Linearization, Gain Scheduling and Regions of Attraction

by - Akash Sharma

- Yash Malandkar

Exercise 1 : Linearization

```
clear all
clc
close all

% Creating a structure for the system's physical properties
drone = struct('m_d',1,'m_c',1,'l',1,'l_d',1,'J',1,'C_D',0.01,'g',10);

syms x u w pn pd T1 T2 vn vd theta thetad gamma gammad wn wd

x = [pn ; pd ; vn ; vd ; theta ; thetad ; gamma ; gammad];
u = [T1 ; T2];
w = [wn ; wd];
x_dot = chardonnay_dynamics(x,u,w,drone);
x_dot = x_dot(3:8);
x = x(3:8);

% Defining trim point
xt = zeros(6,1);
ut = 10*ones(2,1);
wt = zeros(2,1);

% Creating a Jacobian
A = jacobian(x_dot,x);
B = jacobian(x_dot,u);
E = jacobian(x_dot,w);

% Substituting Trim points in the Jacobian Matrix to get linearized model coefficients
At = subs(A,[x ; u ; w],[xt ; ut ; wt])
```

At =

$$\begin{pmatrix} -\frac{1}{100} & 0 & -10 & 0 & 10 & 0 \\ 0 & -\frac{1}{200} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{100} & 0 & 0 & 0 & 20 & 0 \end{pmatrix}$$

```
Bt = subs(B,[x ; u ; w],[xt ; ut ; wt])
Bt =
```

$$\begin{pmatrix} 0 & 0 \\ -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \\ 1 & -1 \end{pmatrix}$$

```
Et = subs(E,[x ; u ; w],[xt ; ut ; wt])
```

```
Et =
```

$$\begin{pmatrix} \frac{1}{100} & 0 \\ 0 & \frac{1}{200} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{100} & 0 \end{pmatrix}$$

```

% Finding Delta x_dot
delta_x_dot = At*(x - xt) + Bt*(u - ut) + Et*(w - wt);
```

Answer:

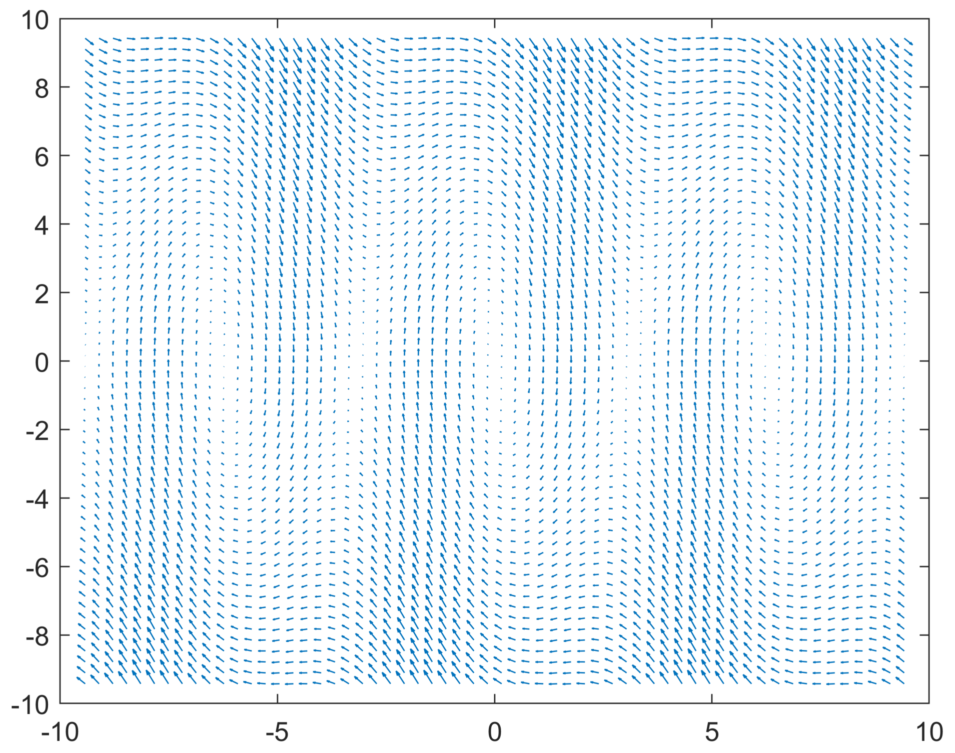
We have choosen the taylor expansion technique for linearizing the model. This method was selected because of its simplicity in terms of coding, which in complex step method would have required us to create multiple for loops. Also, taylor series expansion is efficient in linearizing a multi variable function which in this case suits us well.

The values of A, B, E are termed as At, Bt, Et which can be seen above.

Exercise 2 : Phase Planes and Linearization

```
clear all
clc
close all

% Creating a structure of the simple pendulum
pendulum = struct('g',10,'l',1,'b',1,'m',1);

% Phase Portrait

input = 0;
```

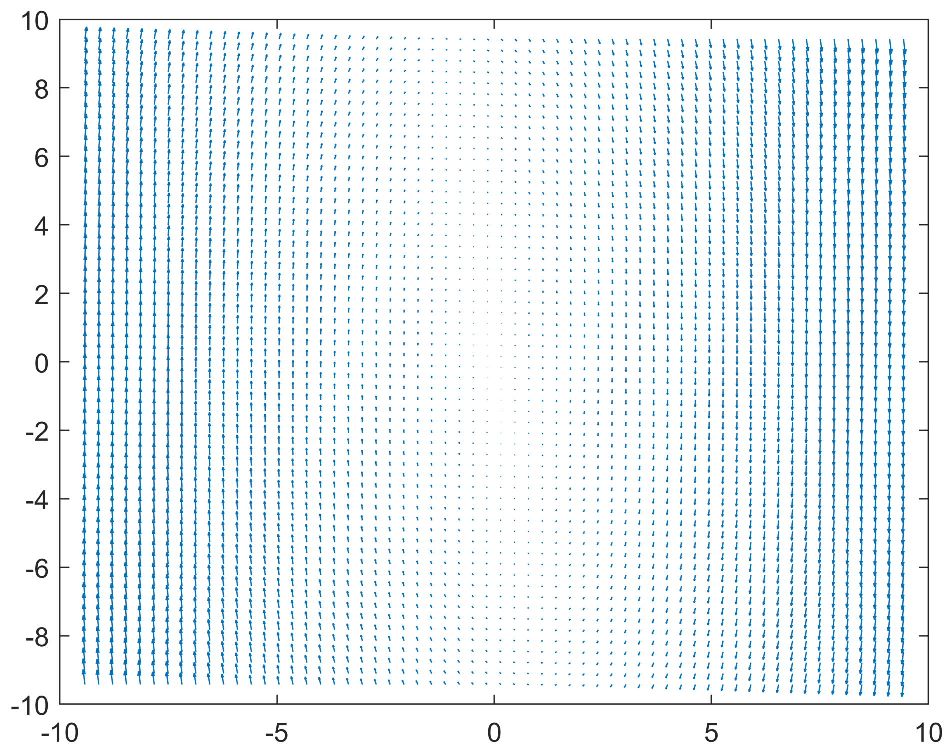
```
scale = 1; % quiver visualization parameter
N = 60; % quiver visualization parameter
n = 3; % quiver visualization parameter
theta = linspace(-n*pi,n*pi,N);
thetad = linspace(-n*pi,n*pi,N);

% Creating a coordinate matrix for the quiver plot
[X,Y] = meshgrid(theta,thetad);

%Change rates in X and Y coordinates respectively
U = zeros(size(X));
V = zeros(size(Y));

for i = 1:numel(X)
        xprime = pendulum_dynamics([X(i) ; Y(i)],input,pendulum);
        U(i) = xprime(1);
        V(i) = xprime(2);
end

quiver(X,Y,U,V,scale)
```



Model Linearization and Phase Portrait

```
syms x u theta thetad

x = [theta ; thetad];
```

```
xt = [0 ; 0];
ut = 0;

x_dot = pendulum_dynamics(x,u,pendulum);

% Linearizing the model
A = jacobian(x_dot,x);
B = jacobian(x_dot,u);

At = subs(A,[x ; u],[xt ; ut]);
Bt = subs(B,[x ; u],[xt ; ut]);

delta_x_dot = At*(x - xt) + Bt*(u - ut);

for i = 1:numel(X)
    xprime = subs(delta_x_dot,[x ; u],[[X(i) ; Y(i)] ; 0]);
    U(i) = xprime(1);
    V(i) = xprime(2);
end

% Plotting the data
quiver(X,Y,U,V,scale)
```



Answer:

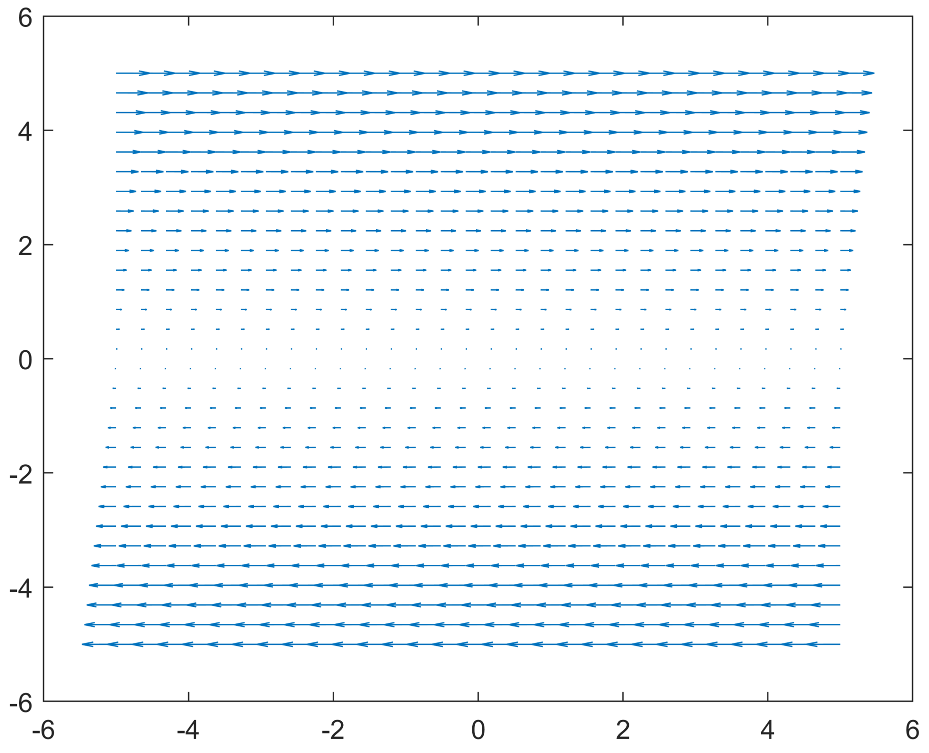
The difference between the non-linear and linear model's quiver plot is that the non-linear plot shows all possible trim points or theta for that model i.e. thetad = 0, whereas the linear model plot just shows the variation of theta and thetad around the trim point i.e. theta=0 at which it was linearlized.

Also, whem the trim point is changed from 0 to pi, a shift in the quiver plot was noticed. The plot now shows the variation of theta and thetad around the new trim point i.e. theta=pi.

Althought both the linearized models trimed at theta=0 and theta=pi are different models individually.

Exercise 3 : Phase Portraits and Regions of Attractions

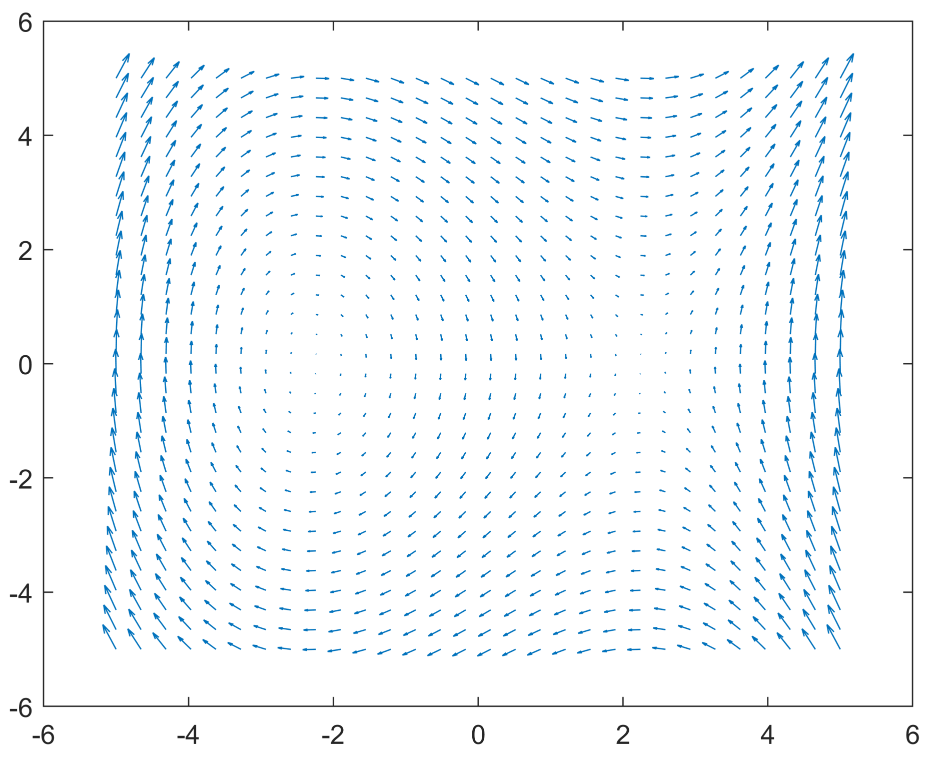
```
clear all
clc
close all

params = struct('alpha',1,'kp',100,'kd',10000);

N = 30;
scale = 1;

x = linspace(-5,5,N);
xd = linspace(-5,5,N);


[X,Y] = meshgrid(x,xd);

U = zeros(size(X));
V = zeros(size(Y));


for i = 1:numel(X)
        xprime = NL_system([X(i) ; Y(i)],params);
        U(i) = xprime(1);
        V(i) = xprime(2);
end

quiver(X,Y,U,V,scale)
```



```
% the commented section below is done for linearizing the model.
%{
syms X x xd

X = [x ; xd];

Xt = [0 ; 0];

X_dot = NL_system(X,params);

[Xq,Yq] = meshgrid(linspace(-5,5,N),linspace(-5,5,N));

U = zeros(size(Xq));
V = zeros(size(Yq));

A = jacobian(X_dot,X);

At = subs(A,X,Xt);

delta_X_dot = At*(X - Xt);

for i = 1:numel(Xq)
    xprime = subs(delta_X_dot,[x ; xd],[Xq(i) ; Yq(i)]);
    U(i) = xprime(1);
    V(i) = xprime(2);
end

subplot(2,1,2)
quiver(Xq,Yq,U,V,scale)
%}
```

Same Code with another Kp and Kd:

```
clear all
clc
close all

params = struct('alpha',1,'kp',16,'kd',4);

N = 30;
scale = 1;

x = linspace(-5,5,N);
xd = linspace(-5,5,N);


[X,Y] = meshgrid(x,xd);

U = zeros(size(X));
V = zeros(size(Y));


for i = 1:numel(X)
        xprime = NL_system([X(i) ; Y(i)],params);
        U(i) = xprime(1);
        V(i) = xprime(2);
end

quiver(X,Y,U,V,scale)
```



Answer:

The controller is locally stable. The region of attraction is at xdot=0 for a wide range of x. This range of x depends on the value of Kd while Kp decides how far the adjacent trim points are from each other.

Expression:

$$x = \pm\sqrt{\frac{K_p}{\alpha}}$$

Functions

```
function x_dot = pendulum_dynamics(x,u,pendulum) % Dynamics for the pendulum
% x = [theta ; thetad];

g = pendulum.g;
l = pendulum.l;
b = pendulum.b;
m = pendulum.b;

x_dot = [x(2) ; (-g/l)*sin(x(1)) - (b/(m*l^2))*x(2) + 1/(m*l^2)*u];

end
```

```matlab
function X_dot = NL_system(X,params) % Dynamics for the last question
% X = [x ; xd]

alpha = params.alpha;
kp = params.kp;
kd = params.kd;

X_dot = [X(2) ; (3*alpha/(1+kd))*X(1)^2 - kp/(1+kd)];

end
```