# METR7203 Problem Based Assignment 2

Callum Rohweder s4357594

August 2017

## 1 Question 1

The inverted pendulum on a cart system is illustrated in figure 1. It can be seen that the coordinates of interest for this are p (linear position), and theta (the pendulum angle with respect to the upwards vertical axis). To control this system, an input F is applied on the cart, allowing it to compensate for the movement of the pendulum; altering the cart's position.

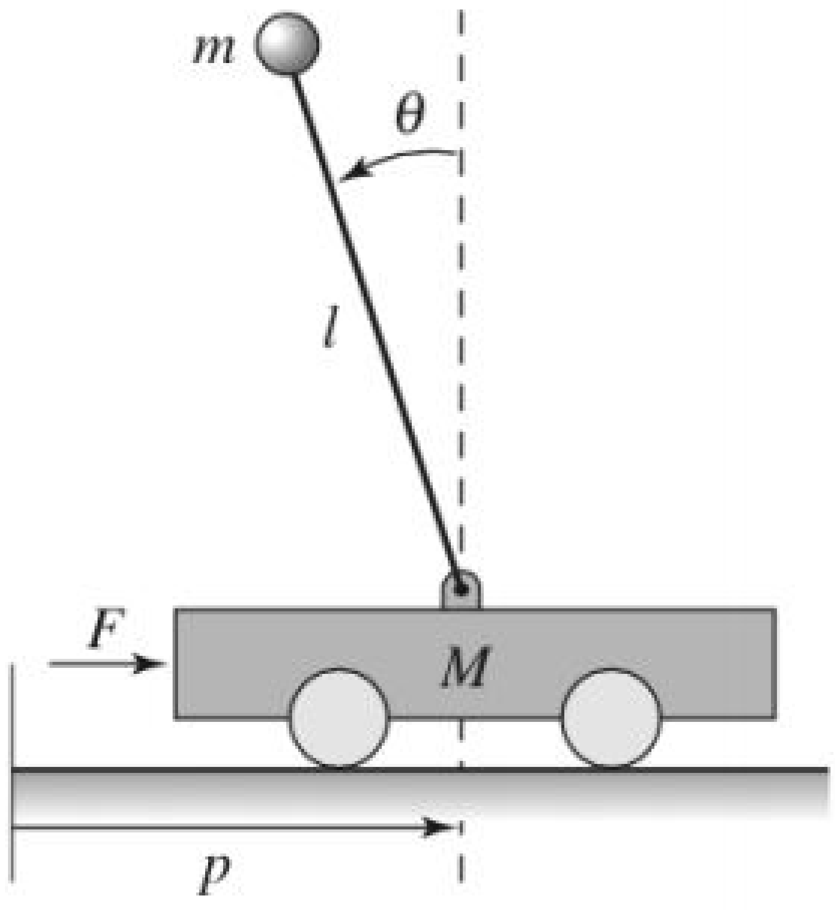

Figure 1: Inverted Pendulum on a cart diagram

// The parameters of this system are:
$\rightarrow thedampingofthependulum(\rho)$
$\rightarrow thedampingofthecart$(c)
$\rightarrow themassofthecart$(M)
$\rightarrow themassofthependulumasapointload$(m)$atdistance($
$\rightarrow theaccelerationsuretogravity$(g)
$\rightarrow thepositionofthecart$(p)$andderivatives$(p$_d$, $p_{dd}$)
$\rightarrow theangleofthependulum$(th)$andderivatives$(th$_d$, $th_{dd}$)
Using Lagranges equations, the equations of motion for this system can be expressed.

## 2 Question 2

In assignment one, a block diagram model of cruise control was made using Simulink. This question is related to that model, as it is essentially a scripted version of that simulation in a function, *cruise_control.m*, shown below. The objective in this activity was to construct a phase portrait representing the velocity and integral of the PI controller states, to show how selected initial conditions and internal functions of the controller can lead to asymptotic stability at an equilibrium; at the desired velocity.

The formulas used in calculation are represented in the matlab code, the most important equation to note is the actuator equation lambda,

$$\lambda = \begin{cases} 0 & u \leq 0 \\ u & 0 \leq u \leq 1 \\ 1 & 1 \leq u \end{cases}$$

Where u is the controller output, effectively the throttle command, and lambda is the throttle actuation. In this case, 1 represents the throttle being completely on and 0 represents no throttle. Thus, the car is limited to only going forward, as $\lambda$ and the applied torque are always positive, and there is no braking system (apart from drag, friction and gravitational forces).

The controller output, u, is made from a PI controller which corrects the error between a regulation velocity $Vr$ and the actual velocity of the car body $v$. The state variables used for the phase portrait are $v$ and $z$, where $z$ is the integral state of the PI controller and thus their derivatives are the body's acceleration, $a$, and the velocity error, $e$, respectively.

Using the Simulink model from assignment 1, the following code could be produced, which computes the derivative states of inputs $[v, z]$, as per the state-space representation of the control system $x' = F(x)$

```
function dy = cruise_control(t, y)
% computes and outputs the derivative of velocity and integral
% (of PI controller)states
%% dy = [v', z'] = [a, e];
% where v is the velocity and z is the integral state
% y = [v, z], which are the initial states
dy = zeros(2,1);    %define output
% parameters from assignment 1
Tm = 200;
wm = 450;
beta = 0.35;
alpha = {40, 35, 16, 12, 10};
g = 9.81;
Cr = 0.01;
p = 1.3;
Cd = 0.36;
A = 2.2;
% new parameters
Vr = 20;
Kp = 0.5;
Ki = 0.1;
m = 1000;
gear = 3;
theta = 0;
```

```
v = y(1);
z = y(2);

% controller
e = Vr - v;       %error between current velocity and regulated velocity
Gi = Ki * z;      %integral control component
Gp = Kp * e;      %proportional control component
u = Gp + Gi;      %controller output

% actuator
lambda = 0;
if u > 1
    lambda = 1; %max actuation / pedal to the floor
elseif u < 0
    lambda = 0; %minimum actuation / no pedal
else
    lambda = u;
end

% Trottle and Engine
alpha_ = alpha{gear};
w = alpha_*v;         % angular velocity
T = lambda*Tm*(1-beta*((w/wm)-1)^2);     %applied torque

% Applied force
F_app = alpha_*T;

% Disturbance forces
theta = theta*pi/180;
Ff = cos(theta)*m*9.81*Cr*sign(v);       %force due to friction
Fg = sin(theta)*m*9.81;                  %force due to gravity
Fa = 0.5*p*Cd*A*v^2;                     %force due to drag / air
F_d = Ff+Fg+Fa;                          %total disturbance force

% Total / Net force
F = F_app - F_d;

% Body
a = F/m;

% Outputs
dy(1) = a;        %derivative of velocity
dy(2) = e;        %derivative of integral state is current error

end
```

Script 1: cruise_control.m

Following this, a script was created to plot the states in time in time, using the state-space equation results; from *cruise_control.m*. This was done using the ode45 function, an ODE solver which outputs the response in time. A phase-portrait is a 2D graph visualising a two states change in time, where an asymptotically stable equilibrium point represents the two states coming to their steady state values; as time goes to infinity. Thus it is necessary to gather information on the state's values in time, and plotted on a phase portrait, which is done in the following matlab script:

```
%% Problem 1 script
%This script is used to obtain and display data in the form of a phase
% portrait

close all;                    % close any open figures
figure(1);                    % open figure 1
init = [1, 10]                % Initial cruise control conditions [v, z]
%init = [-100, -100]          % Initial Conditions for instability (negative
    vel)
%% compute the states in time using ODE solver
% t is time elapsed for y = [v', z']
[t, y] = ode45('cruise_control', [0,100], init);
% plot velocity in blue
plot(t, y(:,1), 'b-o');
hold on;
% plot integral state in red
plot(t, y(:,2), 'r-o');
legend('v', 'z');
title('State response in time');
xlabel('time (seconds)', 'FontSize', 20);
ylabel('State change (blue: m/s, red: m)', 'FontSize', 20);
%% Plot the phase portrait
figure(2)
x = [-10, 50, 10];            % v-axis conditions (limits)
y = [-40, 40, 10];            % z-axis conditions (limits)
phaseplot('cruise_control', x, y, 1, boxgrid(x,y))
title('Phase Portrait');
xlabel('v', 'FontSize', 20);
ylabel('z', 'FontSize', 20);

figure(3)
x = [-30, 90, 10];
y = [-100, 100, 10];
phaseplot('cruise_control', x, y, 1, boxgrid(x,y))
title('Phase Portrait');
xlabel('v', 'FontSize', 20);
ylabel('z', 'FontSize', 20);

figure(4)
x = [-600, 600, 10];
y = [-600, 600, 10];
phaseplot('cruise_control', x, y, 1, boxgrid(x,y))
title('Phase Portrait');
xlabel('v', 'FontSize', 20);
ylabel('z', 'FontSize', 20);

%% generate an interactive phase-portrait
% figure(5);
% interactivePhasePortrait('cruise_control', [-1000,1000;-1000,1000], 0,
    1, 30, 'cruise_control');
% grid on;
```

Script 2: PBA2_script.m

The response of both states in time to an input $[v, z] = [1, 10]$ is shown in the figure below, where it can be seen that both the velocity and integral controller state reach a steady-state value within 100 seconds.

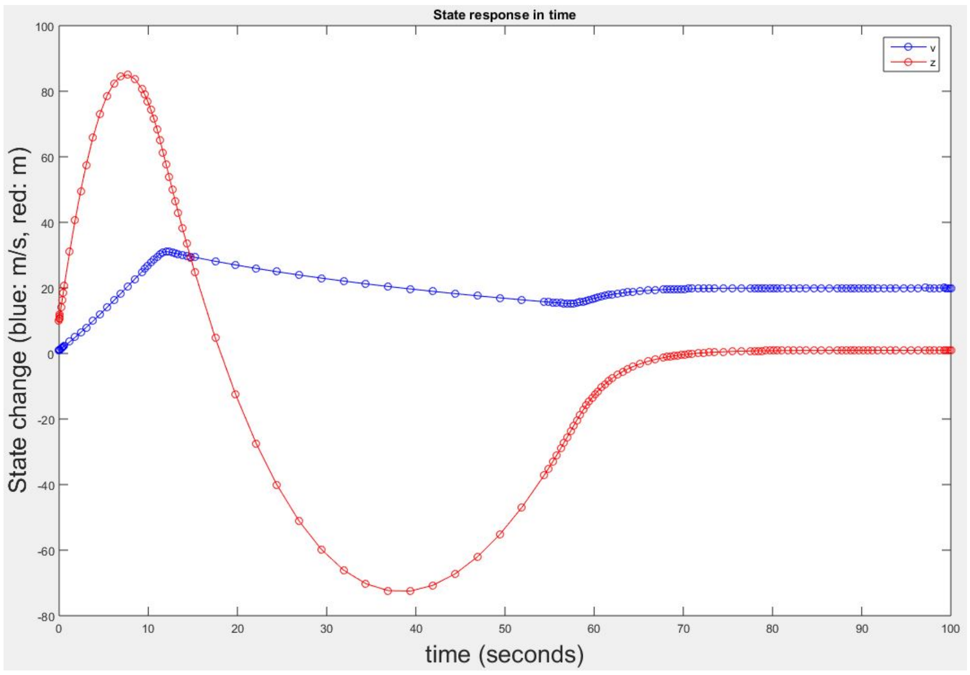

Figure 2: States change in time

Below are several views of the corresponding phase portrait; with different levels of zoom to show the effect of the controller trying to reach a desired negative velocity (remember torque can only be positive, there is no reverse) which results in instability. In addition, there is a asymptotically stable equilibrium where V = Vr (= 20) and a corresponding integral state.

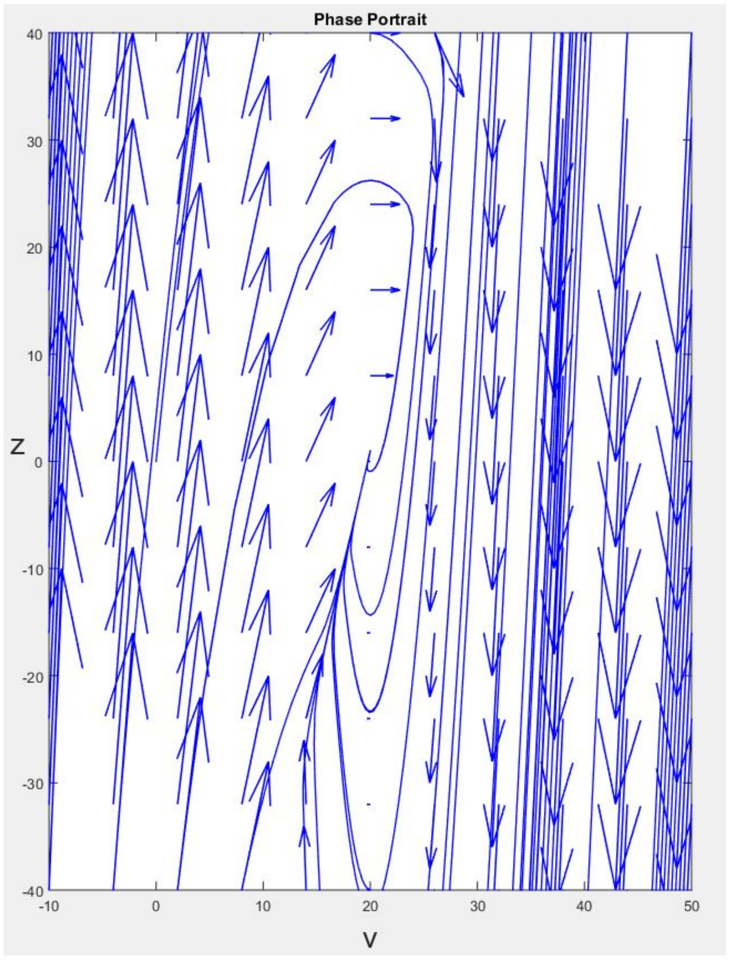

Figure 3: Phase Portrait of state variables v and z - note the stable equilibrium point at the regulated velocity

Zooming out further, it appears that the phase portrait represents a stable spiral for all initial conditions.

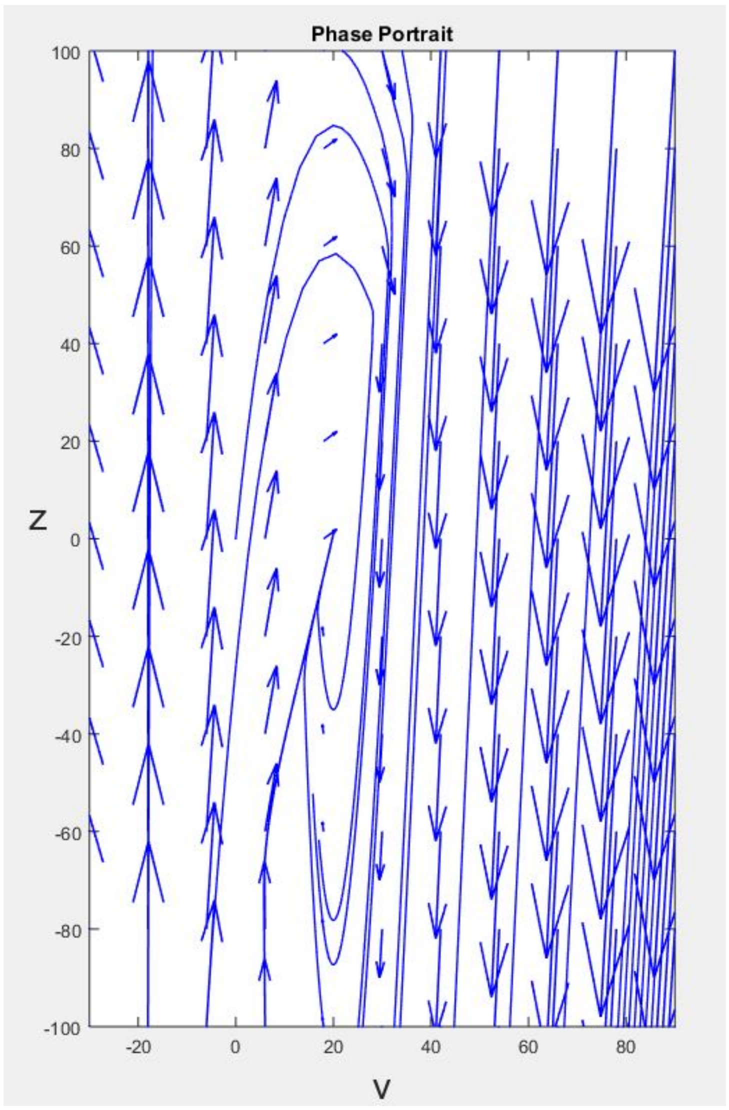

Figure 4: Phase Portrait of state variables v and z - note the stable equilibrium point at the regulated velocity

However, this is not true, it can be seen that for large velocities, where the angular velocity is exceeded for gear 3, it is indeterminable as to whether the trajectories return to the aforementioned equilibrium. This is shown in the figure below, which also shows that for negative velocities the system is unstable, as all initial conditions result in the two states growing with time.