]))
   ...: G_f=np.vstack((gz.B,zeros((1,1))))
   ...:

In [9]: # Pole placement
   ...: k=pplace(Phi_f,G_f,cl_polesd)

In [10]: # Observer design - reduced order observer
    ...: poli_o=5*cl_poles[0:2]
    ...: poli_oz=sp.exp(poli_o*ts);
    ...:
    ...: disks = ss(A,B,C,D)
    ...: disksz = StateSpace(gz.A,gz.B,C,D,ts)
    ...: T=[[0,1,0,0],[0,0,0,1]];
    ...: r_obs = red_obs(disksz ,T, poli_oz)
    ...:

In [11]: # Controller and observer in the same matrix -  Compact form
    ...: contr_I=comp_form_i(disksz,r_obs,k,ts,[0,1])

In [11]:

In [12]: # Implement anti windup
    ...: [gss_in,gss_out]=set_aw(contr_I,[0.1,0.1,0.1])

In [13]: # simulation
    ...: sq1 = squareBlk(1,2,10,5,0,0)
    ...: ctrin = dssBlk([1,6,7],2,gss_in)
    ...: sum1 = sumBlk([2,5],3,[1,1])
    ...: sat = saturBlk(3,4,3000,-3000)
    ...: ctrfbk = dssBlk(4,5,gss_out)
    ...: plant = cssBlk(4,[6,7],disks)
    ...: prnt = printBlk([1,7])
    ...:
    ...: fname = 'disks'
    ...: simblks = [sq1, plant, ctrin, ctrfbk, sum1, sat, prnt]
    ...: genCode(fname,7,ts,simblks)
    ...: genMake(fname)
    ...:
    ...: !make
    ...: simul('disks',30)
    ...:
cp /home/bucher/CACSD/python/CodeGen/src/python_main.c .
gcc -g -O2 -I/home/bucher/CACSD/python/CodeGen/include   -DMODEL=disks    -c -o
python_main.o python_main.c
gcc -g -O2 -I/home/bucher/CACSD/python/CodeGen/include   -DMODEL=disks    -c -o
disks.o disks.c
gcc -static -o disks  python_main.o disks.o
/home/bucher/CACSD/python/CodeGen/lib/libpyblk.a -lrt -lpthread -lm
### Created executable: disks
```




```
In [14]: # RT execution
    ...: sq1 = squareBlk(1,2,10,5,0,0)
    ...: ctrin = dssBlk([1,6,7],2,gss_in)
    ...: sum1 = sumBlk([2,5],3,[1,1])
    ...: sat = saturBlk(3,4,3000,-3000)
    ...: ctrfbk = dssBlk(4,5,gss_out)
    ...: act1 = maxon_MotBlk(4,0x01,5,8000,4000)
    ...: sens1 = maxon_EncBlk(6,1,5,500)
    ...: sens2 = maxon_EncBlk(7,2,5,500)
    ...: prnt = printBlk([1,6,7])
    ...:
    ...: blks = [sq1, ctrin, sum1, sat, ctrfbk, act1, sens1, sens2, prnt]
    ...:
    ...: fname = 'disksRT'
    ...: genCode(fname, 7, ts, blks)
    ...: genMake(fname, '_rt')
    ...: !make
    ...:
cp /home/bucher/CACSD/python/CodeGen/src/python_main_rt.c .
gcc -g -O2 -I/home/bucher/CACSD/python/CodeGen/include   -DMODEL=disksRT    -c -
o python_main_rt.o python_main_rt.c
gcc -g -O2 -I/home/bucher/CACSD/python/CodeGen/include   -DMODEL=disksRT    -c -
o disksRT.o disksRT.c
gcc -static -o disksRT  python_main_rt.o disksRT.o
/home/bucher/CACSD/python/CodeGen/lib/libpyblk.a -lrt -lpthread -lm
### Created executable: disksRT

In [15]: simul('disksRT',30)
```



In [**16**]: