METR7203 Problem Based Assignment 2

Callum Rohweder s4357594

August 2017

1 Question 1

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 λ 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.

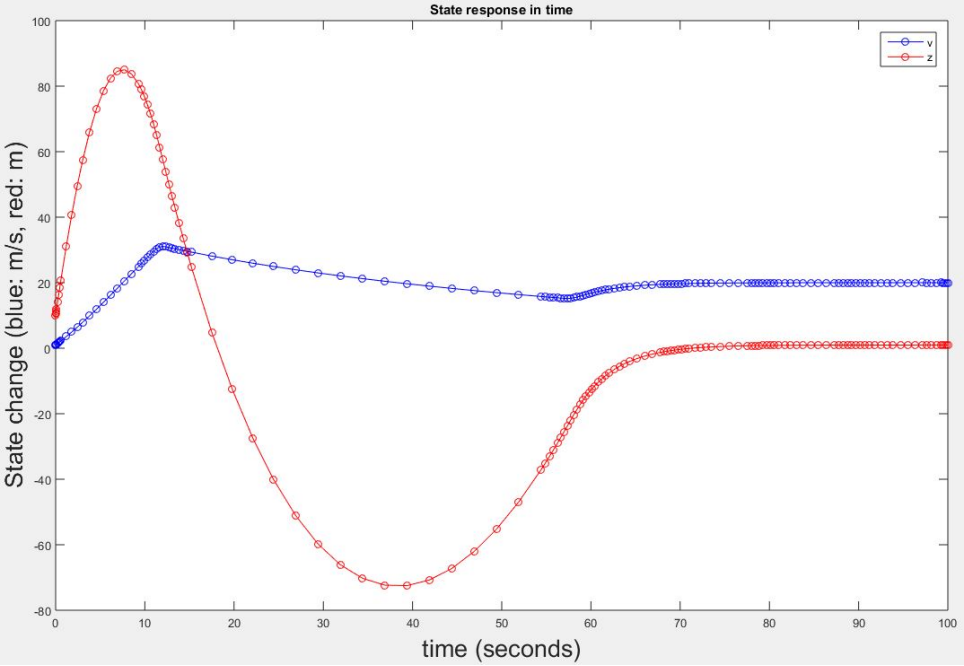

Figure 1: States change in time

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.

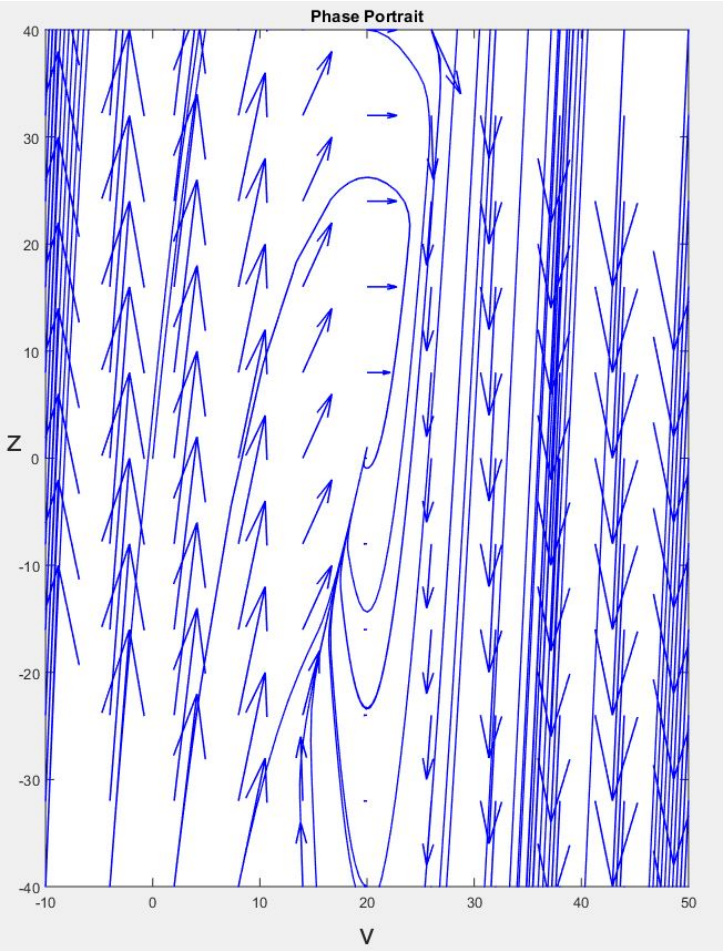

Figure 2: Phase Portrait of state variables v and z - note the stable equilibrium point at the regulated velocity

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

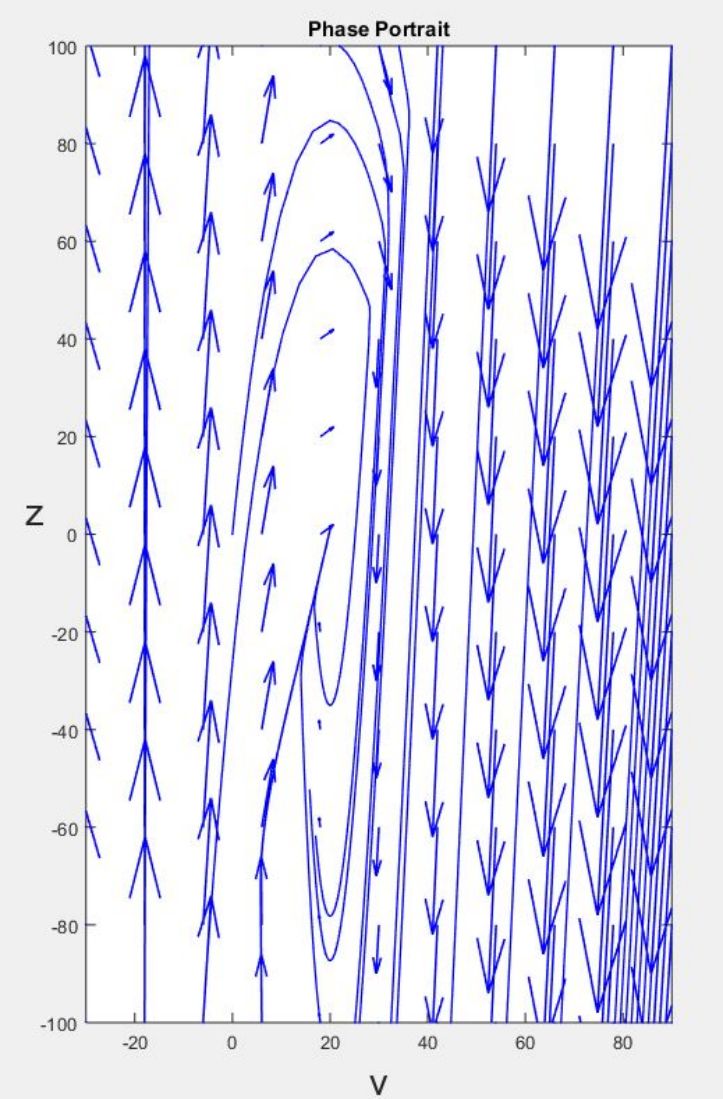

Figure 3: Phase Portrait of state variables v and z - note the stable equilibrium point at the regulated velocity

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.

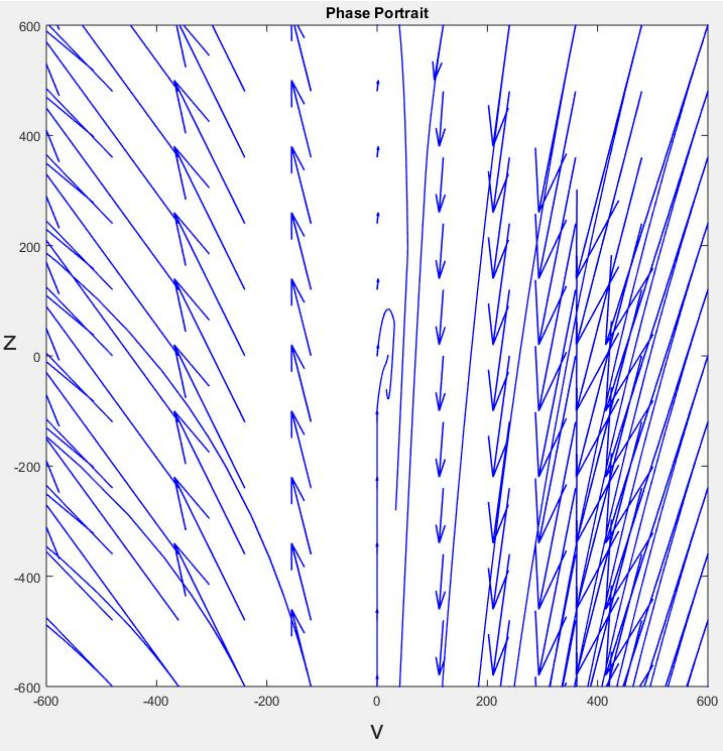

Figure 4: Phase Portrait of state variables v and z - illustrates the unstable and stable regions of the two states v and z

The position of the stable equilibrium is positioned at $v = 20$, as found when reviewing the figures above (in particular figure 1). The corresponding value of z is harder to recognise, however from vooming into figure 1 for large time (90-100 seconds), it can be seen that z is reaching a steady state value of approximately $z = 0.97$, as shown in the figure below, and recognisable in figure 2 as the point where all lines meet (and $v = 20$).

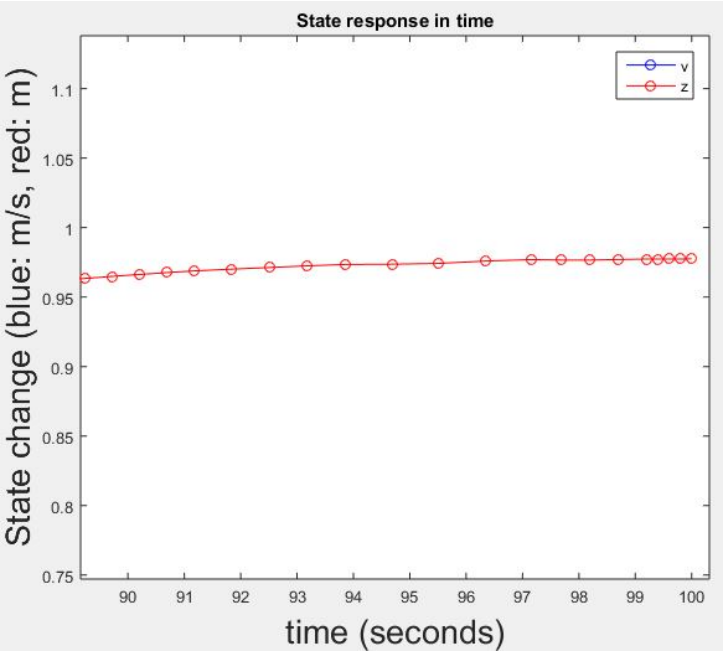

Figure 5: Steady State Response of 'z'

2 Question 2

From the previous assignment, it is known that;

$T(\omega, \lambda) = \lambda * T_m * (1 - \beta * (\frac{\omega}{\omega_m} - 1)^2)$

$\lambda = sat(u)$

$u = K_p(v_r - v) + K_i \int_0^t (v_r - v) d\tau$

$F_{applied} = \alpha T(\omega, \lambda)$

$F_{total} = F_{applied} - (mgC_r sign(v) + \frac{1}{2}\rho C_d A v^2$

$\therefore F_{total} = m * a = F_{applied} - F_{disturbance}$

$\rightarrow v' = \frac{F_{applied}}{m} - g * C_r * sign(v) - \frac{1}{2*m}\rho C_d A v^2$

where $F_d = g * C_r * sign(v) + \frac{1}{2*m}\rho C_d A * v^2$ v' can be written as;

$v' = \frac{\alpha}{m} sat(K_p(v_r - v) + K_i z) T_m (1 - \beta(frac\omega\omega_m - 1)^2) - Fd$

as z is the integral state of the controller, or the integral of the error between the regulated velocity and the actual velocity, z' is the derivative of the integral of the error. Thus, z is simply the error in velocity;

$z = v_r - v$

Thus, the state-space equation, $x' = F(x)$, can be formed;

$$\begin{bmatrix} v' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{\alpha}{m} sat(K_p(v_r - v) + K_i z) T_m (1 - \beta(\frac{\omega}{\omega_m} - 1)^2) - (gC_r sign(v) + \frac{1}{2*m}\rho C_d A * v^2) \\ v_r - v \end{bmatrix}$$

Systems have equilibrium points where the states stop changing in time; as seem in part 1 of this assignment, stable systems state's can reach a steady-state value. Thus, there exists values of $x = [v, z]$ such that $[v', z'] = [0, 0] = F(x)$.

Computing row two of the state-space equation for v corresponding to $v' = 0$ gives;

$0 = v_r - v$
$\therefore v = v_r$

similarly, row one can be solved for z give it is known that an equilibrium point is at $v = v_r$

$0 = \frac{\alpha}{m} * u * T_m * (1 - \beta(\frac{\omega}{\omega_m} - 1)^2) - F_d$

$0 = \frac{\alpha}{m} * [K_p(v_r - v_r) + K_i z] * T_m * (1 - \beta(\frac{\omega}{\omega_m} - 1)^2) - F_d$

$$\therefore z = \frac{m*F_d}{\alpha K_i T_m (1-\beta(\frac{\omega}{\omega_m}-1)^2)}$$

thus, there is an equilibrium point at;

$$[v, z] = [v_r, \frac{m*F_d}{\alpha K_i T_m (1-\beta(\frac{\omega}{\omega_m}-1)^2)}]$$

Checking that V(x) is a Lyapunov function for the non-linear system requires computation of a formula which only holds if the equilibrium is at the origin; $[v, z] = [0, 0]$. Additionally, the eigenvalues of A can be used to test if the origin is asymptotically stable; if they have negative and real parts. Thus, to determine whether the system is stable by Lyapunov, a change in coordinates is required.

For this, x and y will define the new coordinate system such that;

$$x = v - v_r \text{ and } y = z - \frac{m*F_d}{\alpha K_i T_m (1-\beta(\frac{\omega}{\omega_m}-1)^2)}$$

Thus, $F(x)$ is now a function of $[x, y]$, and the state-space input equation can be rewritten.

Matlab is used for the following computations, as they become quite complicated. The final code is illustrated in the figure below, sections are seperated by

```
%% Computes the positive definite Lyapunov function for the cruise control
    system
%% Change coordinate system and set up x' = F(x)
clear all;

syms p Cd A Vr m g Cr Ki alpha Tm beta wm Kp v z Fd u L real


L = Kp*(Vr-v)+Ki*z;                                  %Lambda
T = Tm*(1-beta*((alpha*v/wm)-1)^2);                  %Torque
Fd = g*Cr*sign(v) - (1/(2*m))*p*Cd*A*v^2;            %Disturbance

% x' = F(x)
F_vz = [(alpha/m)*L*T - Fd;
        Vr - v];

% evaluate a change in coordinates of F(x)
syms x y
F_vy = simplify(subs(F_vz, z, y + (m*Fd/(alpha*Ki*T))));
F_xy = simplify(subs(F_vy, v, x + Vr));

%collect terms for seperation
F_xy = collect(F_xy, [x, y]);

%% construct the function F(x) = Ax + F_(x)
% higher order terms found in F_xy; a.k.a F_(x)
F_ = [((Kp*Tm*alpha^3*beta)/(m*wm^2))*x^3 +
    (-(Ki*Tm*alpha^3*beta)/(m*wm^2))*x^2*y + (-(Kp*Tm*alpha*(-
    2*Vr*beta*alpha^2 + 2*beta*wm*alpha))/(m*wm^2))*x^2 + ((Ki*Tm*alpha*(-
    2*Vr*beta*alpha^2 + 2*beta*wm*alpha))/(m*wm^2))*x*y;
        0];
% first order terms found in F_xy; A
```

```
A_1 = [((Kp*Tm*alpha*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta -
    2*Vr*alpha*beta*wm))/(m*wm^2)), -Ki*Tm*alpha*(beta*wm^2 - wm^2 +
    Vr^2*alpha^2*beta - 2*Vr*alpha*beta*wm)/(m*wm^2);
              -1, 0];
% by letting a_11 = (Kp*Tm*alpha*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta -
    2*Vr*alpha*beta*wm))/(m*wm^2)
        %     a_12 = -Ki*Tm*alpha*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta -
    2*Vr*alpha*beta*wm)/(m*wm^2)
        % and
        % G = Tm*alpha/(m*wm^2)       > 0
        % H = Vr*alpha*beta           > 0
        % gives:
        % a_11 = Kp * G *(wm^2 *(beta-1) + alpha*Vr*H - 2*H*wm)
        % a_12 = Ki * G *(wm^2 *(beta-1) + alpha*Vr*H - 2*H*wm)
%% find P_ for the Lyapunov function (As ^T)P_ +P_ As = Q, symbolically
% where Q is the negative identiy matrix. In this case, 'As' contains
    unknown
% real, positive, coefficents

syms a_11 a_12 a_22 P_11 P_12 P_22 G H real
As = [a_11, -a_12; -1, 0];
P_ = [ P_11, P_12; P_12, P_22];
Q = -eye(2);
lyp = transpose(As)*P_+P_*As;
solve(lyp == Q, [P_11, P_12, P_22]);
P_ = simplify([ans.P_11, ans.P_12; ans.P_12, ans.P_22]);
D_ = det(P_);
E_ = simplify(eig(P_));              % check the eigenvalues of P > 0
T_ = simplify(trace(P_));            % check the trace of P > 0
EA_ = simplify(eig(As));             % check the eigenvalues of A < 0
%% Find P for the Lyapunov function (A^T)P+PA = Q, symbolically, using
    cruise control parameters
syms p_11 p_12 p_22 real

P = [ p_11, p_12; p_12, p_22];
Q = -eye(2);
lyp = transpose(A_1)*P+P*A_1;
solve(lyp == Q, [p_11, p_12, p_22]);
P = simplify([ans.p_11, ans.p_12; ans.p_12, ans.p_22]);
E = simplify(eig(P));                % check the eigenvalues of P > 0
T = simplify(trace(P));              % check the trace of P > 0
A_val = simplify(eig(A_1));
%check numerical results
EIG = subs(E, [Tm, wm, beta, alpha, g, Cr, p, Cd, A, Vr, Kp, Ki, m],...
        [200, 450, 0.35, 16, 9.81, 0.01, 1.3, 0.36, 2.2, 20, 0.5, 0.1,
    1000]);
TRACE = subs(T, [Tm, wm, beta, alpha, g, Cr, p, Cd, A, Vr, Kp, Ki, m],...
        [200, 450, 0.35, 16, 9.81, 0.01, 1.3, 0.36, 2.2, 20, 0.5, 0.1,
    1000]);
A_EIG = subs(A_val, [Tm, wm, beta, alpha, g, Cr, p, Cd, A, Vr, Kp, Ki,
    m],...
        [200, 450, 0.35, 16, 9.81, 0.01, 1.3, 0.36, 2.2, 20, 0.5, 0.1,
    1000]);
% gives eigenvalues of P of 0.3506 and 3.9294 > 0
% trace of 4.28 > 0
% eigenvalues of A of -0.2358 and -1.3175 < 0
% therefore asymptopically stable
```

```
%% check for global asymptopic stability
% check lim x-> ||F(x)||/||x|| is determinant

F_mag = (F_(1)^2+F_(2)^2)^0.5;                                  %magnitude of
    higher order terms
x_mag = (x^2 + y^2)^0.5;                                        %magnitude of
    equ variables
Lim_x = limit(F_mag/x_mag, x, 0);
Lim_xy = limit(Lim_x, y, 0);
AS = subs(F_mag/x_mag, [x, y], [0.00000001,0.00000001]);        %computed for
    values of x and y close to 0
GAS = subs(AS, [Tm, wm, beta, alpha, g, Cr, p, Cd, A, Vr, Kp, Ki, m],...
        [200, 450, 0.35, 16, 9.81, 0.01, 1.3, 0.36, 2.2, 20, 0.5, 0.1,
    1000]);
% from Lim_xy,
% Lim F_mag/x_mag | x,y -> 0,0 = 0 and
% GAS = 8.0433e+17 / 1.2360e+28 = 6.5075e-11
% therefore numerator goes to 0 first -> V(x) is a Lyapunov function
% for the non-linear system
%% show lyapunov function
V = collect([x, y]*P*[x; y], [x, y]);
% V = (m*wm^2*(m*wm^2 + Ki*Tm*alpha*wm^2 - Ki*Tm*alpha*beta*wm^2 -
%    Ki*Tm*Vr^2*alpha^3*beta +
    2*Ki*Tm*Vr*alpha^2*beta*wm)*x^2)/(2*Ki*Kp*Tm^2*
%    alpha^2*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta - 2*Vr*alpha*beta*wm)^2)
    +
%    (m*wm^2*x*y)/(Ki*Tm*alpha*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta - 2*Vr*
%    alpha*beta*wm)) - ((Ki*m*wm^2 + Ki^2*Tm*alpha*wm^2 +
    Kp^2*Tm*alpha*wm^2 -
%    Ki^2*Tm*Vr^2*alpha^3*beta - Kp^2*Tm*Vr^2*alpha^3*beta - Ki^2*Tm*alpha*
%    beta*wm^2 - Kp^2*Tm*alpha*beta*wm^2 + 2*Ki^2*Tm*Vr*alpha^2*beta*wm +
%     2*Kp^2*Tm*Vr*alpha^2*beta*wm)*y^2)/(2*Ki*Kp*Tm*alpha*(beta*wm^2 -
    wm^2 +
%    Vr^2*alpha^2*beta - 2*Vr*alpha*beta*wm))


V_ = collect([x, y]*P_*[x; y], [x, y]);
% V_ = ((1 - a_12)*x^2)/(2*a_11*a_12) + (x*y)/a_12 +
%        ((a_11^2 + a_12^2 - a_12)*y^2)/(2*a_11*a_12)
%
% let m = (1 - a_12), n = 1/(2*a_11*a_12), p = a_11^2
% V_ = m*n*x^2 + (p + a_12*m)*n*y^2 + x*y/a_12 ----> which is 0 only when
% x,y = 0,0 ----> therefore is positive definite
%% match the parameter result to the symbolic result
% equating the symbolib and parameter coefficent expressions for x^2
% a_11 and a_12 can be found, represented by b_11 and b_12 respectively
%syms b_11 b_12 real
G = Tm*alpha/(m*wm^2);
H = Vr*alpha*beta;
b_11 = Kp * G *(wm^2 *(beta-1) + alpha*Vr*H - 2*H*wm);
b_12 = Ki * G *(wm^2 *(beta-1) + alpha*Vr*H - 2*H*wm);
E_solv = subs(E_, [a_11, a_12], [b_11, b_12]);
EIG_solv = simplify(subs(E_solv, [Tm, wm, beta, alpha, g, Cr, p, Cd, A,
    Vr, Kp, Ki, m],[200, 450, 0.35, 16, 9.81, 0.01, 1.3, 0.36, 2.2, 20,
    0.5, 0.1, 1000]));

% as EIG_solv = EIG, it is therefore determined that the results are the
    same.
```

```
%% Results
% P_11 =
% (m*wm^2*(m*wm^2 + Ki*Tm*alpha*wm^2 - Ki*Tm*alpha*beta*wm^2 -
    Ki*Tm*Vr^2*alpha^3*beta +
    2*Ki*Tm*Vr*alpha^2*beta*wm))/(2*Ki*Kp*Tm^2*alpha^2*(beta*wm^2 - wm^2 +
    Vr^2*alpha^2*beta - 2*Vr*alpha*beta*wm)^2)
%
% P_12 =
% (m*wm^2)/(2*Ki*Tm*alpha*(beta*wm^2 - wm^2 + Vr^2*alpha^2*beta -
    2*Vr*alpha*beta*wm))
%
% P_22 =
% -(Ki*m*wm^2 + Ki^2*Tm*alpha*wm^2 + Kp^2*Tm*alpha*wm^2 -
    Ki^2*Tm*Vr^2*alpha^3*beta - Kp^2*Tm*Vr^2*alpha^3*beta -
    Ki^2*Tm*alpha*beta*wm^2 - Kp^2*Tm*alpha*beta*wm^2 +
    2*Ki^2*Tm*Vr*alpha^2*beta*wm +
    2*Kp^2*Tm*Vr*alpha^2*beta*wm)/(2*Ki*Kp*Tm*alpha*(beta*wm^2 - wm^2 +
    Vr^2*alpha^2*beta - 2*Vr*alpha*beta*wm))
%
```

Script 3: PBA2_2_3script.m

In the third line of code, all of the cruise control variables are defined as real and symbolic; so they can be carried through equations.

The equation for λ is then defined, assuming small u, such that there is no saturation required. Following, the equation for torque and disturbance forces are represented by T and Fd respectively.

The state-space equation is then written for $[v', z'] = F([v, z])$, as calculated above. From this, the change of coordinate system is performed to calculate $[x', y'] = F([x, y])$, denoted F_xy, and the x and y terms are collected such that F_xy is written with the variable order factored. This results in;

$$
\begin{aligned}
x' = {} & \frac{\left(\mathrm{Kp\,Tm\,alpha^3\,beta}\right)}{\left(m\,\mathrm{wm^2}\right)}\,x^3 + -\frac{\mathrm{Ki\,Tm\,alpha^3\,beta}}{m\,\mathrm{wm^2}}\,x^2\,y + -\frac{\mathrm{Kp\,Tm\,alpha}\left(2\,\mathrm{alpha\,beta\,wm} - 2\,\mathrm{Vr\,alpha^2\,beta}\right)}{m\,\mathrm{wm^2}}\,x^2 \\
& + \frac{\left(\mathrm{Ki\,Tm\,alpha}\left(2\,\mathrm{alpha\,beta\,wm} - 2\,\mathrm{Vr\,alpha^2\,beta}\right)\right)}{\left(m\,\mathrm{wm^2}\right)}\,x\,y + \\
& \frac{\left(\mathrm{Kp\,Tm\,alpha}\left(\mathrm{beta\,wm^2} - \mathrm{wm^2} + \mathrm{Vr^2\,alpha^2\,beta} - 2\,\mathrm{Vr\,alpha\,beta\,wm}\right)\right)}{\left(m\,\mathrm{wm^2}\right)}\,x + \\
& - \frac{\mathrm{Ki\,Tm\,alpha}\left(\mathrm{beta\,wm^2} - \mathrm{wm^2} + \mathrm{Vr^2\,alpha^2\,beta} - 2\,\mathrm{Vr\,alpha\,beta\,wm}\right)}{m\,\mathrm{wm^2}}\,y \\
y' = {} & -x
\end{aligned}
$$

All represented by F_xy, where the higher order terms are copied into a matrix $F\_$ in the following section, and the linear terms are stored in the matrix A_1.

It can be seen from the above expressions that matrix A_1 will contain a positive first row first column element (because of the sign infront of the x term), and a negative first row second column element (because of the sign in front of the y term). Computing A_1 shows that the second row is $[-1, 0]$, corresponding to the second equation above.

Following this, a symbolic representation of the Lyuponuv function is computed, using the symbols a_11 to denote the first (row and column) expression in A_1, a_12 denotes the expression in the first row and second column of A_1. As, the symbolically simplified representation of $A\_$ is then formed; as seen in the Matlab code.

From this, the P matrix in the Lyapunov function can be defined and values computed using the Lyapunov equation;

$$V(x) = x^T P x$$

$$A^T P + PA = -Q,$$

where $Q = I$; I is the identity matrix and Q is thus positive definite.

This is done in Matlab using the function *solve*(), which equates the above expression to find the coefficients of $P\_$; P_11, P_12 and P_22. These are then reassigned into the P matrix, where the determinant, eigenvalues and trace of P are computed.

$$P\_ = \begin{pmatrix} -\frac{a_{12}-1}{2\,a_{11}\,a_{12}} & \frac{1}{2\,a_{12}} \\ \frac{1}{2\,a_{12}} & \frac{{a_{11}}^2+{a_{12}}^2-a_{12}}{2\,a_{11}\,a_{12}} \end{pmatrix}$$

$$EA\_ = \begin{pmatrix} \frac{a_{11}}{2} - \frac{\sqrt{{a_{11}}^2+4\,a_{12}}}{2} \\ \frac{a_{11}}{2} + \frac{\sqrt{{a_{11}}^2+4\,a_{12}}}{2} \end{pmatrix}$$

note that for the linear system to be asymptotically stable, A must have real, negative parts. From above, it can be seen that this is only possible if a_11 is negative. (Which implies that the expression for As would be better having $+a\_11$ in the first element)

$$D\_ = -\frac{{a_{11}}^2+{a_{12}}^2-2\,a_{12}+1}{4\,{a_{11}}^2\,a_{12}}$$

$$E\_ = \begin{pmatrix} \frac{{a_{11}}^2-\sqrt{({a_{11}}^2+{a_{12}}^2-2\,a_{11}+1)\,({a_{11}}^2+{a_{12}}^2+2\,a_{12}+1)}-2\,a_{12}+{a_{12}}^2+1}{4\,a_{11}\,a_{12}} \\ \frac{\sqrt{({a_{11}}^2+{a_{12}}^2-2\,a_{12}+1)\,({a_{11}}^2+{a_{12}}^2+2\,a_{12}+1)}-2\,a_{12}+{a_{11}}^2+{a_{12}}^2+1}{4\,a_{11}\,a_{12}} \end{pmatrix}$$

$$T\_ = \frac{{a_{11}}^2+{a_{12}}^2-2\,a_{12}+1}{2\,a_{11}\,a_{12}}$$

It can be seen that the trace and determinant of P are both negative given $a\_11 < 0$; however this can be fixed if both a_11, $a\_12 < 0$. Additionally, this would give positive eigenvalues for P (expression $E\_$); implying the Lyapunov function is positive definite. This can be explained when analysing the A_1 elements in detail, where it appears the negative terms in the expressions will be computed to be larger, thus making then negative and real, unlike that assumed. Nonetheless, the math will come out the same in the end.

Thus, assuming all coefficients are real and positive, potentially, a better A matrix to assume (for stability) could be;

$$A = \begin{bmatrix} -a_{11} & a_{12} \\ -1 & 0 \end{bmatrix}$$

Following from this, the next section computes P with the same method, but where the A matrix (denoted A_1) contains the cruise control model parameters. The eigenvalues of P are then found by substituting in the model parameters used in part 1, giving positive real results of 0.3506 and 3.9294, with a trace of 4.28. Similarly, it was seen that A

contained negative eigenvalues, implying the linearised model is asymptotically stable.

To check if the proposed Lyapunov function is valid for the non-linear system, the following function need be computed using the matrix comprising of the higher order terms of F([x, y]);

$$\lim_{||x||\to 0} \frac{||F(x)||}{||x||} = 0$$

This is computed in the next section of code in two ways, using the limit function, and calculating the limit manually where $[x, y] = [0.00000001, 0.00000001]$. Both of these arrive at the same result, that the limit goes to 0; however the latter test shows that the numerator goes to 0 first, and thus the solution is determinant, as proved by the limit function.

The Lyapunov function for the used P in the code, with all model parameters, and with the substitution, are displayed further down in the code image. Where the symbolic $V(x)$ came to be;

$$V(x) = \left(\frac{1-a_{12}}{2\,a_{11}\,a_{12}}\right)\, x^2 + \frac{x\,y}{a_{12}} + \left(\frac{{a_{11}}^2+{a_{12}}^2-a_{12}}{(a_{11}\,a_{12}*2)}\right)\, y^2$$

which assuming both a_11 and a_12 are negative, as determined earlier, produces a positive Lyapunov function for the cruise control system; if $[x, y] > 0$.

The symbolic and parameter solutions were equated in the end of the Matlab code, that verified that they were equal.