Homework 4

Robot Dynamics and Control

Erivelton Gualter dos Santos | 4/4/2019

# LQR design with linearized plant

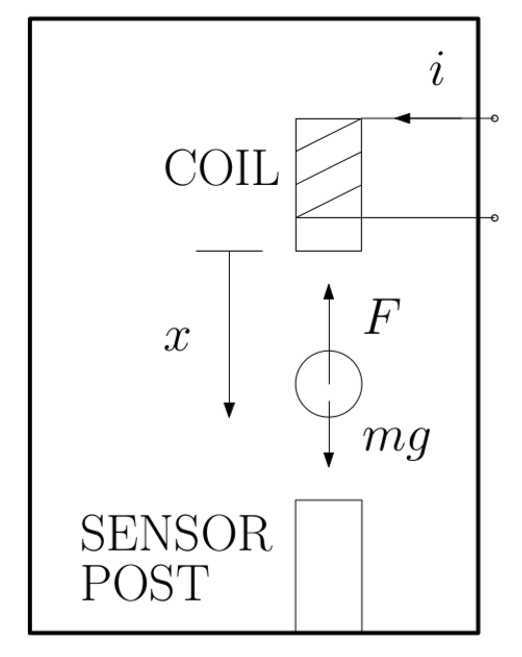
The magnetic levitation system in Figure 1 can be represented by the following differential equation:

Figure - Magnetic Levitation System

Applying Newton’s Second Law:

Where

The equilibrium point of the system can be obtained by solving the differential equation for a null acceleration:

Therefore, it results in:

In order to design a linear state feedback controller, it is necessary to find a linear representation of the nonlinear plant. The following correspond to the Taylor expansion series of the differential equation:

Which results in:

Replacing by

It results in:

In order to find the transfer function, we have:

Choosing the state variables , we have:

It can also be represented in the following state space:

For an equilibrium position as mm , we can:

% Parameters  
m = 0.068; % Mass of the ball, kg  
k = 6.5308\*1e-5; % Nm^2/A^2  
g = 9.81; % Acceleration of gravity, m/s^2  
  
kff = sqrt(2\*m\*g/k)  
  
% Equilibrium Point  
xeq = 6e-3;  
ieq = kff\*xeq

kff =  
  
 142.9291  
  
ieq =  
  
 0.8576

# Control Design by LQR

To control the system, a state feedback gain were found by using LQR solution. The parameters for *Q* and *R* are:

Which results in the following feedback gain:

K =

-285.8932 -5.0983

Eigenvalues =

-46.8986

-69.7420

The linear and nonlinear plant were simulated for the found feedback gain. The following pictures illustrates the block diagram for both simulation.

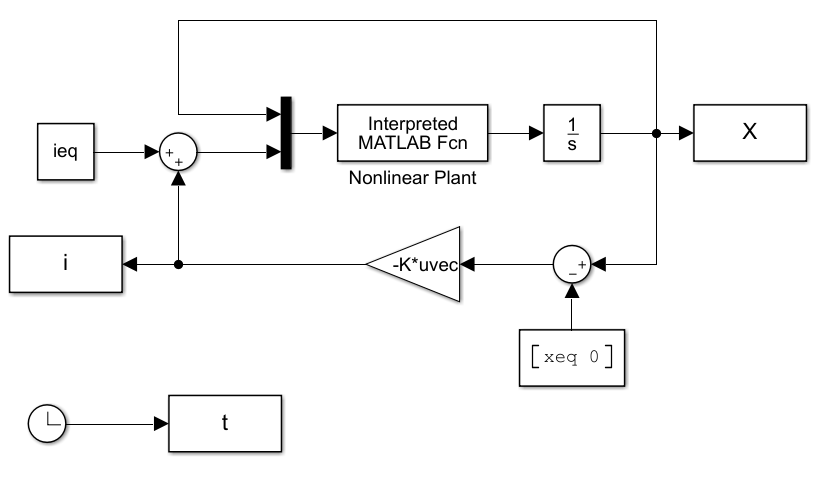


Figure - Block diagram for Nonlinear Plant

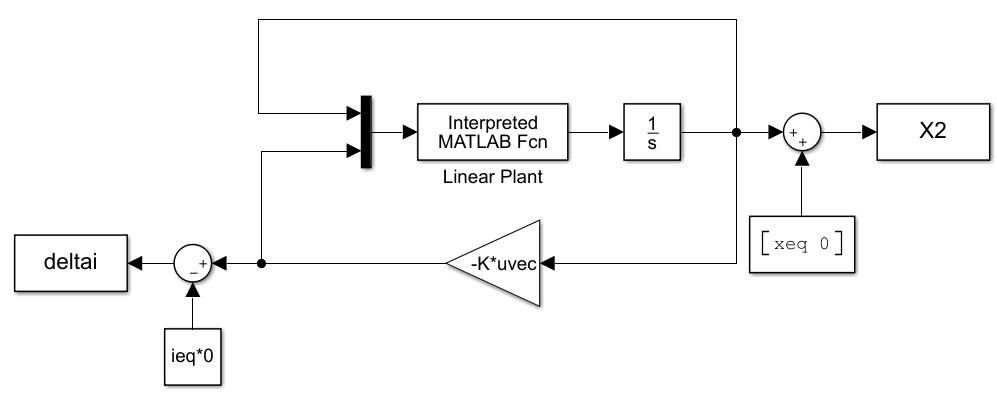


Figure - Block Diagram for Linear Plant

Since, the feedback gain were determined for the linear plant, it is necessary to shift the output and control input for the equilibrium points. The following picture shows the state response and control input.

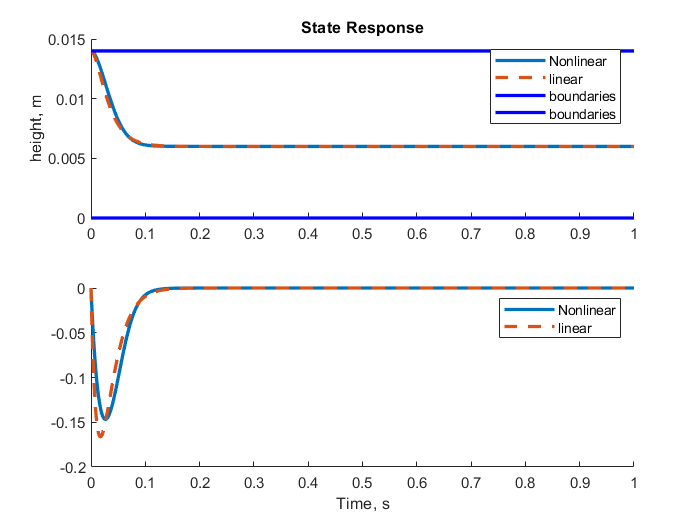


Figure - State response for an initial position at 0.014 mm

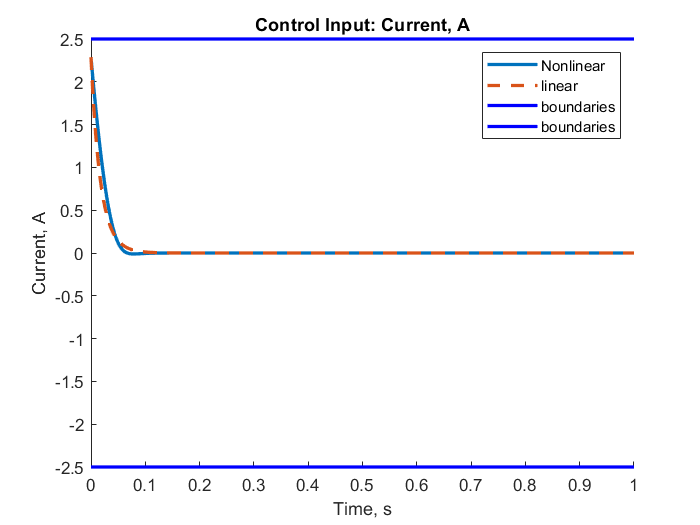


Figure - Control Input

Note that the control is between the current limit. Additionally, the nonlinear and linear system presents a slight difference until 0.1 s, then after that is almost the same.

# Quadratic Lyapunov Function and Region of Attraction

The chosen Lyapunov Function is , where P is found by using the closed loop A matrix ( and solving the Llyapunov equation:

For a identity Q, we have the following contour map. The yellow color corresponds to the negative and blue corresponds to the positive . Note, Lyapunov method is only a sufficient condition and not necessarily show the true stability of the system. For instance, according to the Lyapunov method, needs to be negative semi definite or negative definite to be stable. However, for the following contour map, the green region should be stable and yellow should not be stable. Knowing that the black dots correspond to the initial condition it results in an unstable response, and the red dots correspond to the initial condition which results in a stable response, it is clear that the chosen Lyapunov function presents a large area.

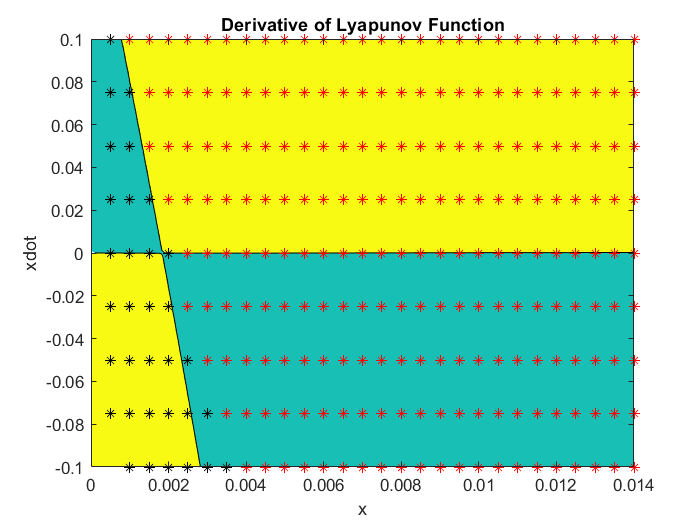


Figure Derivative of Lyapunov Function. Yellow region is negative, Blue region is Positive. Red Dots correspond to initial position which results in a stable response. Black dots correspond to initial condition which results in unstable response.

In order to find a better Lyapunov function, *Q* matrix were found by minimizing the trace of matrix *P,* which results in the following counter map. Note, that it still presents some regions that not represent the true stability; however, it covers a bigger region than the previous Lyapunov function. One way to mitigate it, is finding a different Lyapunov function more evolved, instead of a quadratic function.

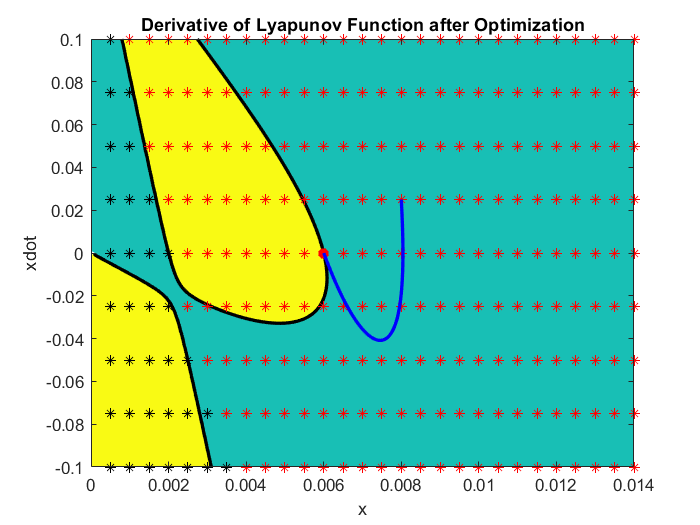
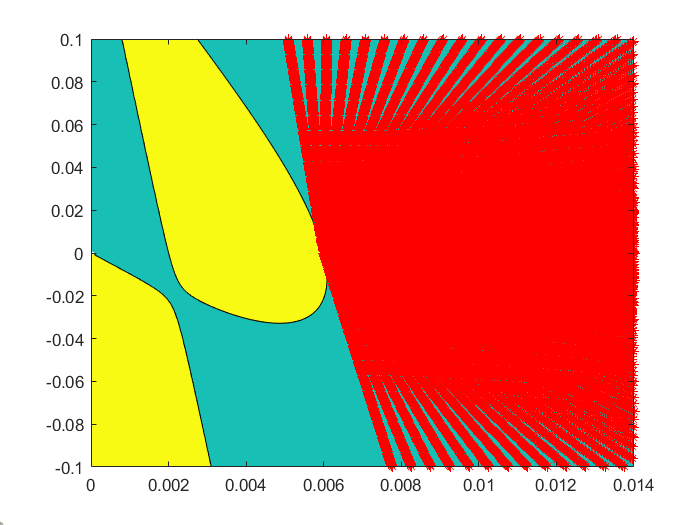
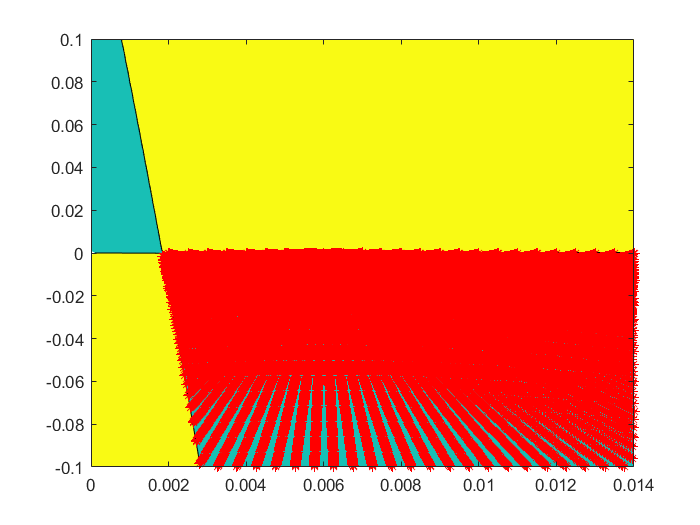


Figure - Derivative of Lyapunov Function. Yellow region is negative, Blue region is Positive. Red Dots correspond to initial position which results in a stable response. Black dots correspond to initial condition which results in unstable response.

Following figures show the polar scan from the equilibrium point:



# Codes

## Part I

Simulation

m = 0.068; % Mass of the ball, kg  
k = 6.5308\*1e-5; % Nm^2/A^2  
g = 9.81; % Acceleration of gravity, m/s^2  
param = [m; k; g];  
  
kff = sqrt(2\*m\*g/k)  
  
% Equilibrium Point  
xeq = 6e-3;  
ieq = kff\*xeq  
  
% Initial Condition  
X0 = [14e-3; 0 ];  
  
% State-Space Representation  
A = [0 1; 2\*g/xeq 0];  
B = [0; -2\*g/ieq];  
% Alternative:  
% A = [0 1; k\*ieq^2/xeq^3/m 0];  
% B = [0; -k\*ieq/xeq^2/m];  
  
% LQR Controller  
Q = diag([10 1]);  
R = 1;  
K = lqr(A,B,Q,R)  
  
% Simulation  
sim('maglev.slx',1);  
  
% Plots  
figure('Name','Nonlinear and Linear State Response');  
ax1 = subplot(211); hold on; plot(t, X(:,1), 'linewidth', 2); plot(t, X2(:,1), '--', 'linewidth', 2); title('State Response');  
ax2 = subplot(212); hold on; plot(t, X(:,2), 'linewidth', 2); plot(t, X2(:,2), '--', 'linewidth', 2);  
legend('Nonlinear','linear');  
  
% Plot boundaries  
plot(ax1, t,zeros(size(t)), '-b', 'linewidth', 2); plot(ax1, t, 0.014\*ones(size(t)), '-b', 'linewidth', 2)  
legend(ax1, 'Nonlinear','linear','boundaries','boundaries');  
  
figure('Name','Control Input');  
title('Control Input: Current, A');  
hold on; plot(t, i, 'linewidth', 2); plot(t, deltai, '--', 'linewidth', 2); legend('Nonlinear','linear');  
  
% Plot boundaries  
plot(t,-2.5\*ones(size(t)), '-b', 'linewidth', 2); plot(t, 2.5\*ones(size(t)), '-b', 'linewidth', 2)  
legend('Nonlinear','linear','boundaries','boundaries');

## Part II

Quadratic Lyapunov Function and Region of Attraction

Q = diag([1 1]); % Worse  
Q = getBestQ(); % Best  
Acl = A-B\*K;  
P = lyap(Acl',Q);  
  
syms x1 x2  
V = 0.5\*[x1 x2]\*P\*[x1; x2];  
V\_sym = matlabFunction(V, 'vars', [x1,x2]);  
  
u = ieq - K\*[x1-xeq; x2];  
x1d = x2;  
x2d = (m\*g-k.\*u^2/2/x1^2)/m;  
  
Vdot = diff(V,x1)\*x1d + diff(V,x2)\*x2d;  
Vdot\_sym = matlabFunction(Vdot, 'vars', [x1,x2]);  
  
[X1, X2] = meshgrid(0:1e-4:.014, -.1:1e-3:.1);  
Vdot = Vdot\_sym(X1, X2);  
  
figure;  
contourf(X1,X2,Vdot, [-Inf 0]); hold on;  
  
for x10 = .0005:.0005:.014  
 for x20 = -0.1:0.025:0.1  
 X0 = [x10; x20];  
 sim('maglev.slx',1);  
 if ((abs(X(end,1)) - 6e-3) < 1e-3)  
 % plot(X(:,1), X(:,2), '--', 'linewidth', 2);  
 plot(x10, x20, 'r\*')  
 else  
 plot(x10, x20, 'k\*')  
 end  
 end  
end  
x10 = 8e-3;  
x20 = 25e-3;  
X0 = [x10; x20];  
sim('maglev.slx',1);  
plot(X(:,1), X(:,2));  
  
% Polar Scan  
P1 = [];  
P2 = [];  
for th = 0:.005:2\*pi  
 for r = 0:.0005:0.1  
 x1 = r\*cos(th)+6e-3;  
 x2 = r\*sin(th);  
 vdot = Vdot\_sym(x1,x2);  
 if vdot > 0  
 P1 = [P1 [x1;x2]];  
 break;  
 else  
 P2 = [P2 [x1;x2]];  
 end  
 end  
end  
plot(P2(1,:),P2(2,:), 'r\*');  
xlim([0 .014])

## State Derivative Function

function xDer = maglevDer(X, u, param)  
 % Extract parameters  
 m = param(1);  
 k = param(2);  
 g = param(3);  
  
 x1 = X(1);  
 x2 = X(2);  
  
 x1D = x2;  
 x2D = (m\*g-k\*u^2/2/x1^2)/m;  
  
 xDer = [x1D; x2D];  
end

## Optimization

function [Qnew] = getBestQ()  
  
 % Simulation  
 m = 0.068; % Mass of the ball, kg  
 k = 6.5308\*1e-5; % Nm^2/A^2  
 g = 9.81; % Acceleration of gravity, m/s^2  
 kff = sqrt(2\*m\*g/k);  
  
 % Equilibrium Point  
 xeq = 6e-3;  
 ieq = kff\*xeq;  
  
 % Plant  
 A = [0 1; 2\*g/xeq 0];  
 B = [0; -2\*g/ieq];  
 C = eye(2);  
 D = 0;  
  
 % LQR Controller  
 K = lqr(A, B, diag([10 1]),1);  
 Acl = A-B\*K;  
  
 % Lyapunov Function  
 Q = eye(2);  
 P = dlyap(Acl', Q);  
  
 nonlcon = @(X) nonlconstraints (X, Acl);  
 x0 = reshape(Q,4,1);  
  
 options = struct('MaxFunctionEvaluations', 1000, 'MaxIterations', 1000);  
  
 fun = @(X) objFunction (X, Acl);  
 A = [];  
 b = [];  
 Aeq = [];  
 beq = [];  
 lb = [];  
 ub = [];  
 x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options);  
  
 Qnew = reshape(x,2,2);  
  
  
function out = objFunction (X, Acl)  
 Q = reshape(X, 2,2);  
 P = lyap(Acl', Q);  
 out = trace(P);  
  
function [c, ceq] = nonlconstraints (X, Acl)  
 c = [];  
 Q = reshape(X, 2,2);  
  
 P = lyap(Acl', Q);  
 gama1 = sqrt([1 0]\*inv(P)\*[1 0]');  
 gama3 = sqrt([0 1]\*inv(P)\*[0 1]');  
  
 ceq = X(2)-X(3);  
 c = [c; -Q(1,1); -Q(1,1)\*Q(2,2)+Q(2,1)\*Q(1,2); gama1-1; gama3-1];