
# Assignment 3 - Path Following

Vehicle Dynamics and Control
(RO47017)

**Author:**

Nicola Visentin (6354815)

June 6, 2025

# Contents

# Introduction

This assignment addresses lateral control of a vehicle for the tracking of a certain desired path. In particular, an MPC is used to compute the steering input necessary for bringing to zero the error on the lateral position of a car with respect to a reference, which is shown in Figure 7, Appendix A. Simulink and Matlab, along with ACADO optimization toolkit, are employed to run simulations using a 9 DOF vehicle model with nonlinear tire dynamics. All vehicle, simulation and controller parameters are reported in Appendix A.
First, a simple kinematic model of the vehicle is used for implementing the MPC regulator: this is presented in section 1. In section 2 a more refined representation is considered, also including (linear) tires dynamics. Then, in section 3, simulation results for both the tested solutions are compared. Finally, conclusions and considerations are drawn in section 4.

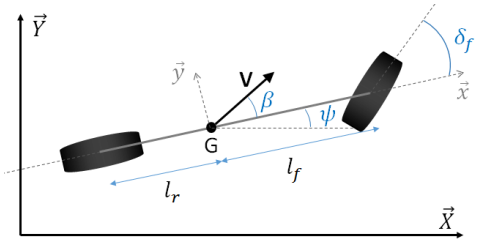
# 1 Default MPC path-follower

In first place, an MPC regulator relying on a kinematic bicycle model is used. MPC theory and derivation is not covered here, as it is not the purpose of the assignment, however the very basic idea consists in solving an optimization problem to compute the best control strategy that brings the car from the current position to the desired position in the next $N_p$ time steps (where $N_p$ is the prediction horizon). Then, only the first control input of the sequence is applied to the real vehicle and the procedure is repeated after $T_s$ seconds, where $T_s$ is the controller sampling time.
Obviously, repeating the optimization and computing the predicted dynamics at each time step makes this control strategy particularly intensive from the computational point of view, and also requires a model of the system and a certain cost function (weights). Our focus will be from this moment on those two elements.

**Model** A nonlinear kinematic bicycle model with 4 states is considered:

$$\begin{cases} \dot{u} = 0 \\ \dot{\psi} = \dfrac{u}{l_r} \sin\beta \\ \dot{x} = u\cos(\psi + \beta) \\ \dot{y} = u\sin(\psi + \beta) \end{cases}$$

$$\beta = \arctan\left(\frac{l_r}{L}\tan\delta\right)$$




where $x$, $y$ are the coordinates in the global frame, $\psi$ is the yaw angle (heading), $u$ the longitudinal constant velocity and $\beta$ the vehicle sideslip angle. Notice that this model is purely kinematic, and no forces nor accelerations are considered. Our input variable is the steering angle of the wheel $\delta$.

**Weights and constraints** The previous equations are used by the MPC to predict the future behaviour of the car. for this reason, in this type of controller we also have the possibility to impose some constraints on the states and the input (this is another big advantage of MPC with respect to other regulators). In particular, we imposed:

- $|\delta_{\text{steering wheel}}| < 360^o$, which corresponds to a limit on the steering angle of the wheel $\delta$ of about $\pm 0.41$ rad.
- $|\beta| < 10^o$ to avoid having critical sideslip angles, that are problematic.

Finally, since MPC is an optimal controller, we also need to define a cost function for the minimization. A quadratic cost is chosen (so the problem is convex and QP algorithms can be used), with a penalization

on the lateral error initially set to $q_y = 0.005$ and a weight $R = 0.1$ on the control input $\delta$. All the other states are not penalized, and the final weight matrix is null as well[1].

**Simulations** Results of the simulation with the parameters mentioned above is shown in Figure 1. The Root Mean Square Error (RMSE) is used to quantify the tracking accuracy, and in this case it is 0.1275 m. The control input $\delta$ is not too aggressive nor unrealistic, and the steering angle rate is limited as well.

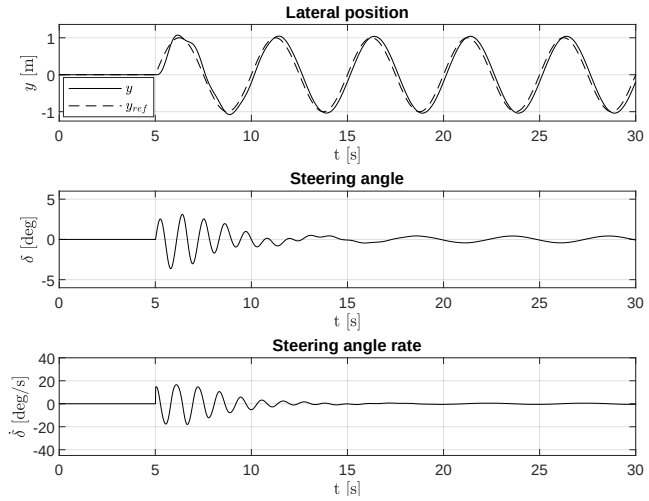



*Figure 1: Kinematic model. $q_y = 0.005$ and $R = 0.1$*

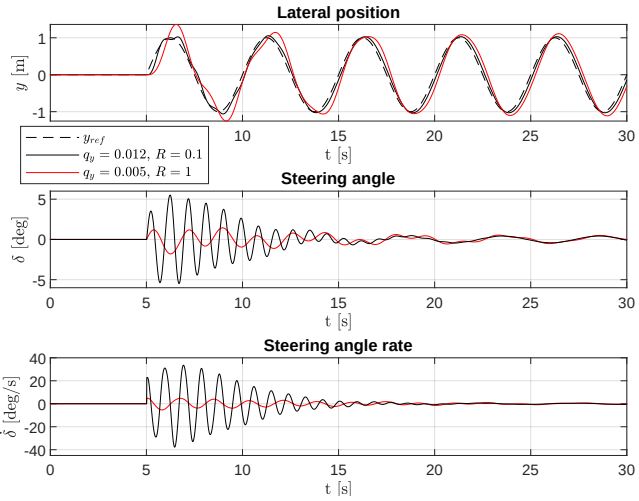



*Figure 2: Kinematic model. Varying the weights*

Then the system was simulated with different weights to test their effect on the performance. As we can see from Figure 2 (black line), increasing the penalization on the lateral error ($q_y = 0.012$) leads to a better tracking (RMSE=0.0968 m), but causes a more aggressive steering (max amplitudes of $\delta$ and $\dot{\delta}$ are more or less $5.5^o$ and $38^o/s$ in this case, compared to $3.5^o$ and $18^o/s$ of the previous case). It was also observed that decreasing $R$ has the same effect of increasing $q_y$: this is expected, if one thinks of how the cost function is defined. On the other hand, an higher $R$ (or lower $q_y$) penalizes more the control, leading to a clearly "more calm" input (red line in Figure 2), but a worst tracking performance (RMSE=0.227 m).

For values of $q_y$ that are too large ($q_y = 0.050$), the controller becomes unstable (black line in Figure 3), as it tries to converge too aggressively. In reality, the stability of an MPC controller depends on many other factors, like the prediction horizon, the constraints, the initial condition and the terminal weight matrix [1]. Just as an example, a simulation with a terminal weight $p_y = 100$ on the lateral error is presented in Figure 3 (red line).

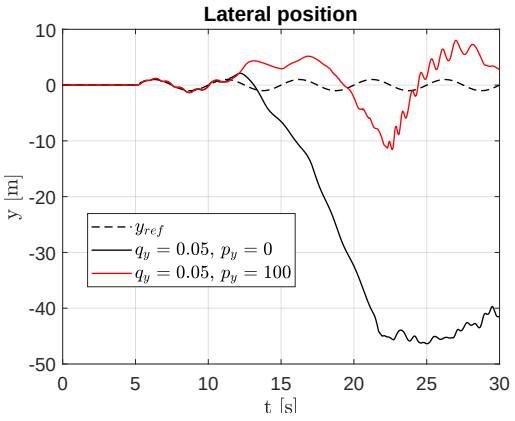



*Figure 3: Kinematic model. Effect of the terminal weight*

All the various simulations results and parameters are collected in Table 1.

[1]We are only penalizing the lateral error since that's the only thing we are interested in having close to zero. The terminal cost matrix is also null: usually this matrix, along with the terminal constraint, is tuned in order to guarantee stability [1]. No terminal constraints are included.

| $q_y$ | R | $p_y$ | RMSE [m] | Control effort | Notes | Figure |
|---|---|---|---|---|---|---|
| 0.005 | 0.1 | 0 | 0.1275 | limited | | 1 |
| 0.012 | 0.1 | 0 | 0.0968 | less limited | | 2 |
| 0.005 | 1 | 0 | 0.2270 | very limited | | 2 |
| 0.05 | 0.1 | 0 | [unstable] | - | immediate severe divergence | 3 |
| 0.05 | 0.1 | 100 | [unstable] | - | divergence is more contained | 3 |

*Table 1: Simulations of the kinematic-based MPC*

## 2 Designed MPC path-follower

Now a more refined model is used for the MPC formulation. This means that its predictions will be more accurate and realistic, potentially leading to better results, but this of course comes at the price of an higher computational effort. Moreover, having a more complex dynamics means that we need to include more parameters, so we need to have a good estimation/knowledge of them.

**Model** The model in this case is a 7 DOF dynamic bicycle that includes linear tires behaviour. The equations are the following:

$$\begin{cases} \dot{u} = v\ r \\ \dot{v} = -\dfrac{C_{\alpha f} + C_{\alpha r}}{m\ u}\ v + \left(\dfrac{l_r C_{\alpha r} - l_f C_{\alpha f}}{m\ u} - u\right)\ r\ + \dfrac{C_{\alpha f}}{m}\delta \\ \dot{r} = \dfrac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z\ u}\ v\ - \dfrac{l_r^2 C_{\alpha r} + l_f^2 C_{\alpha f}}{I_z\ u}\ r\ + \dfrac{l_f C_{\alpha f}}{I_z}\delta \\ \dot{\psi} = r \\ \dot{x} = u\cos\psi - v\sin\psi \\ \dot{y} = u\sin\psi + v\cos\psi \\ \dot{\delta} = \dot{\delta} \end{cases}$$

$$\beta = \arctan\frac{v}{u}$$

where now we have additional states. $r$ is the yaw rate, $v$ is the lateral velocity, steering angle $\delta$ is now a state, while our control input is the steering angle rate $\dot{\delta}$.

**Weights and constraints** Having more states means that we need to define bigger weight matrices. However, also in this case the only state we want to penalize is the lateral position, so all the additional weights are set to zero. Again, we have only one input, but in this case it is different ($\dot{\delta}$ instead of $\delta$), so $R$ needs to be re-tuned: a value of R$= 10^{-4}$ is set.
We chose to have the steering wheel rate as input so that we are actually able to also impose a constraint on that. Our states and input bounds, with this new formulation, are:

- $|\delta_{\text{steering wheel}}| < 360^o$, as before.
- $|\beta| < 10^o$ as before.
- $|\dot{\delta}_{\text{steering wheel}}| < 540^o/s$, i.e. $|\dot{\delta}| < 0.61$ rad/s approximately. This limit corresponds to the driver[2] moving the steering wheel at 1.5 revolutions per second, and was decided drawing inspiration from [2] (here is reported $\sim 570^o/s$ in emergency situations).

Notice that now we also have the possibility to impose a constraint on $\dot{\delta}$, which was not possible with the previous kinematic model, leading in some cases to critical steering rates (e.g. Figure 2).

[2]even if in this case we don't have a driver, as the MPC is controlling the steering

**Simulations** Simulation results for the new dynamic bicycle-based MPC with the settings above are shown in Figure 4. Also in this case, the tracking is quite accurate and the control signal is very contained.

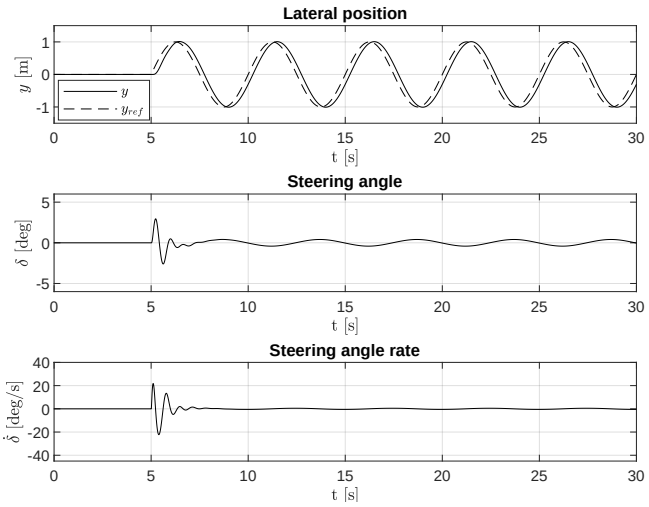



*Figure 4: Dynamic model.* $q_y = 0.005$ *and* $R = 10^{-4}$

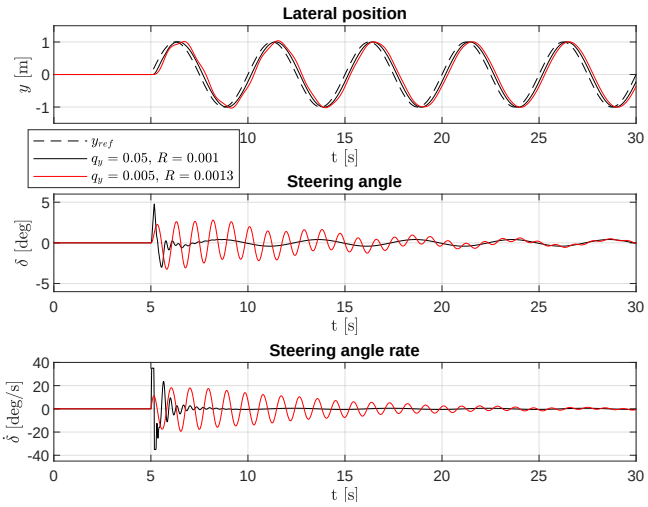



*Figure 5: Dynamic model. Varying the weights*

Similarly to before, weights were varied, and the same considerations of the previous section still hold. Results are reported in Figure 5 and in Table 2. Notice how the constraints on the new control input are now limiting the maximum amplitude of $\dot{\delta}$ when the control action is more aggressive (black line, Figure 5), which was not possible with the first MPC formulation. Moreover, we notice that when we increase the penalization on the control input (red line in Figure 5), now the input $\dot{\delta}$ seems to be more oscillatory than before, even if we expect it to be more contained. This may be due to the fact that the higher penalization keeps the steering rate low at the beginning (in fact the first "oscillation" has a lower amplitude), and then we need to "oscillate more" to make up for this.
Finally, also in this case instability of the controller is observed when the weights are too demanding.

| $\mathbf{q_y}$ | **R** | **RMSE [m]** | **Control effort** | **Notes** | **Figure** |
|---|---|---|---|---|---|
| 0.005 | 0.0001 | 0.1960 | very limited | | 1 |
| 0.05 | 0.0001 | 0.1419 | more aggressive | high frequency and saturated input | 2 |
| 0.005 | 0.0013 | 0.2270 | limited | | 2 |
| 0.005 | 0.01 | [unstable] | - | | - |

*Table 2: Simulations of the dynamic-based MPC*

## 3 Comparison

In this section we present a comparison between the results obtained with the two different MPC controllers.
In both cases, with a proper tuning of the weights in the cost function, it is possible to achieve a good tracking of the reference. In general, the dynamic-based MPC is able to guarantee a lower control effort, having both $\delta$ and $\dot{\delta}$ with much lower amplitudes and less oscillations: this is clear if we compare Figure 1 and Figure 4.
The tracking performance is comparable, but in general seems to be slightly better for the original MPC (lower RMSE). This could be due to different reasons. First of all, the re-designed MPC also includes an additional constraint on the steering angle rate, which previously was not accounted for. This limits the capability of the controller to produce an aggressive control action that can indeed reduce the tracking

error. We can confirm this by looking at Figure 2 and Figure 5: here, the black lines are the ones corresponding to the increased penalization on the lateral error. For the kinematic MPC, we have $|\dot{\delta}|$ that almost hits $40^o/s$, while the dynamic MPC is not allowed to exceed $\sim 34^o/s$ (and in fact it saturates). Second reason is the fact that the control input of the model (i.e. output of the MPC) is different ($\delta$ in the first case and $\dot{\delta}$ in the second), and this may influence somehow the optimization/prediction procedure. To check this and have a "fair" comparison, a simulation was run using a "modified" dynamic-based MPC which 6 states and $\delta$ as control input: results in Figure 6 clearly show that now the dynamic-based model outperforms the kinetic one in terms of RMSE (0.0436 m vs 0.0968 m). Obviously, having such a controller does not allow to set any limitation on the steering rate, and the resulting control action is in fact extreme (notice how the constraint on $\delta$ was maintained).

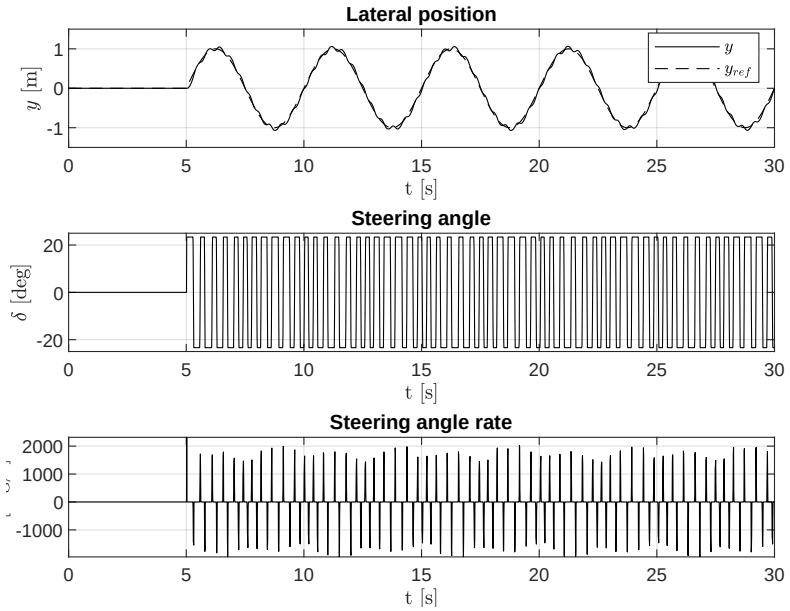



*Figure 6: "Modified" dynamic model, with $\delta$ as input and no constraints on the steering rate*

Finally, some considerations can be done about stability. In general, it was noticed that the re-designed MPC was much more robust, allowing to vary more "freely" the weights, while the kinematic-based MPC tended to go unstable more easily.

# 4 Self reflection

The main challenge faced during the assignment was probably related to understanding the underlying behaviour of the MPC, and in particular trying to justify some unexpected results (as mentioned, the lower RMSE for the kinematic model and the fact that the control effort seemed to increase when more penalized, in the re-designed MPC). However, after many tests, some possible explanations were found. Also the tuning of the weights for the re-designed MPC was a bit challenging, as the different choice of the model didn't allow to use the previous ones.

# Conclusions

In this assignment, the effect of the optimization weights on a kinematic-based MPC was explored. Then a dynamic bicycle model was used to design a more refined MPC. The main differences between the two controllers are reported here:

- **Complexity**: the kinematic-based MPC relies on a simpler model with less states. The dynamic-based MPC additionally models (linear) tires dynamics, and includes more states and parameters; this also means that we need to have a good knowledge of those parameters.
- **Flexibility**: the second implementation offers the possibility to define more constraints, as it includes more variables. This allows us to set some "realistic constraints" that make the MPC predictions reliable. Moreover, we are able to set more weights, potentially having better "control" on how we want our regulator to behave.
- **Computational effort**: kinematic equations are much simpler, thus offering a lower computational load.
- **Control input (MPC output)**: in the first case steering angle $\delta$ was chosen, while $\dot{\delta}$ was selected in the second one. This was driven by the possibility of including an additional bound on the steering rate.

The two MPCs were compared by mean of simulations. In general, the more complex model allows for additional flexibility and a better control on the control effort, but may slightly sacrifice tracking accuracy in some cases.
Finally, in a practical application we always need to choose the controller considering a trade off between a very refined design and the improvements that it is able to offer. For example, the kinematic-based MPC is much faster, thus opening the doors for having larger prediction horizons $N_p$, that in same cases can be beneficial. Moreover, it can run at higher frequencies and it is more suitable for real-time implementation.

# References

[1] J. Rawlings and D. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2008.

[2] Francesco Comolli, Massimiliano Gobbi, and Gianpiero Mastinu. Study on the driver/steering wheel interaction in emergency situations. *Applied Sciences*, 10(20), 2020.

# A Appendix - Parameters

| Parameter | Value | Unit | Description |
|---|---|---|---|
| $g$ | 9.81 | $m/s^2$ | Gravitational acceleration |
| $V_0$ | 100 | Km/h | Constant speed before the maneuver |
| $m$ | 1380 | kg | Vehicle mass |
| $I_z$ | 2634.5 | $kg \cdot m^2$ | Moment of inertia about z-axis |
| $L$ | 2.79 | m | Wheelbase |
| $l_f$ | 1.384 | m | Distance from front axle to CoG |
| $l_r$ | 1.406 | m | Distance from rear axle to CoG |
| $i_{\text{steer}}$ | 15.4 | - | Steering ratio |
| $m_f$ | 695.4 | kg | Front sprung mass |
| $m_r$ | 684.6 | kg | Rear sprung mass |
| $C_{\alpha f}$ | 120000 | N/rad | Front axle cornering stiffness |
| $C_{\alpha r}$ | 190000 | N/rad | Rear axle cornering stiffness |
| $K_{\text{us}}$ | 0.0022 | $N^{-1}$ | Understeer gradient |

*Table 3: Vehicle parameters*

| Description | Value |
|---|---|
| Simulation duration | 30 s |
| Simulation frequency | 1000 Hz |
| MPC frequency | 100 Hz |
| MPC prediction horizon | 40 time steps (4 s) |
| Maneuver start time | 5 s |
| Maneuver type (reference lateral position) | $y_{ref}(t) = sin(\frac{2\pi}{5}t)$ |

*Table 4: Simulation and controller parameters*

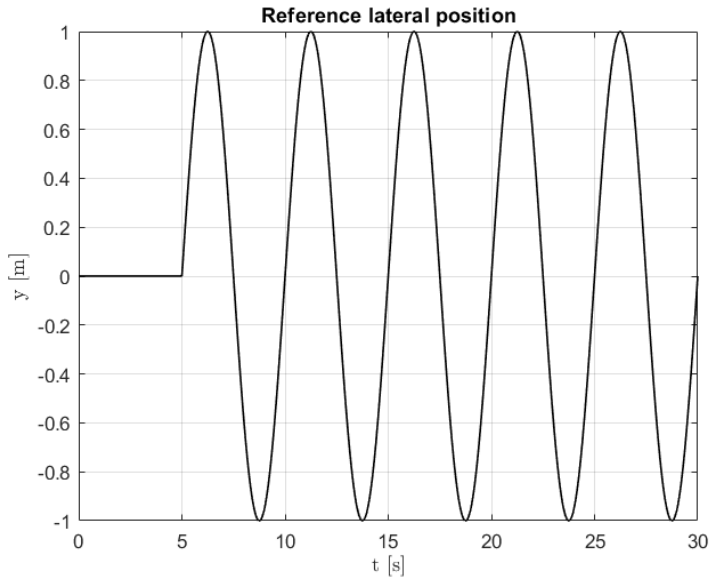



*Figure 7: Desired maneuver*

# B Appendix - Simulink models

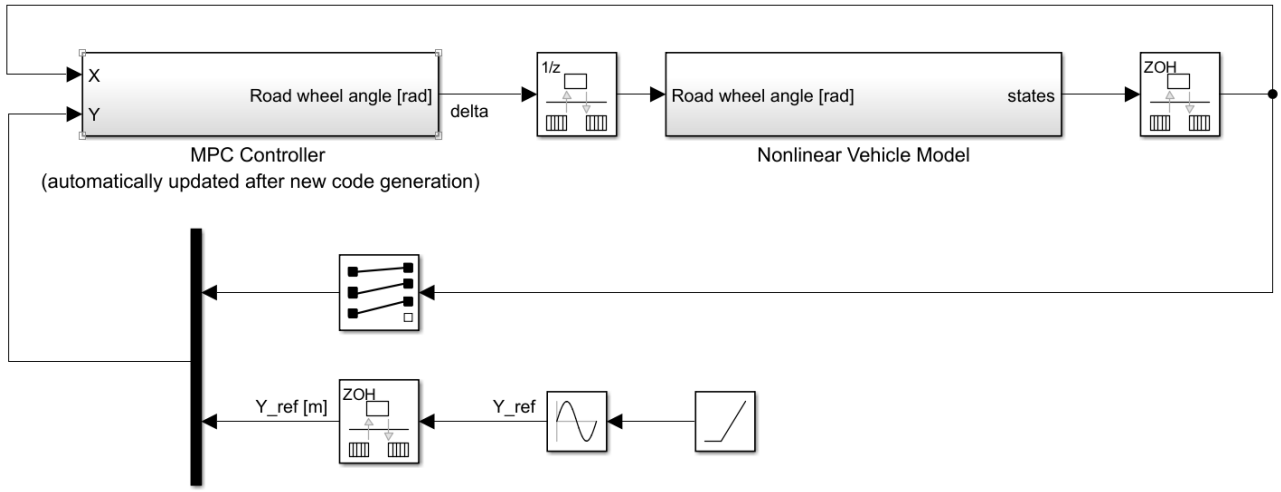



*Figure 8: Simulink architecture for the kinematic-based MPC*

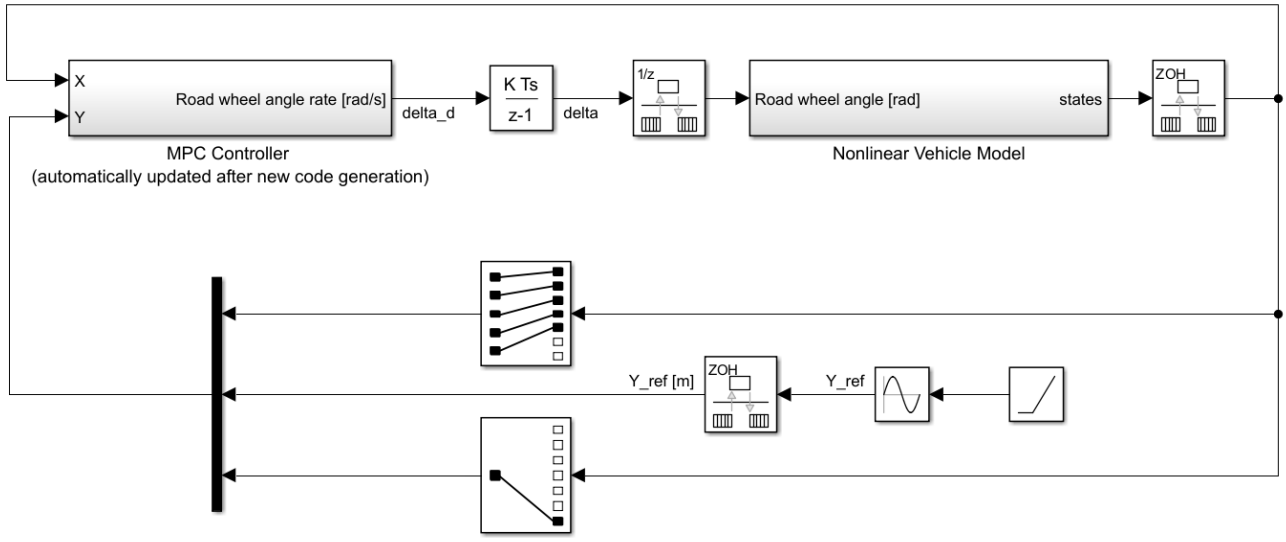



*Figure 9: Simulink architecture for the dynamic-based MPC*