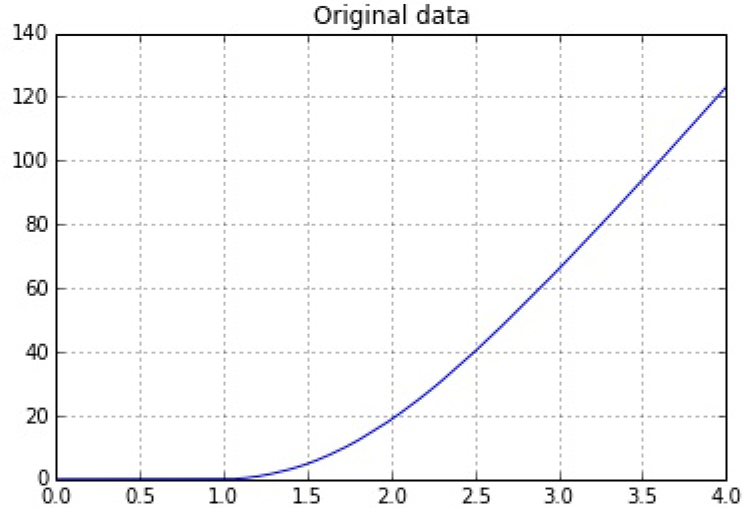
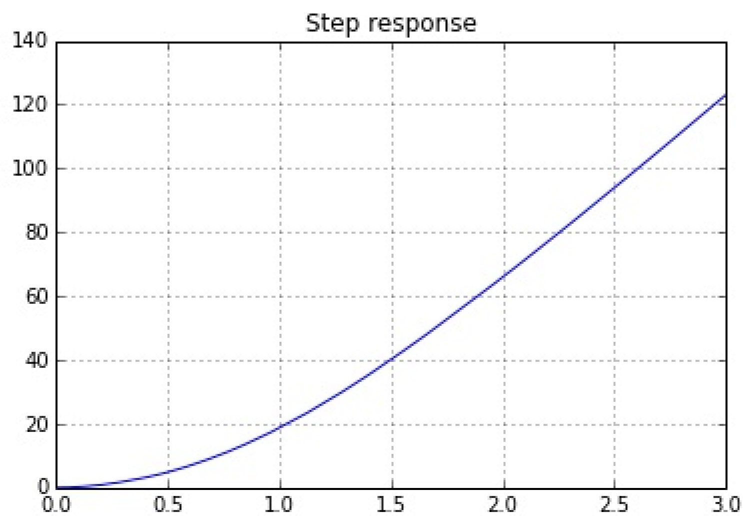
```
Python 2.7.6 (default, Jan 11 2014, 14:34:26)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.2 -- 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.

Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.

In [1]: from yottalab import *
   ...: from scipy.optimize import leastsq
   ...: import numpy as np
   ...: import scipy as sp
   ...: from control import *
   ...: from RCPblk import *
   ...:
   ...: # Motor response for least square identification
   ...: def residuals(p, y, t):
   ...:     [k,alpha] = p
   ...:     err=y+k/alpha**2-k/alpha*t-k/alpha**2*sp.exp(-alpha*t)
   ...:     return err
   ...:
   ...: # Design script
   ...: fname='ID_MOT';
   ...:
   ...: x = loadtxt(fname);
   ...:
   ...: # Read and plot original data
   ...: t=x[:,0]
   ...: y=x[:,1]
   ...: plot(t,y)
   ...: title("Original data")
   ...: grid()
   ...: show()
   ...:
```

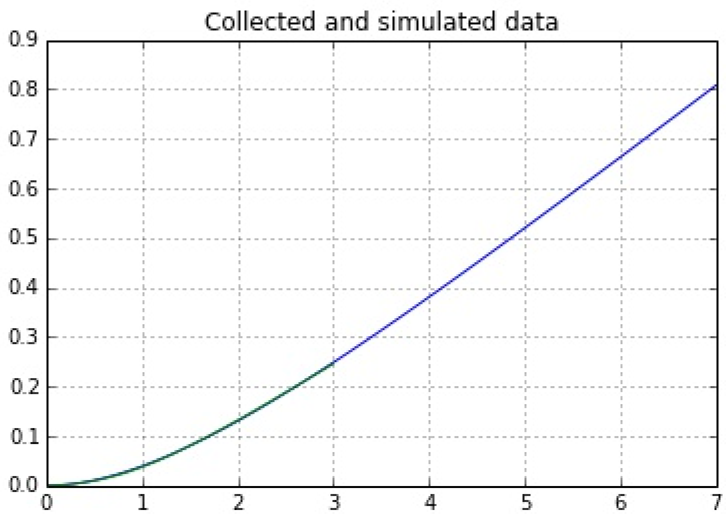



```
In [2]: # Extract and plot step response
   ...: t=t[1000:]
   ...: t=t-t[0];
   ...: y=y[1000:]
   ...: y=y-y[0]
   ...:
   ...: figure()
   ...: plot(t,y)
   ...: title("Step response")
   ...: grid()
   ...: show()
   ...:
```



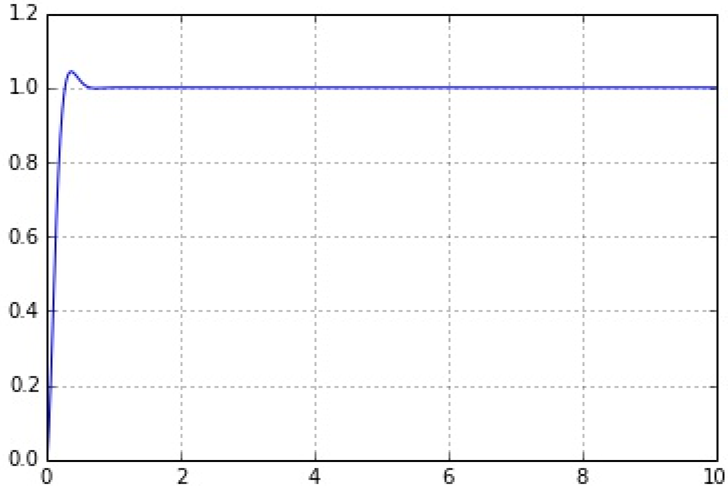

```
In [3]: # Identify the motor transfer function
   ...: p0=[1.0,4.0]
   ...: Uo=500;
   ...: y1=y/Uo
   ...:
   ...: plsq = leastsq(residuals, p0, args=(y1, t))
   ...:
```

```
   ...: # Motor parameters
   ...: Kt=126.0e-6                 # Torque constant
   ...: Jm=Kt/plsq[0][0]            # Motor Inertia
   ...: Dm=plsq[0][1]*Jm            # Motor friction
   ...:
   ...: g=tf([Kt/Jm],[1,Dm/Jm,0])  # Transfer function
   ...:
   ...: a=[[0,1],[0,-Dm/Jm]]
   ...: b=[[0],[1]]
   ...: c=[[Kt/Jm,0]];
   ...: d=[0];
   ...:
   ...: sysc=ss(a,b,c,d)            # Continous state space form
   ...:
   ...: # Compare step response and collected data
   ...: figure()
   ...: Y, T = step(g)
   ...: plot(T,Y)
   ...: hold
   ...: plot(t,y1)
   ...: title("Collected and simulated data")
   ...: grid()
   ...: show()
   ...:
```



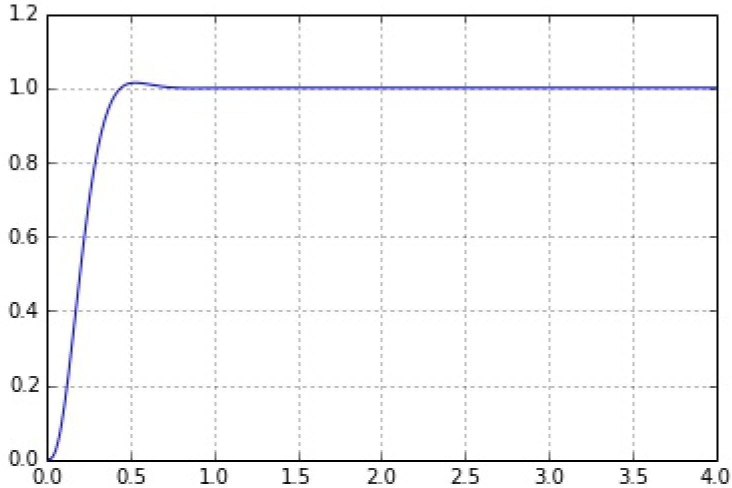

```
In [4]: # Discrete representation
   ...:
   ...: Ts=0.01    # Sampling time
   ...: sysd=bb_c2d(sysc,Ts,'zoh')        # Get discrete state space form
   ...:
   ...: # Control system design
   ...: # State feedback with precompensation
   ...:
   ...: wn=12
   ...: xi=sqrt(2)/2
   ...: cl_poly=[1,2*xi*wn,wn**2]
   ...: cl_poles=sp.roots(cl_poly);  # Desired continous poles
   ...: cl_polesd=sp.exp(cl_poles*Ts)  # Desired discrete poles
   ...: k=place(sysd.A,sysd.B,cl_polesd)
```

```
   ...:
   ...: sysct=StateSpace(sysd.A-sysd.B*k,sysd.B,sysd.C,sysd.D,sysd.dt)
   ...: k_pre=1/(bb_dcgain(sysct)[0,0])
   ...: figure()
   ...: dstep(k_pre*sysct)
   ...:
```




```
In [5]: # State feedback with integral part
   ...:
   ...: wn=12
   ...: xi=sqrt(2)/2
   ...:
   ...: cl_p1=[1,2*xi*wn,wn**2]
   ...: cl_p2=[1,wn]
   ...: cl_poly=sp.polymul(cl_p1,cl_p2)
   ...: cl_poles=sp.roots(cl_poly);     # Desired continous poles
   ...: cl_polesd=sp.exp(cl_poles*Ts)  # Desired discrete poles
   ...:
   ...: sz1=sp.shape(sysd.A);
   ...: sz2=sp.shape(sysd.B);
   ...:
   ...: # Add discrete integrator for steady state zero error
   ...: Phi_f=np.vstack((sysd.A,-sysd.C*Ts))
   ...: Phi_f=np.hstack((Phi_f,[[0],[0],[1]]))
   ...: G_f=np.vstack((sysd.B,zeros((1,1))))
   ...:
   ...: kint=place(Phi_f,G_f,cl_polesd)
   ...:
   ...: # Full order observer
   ...: p_oc=10*(cl_poles[1:3])
   ...: p_od=sp.exp(p_oc*Ts);
   ...:
   ...: f_obs=full_obs(sysd,p_od)
   ...:
   ...: #Reduced order observer
   ...: p_oc=-10*max(abs(cl_poles))
   ...: p_od=sp.exp(p_oc*Ts);
   ...:
   ...: T=[0,1]
```

```
   ...: r_obs=red_obs(sysd,T,[p_od])
   ...:
   ...: contr=comp_form(sysd,r_obs,k)
   ...:
   ...: contr_I=comp_form_i(sysd,r_obs,kint,Ts)
   ...:
   ...: sysct=sysctr(sysd,contr_I)
   ...: figure()
   ...: dstep(sysct,Tf=4)
   ...:
/usr/local/lib/python2.7/dist-packages/slycot/synthesis.py:170: UserWarning: 1
violations of the numerical stability condition occured during the assignment of
eigenvalues
  warnings.warn('%i violations of the numerical stability condition occured
during the assignment of eigenvalues' % warn)
```




```
In [6]:
```