A picture containing fireworks, dark, water, flying

Description automatically generated

College of Engineering

School of Aeronautics and Astronautics

AAE 564

System Analysis and Synthesis

Homework 3

State Space Representation, Linearization, and Transfer Functions

*Author:*

Tomoki Koike

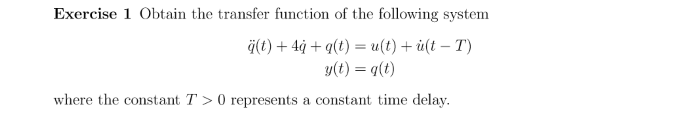
*Supervisor:*

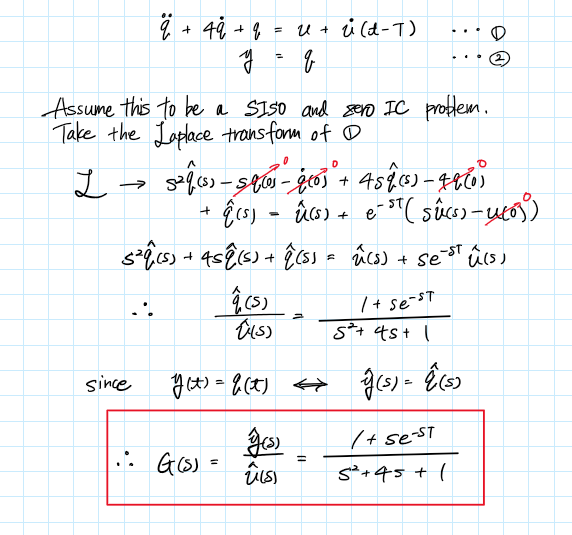
Martin Corless

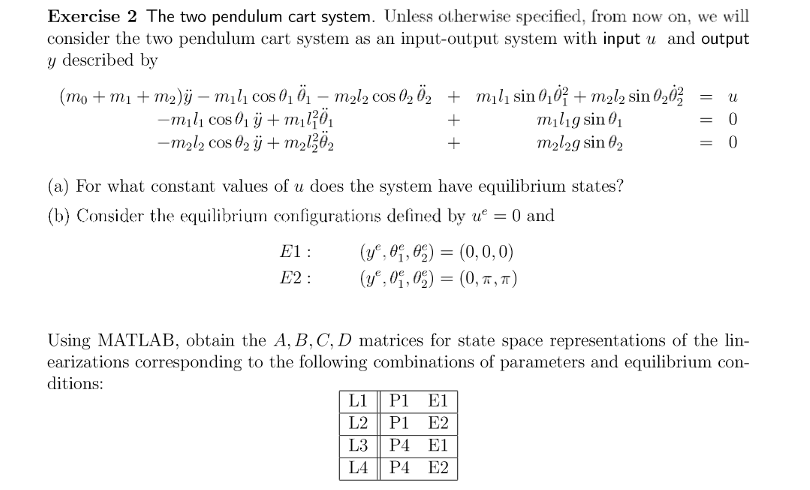
September 18th, 2020 Friday

Purdue University

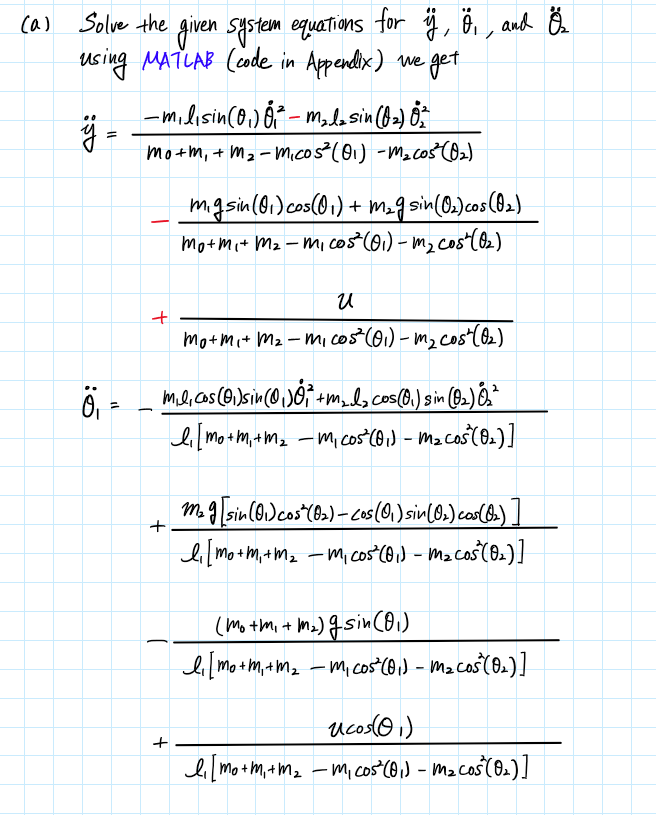
West Lafayette, Indiana

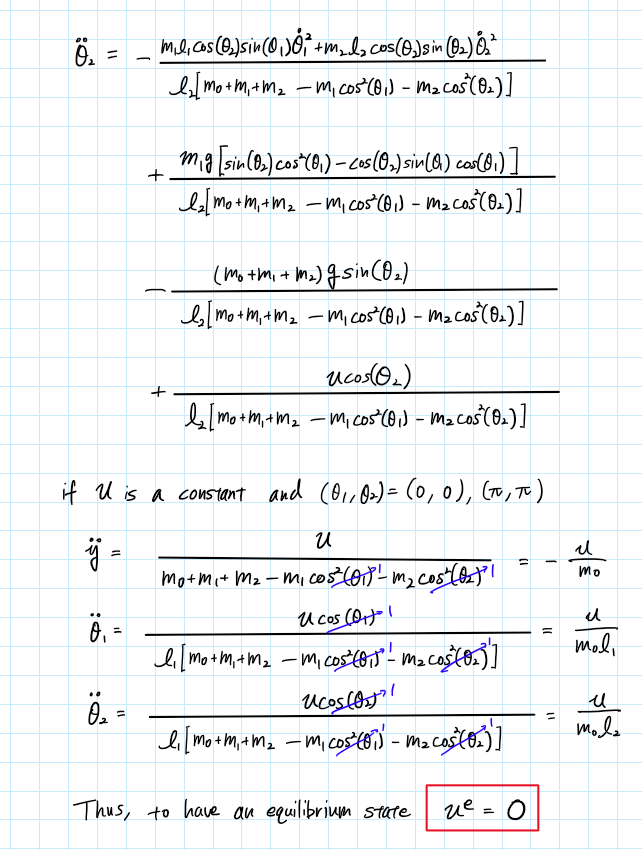






|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
| P1 | 2 | 1 | 1 | 1 | 1 | 1 | 0 |
| P2 | 2 | 1 | 1 | 1 | 0.99 | 1 | 0 |
| P3 | 2 | 1 | 0.5 | 1 | 1 | 1 | 0 |
| P4 | 2 | 1 | 1 | 1 | 0.5 | 1 | 0 |





Solved the complicated algebra using the following MATLAB code

% (a)

syms m\_0 m\_1 m\_2 l\_1 l\_2 y\_ddot theta\_ddot\_1 theta\_ddot\_2 theta\_dot\_1 theta\_dot\_2 theta\_1 theta\_2 u g

eqn1 = (m\_0+m\_1+m\_2)\*y\_ddot-m\_1\*l\_1\*cos(theta\_1)\*theta\_ddot\_1-m\_2\*l\_2\*cos(theta\_2)\*theta\_ddot\_2...

+m\_1\*l\_1\*sin(theta\_1)\*theta\_dot\_1^2+m\_2\*l\_2\*sin(theta\_2)\*theta\_dot\_2^2 == u

eqn2 = -m\_1\*l\_1\*cos(theta\_1)\*y\_ddot+m\_1\*l\_1^2\*theta\_ddot\_1+m\_1\*l\_1\*g\*sin(theta\_1) == 0

eqn3 = -m\_2\*l\_2\*cos(theta\_2)\*y\_ddot+m\_2\*l\_2^2\*theta\_ddot\_2+m\_2\*l\_2\*g\*sin(theta\_2) == 0

eqns = [eqn1, eqn2, eqn3];

S = solve(eqns, [y\_ddot, theta\_ddot\_1, theta\_ddot\_2]);

S.y\_ddot

S.theta\_ddot\_1

S.theta\_ddot\_2

The trim() command for the Simulink model on the next page gives the following results

m = [2, 1, 1]; l = [1, 1]; g = 1;

[x,u,y,dx] = trim('hw3\_p2\_type2');

|  |  |
| --- | --- |
| x = 6×1  0  3.1416  3.1416  0  0  0 | dx = 6×1  10-15 ×  0  0  0  0.1225  -0.2449  -0.2449 |
| u = 0 | y = 0 |

This verifies our results.

(b)

A screenshot of a cell phone

Description automatically generatedFirst, we made a Simulink model of the system

Embedded MATLAB Block – Function (code)

function y = fcn(u, data, data1, data2)

%{

EMBEDDED MATLAB BLOCK FUNCTION

%}

m0 = data(1); m1 = data(2); m2 = data(3); l1 = data1(1); l2 = data1(2);

g = data2;

num = -m1\*l1\*sin(u(1))\*u(3)\*u(3) - m2\*l2\*sin(u(2))\*u(4)\*u(4)...

- m1\*g\*sin(u(1))\*cos(u(1)) - m2\*g\*sin(u(2))\*cos(u(2))...

+ u(5);

den = m0 + m1 + m2 - m1\*cos(u(1))^2 - m2\*cos(u(2))^2;

y = num / den;

end

Embedded MATLAB Block – Function1 (code)

function y = fcn(u, data, data1, data2)

%{

EMBEDDED MATLAB BLOCK FUNCTION1

%}

m0 = data(1); m1 = data(2); m2 = data(3); l1 = data1(1); l2 = data1(2);

g = data2;

num = -(m1\*l1\*cos(u(1))\*sin(u(1))\*u(3)\*u(3) + m2\*l2\*cos(u(1))\*sin(u(2))\*u(4)\*u(4))...

+ m2\*g\*(sin(u(1))\*cos(u(2))^2 - cos(u(1))\*sin(u(2))\*cos(u(2)))...

- (m0 + m1 + m2)\*g\*sin(u(1)) + u(5)\*cos(u(1));

den = l1\*(m0 + m1 + m2 - m1\*cos(u(1))^2 - m2\*cos(u(2))^2);

y = num / den;

end

Embedded MATLAB Block – Function2 (code)

function y = fcn(u, data, data2, data1)

%{

EMBEDDED MATLAB BLOCK FUNCTION2

%}

m0 = data(1); m1 = data(2); m2 = data(3); l1 = data1(1); l2 = data1(2);

g = data2;

num = -(m1\*l1\*cos(u(2,1))\*sin(u(1))\*u(3)\*u(3) + m2\*l2\*cos(u(2))\*sin(u(2))\*u(4)\*u(4))...

+ m1\*g\*(sin(u(2))\*cos(u(1))^2 - cos(u(2))\*sin(u(1))\*cos(u(1)))...

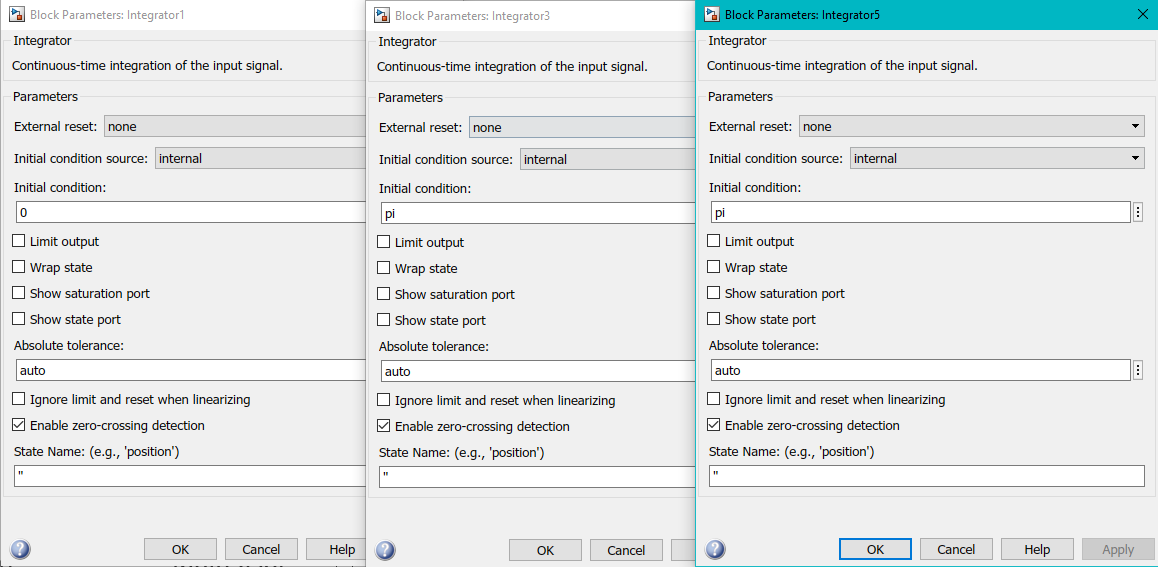
- (m0 + m1 + m2)\*g\*sin(u(2)) + u(5)\*cos(u(2));

den = l2\*(m0 + m1 + m2 - m1\*cos(u(1))^2 - m2\*cos(u(2))^2);

y = num / den;

end

For the conditions E1 and E2, we set the initial conditions of the integrator block of y, , and correspondingly to ; like in the following windows,



Then finally, by running the following code we can get the state space realization for the linearized system.

% (b) Linearizing using simulink

% Set the global variables for the sFunction used in the simulink model

% L1 & L2

m = [2, 1, 1]; l = [1, 1]; g = 1;

[A, B, C, D] = linmod('hw3\_p2\_type2')

% L3 & L4

m = [2, 1, 1]; l = [1, 0.5]; g = 1;

[A, B, C, D] = linmod('hw3\_p2\_type2')

L1:

|  |  |
| --- | --- |
| A = 6×6  0 0 0 1.0000 0 0  0 0 0 0 1.0000 0  0 0 0 0 0 1.0000  0 -0.5000 -0.5000 0 0 0  0 -1.5000 -0.5000 0 0 0  0 -0.5000 -1.5000 0 0 0 | B = 6×1  0  0  0  0.5000  0.5000  0.5000 |
| C = 1×6  1 0 0 0 0 0 | D = 0 |

L2:

|  |  |
| --- | --- |
| A = 6×6  0 0 0 1.0000 0 0  0 0 0 0 1.0000 0  0 0 0 0 0 1.0000  0 -0.5000 -0.5000 0 0 0  0 1.5000 0.5000 0 0 0  0 0.5000 1.5000 0 0 0 | B = 6×1  0  0  0  0.5000  -0.5000  -0.5000 |
| C = 1×6  1 0 0 0 0 0 | D = 0 |

L3:

|  |  |
| --- | --- |
| A = 6×6  0 0 0 1.0000 0 0  0 0 0 0 1.0000 0  0 0 0 0 0 1.0000  0 -0.5000 -0.5000 0 0 0  0 -1.5000 -0.5000 0 0 0  0 -1.0000 -3.0000 0 0 0 | B = 6×1  0  0  0  0.5000  0.5000  1.0000 |
| C = 1×6  1 0 0 0 0 0 | D = 0 |

L4:

|  |  |
| --- | --- |
| A = 6×6  0 0 0 1.0000 0 0  0 0 0 0 1.0000 0  0 0 0 0 0 1.0000  0 -0.5000 -0.5000 0 0 0  0 1.5000 0.5000 0 0 0  0 1.0000 3.0000 0 0 0 | B = 6×1  0  0  0  0.5000  -0.5000  -1.0000 |
| C = 1×6  1 0 0 0 0 0 | D = 0 |



Convert the A, B, C, D matrices to state space using ss(‘sys’), and then convert it to a transfer function using tf(‘sys). Then, using zero(‘sys’) and pole(‘sys’) we obtain the zeros and poles.

sys\_ss = ss(A, B, C, D);

sys\_tf = tf(sys\_ss);

zeros = zero(sys\_tf)

poles = pole(sys\_tf)

L1:

|  |  |
| --- | --- |
| zeros = 4×1 complex  -0.0000 + 1.0000i  -0.0000 - 1.0000i  0.0000 + 1.0000i  0.0000 - 1.0000i | poles = 6×1 complex  0.0000 + 0.0000i  0.0000 + 0.0000i  0.0000 + 1.4142i  0.0000 - 1.4142i  -0.0000 + 1.0000i  -0.0000 - 1.0000i |

L2:

|  |  |
| --- | --- |
| zeros = 4×1 complex  -1.0000 + 0.0000i  -1.0000 - 0.0000i  1.0000 + 0.0000i  1.0000 + 0.0000i | poles = 6×1  0  0  -1.4142  -1.0000  1.4142  1.0000 |

L3:

|  |  |
| --- | --- |
| zeros = 4×1 complex  -0.0000 + 1.4142i  -0.0000 - 1.4142i  -0.0000 + 1.0000i  -0.0000 - 1.0000i | poles = 6×1 complex  0.0000 + 0.0000i  0.0000 + 0.0000i  0.0000 + 1.8113i  0.0000 - 1.8113i  -0.0000 + 1.1042i  -0.0000 - 1.1042i |

L4:

|  |  |
| --- | --- |
| zeros = 4×1  -1.4142  -1.0000  1.4142  1.0000 | poles = 6×1  0  0  -1.8113  -1.1042  1.8113  1.1042 |

