Inverted pendulum - Practical realization

Ing. Roberto Bucher

September 28, 2008

1 Pendulum model

The linearized model of the inverted pendulum is described by

$$\begin{bmatrix} \delta\dot{\varphi} \\ \delta\dot{\omega} \\ \delta\dot{x} \\ \delta\dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m\cdot r\cdot(M+m)\cdot g}{\Theta\cdot(M+m)-m^2\cdot r^2} & 0 & 0 & \frac{-m\cdot r\cdot d}{\Theta\cdot(M+m)-m^2\cdot r^2} \\ 0 & 0 & 0 & 1 \\ \frac{m^2\cdot r^2\cdot g}{\Theta\cdot(M+m)-m^2\cdot r^2} & 0 & 0 & \frac{-\Theta\cdot d}{\Theta\cdot(M+m)-m^2\cdot r^2} \end{bmatrix} \cdot \begin{bmatrix} \delta\varphi \\ \delta\omega \\ \delta x \\ \delta v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{m\cdot r}{\Theta\cdot(M+m)-m^2\cdot r^2} \\ 0 \\ \frac{\Theta}{\Theta\cdot(M+m)-m^2\cdot r^2} \end{bmatrix} \cdot \delta F \tag{1}$$

where
M is the mass of the cart
m is the mass of the pole
r is the fictive length of the pole (distance of center of mass from the joint)
Θ is the Inertial moment of the pole
d is the viscous friction coefficient

2 Substituting the force

The force δF is given by the motor which drives the cart with the pole.
The equation of the motor is given by

$$J\ddot{\varphi} = r_p \cdot \delta F - D_m \cdot dot\varphi + K_t \cdot I_a \tag{2}$$

where r_p is the radius of the motor gear.
If we don't consider the motor inertia and friction (we can integrate them in the identified motor-car model), the previous equation simply become

$$\delta F = K_t \cdot \frac{I_a}{r_p} \tag{3}$$

If we substitute equation (3) in (1) we obtain

$$\begin{bmatrix} \delta\dot{\varphi} \\ \delta\dot{\omega} \\ \delta\dot{x} \\ \delta\dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{m\cdot r\cdot(M+m)\cdot g}{\Theta\cdot(M+m)-m^2\cdot r^2} & 0 & 0 & \frac{-m\cdot r\cdot d}{\Theta\cdot(M+m)-m^2\cdot r^2} \\ 0 & 0 & 0 & 1 \\ \frac{m^2\cdot r^2\cdot g}{\Theta\cdot(M+m)-m^2\cdot r^2} & 0 & 0 & \frac{-\Theta\cdot d}{\Theta\cdot(M+m)-m^2\cdot r^2} \end{bmatrix} \cdot \begin{bmatrix} \delta\varphi \\ \delta\omega \\ \delta x \\ \delta v \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{Kt}{r_p}\cdot\frac{m\cdot r}{\Theta\cdot(M+m)-m^2\cdot r^2} \\ 0 \\ \frac{Kt}{r_p}\cdot\frac{\Theta}{\Theta\cdot(M+m)-m^2\cdot r^2} \end{bmatrix} \cdot I_a \tag{4}$$

3 Cart model

The transfer function of the car can be written as

$$\frac{X(s)}{I(s)} = \frac{\frac{K_t}{M \cdot r_p}}{s \cdot (s + \frac{d}{M})} = \frac{K}{s \cdot (s + \alpha)}$$

From the parametrical model of the car system we can obtain:

$$M = \frac{K_t}{r_p \cdot K}$$

and

$$d = \alpha \cdot M$$

4 Parameter identification

We identify the cart parameters by applying a step current and gettting the output posizion.
Figure 1 shows the response to the step input.

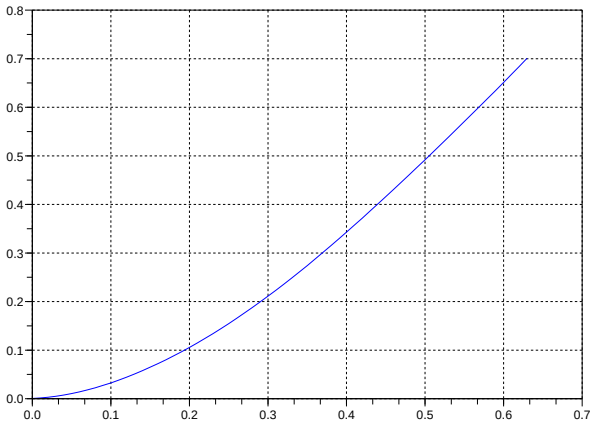

Figure 1: Step responses

The inverse Laplace Transform of $y(t)$ from 3 with an input step is

$$y(t) = -\frac{K}{\alpha^2} + \frac{K}{\alpha}t + \frac{K}{\alpha^2}e^{-\alpha t} \tag{5}$$

We can use the following Scilab script to perform a least-square identification of the parameters $K = p(1)$ and $\alpha = p(2)$. The values from $index_{min}$ and $index_{max}$ can be read from the step answer, where the input step changes his value and where the response is saturated.

```
function z=fun(p,t,y)
z=y+p(1)/p(2)^2-p(1)/p(2)*t-p(1)/p(2)^2*exp(-p(2)*t);
endfunction

x=read(fname,-1,2);
t=x(:,1);
y=x(:,2);
t=t(index_min:index_max);
t=t-t(1);
y=y(index_min:index_max);

xbasc()
plot(t,y)

p0=[1,4];

[ff,p]=leastsq(list(fun,t,y/Uo),p0);
```

Figure 2 shows the measured and the simulated signals.

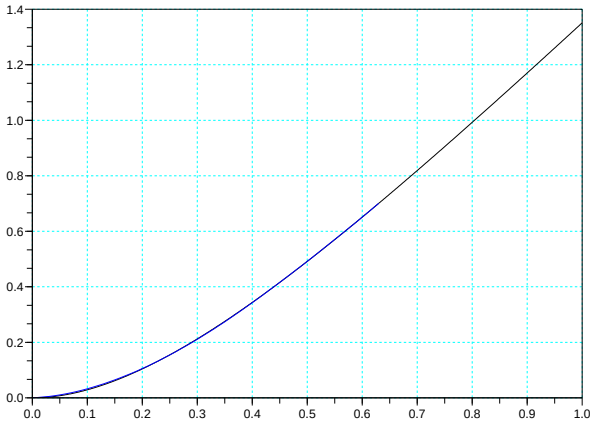

Figure 2: Measured and simulated signals

Now we can calculate the missing motor parameters as

$$M = \frac{K_t}{r_p \cdot p(1)}$$

and

$$d = p(2) * M$$

5 Control of the inverted pendulum

We initialize the control of the inverted pendulum with the following Scilab script

```
// Identification function for leassquare
function z=fun(p,t,y)
z=y+p(1)/p(2)^2-p(1)/p(2)*t-p(1)/p(2)^2*exp(-p(2)*t);
```

```
endfunction

// Fixed and known parameters
rp=0.75/26.17585;       // Motor gear
m=0.08377+0.10834;      // Pole mass
Tosc=29/21;             // Pole swing period
r=0.026+.297;           // Pole center of mass
g=9.81;
Theta=m*g*r*Tosc^2/(4*pi^2);  // Pole inertia
kt=60.3e-6;             // Motor torque constant
k_encp = -1;            // encoder pole gain
Imin=0;
Q=diag([100,1,100,1]);    // LQR Q matrix
R=[1];                    // LQR R matrix
swingAmp = -0.298;        // Sinus amplitude for pendulum swing-up

// Car Identification
fname='SCOPEpicc';
x=read(fname,-1,2);
t=x(:,1);
y=x(:,2);
t=t(index_min:index_max);
t=t-t(1);
y=y(index_min:index_max);

p0=[1,4];

[ff,p]=leastsq(list(fun,t,y/Uo),p0);

g1=syslin('c',p(1)/(s*(s+p(2))));

// Control design - Plant
Ts=5e-3;           // Sampling time

M=kt*N/(rp*p(1))   // Cart mass
d=p(2)*M           // cart viscous friction

A=[  0    1   0   0
     m*r*(M+m)*g/(Theta*(M+m)-m*m*r*r)   0   0   -m*r*d/(Theta*(M+m)-m*m*r*r)
     0    0   0   1
     m*m*r*r*g/(Theta*(M+m)-m*m*r*r)   0   0   -Theta*d/(Theta*(M+m)-m*m*r*r)];
B=N*[0 m*r*kt/(rp*(Theta*(M+m)-m*m*r*r)) 0 kt*Theta/(rp*(Theta*(M+m)-m*m*r*r))]';
C=[1 0 0 0; 0 0 1 0];
D=[0;0];

sys=syslin('c',A,B,C,D);    // Continous linear model of the inverted pendulum
sysd=dscr(sys,Ts);          // discrete model

[Ad,Bd,Cd,Dd]=abcd(sysd);   // Extract state space matrix

// Control design - LQR
k_lqr=bb_dlqr(Ad,Bd,Q,R);   // LQR controller

// Control design - Reduced order observer
preg=max(abs(spec(A)));
poli_oss=exp([-preg,-preg]*10*Ts);
T=[0,0,0,1;0,1,0,0];
[Ao,Bo,Co,Do]=redobs(Ad,Bd,Cd,Dd,T,poli_oss);

// Control design - Controller and observer in compact form
Contr=comp_form(Ao,Bo,Co,Do,Ts,k_lqr);

// Some values for scicos blocks

gs=syslin('c',10/(s+10));
gz=ss2tf(dscr(tf2ss(gs),Ts));

Adnew=Ad-Bd*k_lqr;
Bdnew=Bd;
Cdnew=Cd(2,:);
Ddnew=Dd(2,:);

Gzctr=syslin('d',Adnew,Bdnew,Cdnew,Ddnew);
u=ones(1:1000);
```

```
y=dsimul(Gzctr,u);

Kpregain=1/y($);

// Swing up controller

g_ss=tf2ss(g1);
num=g1.num;
den=g1.den;

p_g=roots(den);
g_dss=dscr(g_ss,Ts);
p_max=max(abs(p_g));

p_c=[-p_max,-p_max,-p_max];
k_poles=10;
p_c=k_poles*p_c;

p_d=exp(p_c*Ts);
[Phi,G,C,D]=abcd(g_dss);

[m1,n1]=size(Phi);
[m2,n2]=size(G);
Phi_f=[Phi,zeros(m1,1);-C(1,:)*Ts,1];
G_f=[G;zeros(1,n2)];

k=ppol(Phi_f,G_f,p_d);

p_oc=[-5*p_max];
p_od=exp(p_oc*Ts);

T2=[0,1];
[Ao2,Bo2,Co2,Do2]=redobs(Phi,G,C,D,T2,p_od);
Contr2=comp_form_i(Ao2,Bo2,Co2,Do2,Ts,k);
Isat=3300;

g_contr=ss2tf(Contr2)

gctr_r=g_contr(:,1);
gctr_y=g_contr(:,2);

g_r=gctr_r.num/z^2
g_y=gctr_y.num/z^2
g_s=1-gctr_r.den/z^2
gin=[g_r,g_y];
gss_in=tf2ss(gin);
gss_fbk = tf2ss(g_s)
```

6 Some useful functions

6.1 bb_dlqr.sci

```
function k=bb_dlqr(fi,H,Q,R)
// Calculates the LQR gains for discrete systems
[n1,d1]=size(fi);
big=sysdiag(Q,R);
[w,wp]=fullrf(big,1e-20);
C1=wp(:,1:n1);
D12=wp(:,n1+1:$);
P=syslin('d',fi,H,C1,D12);
[k,X]=lqr(P);
k=-k;
endfunction
```

6.2 redobs.sci

```
function [A_redobs,B_redobs,C_redobs,D_redobs]=redobs(A,B,C,D,T,poles)
// Find the reduced order observer for the system A,B,C,D,
//  with observer poles "poles"
// T is the matrix needed to get the pair [C;T] invertible
```

```
P=[C;T]
invP=inv([C;T])

AA=P*A*invP

ny=size(C,1)
nx=size(A,1)
nu=size(B,2)

A11=AA(1:ny,1:ny)
A12=AA(1:ny,ny+1:nx)
A21=AA(ny+1:nx,1:ny)
A22=AA(ny+1:nx,ny+1:nx)

L1=ppol(A22',A12',poles)';

nn=nx-ny;

A_redobs=[-L1 eye(nn,nn)]*P*A*invP*[zeros(ny,nn); eye(nn,nn)];
B_redobs=[-L1 eye(nn,nn)]*[P*B P*A*invP*[eye(ny,ny);
          L1]]*[eye(nu,nu) zeros(nu,ny); -D, eye(ny,ny)];
C_redobs=invP*[zeros(ny,nx-ny);eye(nn,nn)];
D_redobs=invP*[zeros(ny,nu) eye(ny,ny);zeros(nx-ny,nu) L1]*[eye(nu,nu) zeros(nu,ny);
                -D, eye(ny,ny)];
endfunction
```

6.3 comp_form

```
function [Contr]=comp_form(A,B,C,D,Ts,K)
// Create the compact form of the Observer ABCD and the
// gain K,
//
// A,B,C,D: Observer matrices
// Ts: sampling time
// K: state feedback gains

Bu=B(:,1);
By=B(:,2:$);
Du=D(:,1);
Dy=D(:,2:$);

X=inv(1+K*Du);

Ac=A-Bu*X*K*C;
Bc=[Bu*X,By-Bu*X*K*Dy]
Cc=-X*K*C;
Dc=[X,-X*K*Dy]
Contr=syslin('d',Ac,Bc,Cc,Dc)

endfunction
```

6.4 comp_form_i

```
function [Contr]=comp_form_i(A,B,C,D,Ts,K,Cy)
// Create the compact form of the Observer ABCD and the
// gain K, using an integrator at the input to eliminate the
// steady state error
//
// A,B,C,D: Observer matrices
// Ts: sampling time
// K: state feedback gains
// Cy: matrix to extract the output for the steady state feedback

[larg,rarg]=argn(0);

if rarg ~= 7 then
Cy = [1];
end

ss_sys=syslin('d',A,B,C,D);
```

```
ss_sys(7)=Ts;
g_sys=ss2tf(ss_sys);

g_int=syslin('d',Ts/(%z-1));
g_int(7)=Ts;

gu=g_sys('num')(:,1)./g_sys('den')(:,1);
gy=g_sys('num')(:,2:$)./g_sys('den')(:,2:$);

nn=size(K,2);

Ke = K(1,nn);
K  = K(1,1:nn-1);

Greg=[-Ke*g_int/(1+K*gu),(Ke*Cy*g_int-K*gy)/(1+K*gu)];
Contr=tf2ss(Greg);

endfunction
```

7 Scicos block diagram of the controller

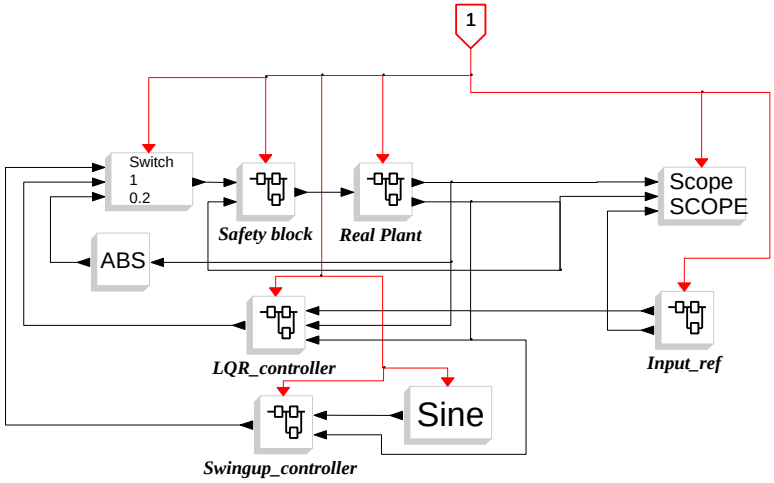

Figure 3: Scicos block diagram for the controller

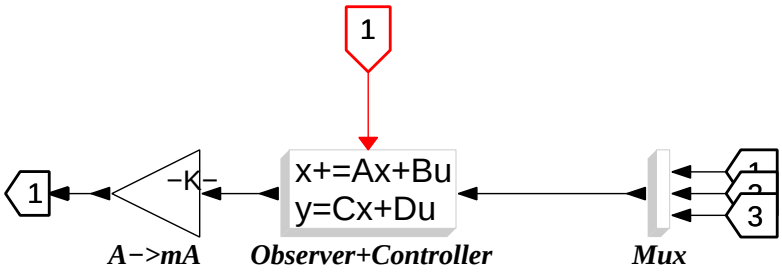

Figure 4: Scicos block diagram for the lqr superblock

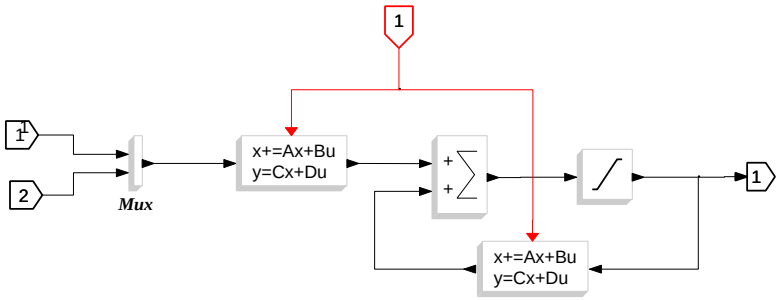

Figure 5: Scicos block diagram for the swingup controller

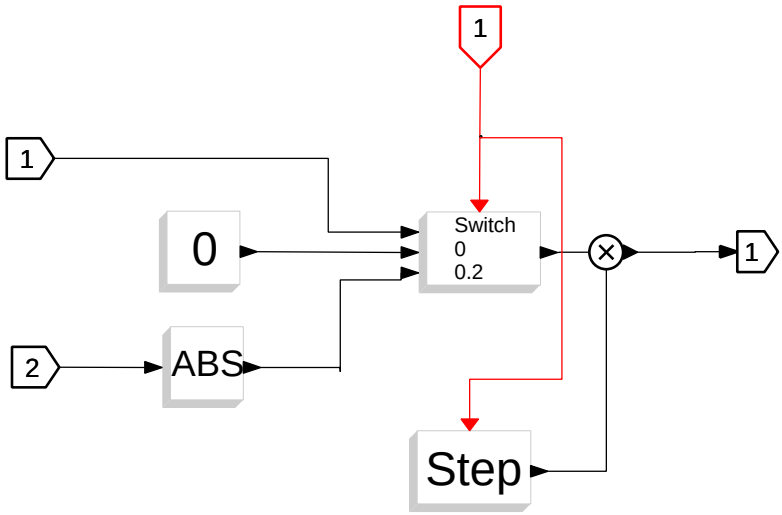

Figure 6: Scicos block diagram for the safety superblock