Robotic Link/Joint Control System:
Dynamic Shaping and Linear State Feedback Control Laws

Aly Khater
Lawrence Lai

Introduction

Our system is a single robotic joint/link driven from a gear by an armature-controlled dc servo motor. Objective for Phase I is to find eigenvalues for our third-order system and compare the step responses between the third-order system and the first-order system. For Phase II, we will design a state feedback control system for our model.

The solution is to use MATLAB for calculations and simulation. We will find the first desired eigenvalue from the given settling time, which would lead us to having all 3 EVs. We can then plot the step response for comparison. For Phase II, we will use Ackerman's formula to get our desired closed-loop response to compare with our open-loop response.

Objective

The objective for the first part of this phase, Phase I, is to find the first desired eigenvalue from the dominant first-order system, with the give settling time of 0.5s. Then we will augment the first EV with two more EVs, with the second EV being 10 times larger than the first, and the third EV being 1 less than the second EV. Plot the step response of these eigenvalues versus the dominant first-order step response for comparison.

For Phase II, the objective is to design state feedback control laws for both cases from Part 2, by using Ackerman's to create the closed-loop state-space system.

Background

Our system involves an RL circuit, a motor outputting rotational motion, connected to a gear that is also rotating, which then leads into our load. The load is connected to the robot link/joint to provide its motions. See Figure 1 for each aspect of the system, and the names assigned to each variable.

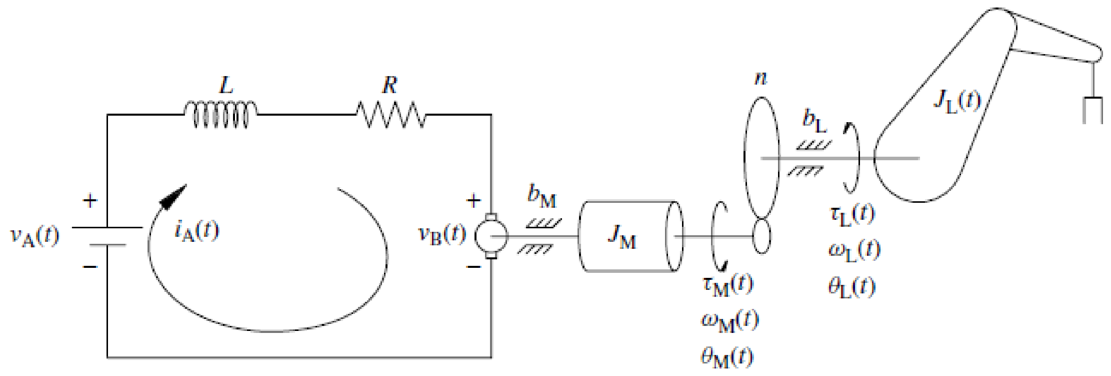


FIGURE 1.16 Diagram for Continuing Exercise 3.

$v_A(t)$	armature voltage	L	armature inductance	R	armature resistance
$i_A(t)$	armature current	$v_B(t)$	back emf voltage	k_B	back emf constant
J_M	motor inertia	b_M	motor viscous damping	$\tau_M(t)$	motor torque
k_T	torque constant	$\omega_M(t)$	motor shaft velocity	$\theta_M(t)$	motor shaft angle
n	gear ratio	$J_L(t)$	load inertia	b_L	load viscous damping
$\tau_L(t)$	load shaft torque	$\omega_L(t)$	load shaft velocity	$\theta_L(t)$	load shaft angle

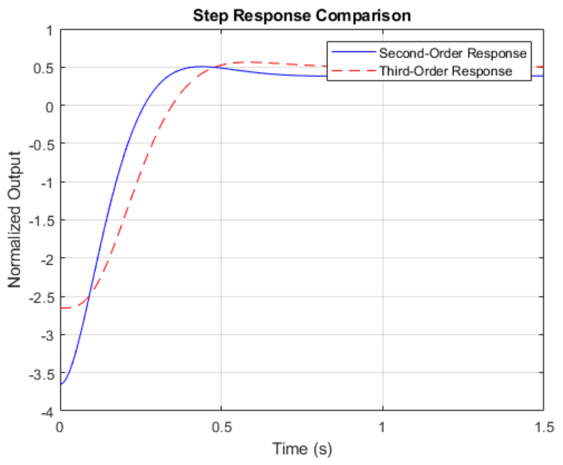
Figure 1: Robot Link/joint system connected to motor and RL circuit

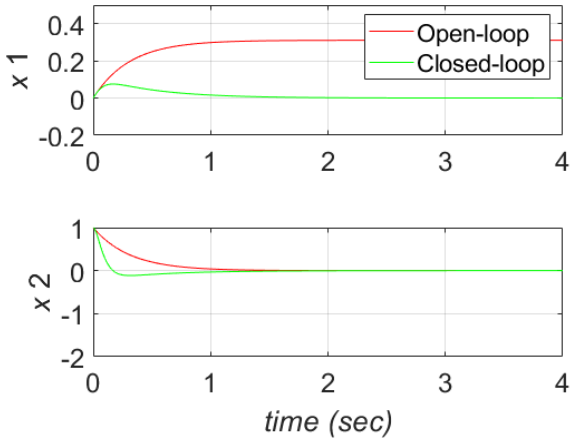
Variables that are controlled in this system are: $L,\ R,\ k_B,\ J_M,\ b_M,\ k_t,\ n,\ J_L,\ b_L$.

Design and Simulation Results

```
domEig =

     2

additional_eig1 =

   -20

additional_eig2 =

   -21

DesEig3 =

    -2
   -20
   -21

K =
```

0.1950 -1.6266 -0.5318





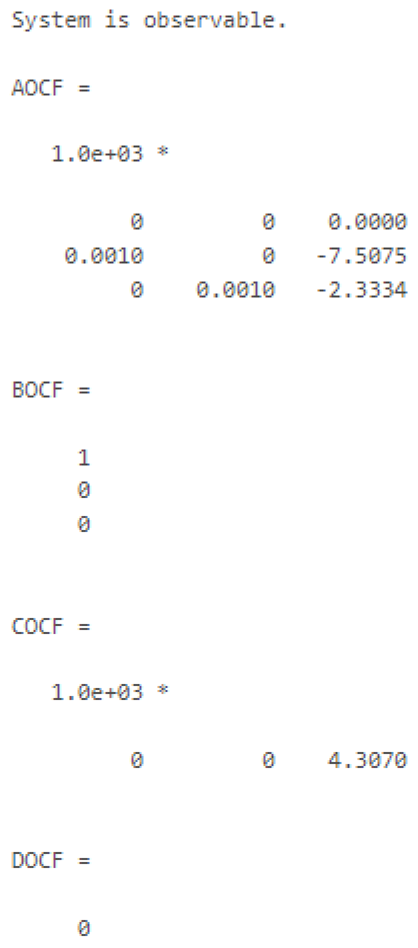
For the left-hand side plot,

```
additional_eig1 = -20  % 10 times the dominant eigenvalue
additional_eig2 = -21  % One less than the first additional eigenvalue
% Create a third-order denominator polynomial from the desired eigenvalues
den3 = poly([DesEig2; additional_eig1; additional_eig2]);
% Create the third-order transfer function.
num3 = [den3(5)]; % Coefficient for the numerator.
Des3 = tf(num3,den3);
% step response
[response2, t2] = step(Des2, td);
[response3, t3] = step(Des3, td);
% normalize step response
nresponse2 = normalize(response2);
nresponse3 = normalize(response3);

plot(t2, nresponse2, 'b-', t3, nresponse3, 'r--');
```

For the right-hand side plots,

```
additional_eig1 = -20  % 10 times the dominant eigenvalue
additional_eig2 = -21  % One less than the first additional eigenvalue

DesEig3 = [-2; additional_eig1; additional_eig2]
K = place(A,B,DesEig3) % Compute state
% feedback gain matrix
% K
Kack = acker(A,B, DesEig3); % Check K via
% Ackerman's formula
Ac = A-B*K;
Bc = B; % Compute closed-loop
% state feedback system
Cc = C;
Dc = D;
RbtRc = ss(Ac,Bc,Cc,Dc); % Create the
% closed-loop
% state-space system
[Yc,t,Xc] = lsim(RbtRc,U,t,X0); % Compare open-loop and
% closed-loop responses
subplot(211), plot(t,Xo(:,1),'r',t,Xc(:,1),'g')
subplot(212), plot(t,Xo(:,2),'r',t,Xc(:,2),'g')
```

Observability



```
System is observable.

AOCF =

   1.0e+03 *

         0         0    0.0000
    0.0010         0   -7.5075
         0    0.0010   -2.3334


BOCF =

     1
     0
     0


COCF =

   1.0e+03 *

         0         0    4.3070


DOCF =

     0
```

```
%-------------------------------------------------------
% Chapter 4. Observability
%-------------------------------------------------------
Q = obsv(Rbt); % Calculate observability
% matrix Q
if (rank(Q) == size(A,1))% Logic to assess
% observability
disp('System is observable.');
else
disp('System is NOT observable.');
end
Q1 = [C; C*A]; % Check Q via the formula
%-------------------------------------------------------
% Chapter 4. Coordinate Transformations and Observer
% Canonical Form
%-------------------------------------------------------
Qocf = inv(Pccfi);
Tocf = inv(Q)*Qocf; % Calculate OCF transformation
% matrix
Aocf = inv(Tocf)*A*Tocf; % Transform to OCF via formula
Bocf = inv(Tocf)*B;
Cocf = C*Tocf;
Docf = D;
[RbtOCF,TOCF] = canon(Rbt,'companion');
% Compute OCF using canon
AOCF = RbtOCF.a
BOCF = RbtOCF.b
COCF = RbtOCF.c
DOCF = RbtOCF.d
```

Analysis

Analyze your results

Conclusions