**Appendices:**

**A.1. Code for Q1**

1. % MEC

2. % Q1E

3. clear;

4.

5. % Parameters

6. m = 1;

7. mu = 0.5;

8. k = 5;

9. t = 5;

10.

11. % Time span

12. tstep = 0.1;

13. t\_vector = 0:tstep:t;

14.

15. % Initial conditions

16. x0 = [0; 5];

17.

18. % Populate A and B matrix

19. A = zeros(2,2);

20. A(1,1) = 0;

21. A(1,2) = 1;

22. A(2,1) = -k/m;

23. A(2,2) = -mu/m;

24.

25. B = [0 ; 1/m];

26.

27. % Solve for unforced linear system

28. x = [];

29. for time = 0:tstep:t

30. curr\_x = expm(A\*time)\*x0;

31. x = [x, curr\_x];

32. end

33.

34. % Plot

35. figure;

36.

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

38. hold on

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

40. title("Plot of unforced system vs. time for spring-mass-damper system");

41. legend("Position of mass","Velocity of mass");

42. xlabel("time");

43. ylabel("x-value");

44. hold off

45.

46. % Q1F

47. % Desired eigenvalues

48. p = [complex(-1,1);

49. complex(-1,-1)];

50.

51. % K matrix

52. K = place(A,B,p);

53. eigs = eig(A-B\*K);

54.

55. % Q1G

56. % Parameters

57. t = 10;

58. x0 = [1; 1];

59.

60. % Time span

61. tstep = 0.1;

62. t\_vector = 0:tstep:t;

63.

64. % Solve for linear state feedback system

65. x = [];

66. for time = 0:tstep:t

67. curr\_x = expm((A-B\*K)\*time)\*x0;

68. x = [x, curr\_x];

69. end

70.

71. % Plot

72. figure;

73.

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

75. hold on

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

77. title("Plot of linear state feedback system vs. time for spring-mass-damper system");

78. legend("Position of mass","Velocity of mass");

79. xlabel("time");

80. ylabel("x-value");

81. hold off

**A.2. Code for Q2B**

1. % MEC

2. % Q2B

3. clear;

4.

5. syms alpha beta gamma D mu u x3 xcdotdot x2 x4 phidotdot

6. eq1 = gamma \* xcdotdot - beta \* phidotdot \* cos(x2) + beta \* x4 \* x4 \* sin(x2) + mu \* x3 == u;

7. eq2 = alpha \* phidotdot - beta \* xcdotdot \* cos(x2) - D \* sin(x2) == 0;

8. sol = solve(eq1, eq2, xcdotdot, phidotdot);

9. disp(sol.xcdotdot);

10. disp(sol.phidotdot);

**A.3. Code for Q2C-G**

1. % MEC

2. % Q2C

3. clear;

4.

5. % Parameters

6. gamma = 2;

7. alpha = 1;

8. beta = 1;

9. D = 1;

10. mu = 3;

11.

12. % Populate A and B 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. % Eigenvalues of A

26. eigs = eig(A);

27.

28. % Q matix for LQR

29. Qu = 10;

30.

31. Qx = zeros(4,4);

32. Qx(1,1) = 1;

33. Qx(2,2) = 5;

34. Qx(3,3) = 1;

35. Qx(4,4) = 5;

36.

37. % LQR

38. [K,S,P] = lqr(A,B,Qx,Qu);

39.

40. % Timespan

41. T = 0.01;

42. tspan = [0 30];

43. t\_vector = 0:T:30;

44.

45. % Initial conditions

46. % x0 = transpose([0, 0.1, 0, 0]);

47. % x0 = transpose([0, 0.5, 0, 0]);

48. % x0 = transpose([0, 1.0886, 0, 0]);

49. % x0 = transpose([0, 1.1, 0, 0]);

50.

51. % Run ode45 for linearized system

52. % [t, x] = ode45(@(t, x) odefun(t, x, A, B, K), t\_vector, x0);

53.

54. % Run ode45 for original non-linear system

55. [t, x] = ode45(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, K), t\_vector, x0);

56.

57. % For the [0, 1.1, 0, 0]^T non-linear system initial state, ode45 is not

58. % able to plot beyond 9.6 secs. Plotting using ode23t instead to 15 secs

59.

60. % tspan = [0 15];

61. % [t, x] = ode15s(@(t, x) odefunnl(t, x, gamma, alpha, beta, D, mu, K), tspan, x0);

62. % t\_vector = 0:(30/(length(x)-1)):30;

63.

64. % Plotting

65. figure();

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

67. hold on

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

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

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

71. % title("State dynamics of linearized system");

72. title("State dynamics of original non-linear system");

73. legend("xc (m)", "phi (rad)", "xcdot (m/sec)", "phidot (rad/sec)");

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

75. ylabel("state");

76. hold off

77.

78. % Create function for linearized ODE

79. function dxdt = odefun(t, x, A, B, K)

80. dxdt = (A - B \* K) \* x;

81. end

82.

83. % Create function for original non-linear ODE

84. function dxdt = odefunnl(t, x, gamma, alpha, beta, D, mu, K)

85. u = -(K \* x);

86. x1 = x(1);

87. x2 = x(2);

88. x3 = x(3);

89. x4 = x(4);

90.

91. dxdt = zeros(4,1);

92. dxdt(1) = x3;

93. dxdt(2) = x4;

94. 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);

95. 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);

96. end

**A.4. Code for Q2H-I**

1. % MEC

2. % Q2H

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. % Eigenvalues of A

28. eigs = eig(A);

29.

30. % Q matix for LQR

31. Qu = 1;

32.

33. Qx = zeros(4,4);

34. Qx(1,1) = 50;

35. Qx(2,2) = 1;

36. Qx(3,3) = 5;

37. Qx(4,4) = 5;

38.

39. % LQR

40. [K,S,P] = lqr(A,B,Qx,Qu);

41.

42. % Timespan

43. T = 0.01;

44. tspan = [0 200];

45. t\_vector = 0:T:200;

46.

47. % Initial conditions

48. x0 = transpose([0, 0, 0, 0]);

49.

50. % Run ode45 for original non-linear system

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

52.

53. % Plotting

54. figure();

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

56. hold on

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

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

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

60. title("State dynamics of non-linear system");

61. legend("xc (m)", "phi (rad)", "xcdot (m/sec)", "phidot (rad/sec)");

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

63. ylabel("state");

64. hold off

65.

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

67. y = C \* transpose(x);

68. figure();

69. plot(t\_vector,y);

70. hold on

71. plot(t\_vector,yd);

72. title("Desired and actual outputs");

73. legend("Actual xc", "Desired xc");

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

75. ylabel("output (in)");

76. hold off

77.

78. % Create function for original non-linear ODE

79. function dxdt = odefunnl(t, x, gamma, alpha, beta, D, mu, K, A, B, C)

80. % Find feedback control from linearized ODE

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

82. v = -inv((C \* inv(A - B \* K) \* B)) \* yd;

83. u = -(K \* x) + v;

84.

85. x1 = x(1);

86. x2 = x(2);

87. x3 = x(3);

88. x4 = x(4);

89.

90. dxdt = zeros(4,1);

91. dxdt(1) = x3;

92. dxdt(2) = x4;

93. 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);

94. 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);

95. end