Perfromance comparison between MPC and PI controller for DC motor driven toy car

Tejas Badakere Gopalakrishna[a]

[a]*University of California San Diego*

Abstract

In this study a toy car driven by a DC motor is studied under two different feedback controller: Propotional Integral (PI) Controller and Model Predictive Controller (MPC). The performance of the feedback controllers are also compared with that of a naive DC car without the feedback controller.The PI Controller is built in MATLAB Simulink environment, while a Non linear MPC algorithm is written in the MATLAB. The PI controller provides more stable control over the DC car and maintains uniform acceleration. In contrast, the MPC algorithm minimizes slip between the front and rear wheels, but the car exhibits non-uniform acceleration and fluctuating speed during steady-state operation.

1. Governing Equations

The governing equations for the combined DC motor and Car are given below. The input to the system is the Voltage **V**.

$$\frac{di}{dt} = -\frac{R}{L}i - \frac{K_b}{L}\omega + \frac{1}{L}V \tag{1}$$

$$\frac{d\omega}{dt} = \frac{K_m}{J}i - \frac{b}{J}\omega - \frac{r_w F_T}{GR \cdot J} \tag{2}$$

$$\frac{d\theta}{dt} = \omega \tag{3}$$

$$\frac{dv}{dt} = \frac{F_T}{m} \tag{4}$$

$$\frac{dx_p}{dt} = v \tag{5}$$

i, ω, θ, v, x_p are the states of the dynamical system. GR is the gear ratio defined by ratio of speeds between the motor and the back wheel, R is the resistance of the motor, L is the length of the resistor, K_b is the Back electromotive force (EMF), K_m is the Motor torque constant, b is the Viscous damping coefficient, rw is the radius of the wheel, is the moment of inertia of wheel, m is the mass of the wheel, is is the traction force given by $F_T = \mu mgQ$. where Q is the weight balance factor, μ is calculated from equation 6.

$$\mu = a_1\left(1 - e^{c_1 s}\right) + a_2\left(1 - e^{c_2 s}\right) + a_3 s \tag{6}$$

where the values of the coefficients are as follows; a1=1.63 a2= -0.9 a3= -0.1 c1= -15 c2= -15; s is slip ratio defined as (2)

$$s = \begin{cases} 1 - \dfrac{r\omega}{v}, & \text{if } v > r\omega \text{ and } v \neq 0 \quad \text{(braking)} \\ \dfrac{v - r\omega}{r\omega}, & \text{if } v < r\omega \text{ and } \omega \neq 0 \quad \text{(driving)} \end{cases}$$

The coefficient of friction is between the back wheel and the ground, since the DC motor is connected to the back wheel. The slow-speeding wheel determines the forward velocity of the entire Car. Most of the time, it would be the front wheel, and hence $\frac{dv}{dt} = \frac{F_t}{m}$ is also used to determine the angular velocity of the front wheel.

2. Model Predictive Control

In this section, the theory behind Model Predictive Control (MPC) is explained. MPC is primarily used in process control within the chemical industry, where it also originated. MPC's are also used in the autonomous vehicle (1). MPC comes under optimal control, whose goal is to either find the optimal set of inputs, inorder to meet the desired set point, or it is used to find the desired set point itself, when trying to maximize certain throughput of the system. In this study, we will be using MPC to try meet the desired setpoint of the DC car. For Set point tracking, MPC tries to predict future inputs, based from the current time step. The number of future steps calculated for the inputs is called future horizon. The current problem involves non linear dynamical system, and hence the theory is shown for the Non linear MPC.

Consider the dynamical system,

$$\frac{dx}{dt} = f(x,t,u) \tag{7}$$

$$y_{observed} = c(x) \tag{8}$$

$$\tag{9}$$

subjected to constraints (10)

$$k(\mathbf{u}) \leq 0 \tag{11}$$

$$\text{keq}(\mathbf{u}) = 0 \tag{12}$$

$$A\mathbf{u} \leq \mathbf{b} \tag{13}$$

$$A_{\text{eq}}\mathbf{u} = \mathbf{b}_{\text{eq}} \tag{14}$$

$$\mathbf{l}_b \leq \mathbf{u} \leq \mathbf{u}_b \tag{15}$$

Here,

- $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector,
- $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input,
- $f(\cdot)$ represents the nonlinear system dynamics,
- $c(\cdot)$ function mapping states and observable states.

At each time step, only the first part of the optimal control trajectory $\mathbf{u}^*(t)$ is applied to the system (a principle known as receding horizon control), and the optimization is solved again at the next step using updated state measurements.

Nonlinear Model Predictive Control (NMPC) is particularly effective for systems with strong nonlinearities, hard constraints, and multi-variable interactions. However, due to the need to solve nonlinear optimization problems in real time, it typically requires substantial computational resources or customized numerical solvers. Since MPC operates on a discrete-time framework, the given nonlinear dynamical system must be discretized to establish a mapping between the current and next time steps. This discretization can be performed using various ODE integration schemes. In this study, ODE45, a variant of the fourth-order Runge-Kutta method, is used to discretize the dynamical system. The resulting discretized form of the system is expressed as follows:

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2\right) \\
k_4 &= f(t_n + h, x_n + hk_3) \\
x_{n+1} &= x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Let $\mathbf{y}_{\text{desire}} \in \mathbb{R}^p$, where $p \leq n$, be the desired setpoints of the dynamical system. Then, the nonlinear optimization minimizes the following cost function.

$$
\min_{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}} \quad \sum_{i=1}^{h} dt \cdot (\mathbf{y}_{D,i} - \mathbf{y}_i)^T Q_i (\mathbf{x}_{D,i} - \mathbf{x}_i) \tag{16}
$$

Here, Q is a weight matrix of size $(ph) \times (ph)$, where p is the dimension of the observed state and h is the prediction horizon. In this problem, Q is set as $\exp(i)$, where i ranges from 1 to h.

Once the discretized mapping between time steps n and $n+1$ is obtained, the resulting equations are used within a nonlinear optimization algorithm to determine the optimal set of control inputs.

At each time step, a nonlinear optimization problem is solved to find the optimal control inputs over a future prediction horizon h. This optimization is typically carried out using well-established nonlinear optimization methods, such as the active set method or the interior-point method.

A major advantage of MPC is its inherent ability to handle multi-input multi-output (MIMO) systems, which is essential when dealing with real-world systems. In contrast, traditional controllers like the Proportional-Integral are only suitable for Single Input Single Output system (SISO).

3. Propotional-Integral Controller

Since the problem in the present study involves a single output (although it could be extended to a multi-output system), a comparison is made between the MPC and the PI controller. The PI controller is known to be effective and efficient for Single Input-Single Output (SISO) systems, making it a suitable candidate for this study. A conclusion will be drawn based on the comparative performance of the two controllers.

One of the simplest and most effective controllers is the PI (Proportional-Integral) controller, which consists of two components: a proportional component and an integral component. The equation for the PI controller is given by:

$$
u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau
$$

$$
\begin{aligned}
u(t) &: \text{Control input applied to the system} \\
e(t) &: \text{Error signal, } e(t) = r(t) - y(t) \\
K_p &: \text{Proportional gain (response to current error)} \\
K_i &: \text{Integral gain (response to accumulated error)} \\
\int_0^t e(\tau)\,\mathrm{d}\tau &: \text{Integral of the error from time 0 to } t
\end{aligned}
$$

The major advantage of adding the integral term is that it ensures the steady state error is minimized, since only implementing proportional block will not have a zero steady state error. The output of the dynamical system is fed back into the PI controller, which compares the signals between the output and the setpoint and the desired input applied to the control system to minimize the error. The value of K_p and K_p can be precomputed based on the pole placement technique. However, in this study these values are manually tuned to achieve the desired output. For the PI-controlled dynamical system, a MATLAB Simulink-based model is implemented rather than hard-coding the controller equations.

4. MATLAB Simulink model for PI Controller

In this section, a MATLAB Simulink model is shown that is used to calculate the desired wheel speed and minimize the slip using the PI controller. The input to the system is the voltage V that is governed by the PI controller which takes the error signal as its input value. The error is the difference between the simulated wheel speed and the desired wheel speed. The desired wheel speed is gradually increased using a ramp input, until the final desired speed is achieved.

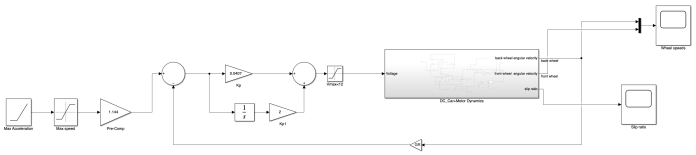

Figure 1: DC Motor and vehicle dynamics associated with the PID system

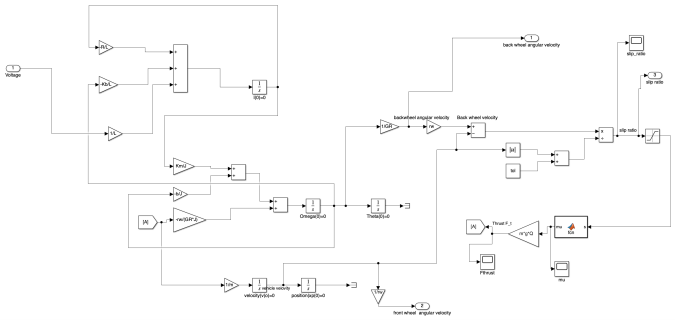

Figure 2: Simulink Model for DC Car and Motor

5. Results

In this section, the results from both the controllers are shown, and a comparison is drawn between them.

5.1. Performance of the DC Motor based car without the controllers

In this subsection, the equations 1- 5 are solved with constant Voltage, without the presence of the feedback based controller. A constant 12V DC power supply is provided to the input. Since, there are no controller, we can see that the slip between the back wheel and the front wheel is large for most of the time, until the car reaches the steady state speed. The car reaches steady state at about 12 seconds, after which the velocity difference between the back wheel and the front wheel is zero. This naive car model without the controller does not take into the account of slip. Also, at 12 V the maximum speed achieved by the wheel is about 800 rad/s which may not be the desired speed needed. We can reduce this by lowering the input voltage, however, this has to be manually implemented and will be laborious when dealing with large scale car model. Therefore, in the next subsections, the results from the two most prominent controllers, PI and MPC are shown and discussed

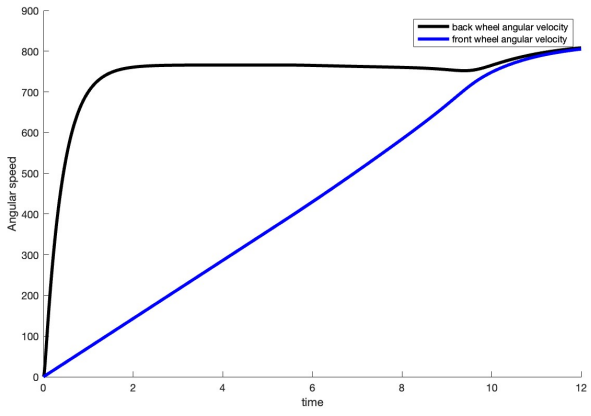

Figure 3: Front and back wheel speeds without feedback controller

5.2. Performance of the DC Motor based car for PI Controller

In this subsection, a PI controller, discussed in section 3 is used to control the speed of the wheels, which also ensures that the slip of the wheels are minimized. The desired speed of the wheel is considered to be 110 rad/s which is achieved by uniformly increasing the velocity (i.e constant acceleration of the car) until it reaches the desired speed. Figure **??** shows the results, obtained from the inclusion of the PI controller. There is a minimal slip between the wheels throughout the time period, and the desire speed is reached, the steady state speed is reached at about 6 seconds and at the steady state, the speed achieved by the car is equal to the maximum desired speed. So, far the PI controller achieved a minimal slip between the wheels.

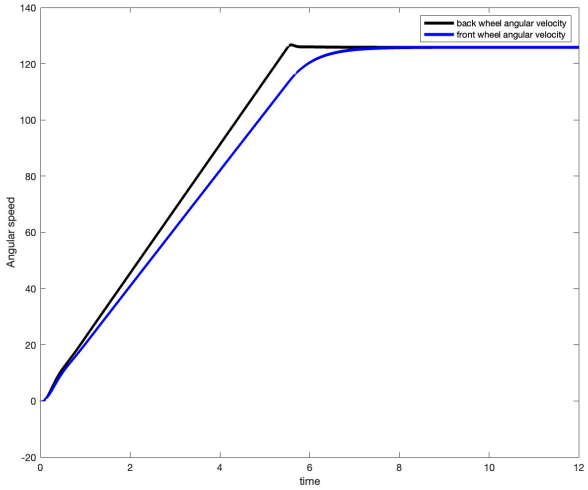

Figure 4: Front and back wheel speeds with PI feedback controller

5.3. Performance of the DC Motor based car for MPC Controller

In this subsection, MPC based controller is applied to the DC motor driven car. MPC, as opposed to PI controller, is time consuming, since it solves an optimization problem at each time step. Hence, it is important to use the MPC only when necessary. However, MPC is discussed here to evaluate only its performance without looking into the computational cost. The problem set-up is similar to the PI controller, with constant increase in the velocity until the desired velocity of the wheels is reached, which is 110 rad/s. Figure 5 shows the wheel speeds of front and back wheel as a function of time. Initially in the beginning the velocity of the back wheel increases at an enororrmous rate and quickly reaches the velocity of the front wheel within 0.2 seconds. When compared to PI controller, MPC has even lower slip between the front and back wheel (barring of the initial 0.5 seconds) through out the time. However, as seen from the graph, there is no steady increase in the speed and the speeds are jagged, which will lead to other problems, causing wear and tear of mechanical components. Nonetheless, the slip between the wheels are minimal in the case of MPC.

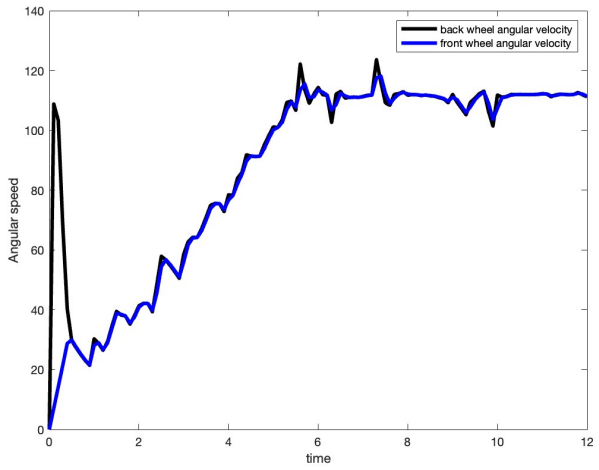

Figure 5: Front and back wheel speeds with PI feedback controller

6. Conclusion

In this study, we compared the performance of the two feedback controllers; a PI controller and an MPC controller. Both the controllers outperformed the naive DC Car implementation, without any feedback controller, which suffered from high slip for most of the time. The implementation of PI controller ensured that the slip is minimized however, the front and back wheel speed were not equal for most of the time before it reached the steady desired speed. On the other hand MPC suffered from high slip for 0.5 seconds before equalling the speed with the front wheel. The slip between the wheels was the lowest in MPC when compared to that with PI controller. Although, MPC is effective when dealing with MIMO systems, PI controller is sufficient when controlling a SISO system. The MATLAB code can be found in the gihub link **??**

References

[1] Shuyou Yu, Matthias Hirche, Yanjun Huang, Hong Chen, and Frank Allgöwer. Model predictive control for autonomous ground vehicles: A review. *Autonomous Intelligent Systems*, 1:1–17, 2021.

[2] Carlos Canudas-de-Wit, Panagiotis Tsiotras, Evagelos Velenis, Michel Basset, and Gilles Gissinger. Dynamic friction models for road/tire longitudinal interaction. *Vehicle System Dynamics*, 39(3):189–226, 2003. `https://doi.org/10.1076/vesd.39.3.189.14152`.