ELEC 3035: Lab 4 — State transfer and pole placement control

Lecturer: Ivan Markovsky

1. *Minimum energy state transfer* Write a function that computes the minimum energy control for transferring the system from a given initial state to a given final state in a given amount of time. Test the function on the system given on page 14 of Lecture 4, reproducing the numerical results shown in the lecture slides.

   Solution: Function for computing the minimum energy state transfer control input:

```
% LN_STATE_TRANSFER - Minimum energy state transfer
function [u,e] = ln_state_transfer(sys,xini,xdes,t)

% Controllability matrix
C = sys.b;
for i = 1:t-1
    C = [C sys.a*C(:,end)];
end

if rank(C) < size(sys,'order')
   disp('The system is not controllabale or the transfer time is less than the order')
   u = NaN; e = NaN;
else
   u = flipud(C' * inv(C*C') * (xdes - a^(t)*xini)); e = norm(u);
end
```

   Using the function on an example:

```
% Define the system
a = [-1.75 -.8; 1 0]; b = [1; 0];
sys = ss(a,b,eye(2),zeros(2,1),1);

xini = [1;1]; % initial state
xdes = [0;0]; % desired state
t = 15;       % transfer time

% Compute the control input and the corresponding output
[u,e] = ln_state_transfer(sys,xini,xdes,t);
y = lsim(sys,[u;0],[],xini)

% Plot the results
stairs(0:(t+2),[U; zeros(3,1)],'linewidth',2)
stairs(0:(t+2),[y(:,1); zeros(2,1)],'linewidth',2)
stairs(0:(t+2),[y(:,2); zeros(2,1)],'linewidth',2)
```

   □

2. *Pole placement* Download and save in your working directory the data file

   `http://www.ecs.soton.ac.uk/~im/elec3035/lab4.mat`

   Load the data in Matlab with the command `load`. Now you have in your workspace a variable `sys` — the control plant you are designing a controller for. The control objective is to achieve a constant output $y_{\text{des}} = \begin{bmatrix} 1 & -1 \end{bmatrix}^\top$ as quickly as possible with "small" control.

   (a) A naive approach to achieve the objective is to ignore the dynamics and apply constant control. See page 13 of the overview lecture for derivation of the constant control. Implement this approach and observe the response of the controlled system to zero initial conditions.

(b) Improve the transient response by design a state feedback controller. Use the pole placement synthesis method, placing the poles close to the origin, so that the response is fast. Pole placement is implemented by the `place` function of the Control System Toolbox. Placing all the poles at the origin (an approach called deadbeat control) ensures convergence of the transient in a finite number of steps (at most n steps, where n is the order of the plant), however, `place` does not accept multiple poles, so you will need to distribute the poles apart from each other.

(c) Design an observer using again pole placement synthesis in order to make the controller output-feedback.

Solution:

```
load lab4.mat % load the data
n = size(sys,'order')

% Constant control
[a,b,c,d] = ssdata(sys);
u = inv(c*inv(eye(n)-a)*b)*[1;-1];

% Simulate the controlled system
T = 250;
y = lsim(sys,u*ones(1,T));

% Plot the trajectories
figure
stairs(y(:,1),'r'), hold on
stairs(y(:,2),'b')

% Output feedback control
k = place(a ,b , linspace(-.5e-1,.5e-1,n));
l = place(a',c', linspace(-.5e-1,.5e-1,n))';
o = estim(sys,l,1:2,1:2); o = o(3:end,:);
C = k * o;

sysa = ss(a,b,[zeros(2,n); c],[eye(2); d],-1);
sysc = feedback(sysa,C);

[ac,bc,cc,dc] = ssdata(sysc(3:4,1:2));
u = inv(cc*pinv(eye(2*n)-ac)*bc)*[1;-1];

% Simulate the controlled system
T = 15;
y = lsim(sysc,u*ones(1,T));

% Plot the trajectories
figure
stairs(y(:,1),'r','linewidth',2), hold on
stairs(y(:,2),'b','linewidth',2)

figure
stairs(y(:,3),'r','linewidth',2), hold on
stairs(y(:,4),'b','linewidth',2)
```

□