METR7203 – Assignment 1

Callum Rohweder
43575943
2017

Equation Background

As per the assignment outlines, cruise control is used to regulate the velocity of an automobile and should be designed to be robust to the road slope and the corresponding force disturbances. Below illustrates the equations governing the cruise control system and the corresponding Matlab Simulink block diagrams.

Body

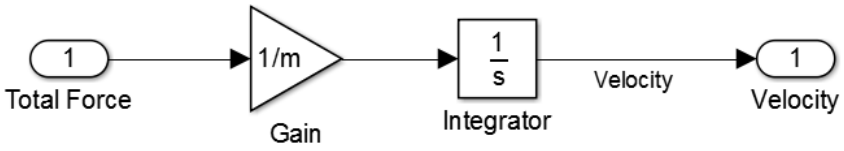
From Newton's second law, it is known that the sum of forces, F_{total}, acting on a body of mass, m, will results in an acceleration $\frac{dv}{dt}$, such that;

$$\sum F = m\frac{dv}{dt}$$
$$\rightarrow v = \frac{1}{m}\int_0^t F\ d\tau$$

Taking the Laplace transform gives

$$V(s) = \frac{1}{ms}\ F(S)$$

Which is illustrated in Simulink by:



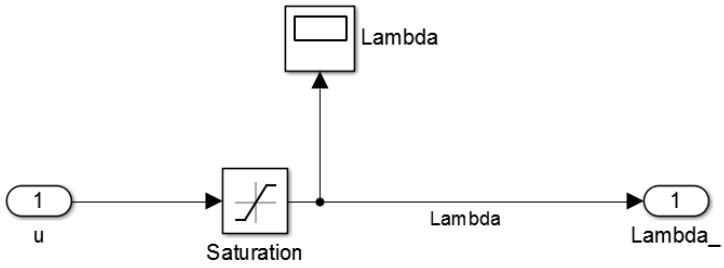
Thus the total force acting on a body can be represented using the transfer function in Simulink.

Actuator

The actuator controls the percentage of which the throttle is open, of which is between un-actuated and fully on, as depicted by λ in response to a controller output u;

$$\lambda = \begin{cases} 0, & u < 0 \\ u, & 0 \leq u \geq 0 \\ 1, & u > 1 \end{cases}$$

This piecewise expression can be represented by a 'saturation' block in matlab. Alternatively a function block of which calls a function that computes the output lambda based on input u based on the three conditions can be used. For simplicity, the Simulink of this system appears as;



Where an oscilloscope block is included to check the outputs of the saturation block; with an upper limit of 1 and lower limit of 0.

Throttle and Engine

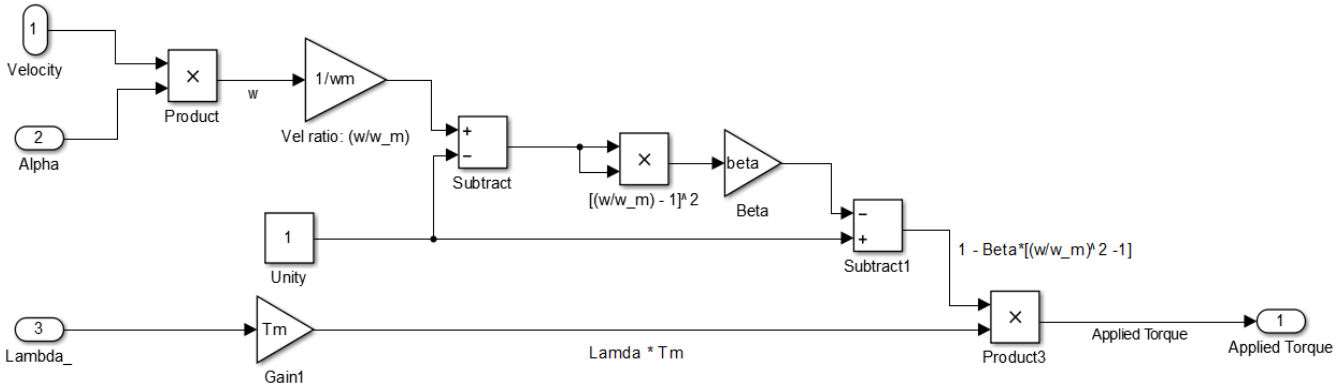
The expression for the torque, T, applied by the motor to achieve a desired engine angular velocity, ω, based on the throttle position represented by λ, is given by;

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

T_m represents the maximum torque the motor can supply, where ω_m represents the achievable engine angular velocity for said torque. Using the fact that the motor velocity can be related to the actual linear velocity, v, by the effective wheel radius, α_n, such that

$$\omega = \alpha_n \times v$$

Then the corresponding Simulink model for applied torque requires real time effective wheel radius / gear, actual velocity and throttle position in order to compute. The above torque formula can be generated using a function block or a combination of gain and operator blocks; the latter is shown below.

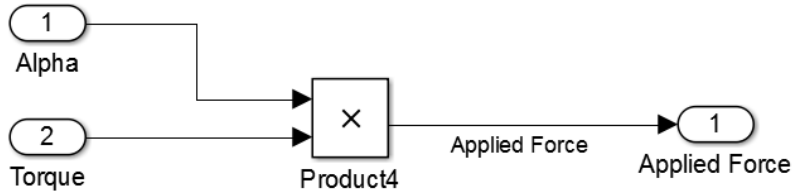


Gears and Wheels

Similarly with the relation of effective wheel radius to velocity outlined above, force can be related to torque by α, such that

$$F = \alpha_n\, T$$

Where the force in this case is the force the car exerts on the road for motion, and appears as;



Disturbance Force

From the aforementioned principle that;

$$\sum F = m\frac{dv}{dt}$$

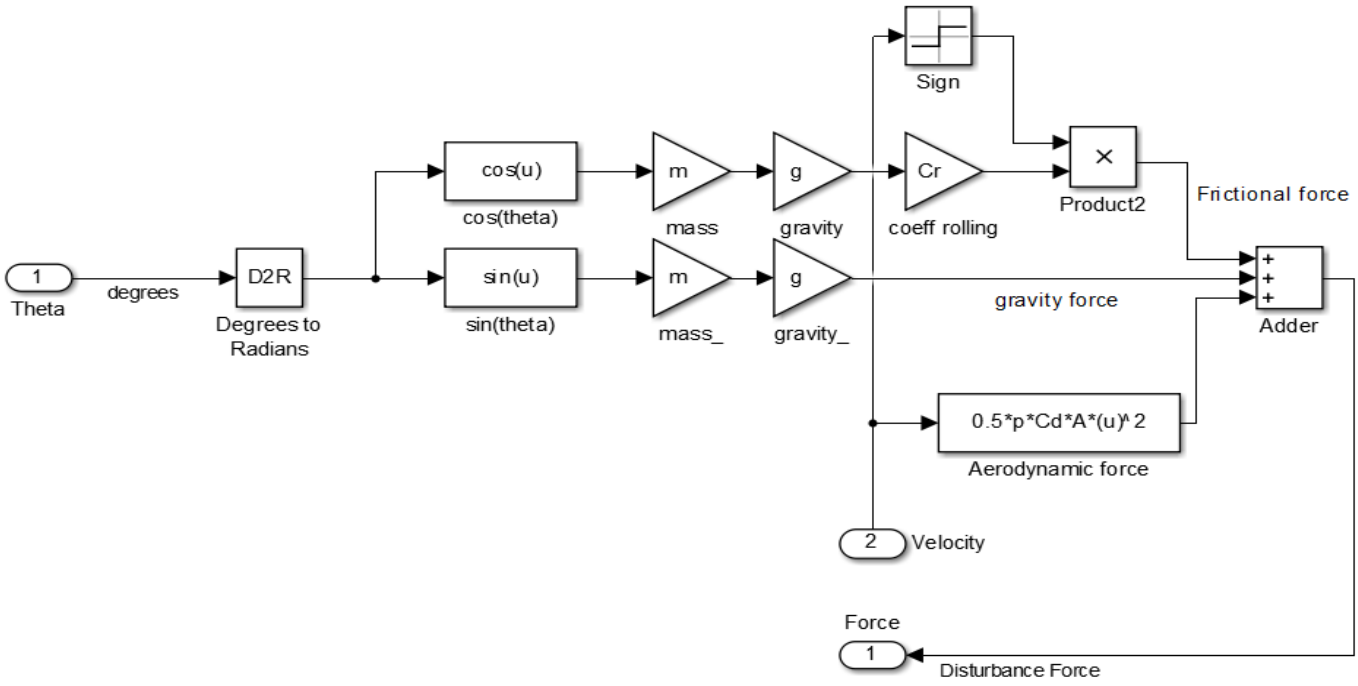
the sum of forces that results in the motion of the car is made up of the applied force, a force exerted by gravity on the car mass, rolling friction between the tyres and the road, and an aerodynamic force related to the car profile. These forces are illustrated in the summation below;

$$\sum F = F_{applied} - F_{gravity} - F_{friction} - F_{aerodynamic}$$

$$F_{total} = \alpha\, T - \left(mgsin(\theta) + mgC_r\cos(\theta)\, sign(v) + \frac{1}{2}\rho C_d A V^2\right)$$

Where, $F_{disturbance} = mgsin(\theta) + mgC_r\cos(\theta)\, sign(v) + \frac{1}{2}\rho C_d A V^2$

Where θ represents the slope of the hill to the horizontal, v is the actual velocity of the automobile, A is the area profile normal to the motion of the vehicle, C_r is the coefficient of rolling friction, and C_d is the drag coefficient.

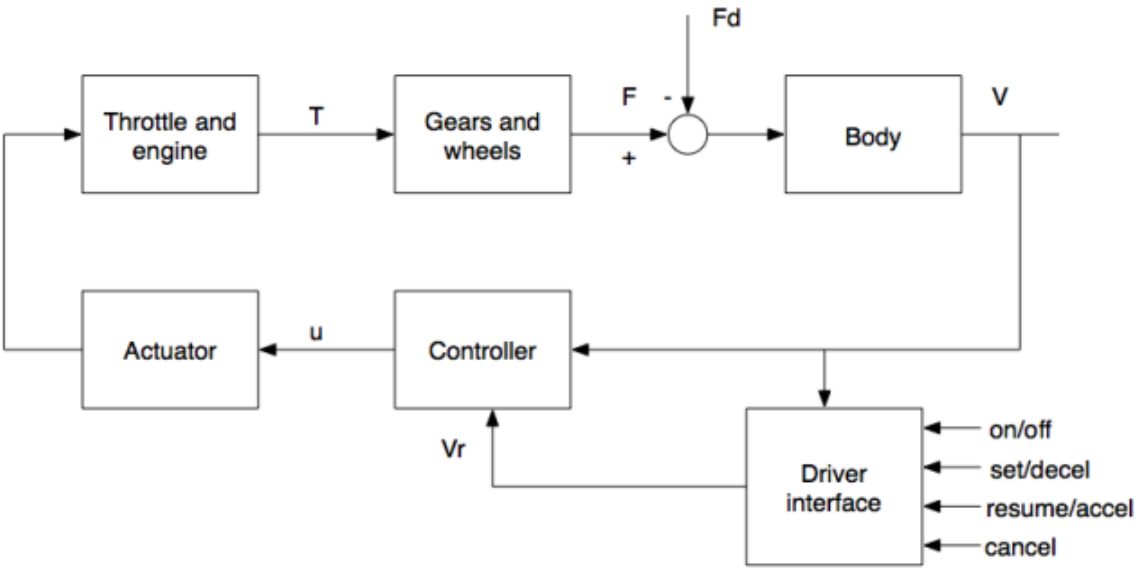


As it can be seen above, using the hill slope and car velocity, the total disturbing force can be computed for each simulation time step. Above, a mixture of function, gain and operation blocks

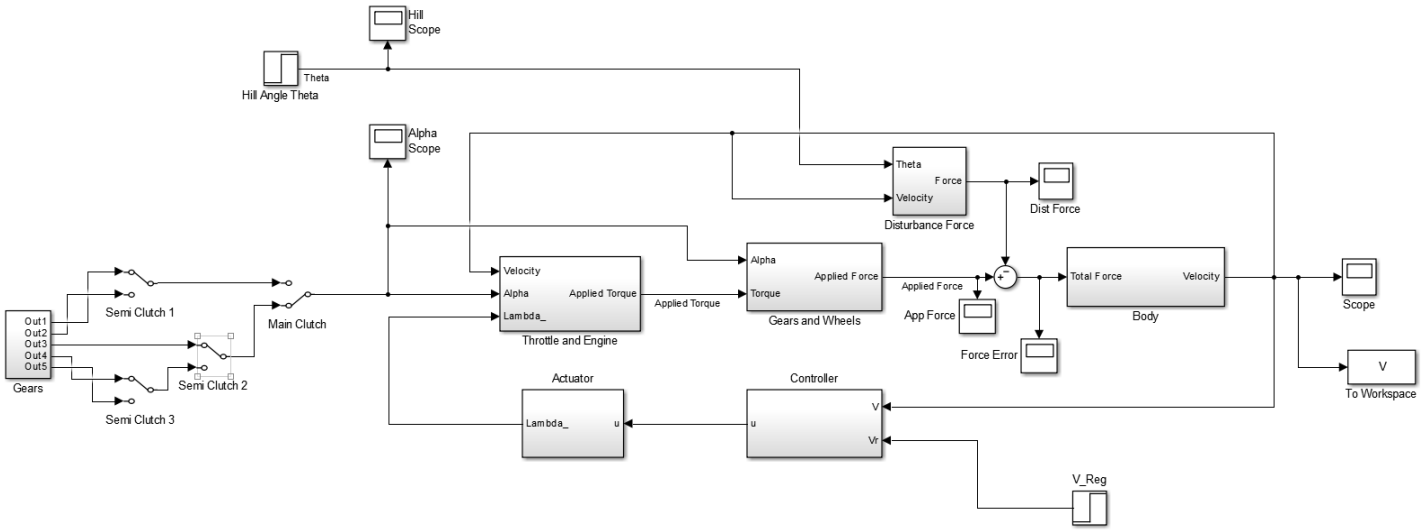
were used. This is then subtracted from the applied force of the motor, which then goes into the body block.

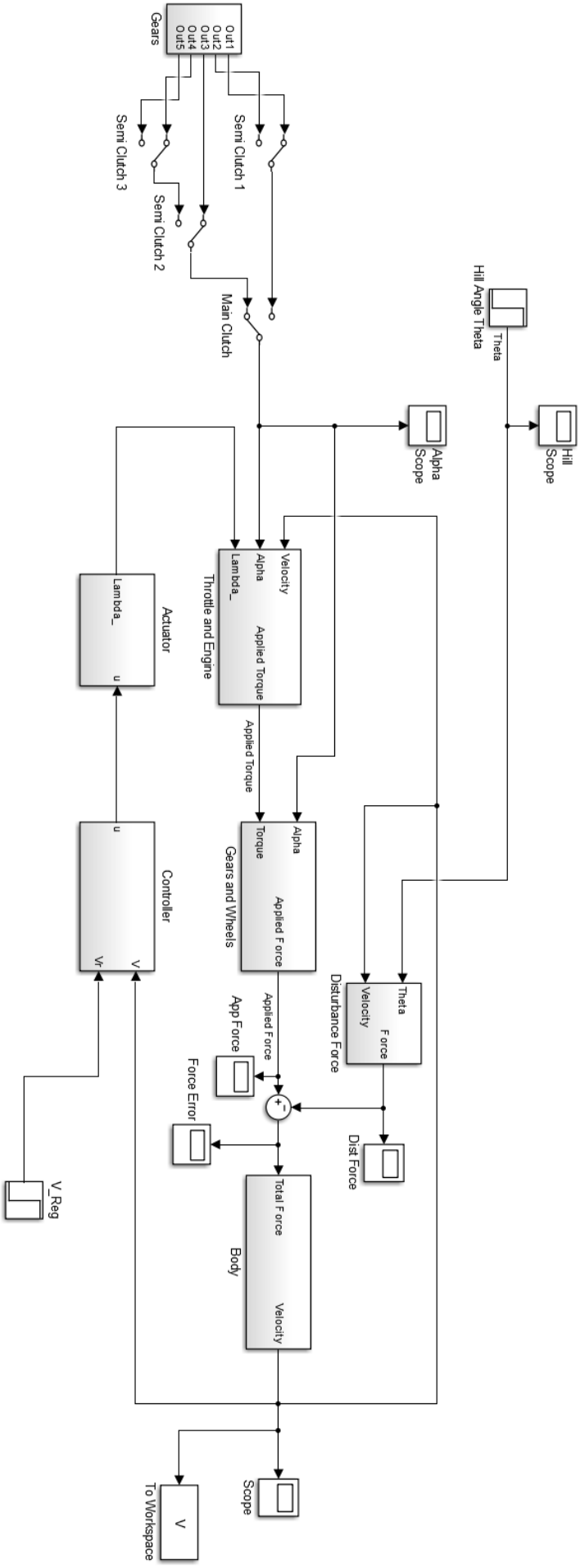
Task 1

Proposed block diagram model of cruise control system:



Corresponding Simulink Model




Gears
Out1
Out2
Out3
Out4
Out5
Semi Clutch 3
Semi Clutch 2
Semi Clutch 1
Main Clutch
Hill Angle Theta
Theta
Alpha Scope
Hill Scope
Velocity
Alpha
Lambda_
Throttle and Engine
Applied Torque
Actuator
Lambda_
u
Applied Torque
Alpha
Torque
Gears and Wheels
Applied Force
Controller
u
Vr
V
Theta
Velocity
Force
Disturbance Force
App Force
Applied Force
Force Error
Dist Force
V_Reg
Total Force
Body
Velocity
Scope
V
To Workspace

Script for declaring variables:

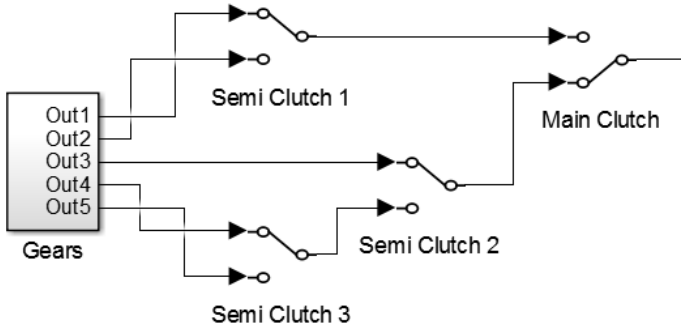
```
%%%  METR7203 PBA1

%% Declare Vaiables
global alpha
global wm

m = 1200;
Tm = 200;
wm(1) = 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;

%% Proportional Integral Controller
% Ziegler-Nichols method values
% Ku = 5
% Tu = 1.1
% Kp = 0.45*Ku
% Ki = 0.54*Ku/Tu
% trial and error values
Kp = 3.5;
Ki = 0.01;

```

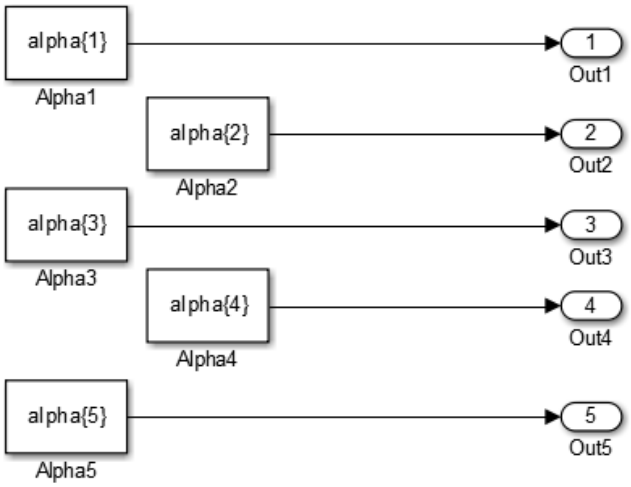
Basic function for gear changing automation; implemented with function block, instead of switching manually

```
function Alpha_ = Get_alpha( Vel )
% Selects the highest gear for the speed of the car, Vel,
% returning the effective wheel radius Alpha
%
    global alpha
    global wm
    %check if velocity is above maximum -> gear change
    s = size(alpha);
    w = 0;
    wn = wm(1);

    for n = 1:s(2)
        w = Vel*alpha{n};
        if w < wn
            Alpha_ = alpha{n};
            %velocity suitable for current gear, n
            return
        end
    end
    % over speed of final gear, use final gear
    if w > wn
        Alpha_ = alpha{s(2)};
    end

end
```

Manual Gearing

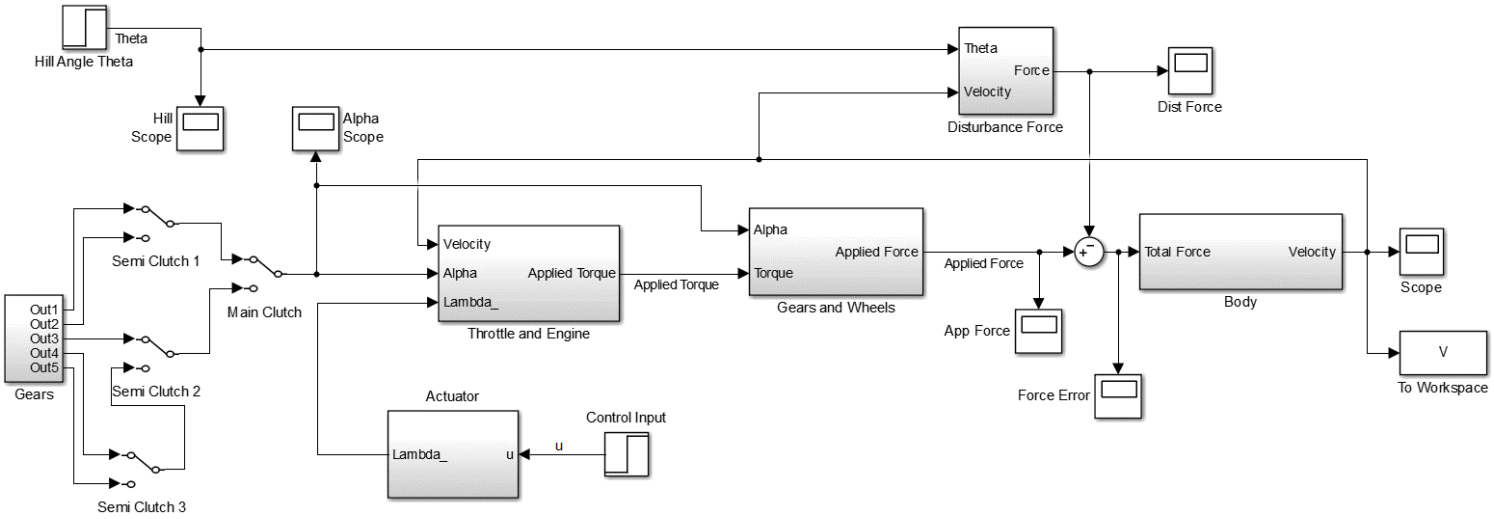
The gear, alpha, is selected using switches, which lets a selected constant through;





Task 2

The open loop system, without a controller, is illustrated below:



With the control input set to a step at 5 seconds to 0.5, and the hill slope set to 0 for all time, and a selection of first gear, the velocity of the car to the input can be observed; at the Scope block. The 'To Workspace' block allows the data, V, from the simulation to be available in the Matlab command line for further computations.

The rise time, and other properties, of the velocity can be found using the stepinfo() command in Matlab. Additionally, the plot() command can be used to illustrate the data in a 2D plot.

Rise time command:

```
stepinfo(V.Data, V.time, 'RiseTimeLimits', [0.05,0.95])
```

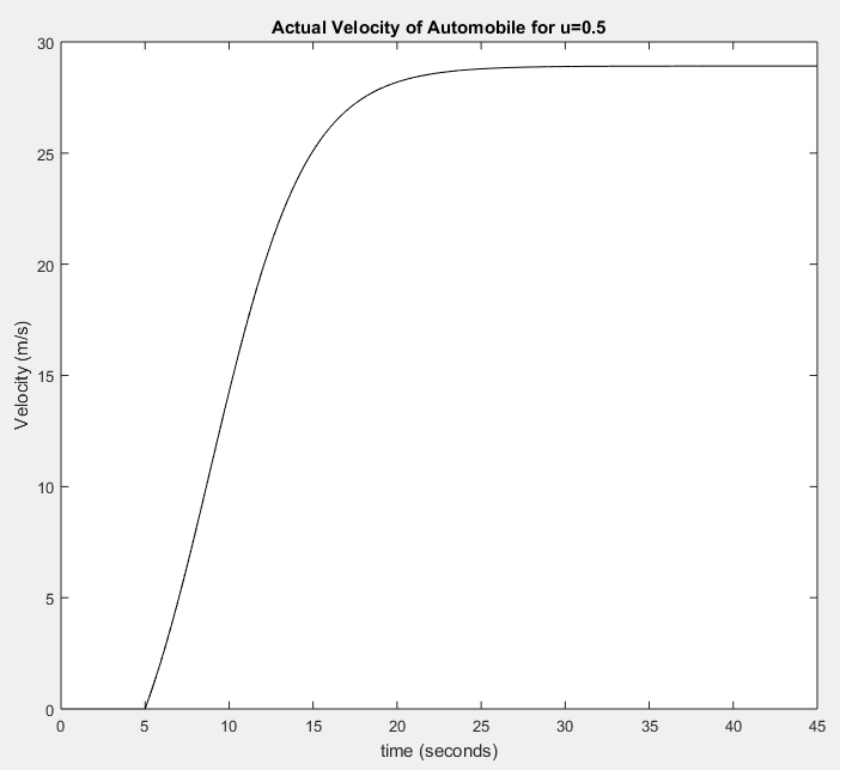
Resulting data:

```
    RiseTime: 12.3107
SettlingTime: 20.6493
 SettlingMin: 27.4715
 SettlingMax: 28.9150
   Overshoot: 0
  Undershoot: 0
        Peak: 28.9150
    PeakTime: 45.0000
```

Plot command:

```
figure;
plot(V.time, V.Data, 'black');
title('Actual Velocity of Automobile for u=0.5');
xlabel('time (seconds)');
ylabel('Velocity (m/s)');
```

Resulting Step Response figure:



As you can see, the rise time is 12.3107 seconds, with a settling time of 20.6493 seconds to reach 28.915 m/s. Additionally the step response to an input of u=0.5 is illustrated.

Task 3

A controller block is included in the model to regulate the actual velocity, to a desired velocity, V_r. The time domain representation of the controller output u(t) is;

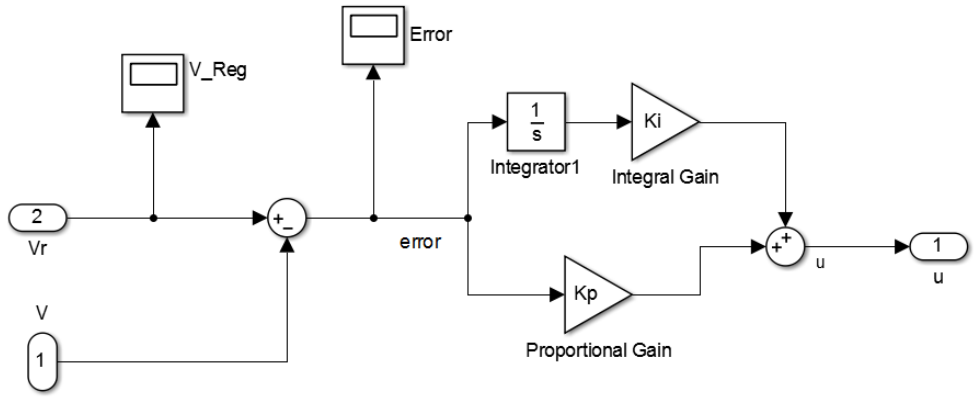
$$u(t) = k_p e(t) + k_i \int_0^t e(\tau) d\tau$$

Where

- e(t) is the error between the actual velocity and the regulated/ desired / design velocity
- K_p is the proportional gain; multiplier to the error to form the output, has effects such as increase risetime whilst increasing steady-state error. Thus, having a large proportional gain can lead to an undesirable steady-state velocity for cruise control; may be lead to speeding, or a velocity slower than desired by the driver.
- K_I is the integral gain; a multiplier to the accumulated error in past time, as described by the integral above. The integral gain thus adds to the controller output with the proportional term for large error, thus causing a further increase in rise time, however this can lead to overshooting the desired velocity. Having this extra term however allows the correction of steady-state error such that the actual velocity is closer to the desired velocity than having a proportional controller alone.

For the cruise control model, it is desirable to regulate the actual velocity to that desired to within 1m/s (accuracy determined by speeding laws in Queensland), without significant overshoot which may cause a speeding fine. Additionally, the rise time need be comfortably quick, such that small disturbances such as hills doesn't make a noticeable change to the velocity by the driver.

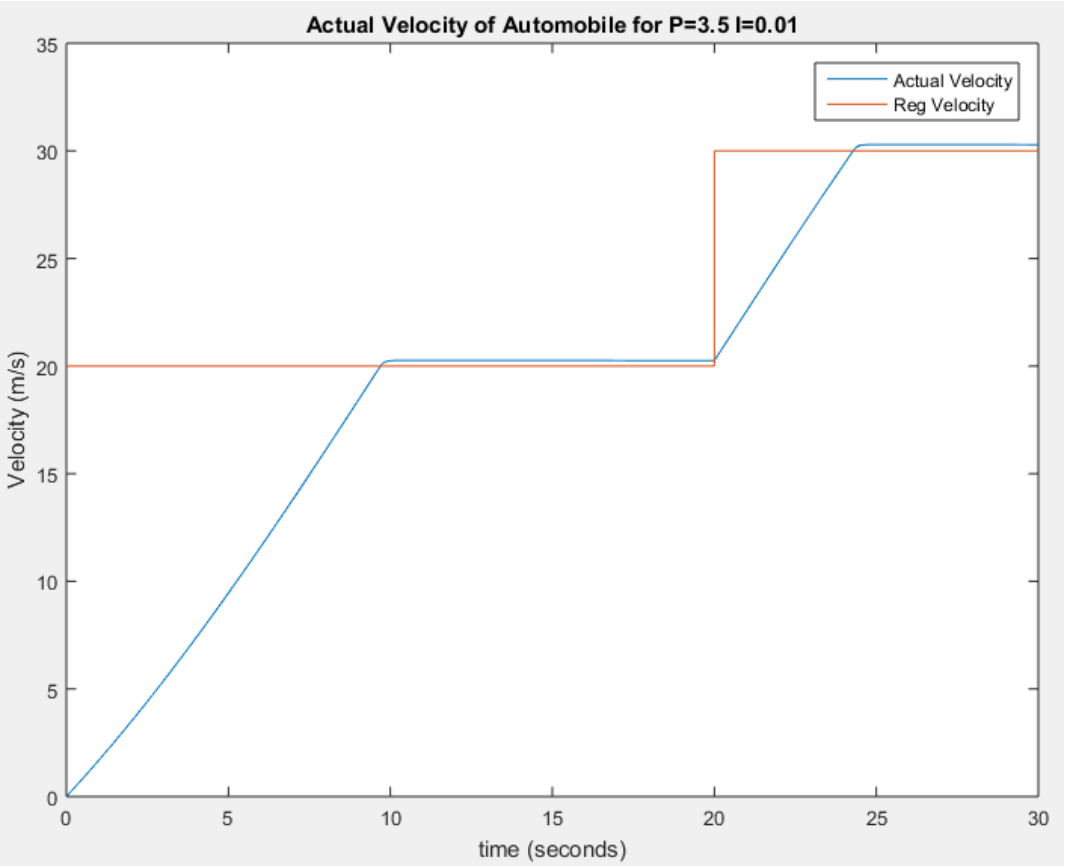
The proportional and integral controller block can be created in Simulink using the PID block or by manipulating the error between V and V_r (similar to that of f=ma earlier) as follows;



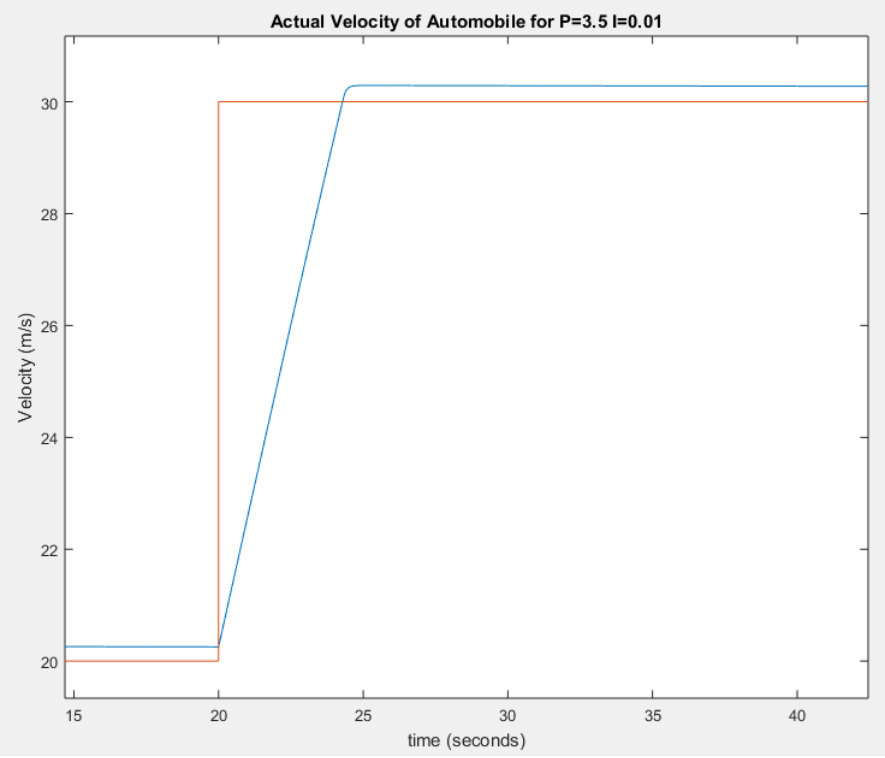
As seen in the simulation model in Task 1, the velocity is fed-back from the output to the controller block, which has a step input block representing the driver 'switching on' a desired velocity. This block has the output u.

It was found that two possible sets of gain values can be used. K_p = 3.5 and K_I = 0.01 were determined after attempting to implement Ziegler-Nichol's tuning method to determine the appropriate values. This method led to significant overshoot, as expected with the method, thus the

terms (more so the integral gain) were tuned down enough to suit the above criteria. This is illustrated in the plot below:

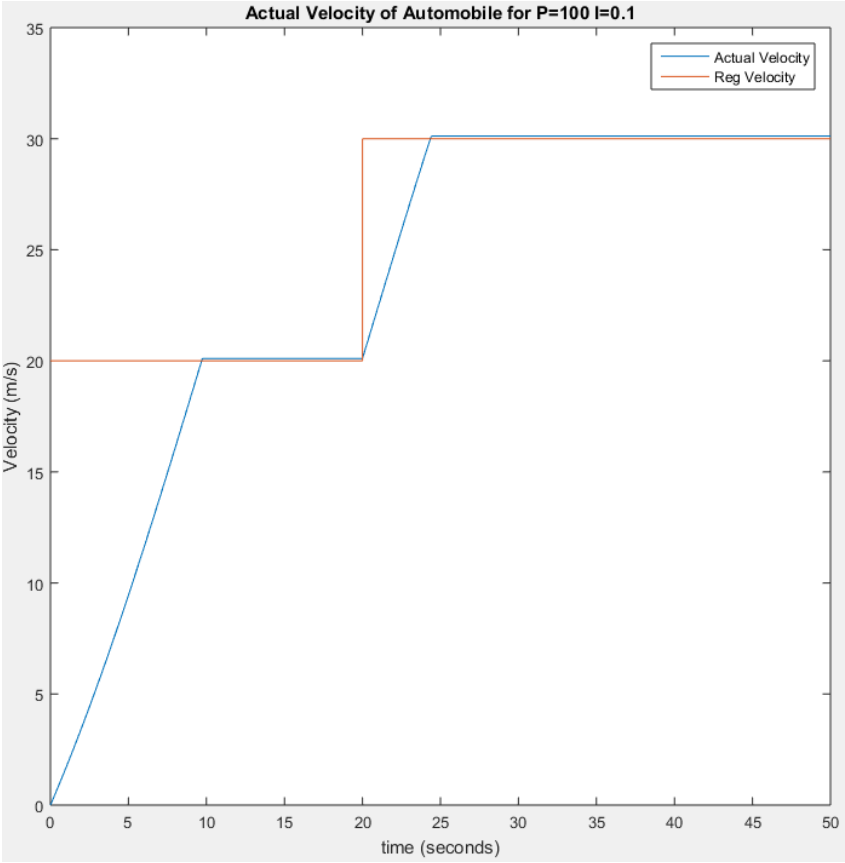


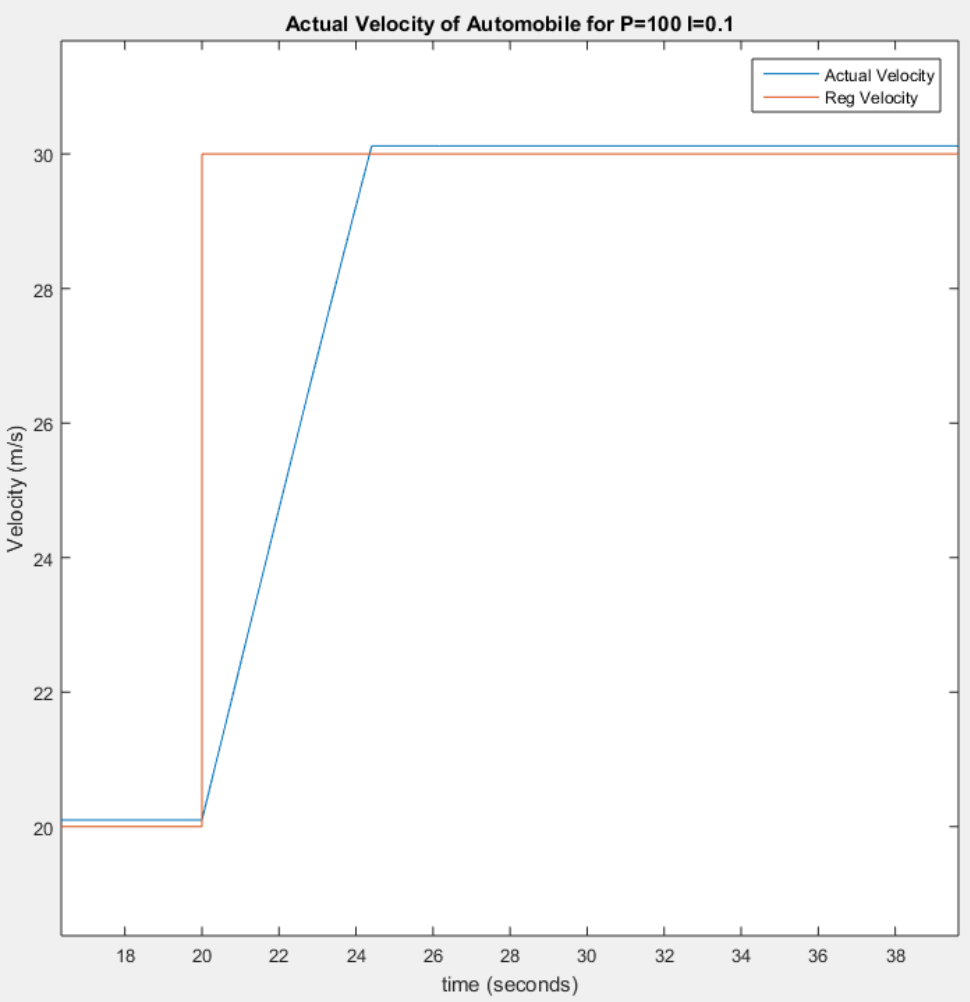
The regulated velocity command was able to be inserted in this step response curve using a 'To Workspace' block at the regulated input.



It can be seen that the steady-state error is under the required 1 m/s, and that the rise time is improved to approximately 3.88 seconds.

After more modification of gains, a better steady-state error was achieved with the gains of $K_p = 100$ and $K_I = 0.1$, the results are illustrated below:



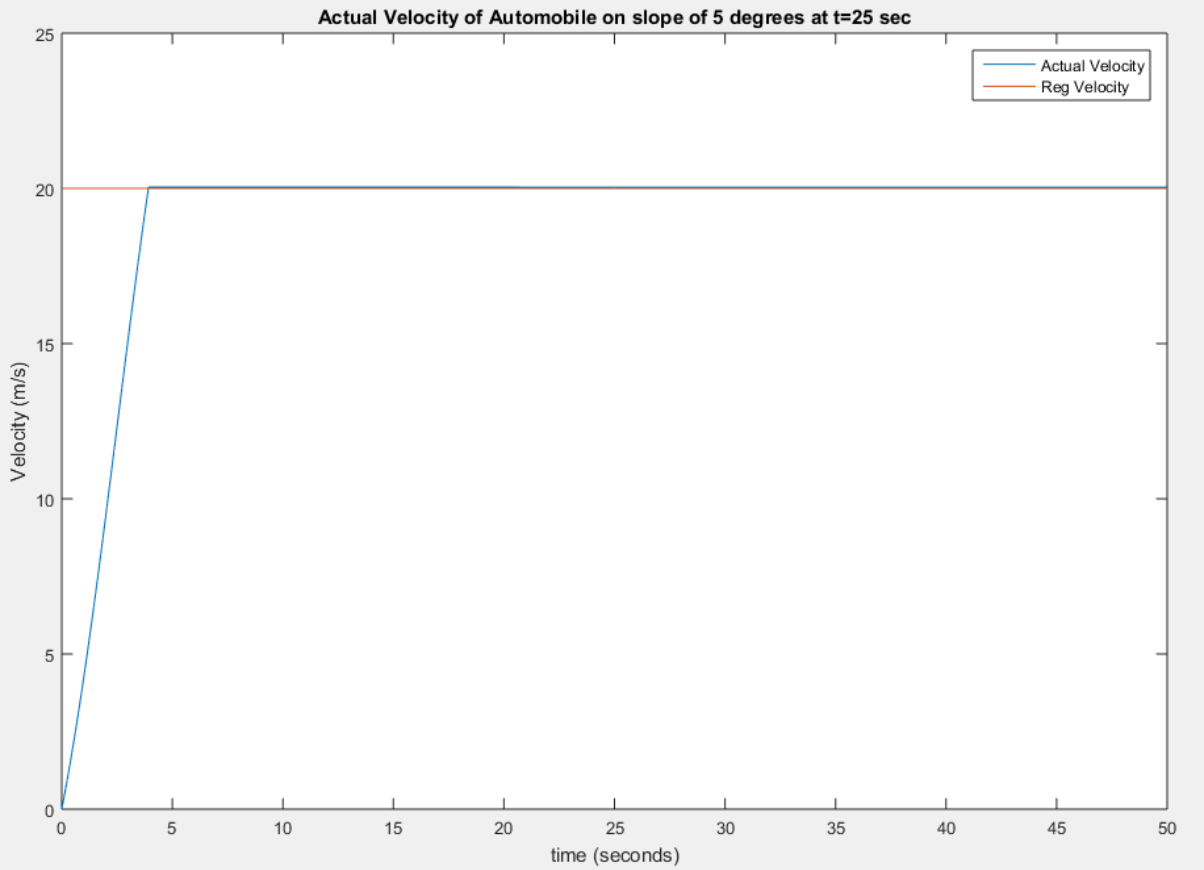


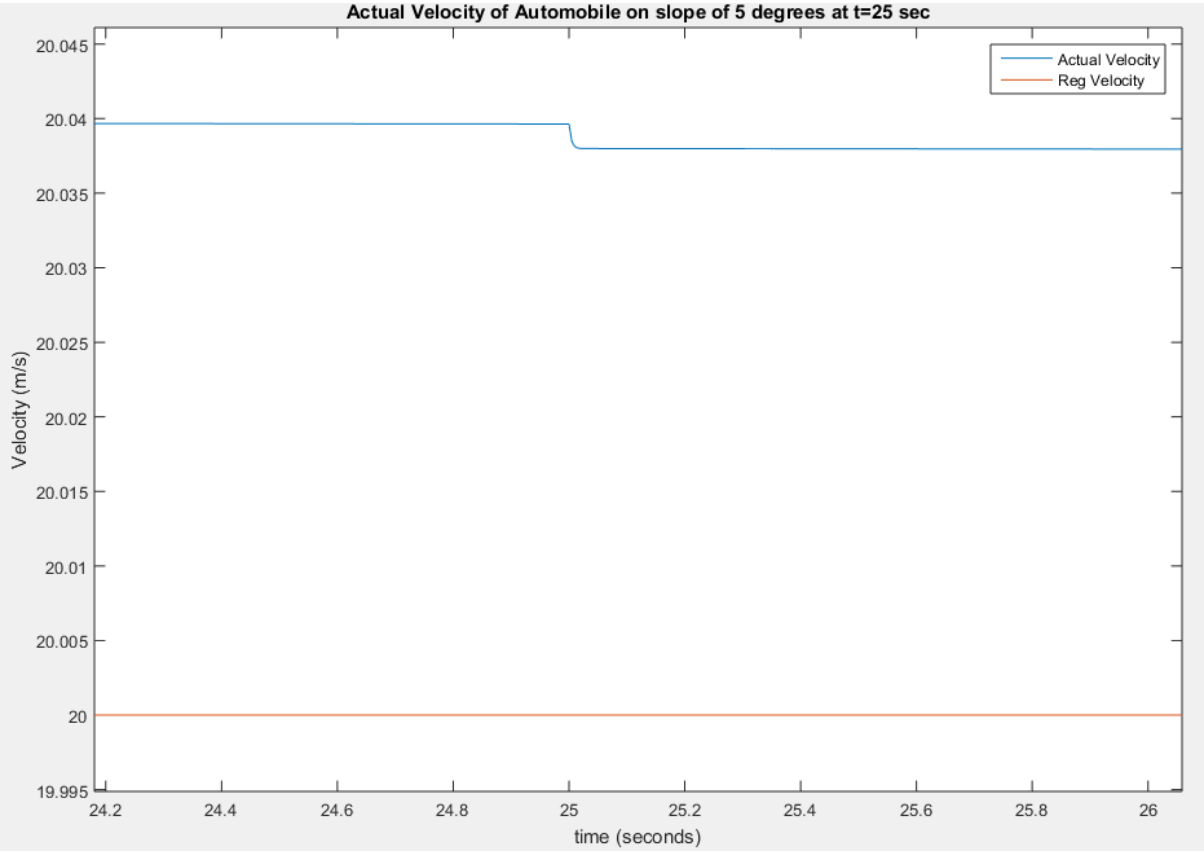
The rise time in this case is almost unchanged from the last set of gains however.

Task 4

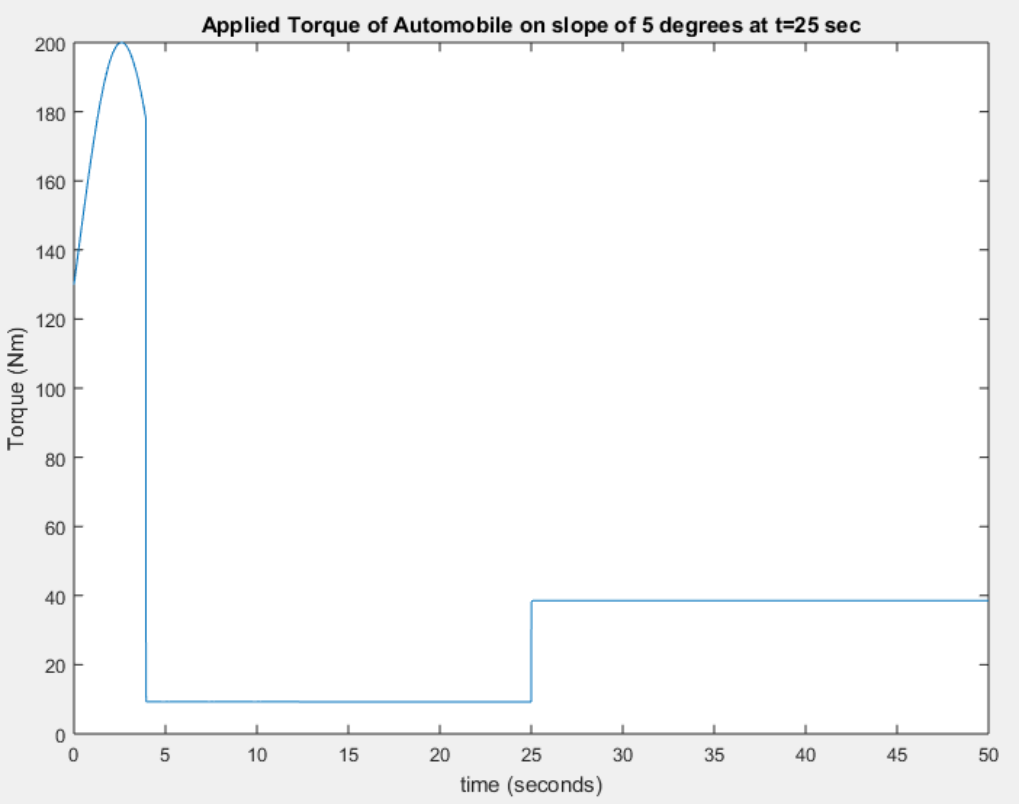
If a car was to approach an immediate slope of 5 degrees, the angle of the car would increase in time until all wheels are on the slope. Thus, the distribution of weight changes in time with the speed of the car, causing the initial stage of an incline to appear as a ramp increase of slope to 5 degrees. Even in this condition, it is unlikely that a road will instantly change slope to such a value, instead they typically increase with distance. For this model, it is appropriate to ensure that cruise control can regulate a disturbance of the worst case scenario; an instantaneous change of gradient to 5 degrees. Modelling the hill as a gradual ramp, before stopping at a constant 5 degrees inclination will show a longer duration response to that of an immediate step.

Therefore, as seen in the simulation figure in task 1, theta is modelled as a step input of 5 degrees at the time of 25 seconds; after the automobile has reached a steady-state velocity of approximately 20 m/s. The previous values of controller were used, $K_p = 100$ and $K_I = 0.1$, with the car in gear 2.

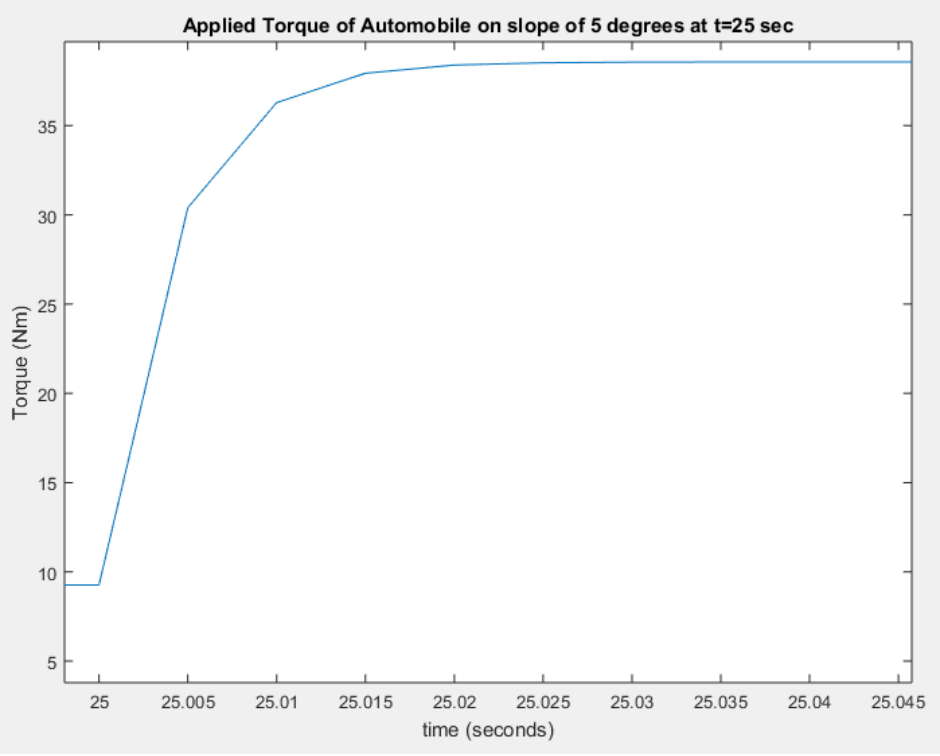

Actual Velocity of Automobile on slope of 5 degrees at t=25 sec
Actual Velocity
Reg Velocity
Velocity (m/s)
time (seconds)
0
5
10
15
20
25
30
35
40
45
50


Actual Velocity of Automobile on slope of 5 degrees at t=25 sec
Actual Velocity
Reg Velocity
Velocity (m/s)
time (seconds)
20.045
20.04
20.035
20.03
20.025
20.02
20.015
20.01
20.005
20
19.995
24.2
24.4
24.6
24.8
25
25.2
25.4
25.6
25.8
26

It can be seen that the step input of 5 degrees has minimal effect on the velocity. This is due to the quick compensation of the controller, resulting in a spike in applied torque by the motor. This is illustrated below:



It can be seen that the chosen gear does not cause the motor to supply more than the maximum 450Nm. Noticeably, it may be impractical for a motor to produce a step in torque at such as quickly as this one has at 25 seconds; approximately 4226.2 Nm/s with error due to discretisation.



Thus, a realistic model would require knowledge of the motor's response time and a more detailed calibration of the controller gains.