16-642

PS2

Boxiang Fu

boxiangf

10/05/2024

**Q1.**

Using the parameters as Problem 2H in Problem Set 1 and the estimated starting state of , we design the error dynamics eigenvalues to be

Note that we could not set the error dynamics eigenvalues to be too far to the left since this will cause the system to be unstable (refer to Problem 2F in Problem Set 1). The state vs. time plots are as follows:

A graph of a function

Description automatically generated

A graph of a state dynamics

Description automatically generated

A graph of a state dynamics

Description automatically generated

A graph of a state dynamics

Description automatically generated

Note that all the estimated states converge to the actual states within a relatively short time (i.e. less than 2 seconds).

**Q2.**

Given the plant model

**A.**

Applying the Laplace transformation gives:

Where is the open-loop transfer function for the system.

**B.**

Closing the loop with unity negative feedback and defining , we get

**C.**

Using MATLAB, we find the poles and zeros of to be:

**D.**

Since the poles of are in the negative half-plane (i.e. its real values are negative), it means the system is asymptotically stable. The system will also oscillate since the imaginary value of the poles is non-zero (and relatively large). This is because when applying the inverse Laplace transform back into the time domain, the real part of a pole forms the exponentiation term, while the imaginary part forms the sinusoidal term (Lecture 7 Slide 24). So, a large imaginary value will cause a highly oscillatory term. Finally, since the zeros are relatively far from the poles, it does not affect much of the dynamics of the system. There might be a slight decrease in the total response and could potentially increase maximum percent overshoot (Lecture 8 Slide 27), but any effects should not be too evident.

**E.**

The step-response of the open-loop system is:

A graph of a step response

Description automatically generated

The step-response of the closed-loop system is:

A graph showing a step response

Description automatically generated

**F.**

For the open-loop system, the Final Value Theorem gives the steady state value as:

For the closed-loop system, the Final Value Theorem gives the steady state value as:

**Q3.**

For the transfer function

We tune the PID controller as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration** |  |  |  | **Rise Time (sec)** | **Max % Overshoot** | **Steady State Error** |
| 1 | 1 | 1 | 1 | 8.1582 | 13.7161 | 0 |
| 2 | 10 | 1 | 1 | 8.7982 | 0 | 0 |
| 3[[1]](#footnote-1) | 50 | 1 | 1 | 0.5669 | 0 | 0 |
| 4 | 50 | 10 | 1 | 0.5492 | 0.2958 | 0 |

We obtain the required specifications on the 4th iteration. The step-response plot with these parameters is displayed below:

A graph showing a step response

Description automatically generated

**Appendices:**

**A.1. Code for Q1**

1. % MEC

2. % Q1

3. clear;

4.

5. % Parameters

6. gamma = 2;

7. alpha = 1;

8. beta = 1;

9. D = 1;

10. mu = 3;

11.

12. % Populate A, B, and C matrix

13. A = zeros(4,4);

14. A(1,3) = 1;

15. A(2,4) = 1;

16. A(3,2) = 1;

17. A(3,3) = -3;

18. A(4,2) = 2;

19. A(4,3) = -3;

20.

21. B = zeros(4,1);

22. B(3,1) = 1;

23. B(4,1) = 1;

24.

25. C = [39.37 0 0 0];

26.

27. % Check controllability

28. Q = [B A\*B A\*A\*B A\*A\*A\*B];

29. cont = rank(Q);

30.

31. % Check observability

32. Q0 = [C; C\*A; C\*A\*A; C\*A\*A\*A];

33. obs = rank(Q0);

34.

35. % Q matix for LQR

36. Qu = 10;

37.

38. Qx = zeros(4,4);

39. Qx(1,1) = 30;

40. Qx(2,2) = 5;

41. Qx(3,3) = 5;

42. Qx(4,4) = 5;

43.

44. % LQR

45. [Kc,S,P] = lqr(A,B,Qx,Qu);

46.

47. % Eigenvalues of closed-loop system

48. eig\_cl = eig(A-B\*Kc);

49.

50. % Design eigenvalues of error dynamics system

51. p0 = [complex(-5,1);

52. complex(-5,-1);

53. complex(-6,-1);

54. complex(-6,1)];

55.

56. K0 = place(A', C', p0)';

57.

58. % Eigenvalues of error dynamics system

59. eig\_ed = eig(A-K0\*C);

60.

61. % Tracking controller

62. Kf = -inv((C \* inv(A - B \* Kc) \* B));

63.

64. % Timespan

65. T = 0.01;

66. tspan = [0 20];

67. t\_vector = 0:T:20;

68.

69. % Initial conditions (first 4 values are actual state, and last 4 values

70. % are estimated state)

71. x0 = transpose([0, 0, 0, 0, 0.01, 0.01, -0.03, 0.01]);

72.

73. % Run ode45 for original non-linear system

74. [t, x] = ode45(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, Kc, K0, Kf, A, B, C), t\_vector, x0);

75.

76. % Plotting

77. figure();

78. plot(t\_vector,x(:,1));

79. hold on

80. plot(t\_vector,x(:,5));

81. title("State dynamics of xc");

82. legend("Actual", "Estimated");

83. xlabel("time (sec)");

84. ylabel("state (m)");

85. hold off

86.

87. figure();

88. plot(t\_vector,x(:,2));

89. hold on

90. plot(t\_vector,x(:,6));

91. title("State dynamics of phi");

92. legend("Actual", "Estimated");

93. xlabel("time (sec)");

94. ylabel("state (rad)");

95. hold off

96.

97. figure();

98. plot(t\_vector,x(:,3));

99. hold on

100. plot(t\_vector,x(:,7));

101. title("State dynamics of xcdot");

102. legend("Actual", "Estimated");

103. xlabel("time (sec)");

104. ylabel("state (m/sec)");

105. hold off

106.

107. figure();

108. plot(t\_vector,x(:,4));

109. hold on

110. plot(t\_vector,x(:,8));

111. title("State dynamics of phidot");

112. legend("Actual", "Estimated");

113. xlabel("time (sec)");

114. ylabel("state (rad/sec)");

115. hold off

116.

117. % Create function for original non-linear ODE

118. function dxdt = odefunnl(t, x, gamma, alpha, beta, D, mu, Kc, K0, Kf, A, B, C)

119. % Find feedback control from linearized ODE

120. yd = 20 \* square(0.02 \* pi \* t);

121.

122. x1 = x(1);

123. x2 = x(2);

124. x3 = x(3);

125. x4 = x(4);

126. x1hat = x(5);

127. x2hat = x(6);

128. x3hat = x(7);

129. x4hat = x(8);

130. x = [x1; x2; x3; x4];

131. xhat = [x1hat; x2hat; x3hat; x4hat];

132.

133. u = -(Kc \* xhat) + (Kf \* yd);

134. y = C\*x;

135.

136. % Correction term for dxdt.hat

137. corr\_term = K0 \* (y - C \* xhat);

138.

139. dxdt = zeros(8,1);

140. % Dynamics of actual system

141. dxdt(1) = x3;

142. dxdt(2) = x4;

143. dxdt(3) = (-alpha\*sin(x2)\*beta\*x4^2 + alpha\*u - alpha\*mu\*x3 + cos(x2)\*sin(x2)\*D\*beta)/(alpha\*gamma - beta^2\*cos(x2)^2);

144. dxdt(4) = (-cos(x2)\*sin(x2)\*beta^2\*x4^2 + u\*cos(x2)\*beta + sin(x2)\*D\*gamma - mu\*x3\*cos(x2)\*beta)/(alpha\*gamma - beta^2\*cos(x2)^2);

145.

146. % Dynamics of estimated system

147. dxdt(5) = x3hat + corr\_term(1);

148. dxdt(6) = x4hat + corr\_term(2);

149. dxdt(7) = (-alpha\*sin(x2hat)\*beta\*x4hat^2 + alpha\*u - alpha\*mu\*x3hat + cos(x2hat)\*sin(x2hat)\*D\*beta)/(alpha\*gamma - beta^2\*cos(x2hat)^2) + corr\_term(3);

150. dxdt(8) = (-cos(x2hat)\*sin(x2hat)\*beta^2\*x4hat^2 + u\*cos(x2hat)\*beta + sin(x2hat)\*D\*gamma - mu\*x3hat\*cos(x2hat)\*beta)/(alpha\*gamma - beta^2\*cos(x2hat)^2) + corr\_term(4);

151. end

**A.2. Code for Q2**

1. % MEC

2. % Q2

3. clear;

4.

5. % Part c

6. num = [1 4 80];

7. denom =[2 17 158];

8.

9. Ts = tf(num, denom);

10.

11. zeros = zero(Ts);

12. poles = pole(Ts);

13.

14. % Part e

15.

16. step(Ts);

17.

18. num\_ol = [1 4 80];

19. denom\_ol = [1 13 78];

20. Gs = tf(num\_ol, denom\_ol);

21. step(Gs);

**A.3. Code for Q3**

1. % MEC

2. % Q3

3. clear;

4.

5. % Transfer function

6. num = [20 17];

7. denom = [1 9 231 400 60];

8. Gs = tf(num, denom);

9.

10. % Create PID controller

11. Kp = 50;

12. Ki = 10;

13. Kd = 1;

14. Tf = 0;

15. C = pid(Kp,Ki,Kd,Tf);

16.

17. % Closed-loop transfer function

18. Tcl = feedback(Gs\*C, 1);

19.

20. % Step Response

21. step\_Tcl = step(Tcl, 50);

22.

23. % Plot

24. step(Tcl);

25.

26. S = stepinfo(Tcl);

27. disp(S);

28.

29. % Steady-state error

30. sse = step\_Tcl(end)-1;

31. disp(sse);

1. While this meets the specifications, the convergence to the steady state is very slow. We increase to decrease convergence time. [↑](#footnote-ref-1)