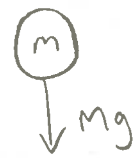
ME 292B | HW #3 | Shawn Marshall-Spitzbart

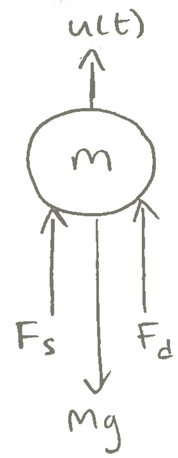
1(a). FBD of Mass in Flight

Assumptions

- Spring and damper are massless.
- Ideal spring



FBD of mass in Stance ($\dot{y} < 0$ stage)



Flight Dynamics

- $\Sigma F = m\ddot{y} \longrightarrow -mg = m\ddot{y} \Rightarrow \ddot{y} = -g$

Stance Dynamics

- $\Sigma F = m\ddot{y} \longrightarrow u(t) + F_s + F_d - mg = m\ddot{y}$
- Expanding we get,

$$u(t) - k(y - l_0) - c\dot{y} - mg = m\ddot{y}$$

- Subsituting x, and solving for $\ddot{y}$ gives,

$$\ddot{y} = \frac{u(t) - k(x_1 - l_0) - cx_2 - mg}{m}$$

$$\therefore \boxed{\ddot{y} = \begin{cases} -g, & \text{if } y > l_0 \\ \dfrac{u(t) - k(x_1 - l_0) - cx_2 - mg}{m}, & \text{if } y \leq l_0 \end{cases}}$$

where $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$

1(b).

For $y > l_0$, $\ddot{y} = -g \rightarrow \int_{y_0}^{y} \ddot{y}\, dy = \int_0^t -g\, dt$

- Integrating we get, $\dot{y} = -gt + \dot{y}_0$
- Assuming $\dot{y}_0 = \emptyset$ and energy conservation, apex could be found when $t = \emptyset$ (defining X_0 for Poincaré surface)

$\therefore$ $\dot{y}$ @ apex $= \emptyset$

- Integrating again we get, $y = -\frac{gt^2}{2} + \dot{y}_0 t + y_0$
- Using same assumptions as before, we find y @ apex from setting $t = \emptyset$

$\therefore$ y @ apex = constant = y (initial height)

$$\boxed{\therefore P(x) = \begin{bmatrix} y \\ \emptyset \end{bmatrix}}$$

1(c).

There is one fixed point at $x^* = \begin{bmatrix} y \\ \emptyset \end{bmatrix}$

- This is truly a fixed point since, $P(x^*) = x^*$ (with energy conservation)
- For linear approximation, $P(x)$ is already linear so this is not applicable.
- x^* is stable iff $\left| \lambda_i \left(\frac{\partial P}{\partial x}\Big|_{x^*} \right) \right| < 1$
- $\frac{\partial P}{\partial x}\Big|_{x^*} = \begin{bmatrix} 1 & \emptyset \\ \emptyset & \emptyset \end{bmatrix}$
- The eigenvalues of $\frac{\partial P}{\partial x}\Big|_{x^*}$ are $\emptyset$ and 1

$\boxed{\therefore \text{ The fixed point is stable in the sense of Lyapunov}}$

```
clc
close all
clear all
```

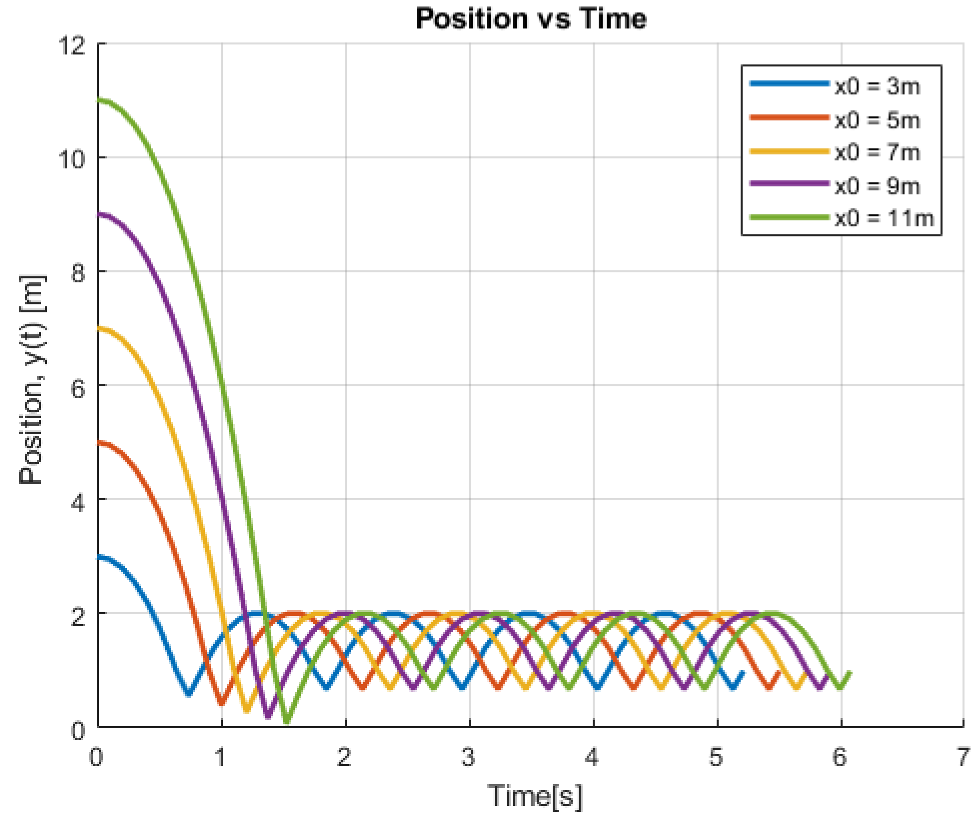
ME 292B - Homework #3 Problem 1

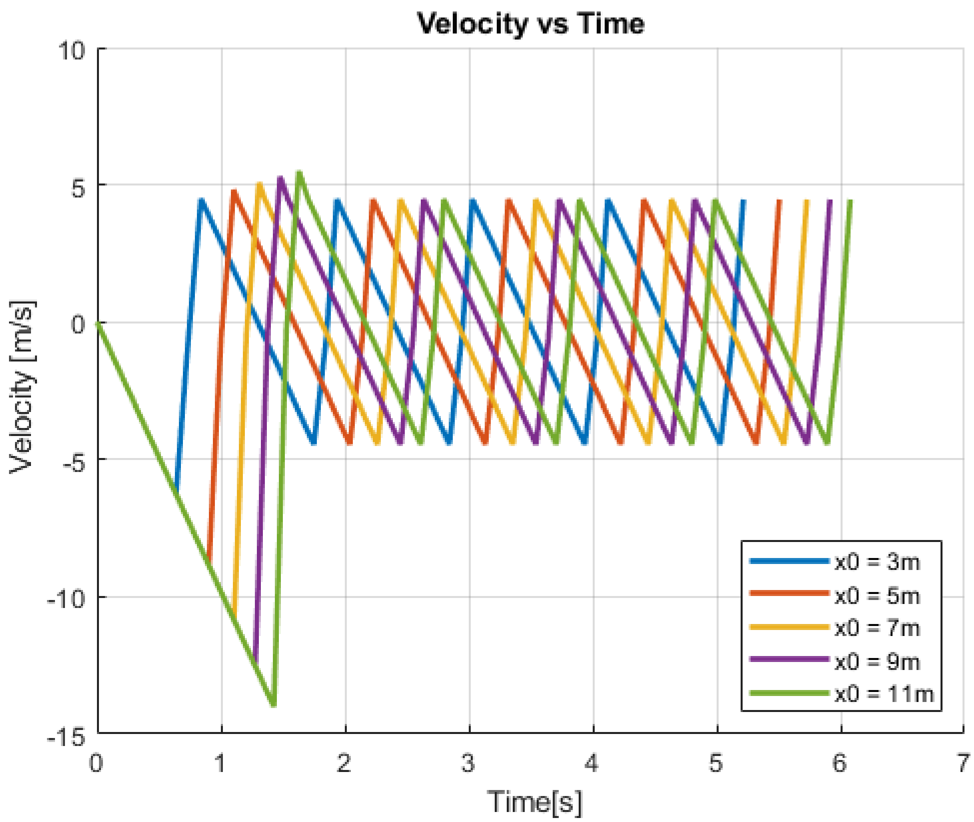
Problem 1(d)

```
% Constants and input defined in stance_dynamics function

% Simulate controlled system for 5 ICs
figure(1)
hold on
figure(2)
hold on
for i = 1:5
sim_10_bounces([2*i + 1;0])
end
figure(1)
title('Position vs Time')
xlabel(' Time[s]')
ylabel(' Position, y(t) [m]')
legend('x0 = 3m', 'x0 = 5m', 'x0 = 7m', 'x0 = 9m', 'x0 = 11m')
figure(2)
title('Velocity vs Time')
xlabel(' Time[s]')
ylabel(' Velocity [m/s]')
legend('x0 = 3m', 'x0 = 5m', 'x0 = 7m', 'x0 = 9m', 'x0 = 11m',...
    'Location','southeast')

function [] = sim_10_bounces(x0)
% Simulate 10 bounces

% Define time range to simulate the system
Tspan = linspace(0,10,100) ;
t0 = 0 ; % Initial Time

% Array to store data for all bounces
t_vec = [] ; x_vec = [] ;

for j=1:5

    % Define the events functions (stop integration when stance stage
 reached)
    options1 = odeset('Events', @ground_contact) ;

    % Simulate the system in flight
    [t_ode x_ode] = ode45(@flight_dynamics, t0+Tspan, x0, options1) ;

    % Save simulation data
    t_vec = [t_vec; t_ode] ;
```

```
    x_vec = [x_vec; x_ode] ;

    % Initialize xo,t for stance stage
    x0 = [x_ode(length(x_ode),1); x_ode(length(x_ode),2)]; t0 =
 t_vec(end);

    % Define the events functions (stop integration when flight stage
 reached)
    options2 = odeset('Events', @leaves_ground) ;

    % Compute stance dynamics

    % Constants
    k = 20000; % N/m
    m = 80; % kg
    L0 = 1; %m
    c = 50000; %Ns/m
    g = 9.81 ;

    % Design input controller , t restarts at each start of stance
 position

    % From energy conservation, we can find target velocity at start
    % of stance dynamics (1/2*m*v^2 = mgh, h = 2m - 1m  = 1m)
    v0_target = sqrt(2*g*1);

    % Calculate real time error from target velocity and actual
 velocity
    error = @(x) v0_target - x(2);

    % PD-Controller
    u = @(t,x) c*x(2) + 7500*error(x)*(x(2)>=0); % only apply u during
    % second half of stance stage (gain of 7500 found from trial/
error)

    xdot = @(t,x) [x(2) ;
    ((u(t,x)-k*(x(1)-L0)-c*x(2)-m*g)/m)] ;

    % Simulate the system in stance stage
    [t_ode x_ode] = ode45(xdot, t0+Tspan, x0, options2) ;

    % Save simulation data
    t_vec = [t_vec; t_ode] ;
    x_vec = [x_vec; x_ode] ;

    % Initialize xo,t for flight stage
    x0 = [x_ode(length(x_ode),1); x_ode(length(x_ode),2)]; t0 =
 t_vec(end);
end

% Plot position and velocity
figure(1)
plot(t_vec, x_vec(:,1), 'LineWidth',2) ; grid on ;
figure(2)
```

```
plot(t_vec, x_vec(:,2), 'LineWidth',2) ; grid on ;
end

% Function that describes flight dynamics
function dx = flight_dynamics(t, x)
    y = x(1) ;
    dy = x(2) ;
    g = 9.81 ;
    dx = [dy ;
          -g] ;
end

% Event function that describes when hopper hits the ground
function [value,isterminal,direction] = ground_contact(t,x)
    y = x(1);
    value = round(y - 1,2) ; % detect when y - L0 == 0
    isterminal = 1 ; % stop integration when y - L0 == 0
    direction = -1 ; % detect zero when function is decreasing
end

% Event function that describes when hopper leaves the ground from
 stance
function [value,isterminal,direction] = leaves_ground(t,x)
    y = x(1);
    value = round(y - 1,2) ; % detect when y - L0 == 0
    isterminal = 1 ; % stop integration when y - L0 == 0
    direction = 1 ; % detect only +ve y to -ve y transitions
end
```


Position vs Time
Position, y(t) [m]
Time[s]
x0 = 3m
x0 = 5m
x0 = 7m
x0 = 9m
x0 = 11m


Velocity vs Time
Velocity [m/s]
Time[s]
x0 = 3m
x0 = 5m
x0 = 7m
x0 = 9m
x0 = 11m

Published with MATLAB® R2019a

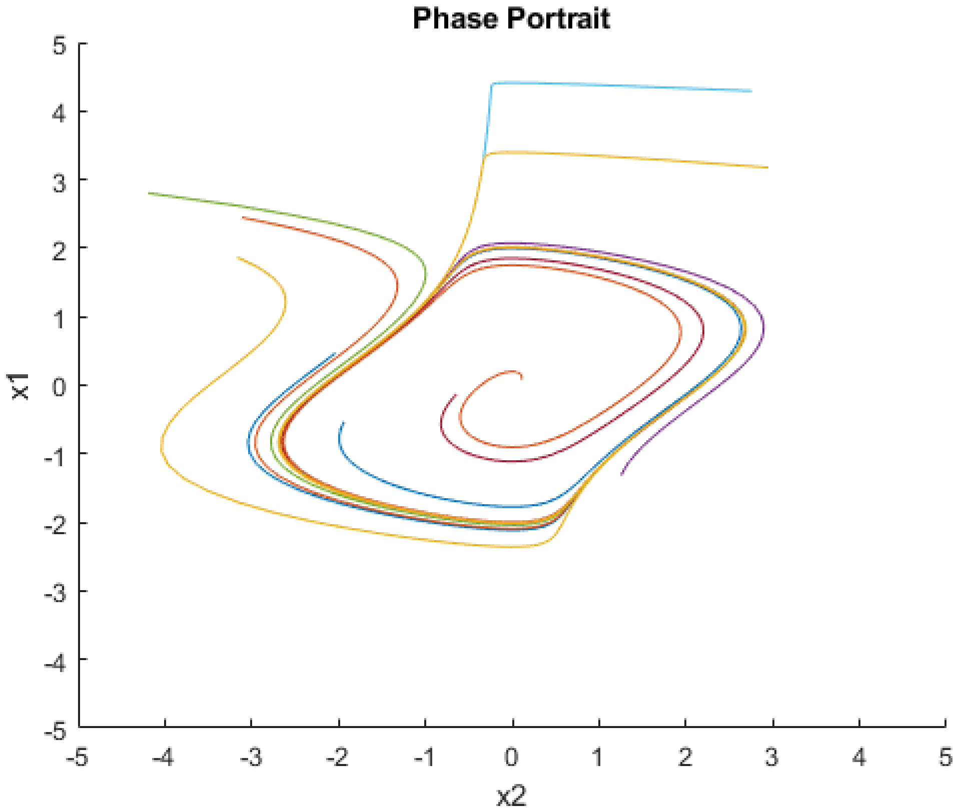
Table of Contents


```
clc
close all
clear all
```

ME 292B - Homework #3 Problem 2

Problem 2(a)

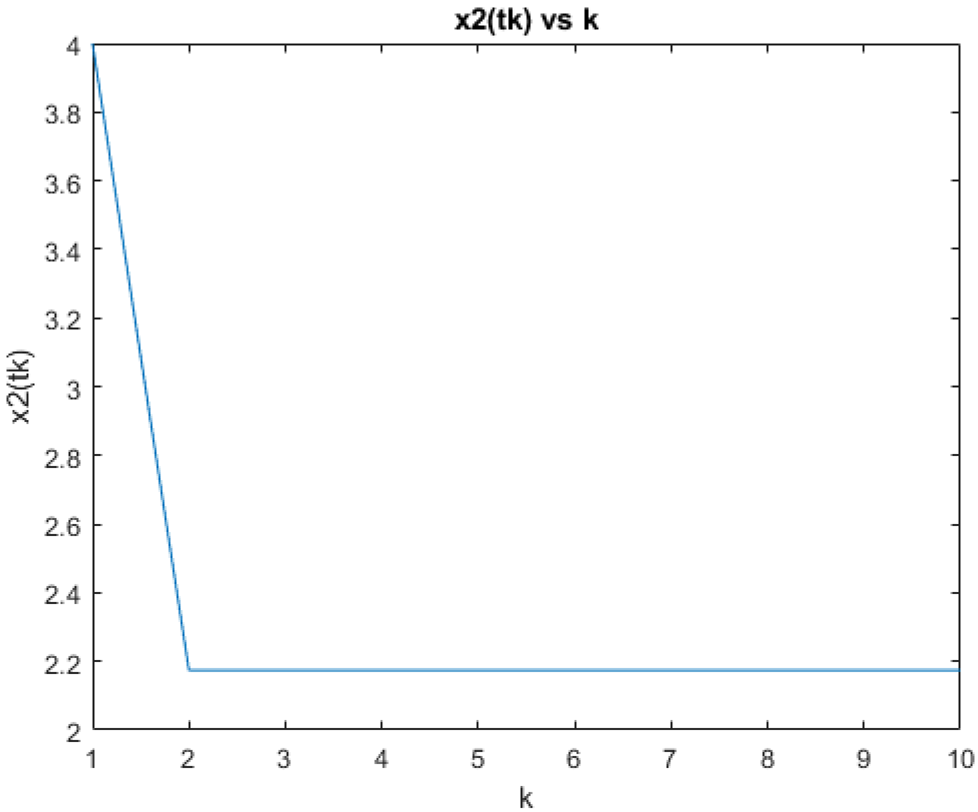
```
mu = 1;
xdot = @(t,x) [x(2); mu*(1 - x(1)^2)*x(2) - x(1)];

ICs = zeros(10,2);
for i = 1:size(ICs,1)
    for j = 1:size(ICs,2)
        ICs(i,j) = 10*rand(1) - 5;
    end
end

tspan = linspace(0,50,1000);
for i = 1:size(ICs,1)
    [time, xplot] = ode45(xdot,tspan,ICs(i,:));
    plot_struct(i).x1 = xplot(:,1);
    plot_struct(i).x2 = xplot(:,2);
    plot_struct(i).time = time;
end
figure()
hold on
axis([-5,5,-5,5])
title('Phase Portrait')
xlabel('x2')
ylabel('x1')
for i = 1:10
    plot(plot_struct(i).x2, plot_struct(i).x1)
end
```



Problem 2(b)

```
% Part i
% VanderPolPoincare is located at end of script

% Part ii
% Chose IC of [0; 4]
x0 = [0; 4];
x1 = VanderPolPoincare(x0)

% Part iii
% Obtain sequence of x's for n = 10 (x0 and x1 already obtained)
xold = x1;
xn(1,:) = x0;
xn(2,:) = x1;
for i = 3:10
    xn(i,:) = VanderPolPoincare(xold);
end

% Part iv
figure()
plot(linspace(1,10,10), xn(:,2))
title('x2(tk) vs k')
xlabel('k')
ylabel('x2(tk)')
```

```
% Show convergence to fixed point of [0 ; 2.1733]
disp(xn(end-5:end,:))
x_fixed = xn(end,:)';


x1 =

    0.0000
    2.1744

    0.0000    2.1739
    0.0000    2.1739
    0.0000    2.1739
    0.0000    2.1739
    0.0000    2.1739
    0.0000    2.1739
```



Problem 2(c)

```
% Compute A numerically using finite difference
delta = 0.01;
A = zeros(2:2);
for j = 1:2
    ej = zeros(2,1);
```

```
    ej(j,1) = 1;
    A(:,j) = ( VanderPolPoincare(x_fixed + delta*ej)...
        - VanderPolPoincare(x_fixed - delta*ej) ) / (2*delta) ;
end

% Display A
A

% Find eigenvalues of A to determine stability
eigA = eig(A)
if abs(eigA) < 1
    disp('The limit cycle of the oscillator is exponentially stable')
end


% Function for 2(b)
function [x1] = VanderPolPoincare(x0)

    mu = 1;
    xdot = @(t,x) [x(2); mu*(1 - x(1)^2)*x(2) - x(1)];

    % Define the events function (stop integration after one complete
 cycle)
    options = odeset('Events', @cycle) ;

    % Define time range to simulate the system
    Tspan = linspace(0,100,10000) ;
    t0 = 0 ; % Initial Time

    % Simulate system
    [t x] = ode45(xdot, t0+Tspan, x0, options) ;

    x1 = [x(end,1);x(end,2)];

    function [value,isterminal,direction] = cycle(t,x)
    value = x(1) ; % detect when x1 == 0
    isterminal = 1 ; % stop integration when y == 0
    direction = 1 ; % can only approach zero while increasing (ccw)
    end
end
```

```
A =

    0.0000   -0.0000
   -0.4795    0.0374


eigA =

    0.0000
    0.0374

The limit cycle of the oscillator is exponentially stable
```

Published with MATLAB® R2019a