.

Nonlinear Systems Toolbox

Arthur J. Krener

ajkrener@nps.edu

Supported in part by AFOSR

The Nonlinear Systems Toolbox is a suite of Matlab routines for the control and estimation of nonlinear systems.

We will discuss two of them for Optimal Stabilization and Optimal Regulation

.

Example of Optimal Stabilization

```
% This is an example of how to optimally
% stabilize a system to an operating point
% using Al'brekht's method to solve a smooth
% infinite horizon optimal control problem.
% E. G. Al'brekht, On the Optimal
% Stabilization of Nonlinear Systems, J.
% Appl. Math. Mech., v. 25, pp. 1254-1266, 1961.
```

```
% This script can be used as a template to
% optimally stabilize any smooth nonlinear
% system.  Smooth means that the system
% can be described using elementary functions
% that MATLAB can symbolically differentiate.

% This example is to stabilize to the upright
% position a double pendulum using  torques
% at each of the pivots.

% Although it is not needed in this example
% in general it is essential to scale the
% state and control variables by dividing
% each variable by its charcteristic value
% so that each variable varies plus/minus
% one unit from its equilibrium value.
```

```
% The code starts by computing the Taylor
% polynomial of the dynamics of degrees
% one through D and the Taylor polynomial
% of the Lagrangian of degrees two through D+1
% at an operating point. It is essential
% that the linear part of the dynamics and
% the quadratic part of the Lagrangian
% constitute a nice LQR problem.
```

```
% Then the code computes the Taylor polynomial
% of the optimal cost PY through degree two to D+1
% and the Taylor polynomial of the optimal
% feedback KA to degree D at the operating
% point.  These are in scaled displacement coordinates
% around the operating point. Then the code
% simulates the close loop response from some
% initial state.
```

```
% Increasing the degree D generally leads to
% a more accurate approximation to the true
% optimal cost and optimal feedback near the
% operating point.  But it does not necessarily
% lead to a larger domain on which the computed
% PY is a valid Lyapunov function for the dynamics
% using the computed feedback KA but generally
% using D=3 is better than D=1.
% The code puts no limit on the degree D nor
% the dimensions N and M of the state and
% control but time and storage limitations
% might effectively do so.
```

```
clear
% Define system
n=4;  % state dimension
m=2; % control dimension
d=3; % degree of optimal feedback
x=sym('x',[n,1]); % state variables, theta1,theta2, th
u=sym('u',[m,1]); %  control variables, torque at firs

l1=1; % length of first massless link
l2=2; % length of second massless link
m1=2;  % mass at end of first link
m2=1; % mass at end of second link
b1=0.5; % damping coefficient at first joint
b2=0.5; % damping coefficient at second joint
g=9.8; % gravitational constant

x0=[pi;pi;0;0];  % equilibrium state, at rest straight
u0=[0;0]; % equilibrium control

xscale=ones(n,1); % state variable charcteristic lengt
uscale=ones(m,1); % control variable charcteristic len
```

```
% inertia matrix

M=[m1*l1^2+m2*l2^2,m2*l1*l2*cos(x(1,1)-x(2,1));.....
    m2*l1*l2*cos(x(1,1)-x(2,1)), m2*l2^2];

% coriolis and centripetal matrix

C=reshape(jacobian(reshape(M,4,1),x(1:2,1))*x(3:4,1),2

% kinetic energy

T=x(3:4,1).'*M*x(3:4,1)/2;

% potential energy

V=g*(m1*l1*(1-cos(x(1,1)))+m2*(l1*(1-cos(x(1,1)))+l2*(

% Lagrangian

L=T-V;
```

```
% dynamics

fsym12=x(3:4,1); % kinematics
fsym34=inv(M)*(jacobian(L,x(1:2,1)).'-C*x(3:4,1).....
    +u-[b1*x(3,1);b2*(x(4,1)-x(3,1))]);
fsym=[fsym12;fsym34];  % dynamics

% The linear part of the dynamics at
% the equilibrium of the must be stabilizable.
```

```
% We choose a control Lagrangian starting with quadrat
% The quadartic part of the Lagrangian and the linear
% of the dynamics must satisfy the standard LQR condit

lsym=( (x(1,1)-x0(1,1))^2+(x(2,1)-x0(2,1))^2....
         +u(1,1)^2+u(2,1)^2)/2;

% If there are state and/or control constraints then p
% terms can be added to the Lagarngian to try to enfor
```

```
% We call hjb_set_up.m to convert the symbolic FSYM ar
% matrices F and L of coefficients of their Taylor pol
% at x0, u0.

tic
[f,l]=hjb_set_up(fsym,lsym,x,u,x0,u0,.....
                            xscale,uscale,n,m,d);
set_up_time=toc
```

```
% We call hjb.m to find the Taylor polynomial PY of th
% to degree D+1 and the Taylor polynomial KA of the op
% to degree D. The routine also returns FK and LK, the
% and the closed loop Lagarngian.


tic
[ka,fk,py,lk]= hjb(f,l,n,m,d);
comp_time=toc
% Note: hjb.m runs much faster the second time.
```

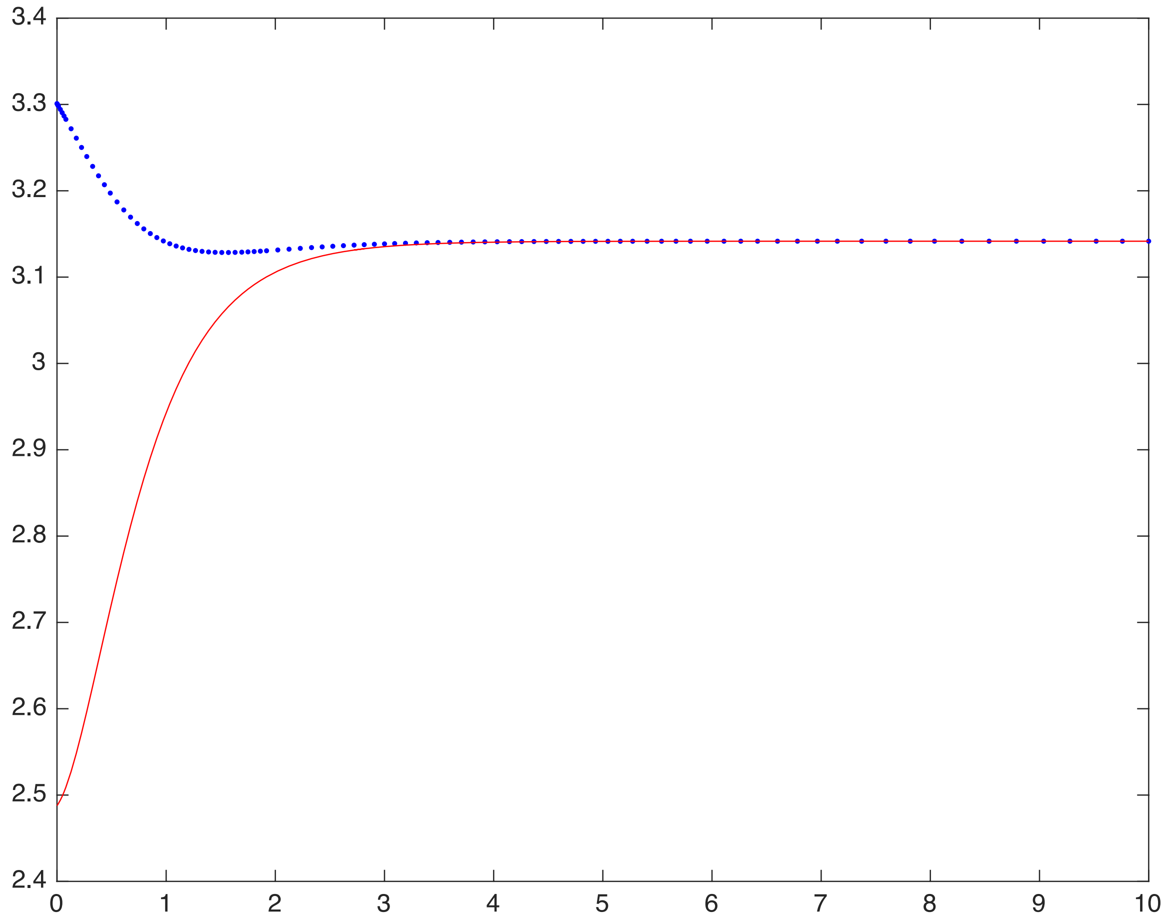
```
% We verify that PY and KA approximately satisfy the
% first HJB equation.  The rseidue of this equation
% is a polynomial of degrees 2 through D+1 and HJB_ERR
% is the 2 norm of its coefficients.

hjb_err=norm(dd(py,[1,n],[2,d+1],fk,[n,n],[1,d],[2,d+1
```

```
% Create anonymous functions of PY and KA.
% Their argument XX is in the original coordinates.

py_fn=@(xx) py*mon((xx-x0)./xscale,n,[2,d+1]);

ka_fn=@(xx) ka*mon((xx-x0)./xscale,n,[1,d]);

% Create an anonymous matlabFunction of FSYM.
ff=matlabFunction(fsym,'file','','vars',{x,u});


% Create an anonymous function
% that ode45.m can integrate
xdot=@(t,xx)ff(xx,ka_fn(xx));
```

```
% Define a random initial state.
x1=0.5*randn(n,1)+x0

% Integrate the closed loop plant f
% rom this initial condition.
[tout,xxout]=ode45(xdot,[0,10],x1);

% Plot the trjectories of two angles

plot(tout,xxout(:,1),'.b',tout,xxout(:,2),'-r')
```

```
 set_up_time =

    3.0611

Solving the HJB equations of degree 1
Solving the HJB equations of degree 2
Solving the HJB equations of degree 3

comp_time =

    0.0544

hjb_err =

   4.2107e-10

x1 =

    3.3010
    2.4877
   -0.2168
    0.1713
```

.


3.4
3.3
3.2
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
0
1
2
3
4
5
6
7
8
9
10