Assignment 3: Model Predictive Control (MPC) for path following

RO47017 Vehicle Dynamics and Control

by

Ben Halliwell

02/06/2023

Mechanical Engineering

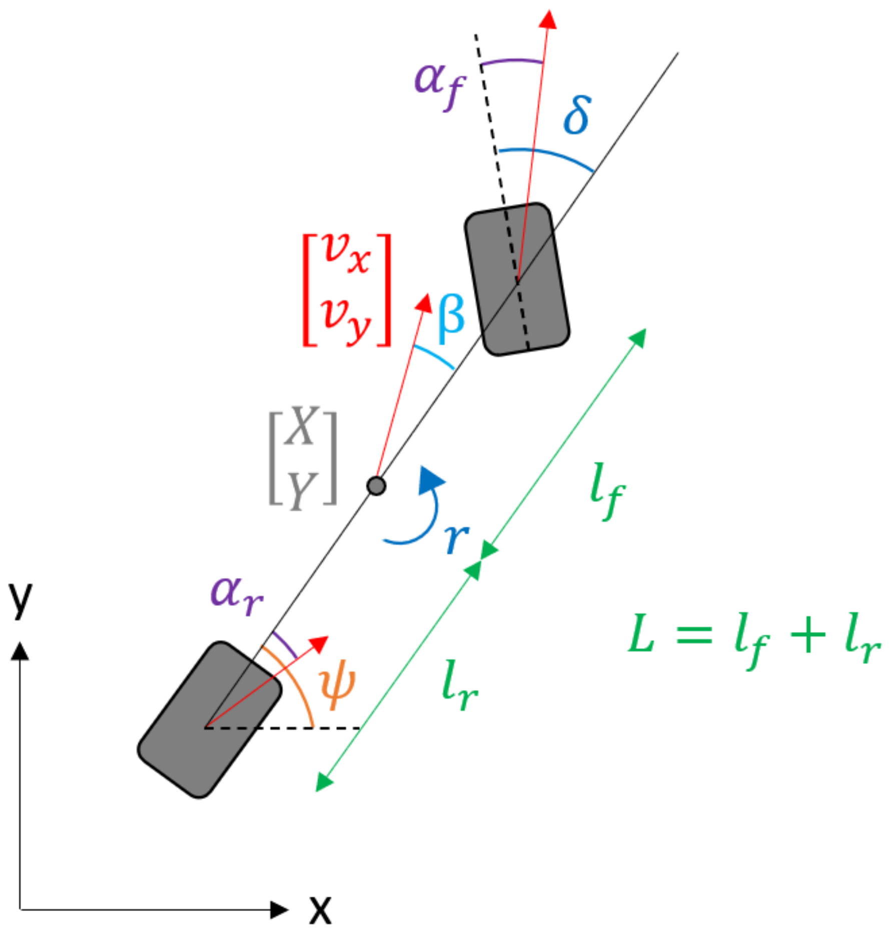
Department of Mechanical, Materials and Maritime Engineering

Contents

1. Introduction

1.1 Bicycle Model

The bicycle model is a simplified representation of a car (Figure 1), as it only considers differences between the front and rear of the car, with no difference between the inner and outer side of the car, just like a bicycle.



α_f
δ
$\begin{bmatrix} v_x \\ v_y \end{bmatrix}$
β
$\begin{bmatrix} X \\ Y \end{bmatrix}$
l_f
r
α_r
$L = l_f + l_r$
ψ
l_r
y
x

Figure 1: Bicycle Model Diagram

The bicycle model assumes a constant magnitude of velocity, as only the lateral forces on the tyres are considered. These tyres are also assumed to behave linearly, as they have a constant cornering stiffness (C_α). The model is 2D, so neglects any effects that require motion in the z-direction, such as pitch, roll or suspension dynamics.

1.2 MPC Controller

Model Predictive Control (MPC) is method used to optimise the performance of a system over a specific time horizon. The main principle is that instead of applying fixed control actions, the control inputs are dynamically adjusted at each time step based on the current state of the system and the predicted future behaviour. This requires minimising a cost function which penalises deviation from the reference signal and large variations in the control output (Equation 1).

$$J = \sum_{i=1}^{H_1} W_{1,i}(Y_{ref,i} - Y_i)^2 + \sum_{i=1}^{H_2} W_{1,i}(\Delta\delta)^2 \tag{1}$$

Prediction is particularly beneficial in path following as it allows the controller to predict where the vehicle should go, allowing it to cut the corner of the path (Figure 2). Prediction also allows an MPC controller to adapt to system disturbances making it suitable for handling uncertain and dynamic systems such as automated driving. It is also easy to add multiple constraints for the individual states of the car, which allow it to output a reasonable control signal. MPC is also useful for multivariable control, as it can optimally control multiple inputs and outputs simultaneously.

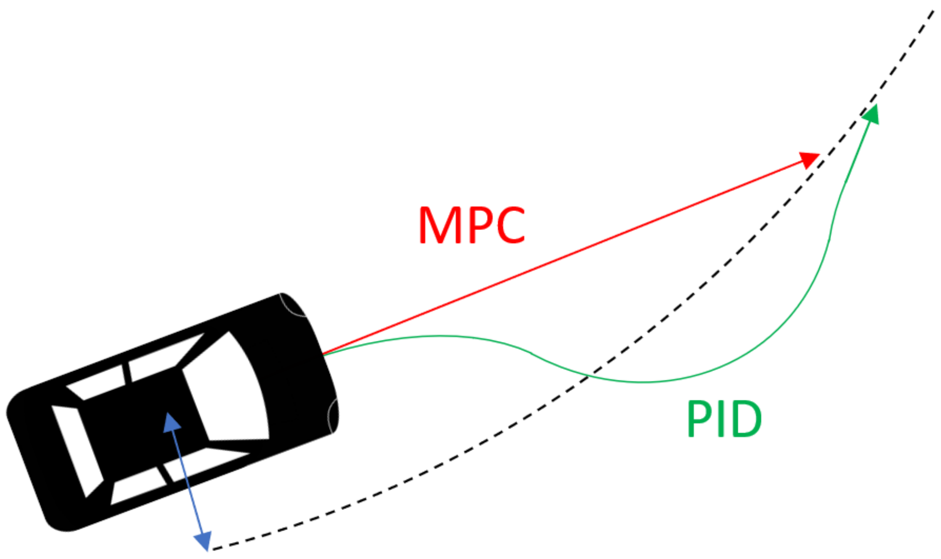


Figure 2: MPC vs (underdamped) PID Diagram

However, MPC is computationally expensive as it requires optimisation to occur at every time-step in real time. For more states and a longer time horizon, the computational power required increases significantly. In the assignment, a time horizon of 40 steps was used and a compiled computing language (C++) was required to run the simulation in a reasonable time. MPC also requires an realistic plant model to make accurate projections, which may be difficult to calculate for a vehicle with 100s of degrees of freedom.

2. Default MPC Path Follower

The default MPC used the kinematic bicycle model (Equation 2) as the theoretical plant for the controller.

$$
\begin{aligned}
\dot{v}_x &= 0 \\
\dot{\psi} &= \frac{v_x}{l_r}\sin(\beta) \\
\dot{X} &= v_x\cos(\psi + \beta) \\
\dot{Y} &= v_x\sin(\psi + \beta) \\
\text{where } \beta &= \arctan\left(\frac{l_r}{L}\tan(\delta)\right) \\
\text{-10} < \beta < 10\text{[deg], -360} &< \delta < 360 \text{ [deg]}
\end{aligned} \tag{2}
$$

The reference signal was a sine input for the lateral position (Y), so the Y-weighting was optimised with a control input weighting of 1 (Figure 3). All other weightings were set to zero. The optimisation was based off minimising the lateral position error (Equation 3) and the chatter (Equation 4). *Note: the error was calculated from 5 to 30 seconds.*

$$e = \sum_{i=1}^{n} \frac{|Y_{ref,i} - Y_i|}{n} \tag{3}$$

$$c = \sum_{i=1}^{n} \frac{1}{n}\left|\frac{d(\delta_i)}{dt}\right| \tag{4}$$

The optimum weighting was found to be 4.5e-2 (Figure 4), although the MPC performed similarly at weights between 1e-2 to 1e-1. At low weights (below 3e-3), the system experienced oscillations at a lower frequency than the reference signal, which caused the steering angle input to experience unstable growth (Appendix 1). The system became unstable even before the sine wave reference began. At high weights (above 3e-1), the system experienced oscillations at a higher frequency than the reference signal. The system remained stable for longer than the low weighting but still became unstable after 2 oscillations.

Also the absolute value of the weightings did not matter, only the ratio between them (Appendix 2).

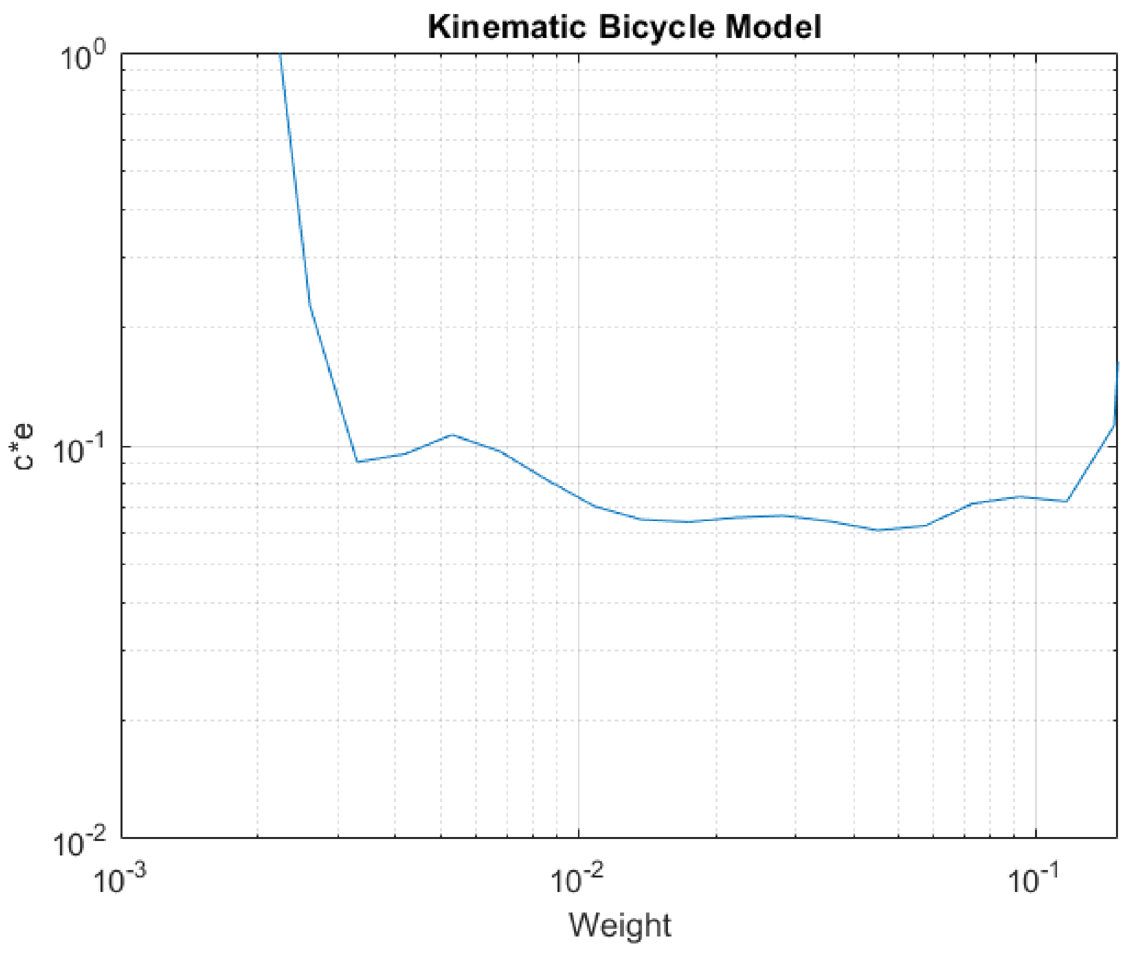


Figure 3: Error Optimisation of Kinematic Bicycle Model MPC

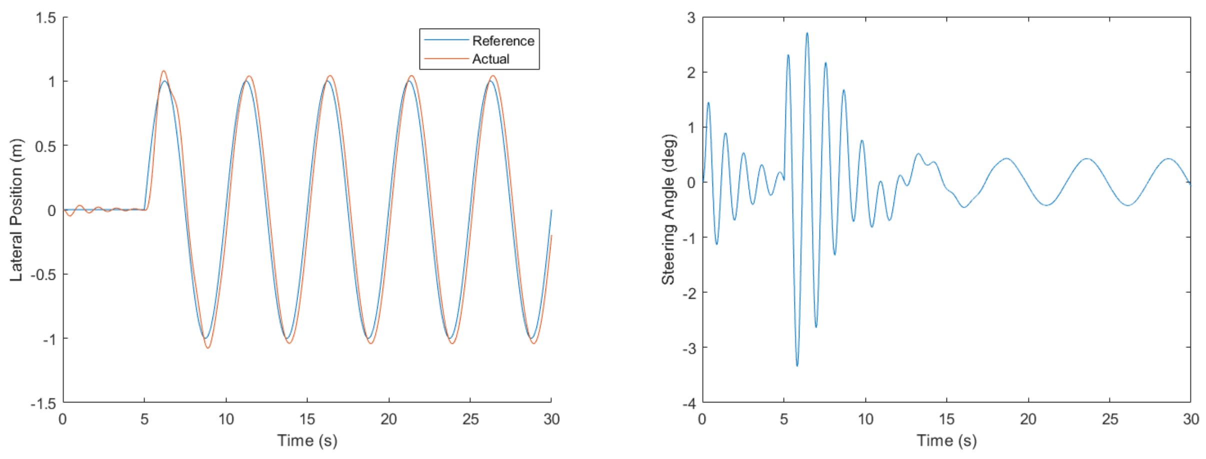


Figure 4: Default MPC Reference Signal (L) & Steering Angle (R) for Y-weight = 4.5e-2

3. Designed MPC Path Follower

The kinematic bicycle model was replaced with the dynamic kinematic model (Equation 5).

$$
\begin{aligned}
\dot{V_x} &= v_y r \\
\dot{V_y} &= -\left(\frac{C_{\alpha,f}+C_{\alpha,r}}{m v_x}\right) v_y + \left(\frac{l_r C_{\alpha,r} - l_f C_{\alpha,r}}{m v_x} - v_x\right) r + \frac{C_{\alpha,f}}{m}\delta \\
\dot{r} &= \left(\frac{l_r C_{\alpha,r} - l_f C_{\alpha,r}}{I_z v_x}\right) v_y + \left(\frac{l_r^2 C_{\alpha,r} + l_f^2 C_{\alpha,r}}{I_z v_x}\right) r + \frac{l_f C_{\alpha,f}}{I_z}\delta \\
\dot{\psi} &= r \\
\dot{X} &= V_x \text{cos}(\psi) - V_y \text{sin}(\psi) \\
\dot{Y} &= V_x \text{sin}(\psi) + V_y \text{cos}(\psi) \\
\dot{\delta} &= d_\delta \\
&\text{-10} < \beta < 10\text{[deg], -360} < \delta < 360\text{[deg], -800} < \dot{\delta} < 800 \text{ [deg/s]}
\end{aligned}
\tag{5}
$$

The same optimisation was used for the dynamic bicycle model, giving an optimum Y-weighing of 3. The error increase at low weightings is much lower than for the kinematic bicycle model. Above a Y-weighting of around 4, the error increases significantly. Unlike the kinematic bicycle model MPC, the dynamic model does not become unstable at suboptimal weights. At low weights, the lateral postion error increases whereas at high weights, the chattering performance is insufficient (Appendix 3).

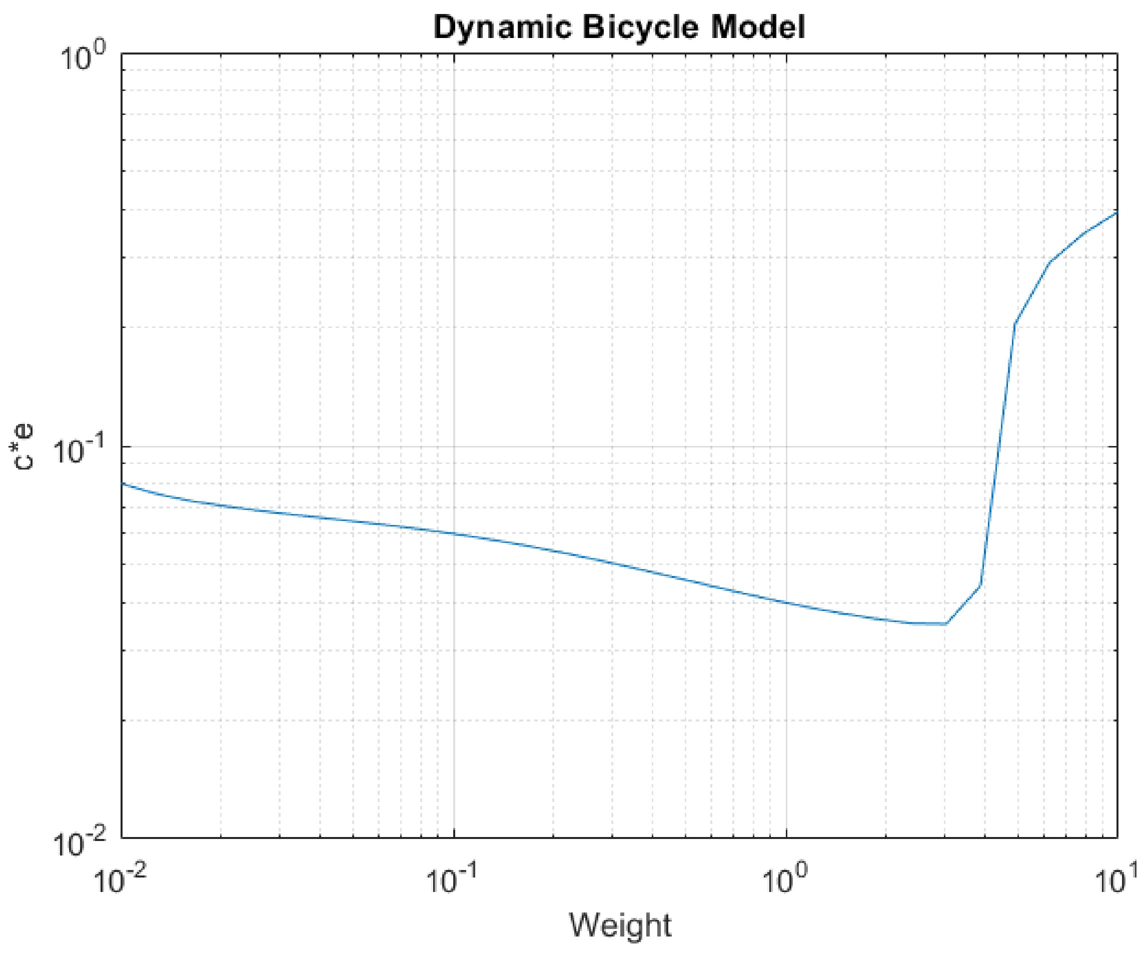


Figure 5: Error Optimisation of Dynamic Bicycle Model MPC

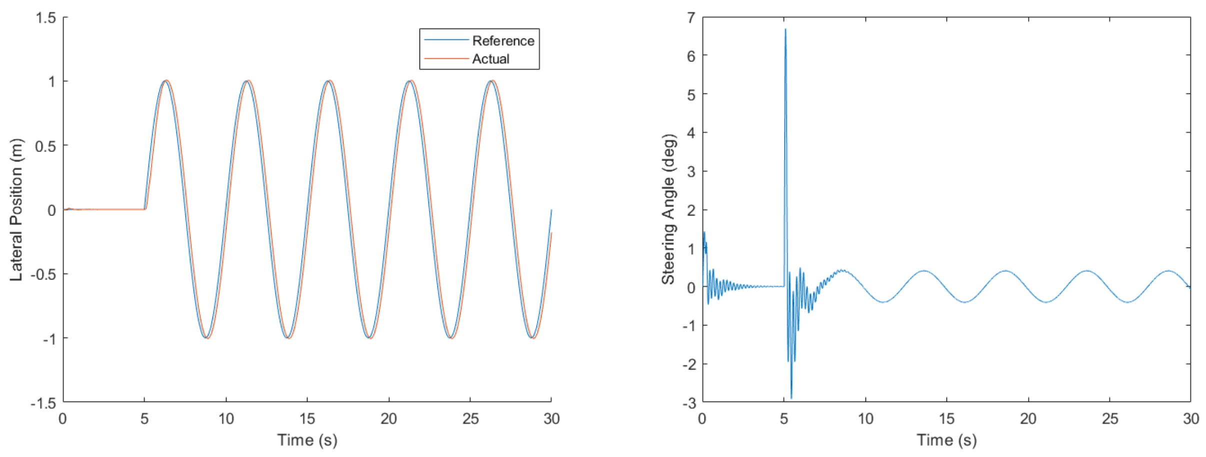


Figure 6: Designed MPC Reference Signal (L) & Steering Angle (R) for Y-weight = 3

4. Controller Comparison

RMS error was used to compare the two controllers for lateral position (Equation 6) and steering angle (Equation 7). Since the steering angle did not have a reference, the RMS was calculated against zero steering angle. *Note: the error was calculated from 5 to 30 seconds.*

$$e_{\text{Y,RMS}} = \sqrt{\frac{\Sigma_{i=1}^{N}(Y_{i,ref} - Y_i)^2}{N}} \tag{6}$$

$$e_{\delta,\text{RMS}} = \sqrt{\frac{\Sigma_{i=1}^{N}(0 - \delta_i)^2}{N}} \tag{7}$$

Table 2: RMS Error

Controller	Y Error (m)	δ Error (deg)
Kinematic	0.1441	0.7414
Dynamic	0.1260	0.5468

As shown in table 2, the dynamic bicycle model MPC was superior in lateral position tracking and steering input. This makes sense because the dynamic bicycle model is more similar to the actual plant dynamics than the kinematic bicycle model. However the dynamic bicycle model requires a detailed understanding of the car's properties, such as the cornering stiffness (C_α) and moment of inertia about the z axis (I_z). The kinematic bicycle model only requires the centre of gravity and the length of the vehicle as inputs.

It is also clear that the dynamic model has a higher bandwidth as it can closely follow the reference signal up to a frequency of 0.35 Hz, whereas the kinematic model is only stable until 0.25 Hz. *Note: all other graphs show a reference signal frequency of 0.2 Hz.*

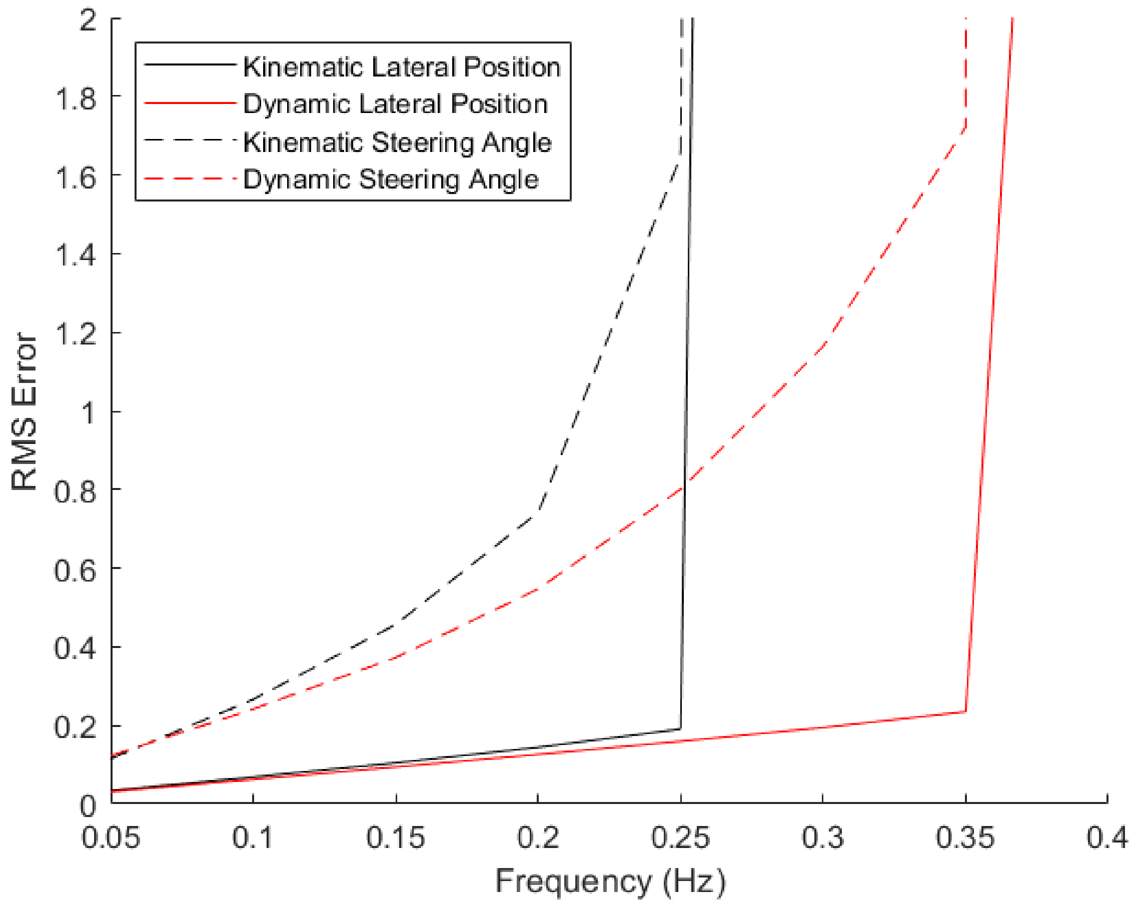


Figure 7: RMS error for kinematic and dynamic bicycle model

5. Reflection

Overall the most challenging part of this assignment was actually getting the MPC controller to control the lateral position for the designed controller. This was because the steering angle output remained at zero even though I changed the weights, order of states and constraints. It was particularly time consuming as writing and compiling the ACADO C++ files took around 45 seconds each time I wanted to make a small adjustment to the initialisation file. In the end I had to start from the template file again. Also, downloading and linking the compiler to MATLAB took a significant amount of time.

On the other hand the optimisation of the weight was very quick, as only the Y-position weight was adjusted with the MPC controller bearing most of the optimisation workload. This is in stark contrast to the previous two assignments where optimisation was the most time-consuming task.

Overall, the main learning outcomes were the ability to use an MPC controller to follow a given path. Unlike the previous assignments, the focus was on extending pre-existing controller dynamics to a more advanced model, instead of simply tuning the controller. This produces its own challenges as it required me to read and understand someone else's MATLAB and Simulink code, which is notoriously difficult.

6. Conclusion

Overall, the dynamic bicycle model surpassed the kinematic bicycle model in reference tracking and control output. However, the controller could be further extended by optimising the weights for the other state variables to give an even better performance. Alternatively, the MPC controller could be tuned to increase the bandwidth beyond 0.35 Hz, as relatively high frequencies could be experienced during evasive manoeuvres.

7. Appendix

7.1 Appendix 1: Default (Kinematic) MPC Path Follower

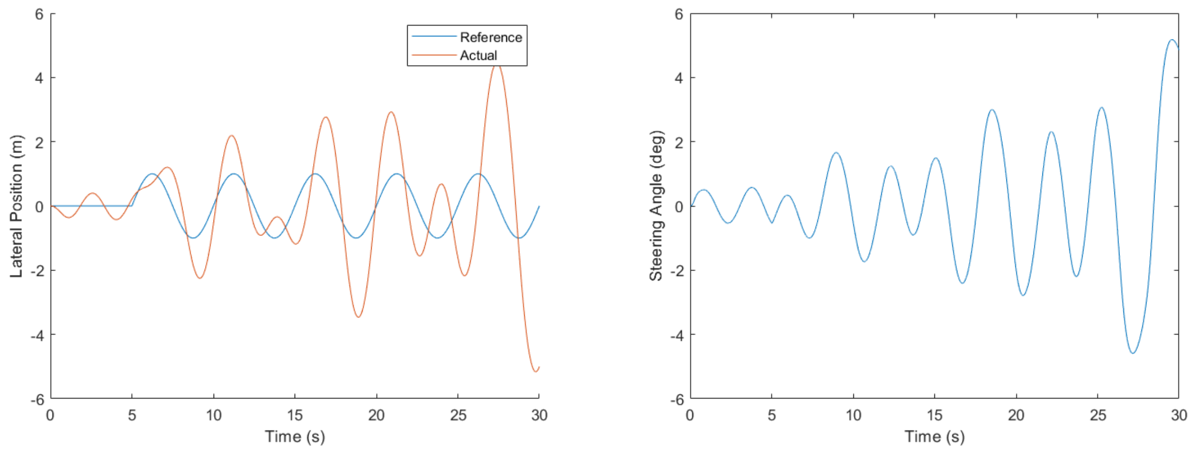


Figure 8: Default MPC Reference Signal (L) & Default MPC Steering Angle (R) for Y-weight = 1e-3

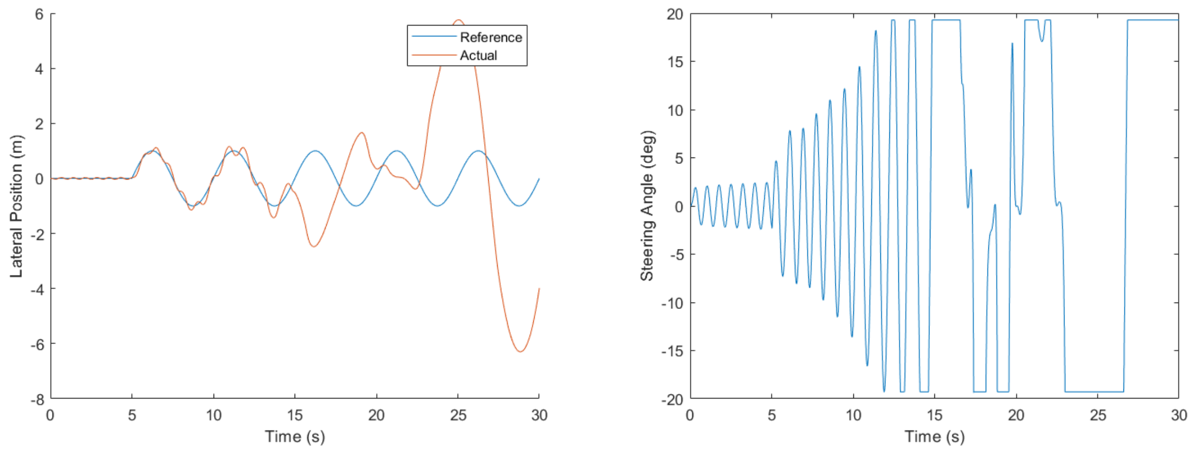


Figure 9: Default MPC Reference Signal (L) & Default MPC Steering Angle (R) for Y-weight = 0.3

7.2 Appendix 2: Absolute Weightings

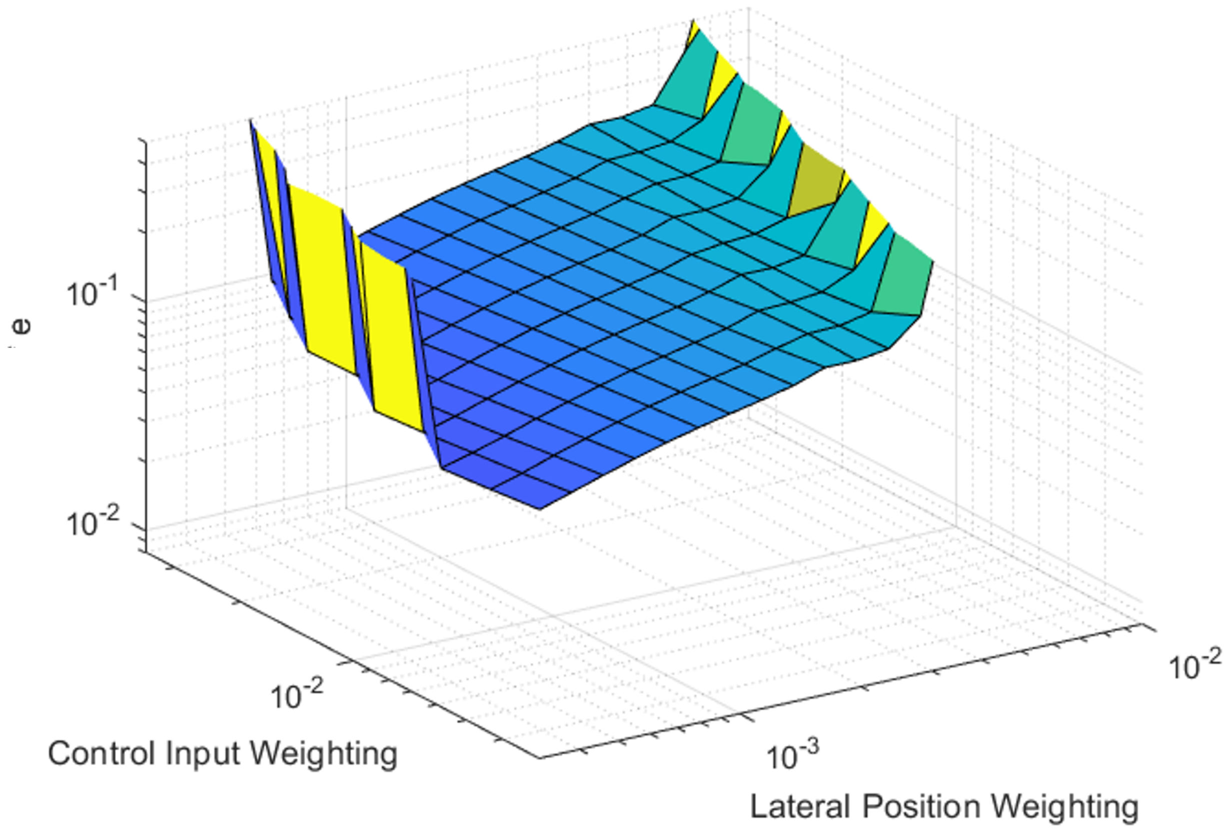


Figure 10: Error optimisation with absolute weightings

7.3 Appendix 3: Designed (Kinematic) MPC Path Follower

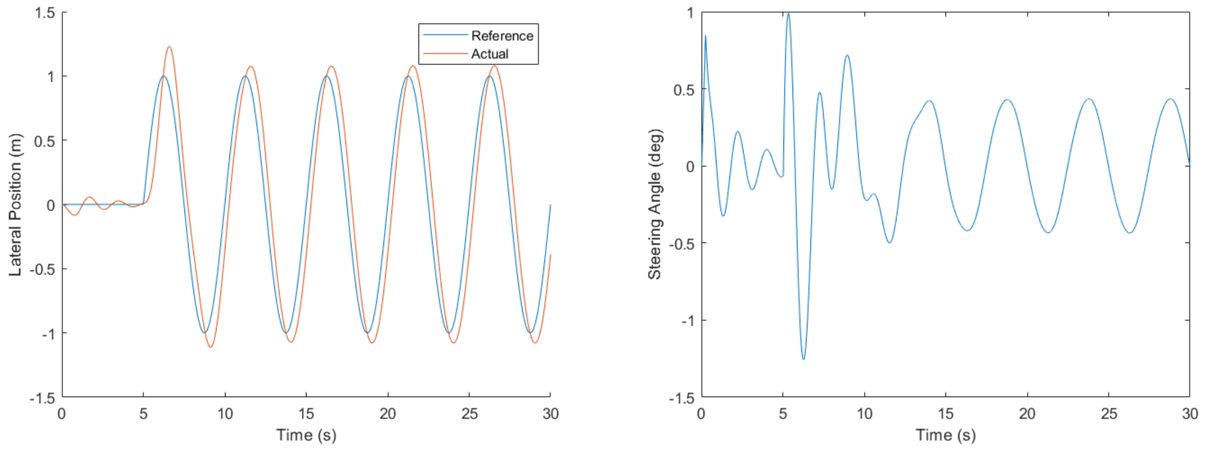


Figure 11: Designed MPC Reference Signal (L) & Default MPC Steering Angle (R) for Y-weight = 0.01

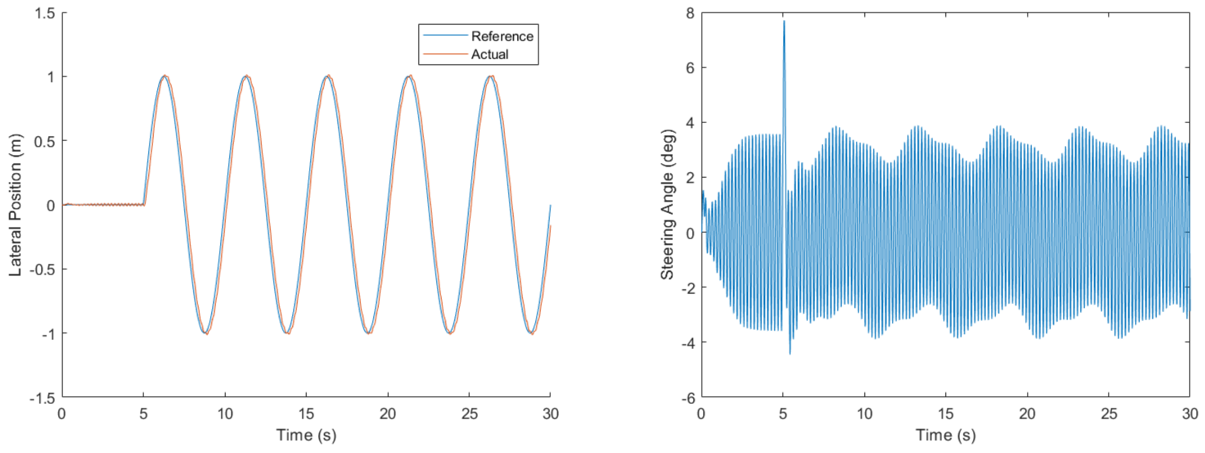


Figure 12: Designed MPC Reference Signal (L) & Default MPC Steering Angle (R) for Y-weight = 5

References