University of Padova

Department of Information Engineering

Master Thesis in Control Systems Engineering

Iterative Learning Control analysis for automotive testbed applications

Supervisor

Prof. Alberto Dalla Libera

University of Padova

Master Candidate

Marco Mustacchi

Student ID

2054137

Academic Year 2023-2024

11th April 2024

AI MIEI GENITORI,
PER ESSERCI SEMPRE STATI
IN TUTTI I MIEI MOMENTI
DI DIFFICOLTÀ

Abstract

Iterative Learning Control (ILC) represents an innovative paradigm in the control of dynamic systems, aimed at progressively improving system performance through iterative learning from repeatable reference tracks. This thesis aims to explore and analyse different ILC approaches to achieve accelerometer reference tracking. The research work first focuses on the dynamic modelling of the Quarter Car model system and the more generic Full Car model. Starting from the differential equations, the State Space model is then derived to facilitate the implementation in the Simulink environment. Then, road profiles will be generated to be used as input to obtain the accelerometer references. These profiles will be generated according to the ISO 8608 classification in order to get data as consistent as possible with the real ones. After introducing the iterative algorithm, the results obtained in simulation using different approaches will be shown and discussed, both in the SISO case using the Quarter Car model and in the MIMO case using the Full Car model. Furthermore, after exploring and evaluating some strategies to increase the robustness of the ILC, the algorithm will be tested in the presence of repetitive and non-repetitive disturbances and compared with more traditional controllers. Finally, some tests of the algorithm in a real environment will be carried out, making a comparison with those obtained previously with simulations. In conclusion, this thesis contributes to a better understanding and implementation of the Iterative Learning Control algorithm by analysing its theoretical foundations and the relative results obtained through simulations and real experiments.

Sommario

L'Iterative Learning Control (ILC) rappresenta un paradigma innovativo nel controllo dei sistemi dinamici, volto a migliorare progressivamente le prestazioni del sistema utilizzando un apprendimento iterativo da tracce di riferimento ripetibili. Questa tesi si propone di esplorare e analizzare diversi approcci ILC per ottenere il tracciamento del riferimento accelerometrico. Il lavoro di ricerca si concentra innanzitutto sulla modellazione dinamica del sistema del modello Quarter Car e del più generico modello Full Car. Partendo dalle equazioni differenziali, viene poi derivato il modello dello spazio di stato per facilitare l'implementazione nell'ambiente Simulink. Successivamente, verranno generati i profili stradali da utilizzare come input per ottenere i riferimenti accelerometrici. Questi profili saranno generati secondo la classificazione ISO 8608, in modo da ottenere dati il più possibile coerenti con quelli reali. Dopo aver introdotto l'algoritmo iterativo, verranno mostrati e discussi i risultati ottenuti in simulazione utilizzando diversi approcci, sia nel caso SISO utilizzando il modello Quarter Car che nel caso MIMO utilizzando il modello Full Car. Inoltre, dopo aver esplorato e valutato alcune strategie per aumentare la robustezza dell'ILC, l'algoritmo verra' testato in presenza di disturbi ripetitivi e non ripetitivi e comparato con controllori piu' tradizionali. Infine, verranno effettuati alcuni test dell'algoritmo in un ambiente reale, facendo un confronto con quelli ottenuti in precedenza con le simulazioni. In conclusione, questa tesi contribuisce a una migliore comprensione e implementazione dell'algoritmo Iterative Learning Control, analizzando le sue basi teoriche e i relativi risultati ottenuti attraverso simulazioni ed esperimenti reali.

Contents

Listing of figures

Listing of tables

1 Introduction

The idea for this thesis was born from the need of the STEP Lab company to reconstruct road profiles from the measurement of accelerometer profiles. In fact, direct road profile measurement is generally a difficult and very expensive action, that requires specific equipment. In contrast, the measurement of acceleration profiles is a much easier data to obtain, as only an accelerometer mounted on each corner of the vehicle chassis is needed. The problem, well known in the literature as road profile inversion from in-vehicle accelerometers, has not yet found a definitive solution.

One of the first approaches chosen was the use of system identification techniques, particularly with a black box approach. However, following difficulties that arose mainly due to insufficient data and multivariable model, it was chosen to abandon this approach.

The project then evolved with the aim of reproducing in laboratories specialized in testing for automotive applications, accelerometer signals measured on the road. These tests make it possible to study the behavior of the vehicle on different types of road profiles so as to improve the driving experience, whether aimed at passenger comfort (in the case of city use of the vehicle) or performance (in the case of racing use). In fact, the vehicle's suspension calibration is a crucial aspect to ensure safe and comfortable traveling.

The focus of this thesis therefore shifted to a tracking problem with high-frequency references. One of the major problems in control for tracking a reference signal is to ensure good transient behavior. However, when the latter turns out to be repetitive, it is possible to exploit the repetitiveness to improve tracking performance.

Particularly in the automotive field, the road profile can be considered as a repetitive signal over time, especially on specific types of roads or under certain conditions. For instance, certain road sections may have consistent patterns of bumps and undulations that repeat at regular intervals.

One of the most widely used algorithms for controlling repetitive signals is Iterative Learning Control. Iterative Learning Control (ILC) algorithm is a control technique that focuses on iterative refinement of the performance of a dynamic system, particularly suitable for tracking repetitive signals. This control method exploits the repetitiveness of the signal to track to incorporate past control information, such as tracking errors and control input signals, into the construction of the next control action. In this way during each iteration, the algorithm learns from past errors and gets closer and closer to the desired reference signal. Finally, the effectiveness of iterative control in tracking the reference can be evaluated by error measures, such as the root mean square (RMS) between the desired reference and the system response.

Therefore, this thesis aims to study the application of the Iterative Learning Control algorithm for tracking a reference signal obtained from a repetitive road profile signal applied to a vehicle.

Once the objectives of the project were defined, we wanted to proceed in a stepwise manner. In fact, it started with the study of the system model, starting from the simplest case single input single output to the most complex case multiple input multiple output. Then a research was made for methods and techniques in order to generate road profiles that were as similar as possible with reality. Having defined the necessary foundations, we went into detail in the study of the Iterative Learning Control algorithm. Finally, after an in-depth analysis of the theory, different implementation approaches were compared with the goal of figuring out which was the most suitable for our situation. Finally, once the possibility of implementation in a real environment was verified in simulation, the algorithm was tested in the laboratories of the STEP Lab company. All simulations were performed in the MATLAB & Simulink environment.

1.0.1 Structure of the thesis

The thesis is organized as follows:

Chapter 2 introduces the main types of suspensions normally found in a vehicle. Then the quarter car and full car model for a vehicle is modeled by differential equations and then by state space representation. Finally, the two models are tested in simulation using a bump road

profile as input.

Chapter 3 focuses on generating a road profile that conforms as closely as possible with reality. After introducing the main standard for road profile classification, a method for obtaining a road profile that conforms with that standard is presented. Finally, simulation results using the road profile signal generated on the two models in the previous chapter are shown.

Chapter 4 introduces the Iterative Learning Control algorithm from a theoretical point of view, illustrating the main assumptions necessary for its proper operation. Then the representation of the algorithm in the time domain using the so-called lifted-domain representation is presented. Finally, the two main theorems of the algorithm, the stability theorem and the monotonic convergence theorem, are shown and proved.

Chapter 5 analyses various algorithm design techniques, in particular some heuristic approaches and a model-based approach. These approaches are also implemented in simulation and the corresponding results are presented and discussed. The robustness of the algorithm is then analyzed, particularly in the presence of repetitive disturbances and non-repetitive disturbances. Finally, a comparison of the Iterative Learning Control algorithm with more classical control techniques is made.

Chapter 6 shows the experimentation of the algorithm using the model-based inversion approach on a real system, consisting of a suspension on which a force is exerted via a current-controlled electrodynamic actuator. The results are finally illustrated and discussed.

Chapter 7 presents the conclusions achieved and suggests some possible future developments.

2 Mathematical Model

The goal of this chapter is to obtain the linear dynamical model of a vehicle system in the form of differential equations using Newton's laws. After introducing the main types of suspensions found in a vehicle and defining the differences, we will focus initially on the one-suspension model, called the quarter car model consisting of 2 DOFs and then the so-called full car model equipped with 7 DOFs, considering in both cases the passive type of suspensions. Finally, after obtaining the state space model for each of the found models, they will be implemented in MATLAB and Simulink environment and a simulation for each model using a bump road profile will be shown.

2.1 Suspension

The Quarter Car model is the simplest mathematical model used to describe the dynamics of a vehicle. The model consists of two masses, the sprung mass m_c which represents $1/4$ of the body of the vehicle, and the unsprung mass M_v which, together with a spring with an elastic constant k_t, represents one wheel of the vehicle. Finally between the sprung mass and the unsprung mass is a suspension, which can be classified according to [1] [2] into three different types represented in figure 2.1:

- passive suspension
- semi-active suspension

- active suspension

2.1.1 Passive suspension

Passive suspensions are the most widely used in the automotive industry, as they are reliable, simple and inexpensive. It includes a spring and a viscous damper. The damper consists of a piston that moves within a hydraulic oil or compressed gas In this case, the characteristics of the two components, i.e. the value of the elastic constant for the spring and the value of the damping coefficient for the shock absorber, are fixed.

The main disadvantage of this type is that the performance is effective only in a limited range of frequencies.

2.1.2 Semi-active suspension

Semi-active suspensions, first proposed around 1970s, include a passive spring and an element variable damper, in which it is possible to control the damping coefficient. They thus consist of an active apparatus, i.e. they must also include an actuator that provides the active force required for operation based on the algorithm of control used.

The advantage of this type, besides operating in a wider range of frequencies than the previous ones, is that in case of failure of the control system for the damper controllable, the semi-active suspension can continue to operate independently in the passive condition.

2.1.3 Active suspension

In the case of active suspension, proposed around the 1990s, it is possible to adjust both the spring and of the damper. This allows not only to dissipate energy as with the type of passive suspensions, but also to add/dissipate energy from the system. Through an actuator, it is possible to control in real time the system based on the information provided by various sensors such as accelerometers, gyroscopes and inertial platforms (IMUs), which are then processed by an internal control unit.

With respect to the passive and semi-active suspension types, active suspensions allow operation in a spectrum of even greater frequencies. In addition, the ability to control not only damping, but also spring elasticity allows real-time management of both damping (as in the case of semi-active suspensions) but also of the vehicle attitude change.

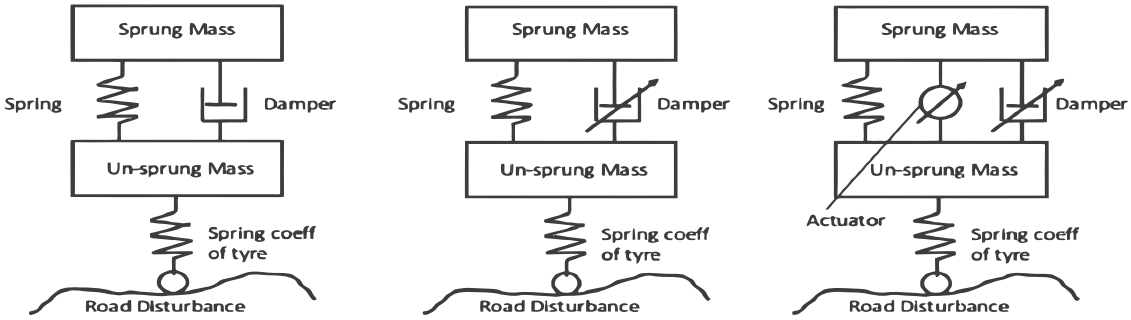

Figure 2.1: Mathematical representation of the three type of suspension systems. From the left: passive suspension, semi-active suspension, active suspension

However because of the high energy required for operation, the significant weight and cost, this type of suspension remains little used to this day. In fact, they are currently fitted only on high end, high-performance vehicles.

2.2 Quarter Car Model

In this section the so-called quarter car model with passive type suspension, as represented in the figure 2.2, will be analyzed. It is composed of the tyre (modelled as a spring with spring constant k_t), the unsprung mass M_v, the passive suspension (modelled as a spring with spring constant k_s and a damper with damping coefficient c_s) and the sprung mass m_c.

2.2.1 Equations of motion

At this point we want to model the dynamical system, and to do this we need to formulate a set of differential equations that describe how the system evolves over time, specifically how the next state is a function of the current state.

In general there are several methods of obtaining the equations of motion of a dynamical system in the form of differential equations:

- Newtonian
- Lagrangian
- Hamiltonian

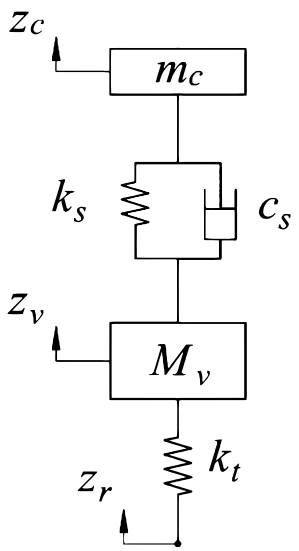

Figure 2.2: Quarter car model

While the Newtonian constructs the forces of motion based on forces and accelerations acting on the system, the Lagrangian and Hamiltonian are based on potential and kinetic energy. To obtain the dynamics of the system, it was chosen to use the Newtonian, which derives the equations of motion using the Newtonian law:

$$m\ddot{x} = \Sigma F \tag{2.1}$$

where m is each mass in the system, $\ddot{x}$ is the acceleration of that mass, and ΣF is the sum of the forces acting on that mass.

Accordingly, we analyze all the forces acting on our system in the form of free body diagrams. Since the quarter car model is a 2 DOF system, it will be necessary to find two equations of motion that completely describe the evolution of the system.

In particular from the laws of physics we recall that :

$$F_{\text{spring}} = -k\Delta x \tag{2.2}$$

where k represents the spring's constant factor (i.e., its stiffness) and Δx is the displacement

or change in length, for the elastic force of a spring using the Hooke's law, and

$$F_{\text{damper}} = -c\dot{x} \tag{2.3}$$

where c is the damping constant and $\dot{x}$ is the velocity, to obtain the force exerted by the damper.

At this point, considering each individual mass in our system and using the relation eq. (2.1) we obtain:

$$\begin{aligned} m_c\ddot{z}_c &= +F_{\text{spring suspension}} + F_{\text{damper suspension}} \\ &= -k_s(z_c - z_v) - c_s(\dot{z}_c - \dot{z}_v) \end{aligned} \tag{2.4}$$

considering the sprung mass m_c, and

$$\begin{aligned} M_v\ddot{z}_v &= -F_{\text{spring suspension}} - F_{\text{damper suspension}} + F_{\text{spring tyre}} \\ &= -k_s(z_v - z_c) - c_s(\dot{z}_v - \dot{z}_c) - k_s(z_v - z_r) \end{aligned} \tag{2.5}$$

considering the unsprung mass M_v. The two relations that completely describe the dynamics of the system are then:

$$\begin{aligned} M_v\ddot{z}_v + c_s\left(\dot{z}_v - \dot{z}_c\right) + k_t(z_v - z_r) + k_s\left(z_v - z_c\right) = 0 \\ m_c\ddot{z}_c + c_s\left(\dot{z}_c - \dot{z}_v\right) + k_s\left(z_c - z_v\right) = 0 \end{aligned} \tag{2.6}$$

which we can finally rewrite as:

$$\begin{aligned} \ddot{z}_v &= \frac{-c_s\left(\dot{z}_v - \dot{z}_c\right) - k_t(z_v - z_r) - k_s\left(z_v - z_c\right)}{M_v} \\ \ddot{z}_c &= \frac{-k_s(z_c - z_v) - c_s(\dot{z}_c - \dot{z}_v)}{m_c} \end{aligned} \tag{2.7}$$

2.2.2 Simulink implementation

Once we get the dynamics of the system, we can model it using blocks in Simulink. A possible implementation by block diagram of the ODE found is shown in figure 2.3.

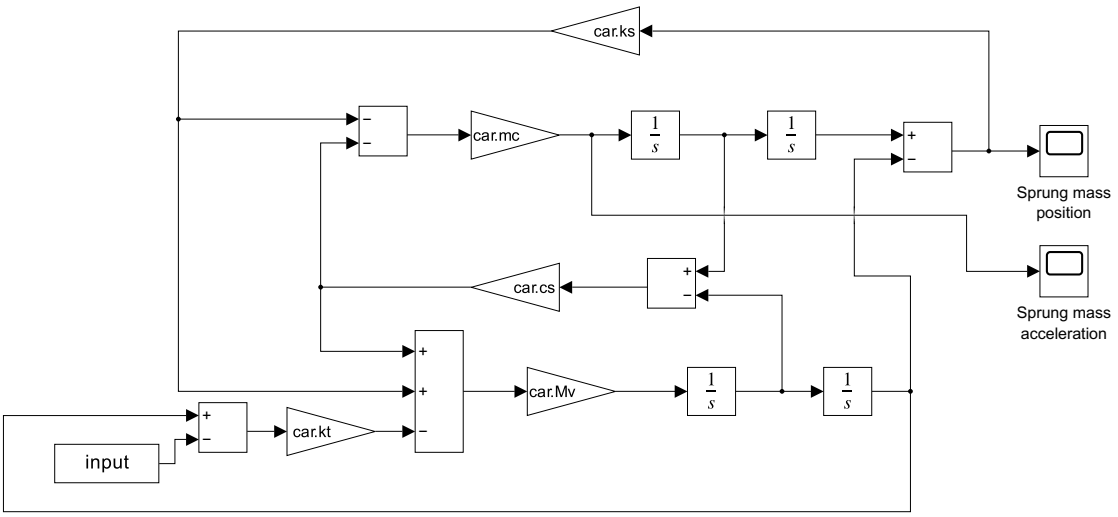

Figure 2.3: Simulink implementation of the quarter car model

2.2.3 State Space representation

A useful representation of a linear system is the state space model, where the relationship between input, system variables and output is given by first-order differential equations.

Considering only the first derivative of each state, we obtain a system of first-order differential equations, which we can package into a matrix form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t) \end{aligned} \tag{2.8}$$

The first equation is called state equation, where $\dot{\mathbf{x}}$ is the state vector, $\mathbf{u}$ is the input vector, the $\mathbf{A}$ matrix describes how the internal states connect to each other the underlying dynamics of the system and the $\mathbf{B}$ matrix describes how the inputs enter into the system and which states are they affecting. Now considering the second equation, which is called the output equation, $\mathbf{y}$ is a vector of the outputs of the system or the part that you're interested in knowing, the $\mathbf{C}$ matrix describes how the states are combined to get the outputs and the $\mathbf{D}$ matrix is used to allow the inputs to bypass the system altogether and feed forward to the output. One important thing consider is that the output are not necessarily the state variables.

One advantage of this representation is the fact that there are many different techniques that are based on the state space representation, including Kalman filter, LQR or MPC control.

It is necessary, however, to define the set of state variables, that is, the minimum set of variables that fully describes the system. We saw earlier that in this case we have a 2 DOF system, so it is fairly straightforward to choose four state variables, two positions and two velocities. We then choose as state variables the position and velocity of the sprung mass and the position and velocity of the unsprung mass. The input of the model is the wheel disturbance z_r.

$$\mathbf{x} = \begin{bmatrix} z_v \\ \dot{z}_v \\ z_c \\ \dot{z}_c \end{bmatrix} \qquad u = z_r$$

Reorganizing the equations of motion found in eq. (2.6), we obtain the first order differential relations of the state variables derivatives.

$$\begin{cases} \ddot{z}_v = \frac{k_s(z_c - z_v) + k_t(z_r - z_v) + c_s(\dot{z}_c - \dot{z}_v)}{M_v} \\ \ddot{z}_c = \frac{k_s(z_v - z_c) + c_s(\dot{z}_v - \dot{z}_c)}{m_c} \end{cases} \tag{2.9}$$

where the first derivative relation is not reported because it is already present as the state of the system.

Finally, the matrices A, B, C and D are obtained by exploiting MATLAB and in particular the Symbolic Math Toolbox:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_s + k_t}{M_v} & -\frac{c_s}{M_v} & \frac{k_s}{M_v} & \frac{c_s}{M_v} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{m_c} & \frac{c_s}{m_c} & -\frac{k_s}{m_c} & -\frac{c_s}{m_c} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{k_t}{M_v} \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{k_s}{m_c} & \frac{c_s}{m_c} & -\frac{k_s}{m_c} & -\frac{c_s}{m_c} \end{bmatrix} \qquad D = \begin{bmatrix} 0 \end{bmatrix}$$

2.2.4 Simulation of Quarter Car model using the bump road profile

As input to test the correctness of the found model, a bump road profile is used.

It is implemented with the formula proposed by the paper [3]:

$$u = \begin{cases} \frac{a}{2}\left(1 - \cos\left(\frac{2\pi V t}{\lambda}\right)\right) & 1 \leq t \leq 1 + \frac{\lambda}{V} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where u is the wheel input disturbance z_r, a is the bump amplitude, λ is the length of the bump, t is the time in seconds and V is the vehicle forward velocity. In our case we define $a = 30$ mm, $V = 10$ m/s and $\lambda = 5$ m in order to get the figure 2.4.

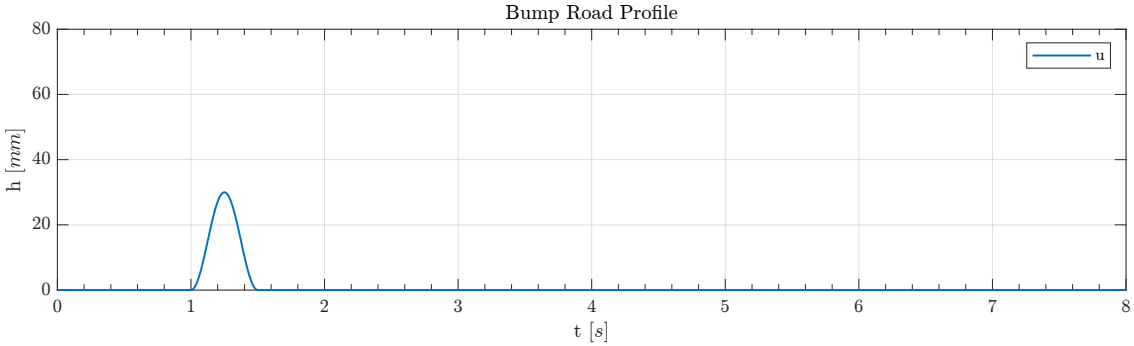

Figure 2.4: Bump road profile for quarter car model

Once we have defined the values of the quarter car model parameters, listed in table B.1 and adapted by [4], it is possible to perform the simulation of the system. The simulation of the system using the newly derived bump road profile as input is obtained using the Simulink model in figure 2.3 with a sample time $Ts = 0.01$ s. The result is shown shown in figure 2.5. Moreover, looking at the result obtained, it can already be guessed that the system is asymptotically stable, since after an initial phase of oscillations, the acceleration settles down to a steady state value of 0.

2.3 Full Car model

The generalization of the quarter car model that includes only one wheel is the so-called full car model, which includes all four wheels and is therefore the most appropriate model for a generic vehicle.

The frontal view of the full car model is represented in figure 2.6, while the whole view is represented in figure 2.7.

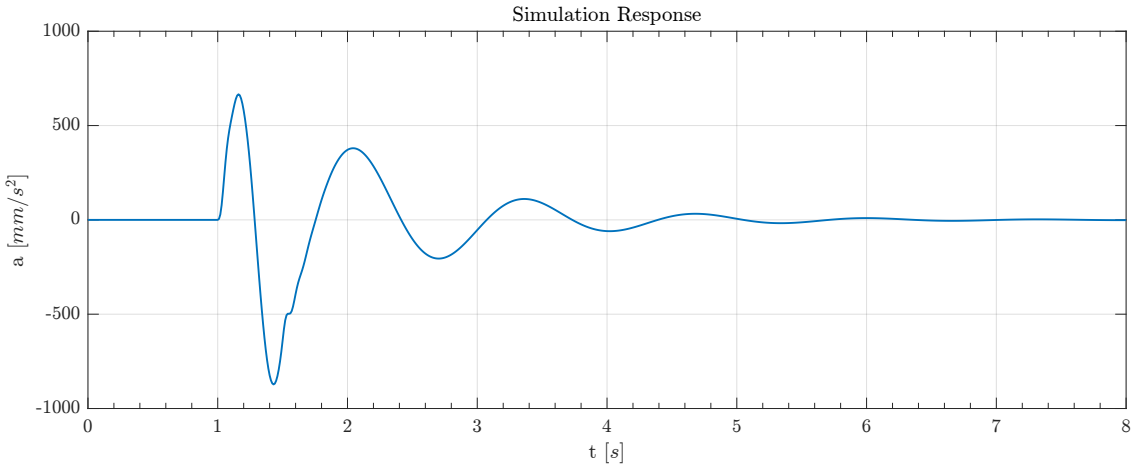

Figure 2.5: Output response of the Quarter Car model using the bump road profile as input

Regarding the motion of the model, z_{v1}, z_{v2}, z_{v3}, and z_{v4} represent the vertical displacement in position of the front and rear unsprung mass on left and right sides, respectively.

For the sprung mass we have z_c that denotes the vertical displacement of the body center of mass; while θ and φ denote the pitch and roll angle displacement.

These 7 parameters represent the 7 DOF of the model.

The outputs of the model are z_{c1}, z_{c2}, z_{c3}, and z_{c4} representing the body vertical displacement at the four corners of the chassis, where the accelerometers are placed.

On the other hand, the inputs of the model are z_{r1}, z_{r2}, z_{r3}, and z_{r4} representing the road irregularity excitation to the front right, rear right, rear left and front left wheel, respectively.

Regarding the parameters of the model, m_c represents the sprung mass; I_x and I_y represent the rotary inertia about the x and y axis, respectively.

M_{v1}, M_{v2}, M_{v3}, and M_{v4} represent the unsprung masses of the front and rear wheels on left and right sides, respectively.

k_{t1}, k_{t2}, k_{t3}, and k_{t4} represent the elastic stiffness of the front and rear tyre on left and right sides, respectively.

k_{s1}, k_{s2}, k_{s3}, and k_{s4} represent the elastic stiffness of the front and rear suspension on left and right sides, respectively; while c_{s1}, c_{s2}, c_{s3}, and c_{s4} represent the damping of the front and rear suspension on left and right sides, respectively.

Lastly l_f and l_r denote the distance of front and rear axle to the body center of mass, while t_r and t_l denote the distance between the left and right wheel to the body center of mass.

Now the system dynamics for the full car suspension model is obtained by adapting it from

the work by Desai et al. in [5].

2.3.1 Equations of motion

Regarding the dynamic differential equation of the vertical motion of the four unsprung mass, we can generalize the result found in the previous section as:

$$M_{v_i}\ddot{z}_{v_i} + k_{t_i}\left(z_{v_i} - z_{r_i}\right) + k_{s_i}\left(z_{v_i} - z_{c_i}\right) + c_{s_i}\left(\dot{z}_{v_i} - \dot{z}_{c_i}\right) = 0 \qquad i = 1, \ldots, 4 \tag{2.11}$$

and hence we find:

$$\begin{aligned}
&M_{v1}\ddot{z}_{v1} + k_{t1}\left(z_{v1} - z_{r1}\right) + k_{s1}\left(z_{v1} - z_{c1}\right) + c_{s1}\left(\dot{z}_{v1} - \dot{z}_{c1}\right) = 0\\
&M_{v2}\ddot{z}_{v2} + k_{t2}\left(z_{v2} - z_{r2}\right) + k_{s2}\left(z_{v2} - z_{c2}\right) + c_{s2}\left(\dot{z}_{v2} - \dot{z}_{c2}\right) = 0\\
&M_{v3}\ddot{z}_{v3} + k_{t3}\left(z_{v3} - z_{r3}\right) + k_{s3}\left(z_{v3} - z_{c3}\right) + c_{s3}\left(\dot{z}_{v3} - \dot{z}_{c3}\right) = 0\\
&M_{v4}\ddot{z}_{v4} + k_{t4}\left(z_{v4} - z_{r4}\right) + k_{s4}\left(z_{v4} - z_{c4}\right) + c_{s4}\left(\dot{z}_{v4} - \dot{z}_{c4}\right) = 0
\end{aligned} \tag{2.12}$$

The dynamic differential equation of the chassis's vertical motion is listed as follows:

$$m_c\ddot{z}_c + F_1 + F_2 + F_3 + F_4 = 0 \tag{2.13}$$

Considering the rotational dynamics of the vehicle, it is possible to use the newton's law for the rotation motion:

$$I\ddot{\alpha} = \Sigma M$$

In this way, the dynamic differential equation for the pitch motion of the body is as follows:

$$I_{cx}\ddot{\varphi} - F_1 t_r - F_2 t_r + F_3 t_l + F_4 t_l = 0 \tag{2.14}$$

and the the dynamic differential equation for the roll motion of the body is:

$$I_{cy}\ddot{\theta} - F_1 l_f + F_2 l_r + F_3 l_r - F_4 l_f = 0 \tag{2.15}$$

where

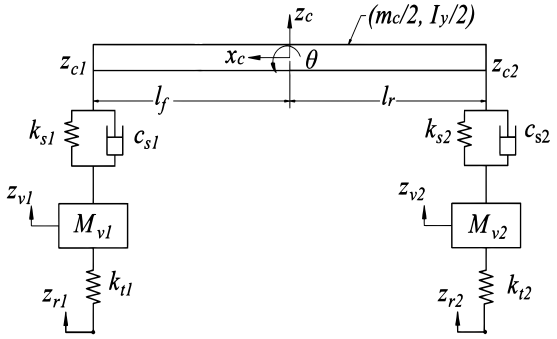

Figure 2.6: Full car model frontal view (half car model)

$$
\begin{aligned}
F_1 &= \left(k_{s1}\left(z_{c1} - z_{v1}\right) + c_{s1}\left(\dot{z}_{c1} - \dot{z}_{v1}\right)\right) \\
F_2 &= \left(k_{s2}\left(z_{c2} - z_{v2}\right) + c_{s2}\left(\dot{z}_{c2} - \dot{z}_{v2}\right)\right) \\
F_3 &= \left(k_{s3}\left(z_{c3} - z_{v3}\right) + c_{s3}\left(\dot{z}_{c3} - \dot{z}_{v3}\right)\right) \\
F_4 &= \left(k_{s4}\left(z_{c4} - z_{v4}\right) + c_{s4}\left(\dot{z}_{c4} - \dot{z}_{v4}\right)\right)
\end{aligned}
$$

By finally regrouping the equations eq. (2.12), eq. (2.13), eq. (2.14), eq. (2.15), we obtain the seven relations that completely describe the system dynamics:

$$
\begin{cases}
M_{v1}\ddot{z}_{v1} + k_{t1}\left(z_{v1} - z_{r1}\right) + k_{s1}\left(z_{v1} - z_{c1}\right) + c_{s1}\left(\dot{z}_{v1} - \dot{z}_{c1}\right) = 0 \\
M_{v2}\ddot{z}_{v2} + k_{t2}\left(z_{v2} - z_{r2}\right) + k_{s2}\left(z_{v2} - z_{c2}\right) + c_{s2}\left(\dot{z}_{v2} - \dot{z}_{c2}\right) = 0 \\
M_{v3}\ddot{z}_{v3} + k_{t3}\left(z_{v3} - z_{r3}\right) + k_{s3}\left(z_{v3} - z_{c3}\right) + c_{s3}\left(\dot{z}_{v3} - \dot{z}_{c3}\right) = 0 \\
M_{v4}\ddot{z}_{v4} + k_{t4}\left(z_{v4} - z_{r4}\right) + k_{s4}\left(z_{v4} - z_{c4}\right) + c_{s4}\left(\dot{z}_{v4} - \dot{z}_{c4}\right) = 0 \\
m_c\ddot{z}_c + F_1 + F_2 + F_3 + F_4 = 0 \\
I_{cx}\ddot{\varphi} - F_1 t_r - F_2 t_r + F_3 t_l + F_4 t_l = 0 \\
I_{cy}\ddot{\theta} - F_1 l_f + F_2 l_r + F_3 l_r - F_4 l_f = 0
\end{cases}
\tag{2.16}
$$

Positions and Velocities relations

It is further possible to derive the relations to obtain the coupling of the system:

$$
z_{c1} = z_c - l_f \sin\theta - t_r \sin\varphi
$$

To keep the system so far found linear, it is possible to approximate the previous relations for small angle values:

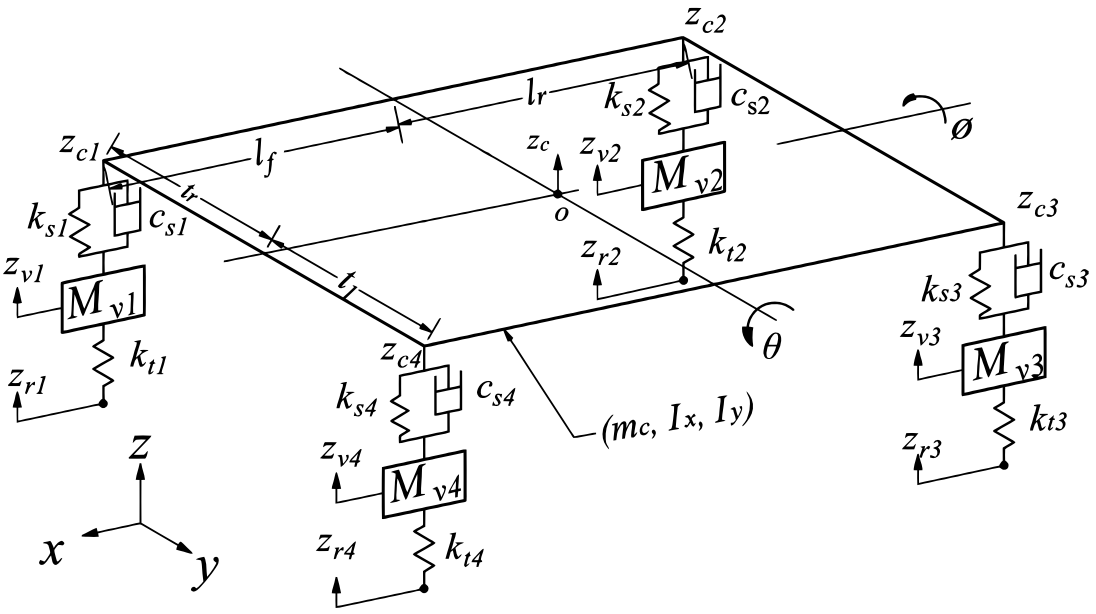

Figure 2.7: Full car model

$$\sin\alpha \approx \alpha$$

Hence, we get:

$$z_{c1} = z_c - l_f\theta - t_r\varphi$$

Finally, considering all the components, the relations are:

$$\begin{aligned} z_{c1} &= z_c - l_f\theta - t_r\varphi & \dot{z}_{c1} &= \dot{z}_c - l_f\dot{\theta} - t_r\dot{\varphi} \\ z_{c2} &= z_c + l_r\theta - t_r\varphi & \dot{z}_{c2} &= \dot{z}_c + l_r\dot{\theta} - t_r\dot{\varphi} \\ z_{c3} &= z_c + l_r\theta + t_l\varphi & \dot{z}_{c3} &= \dot{z}_c + l_r\dot{\theta} + t_l\dot{\varphi} \\ z_{c4} &= z_c - l_f\theta + t_l\varphi & \dot{z}_{c4} &= \dot{z}_c - l_f\dot{\theta} + t_l\dot{\varphi} \end{aligned} \tag{2.17}$$

2.3.2 State Space representation

At this point we want to obtain a representation of the system by State Space equation as done in the previous section.

It, as already seen, it requires the definition of the state variables of the system, and in this case, the four tire positions z_{v1}, z_{v2}, z_{v3}, z_{v4} and the four tire velocities $\dot{z}_{v1}$, $\dot{z}_{v2}$, $\dot{z}_{v3}$, $\dot{z}_{v4}$, the body positions z_c, body velocities $\dot{z}_c$, roll angle φ, roll angular velocity $\dot{\varphi}$, pitch angle θ and pitch angular velocity $\dot{\theta}$, would be a natural choice.

However, in this case, because the outputs of the model are the four body vertical displacements at the four corners of the chassis z_{c1}, z_{c2}, z_{c3}, z_{c4} where the accelerometers are placed, it was chosen to use the relative positions and velocities of the latter instead of the roll, pitch and body COM motion:

$$\mathbf{x} = \begin{bmatrix} z_{v1} & \dot{z}_{v1} & z_{v2} & \dot{z}_{v2} & z_{v3} & \dot{z}_{v3} & z_{v4} & \dot{z}_{v4} & z_{c1} & \dot{z}_{c1} & z_{c2} & \dot{z}_{c2} & z_{c3} & \dot{z}_{c3} & z_{c4} & \dot{z}_{c4} \end{bmatrix}^T$$

while the system inputs are road excitations to each of the four wheels:

$$\mathbf{u} = \begin{bmatrix} z_{r1} & z_{r2} & z_{r3} & z_{r4} \end{bmatrix}^T$$

At this point it is necessary to substitute to obtain the differential equations with respect to the outputs $\ddot{z}_{c1}$, $\ddot{z}_{c2}$, $\ddot{z}_{c3}$, $\ddot{z}_{c4}$ of the system to obtain the model state space.

Substituting the relations found for position, velocity and accelerations, we obtain the first derivative relations:

$$\begin{cases} \ddot{z}_{v1} = \frac{k_{s1}(z_{c1}-z_{v1})+k_{t1}(z_{r1}-z_{v1})+c_{s1}(\dot{z}_{c1}-\dot{z}_{v1})}{M_{v1}} \\ \ddot{z}_{v2} = \frac{k_{s2}(z_{c2}-z_{v2})+k_{t2}(z_{r2}-z_{v2})+c_{s2}(\dot{z}_{c2}-\dot{z}_{v2})}{M_{v2}} \\ \ddot{z}_{v3} = \frac{k_{s3}(z_{c3}-z_{v3})+k_{t3}(z_{r3}-z_{v3})+c_{s3}(\dot{z}_{c3}-\dot{z}_{v3})}{M_{v3}} \\ \ddot{z}_{v4} = \frac{k_{s4}(z_{c4}-z_{v4})+k_{t4}(z_{r4}-z_{v4})+c_{s4}(\dot{z}_{c4}-\dot{z}_{v4})}{M_{v4}} \\ \ddot{z}_{c1} = \frac{F(k_{s1}(z_{v1}-z_{c1})+c_{s1}(\dot{z}_{v1}-\dot{z}_{c1}))+G(k_{s2}(z_{v2}-z_{c2})+c_{s2}(\dot{z}_{v2}-\dot{z}_{c2}))}{D} \\ \qquad\qquad \frac{-H(k_{s3}(z_{v3}-z_{c3})+c_{s3}(\dot{z}_{v3}-\dot{z}_{c3}))+L(k_{s4}(z_{v4}-z_{c4})+c_{s4}(\dot{z}_{v4}-\dot{z}_{c4}))}{D} \\ \ddot{z}_{c2} = \frac{G(k_{s1}(z_{v1}-z_{c1})+c_{s1}(\dot{z}_{v1}-\dot{z}_{c1}))+Z(k_{s2}(z_{v2}-z_{c2})+c_{s2}(\dot{z}_{v2}-\dot{z}_{c2}))}{D} \\ \qquad\qquad \frac{+P(k_{s3}(z_{v3}-z_{c3})+c_{s3}(\dot{z}_{v3}-\dot{z}_{c3}))-H(k_{s4}(z_{v4}-z_{c4})+c_{s4}(\dot{z}_{v4}-\dot{z}_{c4}))}{D} \\ \ddot{z}_{c3} = \frac{-H(k_{s1}(z_{v1}-z_{c1})+c_{s1}(\dot{z}_{v1}-\dot{z}_{c1}))+P(k_{s2}(z_{v2}-z_{c2})+c_{s2}(\dot{z}_{v2}-\dot{z}_{c2}))}{D} \\ \qquad\qquad \frac{+Q(k_{s3}(z_{v3}-z_{c3})+c_{s3}(\dot{z}_{v3}-\dot{z}_{c3}))+T(k_{s4}(z_{v4}-z_{c4})+c_{s4}(\dot{z}_{v4}-\dot{z}_{c4}))}{D} \\ \ddot{z}_{c4} = \frac{L(k_{s1}(z_{v1}-z_{c1})+c_{s1}(\dot{z}_{v1}-\dot{z}_{c1}))-H(k_{s2}(z_{v2}-z_{c2})+c_{s2}(\dot{z}_{v2}-\dot{z}_{c2}))}{D} \\ \qquad\qquad \frac{+T(k_{s3}(z_{v3}-z_{c3})+c_{s3}(\dot{z}_{v3}-\dot{z}_{c3}))+W(k_{s4}(z_{v4}-z_{c4})+c_{s4}(\dot{z}_{v4}-\dot{z}_{c4}))}{D} \end{cases} \tag{2.18}$$

where

$$
\begin{aligned}
F &= I_{c_x} m_c l_f^2 + I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y} \\
G &= I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y} - I_{c_x} l_f l_r m_c \\
H &= I_{c_x} l_f l_r m_c - I_{c_x} I_{c_y} + I_{c_y} m_c t_l t_r \\
L &= I_{c_x} m_c l_f^2 + I_{c_x} I_{c_y} - I_{c_y} m_c t_l t_r \\
P &= I_{c_x} m_c l_r^2 + I_{c_x} I_{c_y} - I_{c_y} m_c t_l t_r \\
Q &= I_{c_x} m_c l_r^2 + I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y} \\
T &= I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y} - I_{c_x} l_f l_r m_c \\
W &= I_{c_x} m_c l_f^2 + I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y} \\
Z &= I_{c_x} m_c l_r^2 + I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y} \\
D &= I_{c_x} I_{c_y} m_c
\end{aligned}
$$

where the first derivative relation is not reported because it is already present as the state of the system.

The matrices obtained for the state space model are reported in Appendix A.

2.3.3 Simulink implementation

Compared to the Quarter Car case, in this case it is more convenient to implement the differential equations using block diagrams in Simulink. In this case we find it particularly convenient to derive the model via State Space. In fact, we can use the Simulink block `StateSpace` for the implementation of the system. The final model implementation for simulation is shown in figure 2.8.

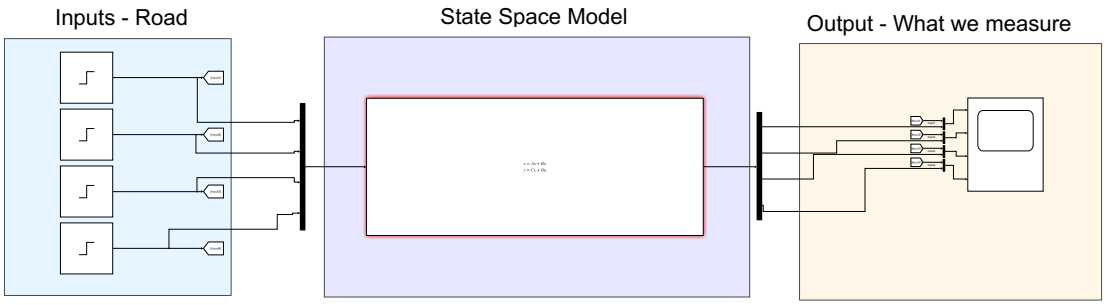

Figure 2.8: Simulink representation of the full car model using State Space

2.3.4 Simulation of Full Car model using the bump road profile

In order to carry out the system simulation it is necessary, as done in the case of the Quarter Car Model, to define the values of the Full Car model parameters, listed in Appendix B.

Also, as in the previous section it was chosen to use a Bump Road Profile as input, but the formula must be appropriately generalized as expressed in equation eq. (2.19) where u_f are the front wheel input disturbances z_{r1} and z_{r4}, while u_r are the rear wheel input disturbances z_{r2} and z_{r3}, a is the bump amplitude, λ is the length of the bump, t_f and t_r are the time in seconds of the bump for the front wheels and the bump for the rear wheels respectively, t_d is the time delay between the front and rear wheels and V is the vehicle forward velocity.

$$
\begin{aligned}
u_f &= \begin{cases} \frac{a}{2}\left(1-\cos\left(\frac{2\pi V t_f}{\lambda}\right)\right) & 1 \le t_f \le 1+\frac{\lambda}{V} \\ 0 & \text{otherwise} \end{cases} \\
u_r &= \begin{cases} \frac{a}{2}\left(1-\cos\left(\frac{2\pi V t_r}{\lambda}\right)\right) & t_d \le t_r \le \frac{\lambda}{V}+t_d \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.19}
$$

As in the previous case, it was chosen to use $a = 30$ mm, $V = 10$ m/s and $\lambda = 5$ m. In addition, the delay time t_d was calculated taking into account the length of the vehicle as:

$$
t_d = T_1 + \frac{(l_f + l_r)}{V} \tag{2.20}
$$

where T_1 corresponds to the start time of the bump while l_f and l_r correspond to the distance of the front and rear axle to the body center of mass. The input obtained using the proposed equations is shown in figure 2.9.

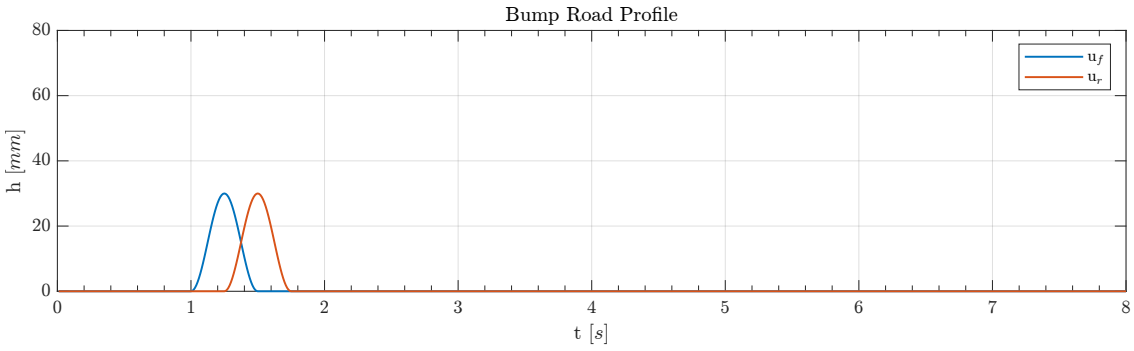

Figure 2.9: Bump road profiles for full car model

The simulation of the system using the bump road profile just derived as input is shown in figure 2.10. To obtain it, the Simulink model was used 2.8 with a sample time $Ts = 0.01$ s.

We can see some symmetry in the simulation result since, as can be verified from the table, the parameters of the front of the vehicle were assumed to be identical, and likewise the parameters of the rear of the vehicle were assumed to be identical.

As in the previous case of the quarter car model, we can already guess that the system is asymptotically stable, since after an initial phase of oscillations, each of the accelerations at the four corners of the chassis settles to a steady state value of 0.

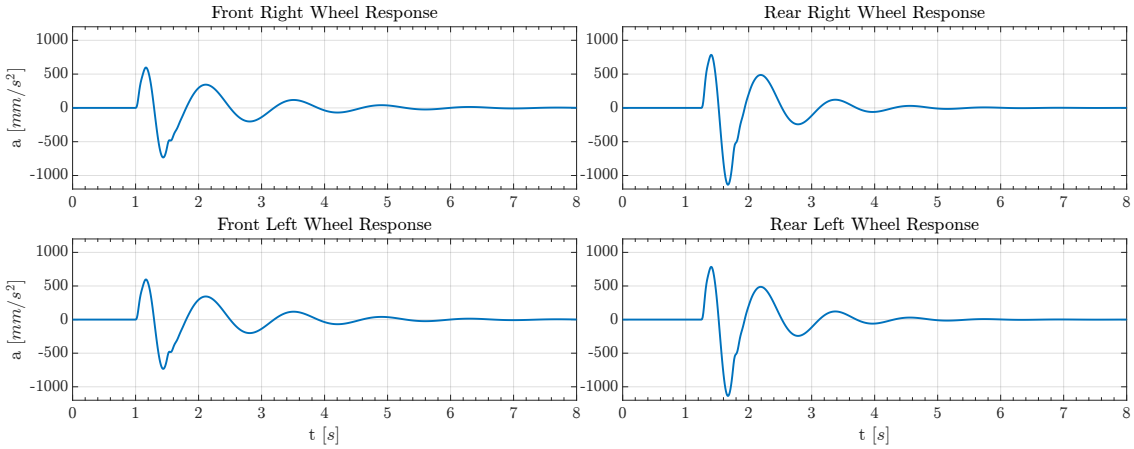

Figure 2.10: Outputs response of the Full Car model using the bump road profiles as inputs

3 Road Profile

Having obtained the Quarter Car model and the Full Car model in chapter 2, in this chapter we will try to generate inputs for our models as consistent with reality as possible. To do this we will rely on the road profile classification based on ISO 8608. After a description of this classification, we will use the so-called Sinusoidal Approximation method to generate via MATLAB the road profiles, obtained initially in the space domain and then in the time domain. Finally, a simulation of the Quarter Car model and Full Car model will be shown using the generated road profiles as input.

3.1 Road profile classification

Once we have obtained the differential equations that define the dynamics of a vehicle, the next step is to generate a reality-consistent signal to use as input to the model found in the previous chapter. Since it is immediately assumed that the quarter car model travels on a road, the goal of this chapter will be to find a way to generate a signal that reproduces a real road. According to [6] a road can be modeled as a piece of a two-dimensional surface taken along an imaginary line. This simplified "modeling" is called a road profile. The road profile is called lateral road profile if the imaginary line follows the cross-road direction, while it is called longitudinal road profile if the imaginary line follows the road direction. In this thesis we will only dwell on the longitudinal road profile.

In general from the literature, most methods for road profile synthesis exploit the characterization of the road profile in terms of roughness and uneveness. In particular, the two most widely used methods for classifying a road profile are:

- IRI
- ISO 8608

The first classification, published in 1986, in the International Road Roughness Experiment by the World Bank aims to establish a standard for methods of evaluating and measuring road roughness. To assess road roughness on a single scale, the authors developed the International Roughness Index (IRI).

The IRI is a statistical measure that is calculated using a virtual quarter car moving at a steady speed over a road profile. However, since IRI is more related to the comfort experienced by a private car, but it is unsuitable for a mathematical description of the road profile, the ISO 8608 classification, which will be discussed in detail in the following section, is almost always taken as a reference for generating a road profile.

3.1.1 ISO 8608 for road profiles classification

Standard ISO 8608 [7] specifies road classification of longitudinal road profiles based on vertical displacement power spectral density (PSD). The PSD is represented on logarithmic scales and it is characterized by two parameters to describe the road profile: the degree of roughness G_d, (defined with respect to the spatial frequency $G_d(n_0)$ or with respect to the angular spatial frequency $G_d(\Omega_0)$), identified as power spectral density (PSD) of the vertical road profile displacement and, as the slope of the line, the waviness w.

$$G_d(n) = G_d(n_0) \cdot \left(\frac{n}{n_0}\right)^{-w} \tag{3.1}$$

Specifically, ISO 8608 proposes to set the waviness $w = 2$, and to classify the road profile based only on the degree of roughness G_d in correspondence of conventional values of spatial frequency $n_0 = 0.1$ cycles/m and angular spatial frequency $\Omega_0 = 1$ rad/m.

$$\begin{aligned} G_d(n) &= G_d(n_0) \cdot \left(\frac{n}{n_0}\right)^{-2} \\ G_d(\Omega) &= G_d(\Omega_0) \cdot \left(\frac{\Omega}{\Omega_0}\right)^{-2} \end{aligned} \tag{3.2}$$

Switching from one convention to another is extremely simple and is achieved by the formula eq. (3.3) showing the relationship between $G_d\,(n_0)$ and $G_d\,(\Omega_0)$).

$$G_d\,(\Omega_0)\,\left(\mathrm{m}^3/\mathrm{rad}\right) = \frac{G_d\,(n_0)}{2\pi}\,\left(\mathrm{m}^3/\mathrm{cycles}\right) \tag{3.3}$$

According to this standard, road profiles are classified into one of 8 classes (A to H), as shown in figure 3.1.

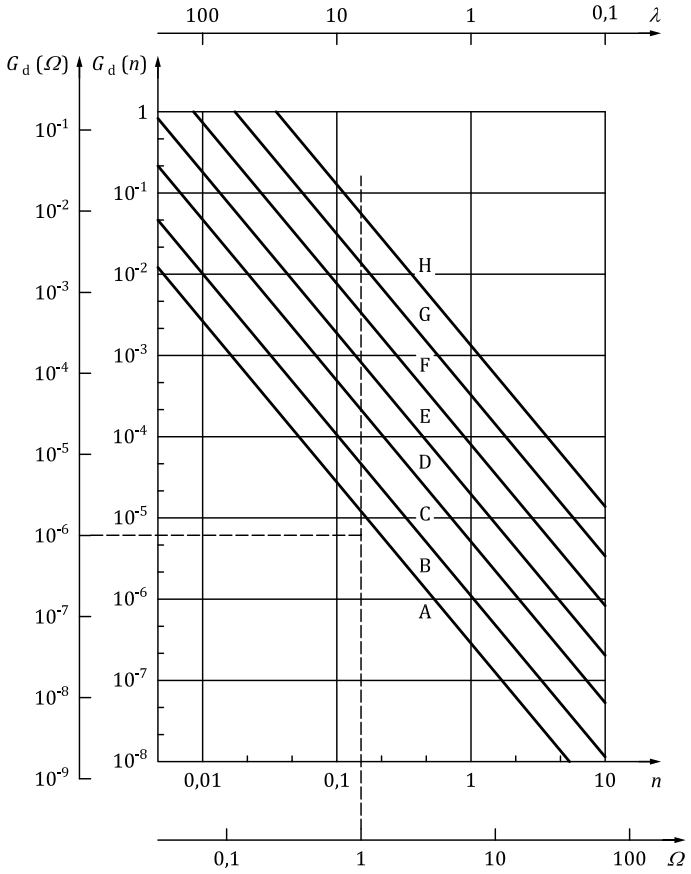

Figure 3.1: Classification of roads profile from classes A to H using PSD according to ISO 8608

Specifically, the values of the lower and upper limit of the degree of roughness G_d for each class are given in table 3.1.

Road class	$G_d(n_0)$ $(10^{-6}\ m^3)$		$G_d(\Omega_0)$ $(10^{-6}\ m^3)$	
	Lower limit	Upper limit	Lower limit	Upper limit
A	-	32	-	2
B	32	128	2	8
C	128	512	3	128
D	512	2048	32	12
E	2048	8192	128	512
F	8192	32768	512	2048
G	32768	131072	2048	8192
H	131072	-	8192	-
	$n_0 = 0.1$ cycles/m		$\Omega_0 = 1$ rad/m	

Table 3.1: Upper and lower limit of PSD values $G_d(n_0)$ and $G_d(\Omega_0)$ of each road classes according to ISO 8608

The ISO Road profile standard also defines the type of road for each class and the maximum permissible speeds for each road profile class as reported by Georges et al. in [8] are shown in table 3.2. In particular, the reference value $G_d(n_0)$ is the average value between the upper and lower bound of the classification.

ISO Class	$G_d(n_0)$ $(10^{-6}\ m^3)$	Description	v_{max} (ms^{-1})
A	16	Airport runways and superhighways	60
B	64	Normal pavements	50
C	256	Unpaved roads and damaged pavements	30
D	1024	Rough unpaved roads	20
E	4096	Enduro tracks	10
F	16384	Off-road tracks	5
G	65536	Rough off-road tracks	5
H	262144	Simulation purpose only	5

Table 3.2: Description of each road classes according to ISO 8608

3.1.2 Longitudinal road profile synthesis based on ISO 8608

According to [9], in general, there are three possible methods for generating a road profile based on ISO 8608

- White Noise filtration
- Sinusoidal Approximation
- Moving average of white noise

The White Noise filtration method is the simplest method, having only three parameters to define. The main advantage is the easy implementation in the Simulink environment, however the PSD generated does not fully reflect that proposed by ISO 8608.

The sinusoidal approximation method makes it possible to generate road profile segments of a certain length, with the possibility then to have repeatability. An advantage of this approach is that there is no risk of numerical errors, since the differentiations are obtained without going through numerical differentiations. In addition, this procedure is composed of the largest number of parameters to be defined. For these last two features, it is among the three the most computational demanding method.

The last method is the Moving average of white noise, which has the advantage (or disadvantage) of having no repeatability of road profile segments and which allows for possible various PSD approximations.

In this thesis it was chosen to use the sinusoidal approximation method, which will be described in detail in the next section.

3.2 Sinusoidal Approximation

A collection of sinusoidal waves in the spatial or temporal domain that have varying wavelengths, amplitudes, and phases can be used mathematically to create the longitudinal road profile. By dividing the one-sided PSD spectrum into multiple frequency bands, the same number of harmonic samples with corresponding band frequency and spectral height with random phase angles ϕ_i are generated using this method. The value of the Power Spectral Density PSD function for the assigned frequency n, given a continuous road profile and a defined value of spatial frequency n centered within a frequency range Δn, is given by the following expression:

$$G_d(n) = \lim_{\Delta n \to 0} \frac{\psi_x^2(n, \Delta n)}{\Delta n} \tag{3.4}$$

where ψ_x^2 is the signal's mean square value within the frequency band Δn for the spatial frequency n. Then the PSD of the road signal according to ISO 8608 is discretized by a sequence of uniformly spaced elevation points and formula in eq. (3.4) can be rewritten as:

$$G_d\left(n_i\right) = \lim_{\Delta n \to 0} \frac{\psi_x^2\left(n_i, \Delta n\right)}{\Delta n} = \frac{\psi_x^2(i \cdot \Delta n, \Delta n)}{\Delta n} \tag{3.5}$$

where i varies between 0 and $N = \frac{n_{\max}}{\Delta n}$. At this point that signal can be described by a simple harmonic function:

$$\begin{aligned} y(x) &= A_i \cos\left(2\pi \cdot n_i \cdot x + \phi\right) \\ &= A_i \cos(2\pi \cdot i \cdot \Delta n \cdot x + \phi) \end{aligned} \tag{3.6}$$

where A_i denotes the amplitude, n_i the spatial frequency, and ϕ the phase angle, the mean square value of this harmonic signal can be proved to be:

$$\psi_x^2 = \frac{A_i^2}{2} \tag{3.7}$$

Substituting the expression eq. (3.7) into the formula in eq. (3.5) we get:

$$G_d\left(n_i\right) = \frac{\psi_x^2\left(n_i\right)}{\Delta n} = \frac{A_i^2}{2 \cdot \Delta n} \tag{3.8}$$

Hence, the amplitude can be obtained as follows:

$$A_i = \sqrt{2 G_{\mathrm{d}}\left(n_i\right) \Delta n} \tag{3.9}$$

Then, the samples are superimposed to generate the road profile in the spatial or time domain by:

$$z_{\mathrm{R}}(x) = \sum_{i=1}^{N} A_i \cos\left(2\pi n_i x + \phi_i\right) \quad \text{with} \quad A_i = \sqrt{2 G_{\mathrm{d}}\left(n_i\right) \Delta n} \tag{3.10}$$

or equivalently as:

$$z_{\mathrm{R}}(x) = \sum_{i=1}^{N} A_i \cos\left(2\pi \Omega_i x + \phi_i\right) \quad \text{with} \quad A_i = \sqrt{2 G_{\mathrm{d}}\left(\Omega_i\right) \Delta \Omega} \tag{3.11}$$

where $N = \frac{\Omega_U - \Omega_L}{\Delta \Omega}$ is the number of frequency bands into which the entire PSD spectrum is divided, Ω_U, Ω_L are the upper and lower spatial frequencies in the PSD spectrum (rad/m),

$\Delta\Omega$ is the width of each frequency band, and $\Omega_i = L + (i-1)\Delta\Omega$ represents the ith harmonic or frequency band's spatial frequency. Finally $\phi_i = \mathcal{U}(0, 2\pi)$ is the random phase of ith harmonic uniformly distributed in the interval $(0, 2\pi)$. The amplitudes A_i are calculated from the straight line fit $G_d(n_i)$ for the N different spatial frequencies n_i.

3.2.1 MATLAB implementation of Sinusoidal Approximation

For the implementation of the previous formulas in MATLAB, the following approximation from [10] has been used:

$$A_i = \sqrt{2G_{\mathrm{d}}\left(n_i\right)\Delta n} \quad = 2^k 10^{-3}\sqrt{\Delta n}\left(\frac{n_0}{n_i}\right) \tag{3.12}$$

Finally, using eq. (3.12) in eq. (3.11), the road profile in spatial domain can be obtained as follows:

$$h(x) = \sum_{i=0}^{N} A_i \cos\left(2\pi n_i x + \phi_i\right) \tag{3.13}$$

where

$$A_i = 2^k 10^{-3}\sqrt{\Delta n}\left(\frac{n_0}{n_i}\right) \quad \text{and} \quad \phi_i = \mathcal{U}(0, 2\pi) \tag{3.14}$$

At this point two options are possible:

- Sample in Spatial domain
- Sample in Time domain

The next two sections will apply the formula for the road profile in spatial domain assuming sampling in both the spatial domain and directly in the time domain.

Sampling in Spatial domain

Considering sampling in spatial domain we need to define the number of data points we want to generate N and the length of the profile L. We can then determine the sampling interval $B = \frac{L}{N}$ and the frequency band $\Delta n = \frac{1}{L}$. Finally we compute the spatial frequency band n which will be spanned by samples between $n \in [\Delta n, \Delta n \cdot N]$ equidistant with spatial frequency Δn. Finally, the amplitudes of the N samples at spatial frequencies equidistant in the space

Class	k	Height Max
A	3	± 15 mm
B	4	± 25 mm
C	5	± 50 mm
D	6	± 100 mm
E	7	± 200 mm
F	8	± 400 mm

Table 3.3: k values for generation of each ISO road class and corresponding height limit

x between 0 and L at intervals defined by the spatial sampling interval B with respect to the conventional spatial frequency $n_0 = 0.1$ cycles/m reference defined by the ISO 8608 standard are calculated.

The k are defined based on the desired road profile according to [10] and reported in table 3.3 as well as the maximum height for each class according to the used procedure.

Then in figure 3.2, we show the comparison of the road profile obtained for each road class. Specifically, the upper plot shows the same road profile of length $l = 50$ m generated for each of classes A through F, while the bottom plot shows the detailed extract of the road profile of class B. Note that class H is not shown in the plot, because as reported in table 3.2, ISO 8608 defines class H only for simulation purpose.

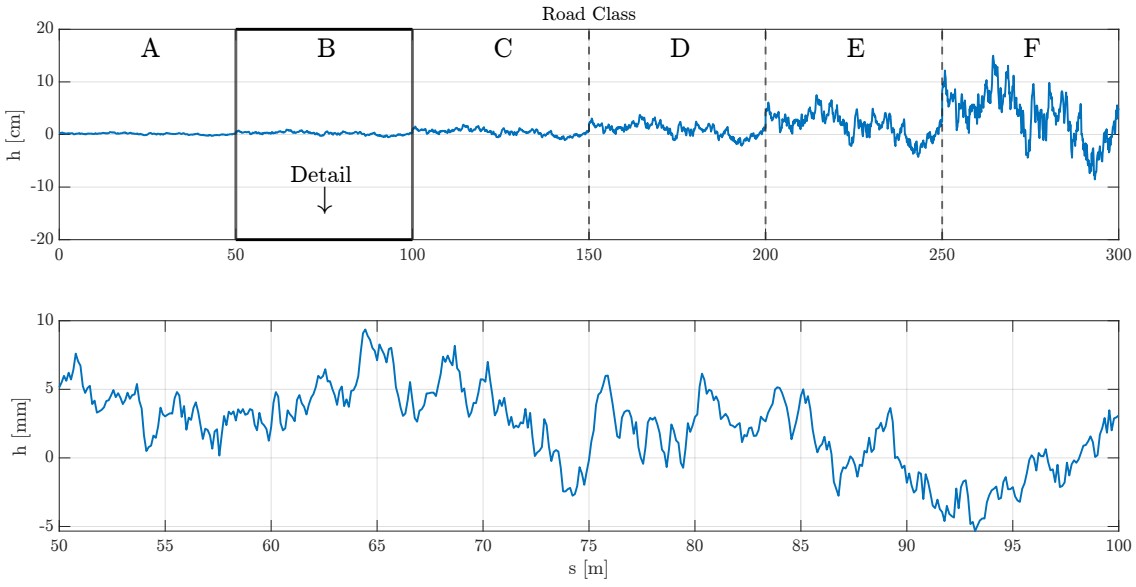

Figure 3.2: Road profiles belonging to road classes A–F and a detailed extract of class B

From Space domain to Time domain

Assuming to sample in the time domain the parameters we will have to initially define will be different. In our case we have assumed to sample at a Sample time $Ts = 0.01$ s ($f = 100$ Hz) on a road of length $L = 250$ m traveling at a constant speed of 40 km/h ≈ 11 m/s. We can then calculate the total duration $T = L/v = 22.50s$, the number of samples $N = fT = 2250$ and the sampling interval $B = L/N = 0.1$ and the spatial frequency band $\Delta n = 1/L = 0.004$ cycles/m. At this point we can go back to what was said in the subsection about generation in the spatial domain. By choosing a $k = 3$ to generate a class A road profile we get that the road profile in the spatial domain represented in figure 3.3.

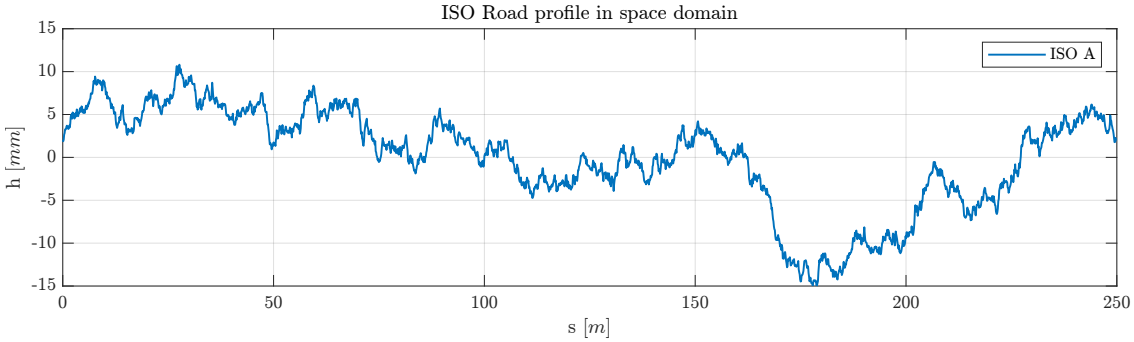

Figure 3.3: Road profile generated using ISO 8608 in spatial domain

At this point we can simply divide the space by the assumed speed to get the generated profile versus time in figure 3.4.

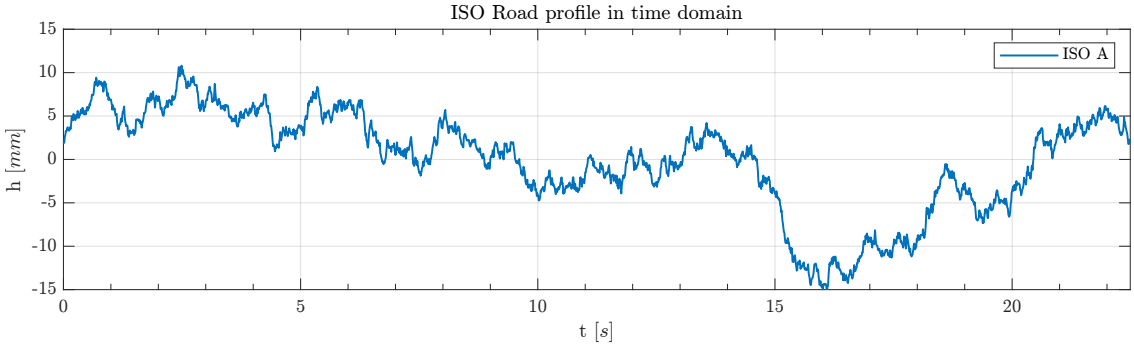

Figure 3.4: Road profile generated using ISO 8608 in time domain

Finally we show the PSD of the road profile generated in figure 3.5 in which we can see that it is correctly classified as class A. Especially for low frequencies, we can see from the PSD plot the equidistant samples $\Delta n = 0.004$ cycles/m.

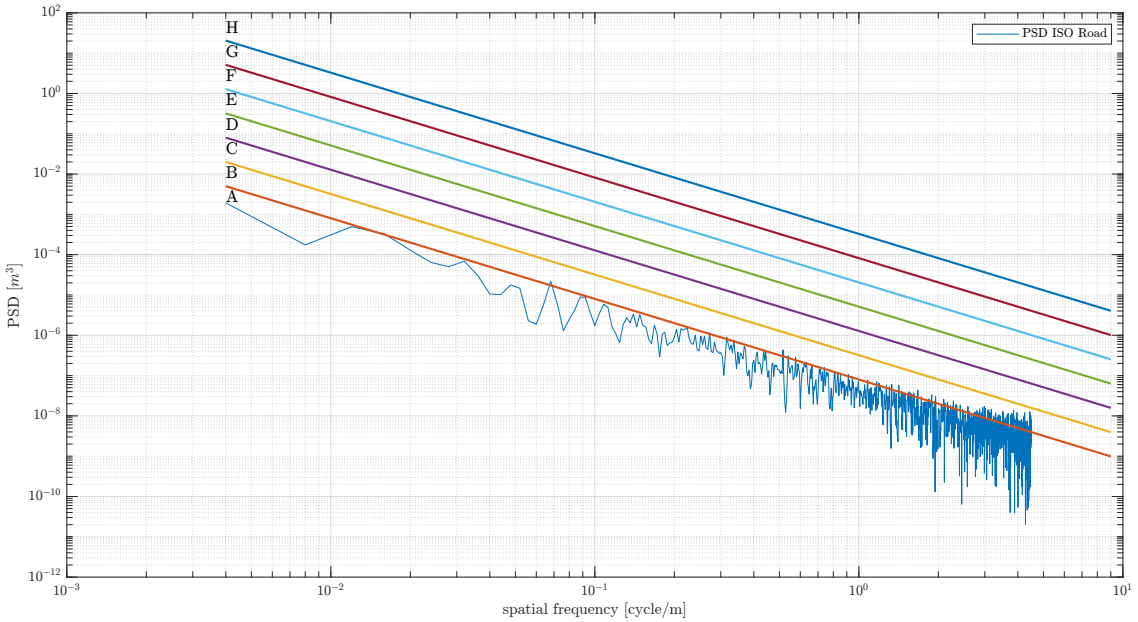

Figure 3.5: PSD classification according to ISO 8608 of the road profile generated

3.2.2 Simulation of Road Profiles

This last section shows the results obtained in simulation by exploiting the sinusoidal approximation method for generating road profiles to be applied to the quarter car model and the full car model. In both cases, the results are obtained using a sample time $T_s = 0.01$ s ed e' stato scelto di utilizzare un profilo di classe A secondo la classificazione ISO 8608.

For the simulation of the quarter car model, the road profile used as input is that obtained in figure 3.4. The result is obtained using the Simulink model represented in figure 2.3 and it is shown in figure 3.6, and the resulting acceleration at the sprung mass is shown at the bottom of the same plot.

For the simulation of the full car model we will need four road profiles based on ISO 8608. Assuming that the rear wheels follow the same road profile as the front wheels, the road excitation in input on the rear wheels is the same as the one for the front wheels but with a time delay τ. Thus, for the right front wheel the same road profile applied on the quarter car model was used, for the left front wheel a new road profile was generated again using the Sinusoidal approximation method with $k = 3$ (road profile of class A). For the road excitations of the rear wheels, using the previous assumptions, the corresponding road profile of the front wheel shifted in time with respect to a time delay $\tau = 1.1$ s was used.

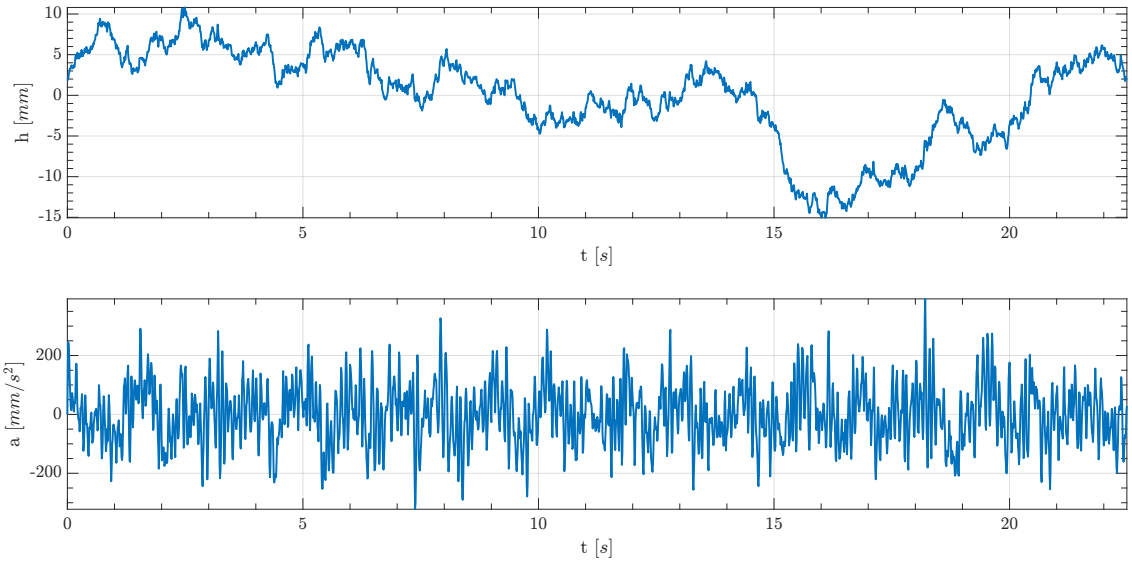

Figure 3.6: Quarter car model output response (bottom figure) using the road profile as input (top figure)

The final result is obtained using the four road profile generated in figure 3.7 as inputs of the Simulink model represented in figure 2.8. The output corresponding to the measurement of the accelerometers at the four corners of the chassis is shown in figure 3.8.

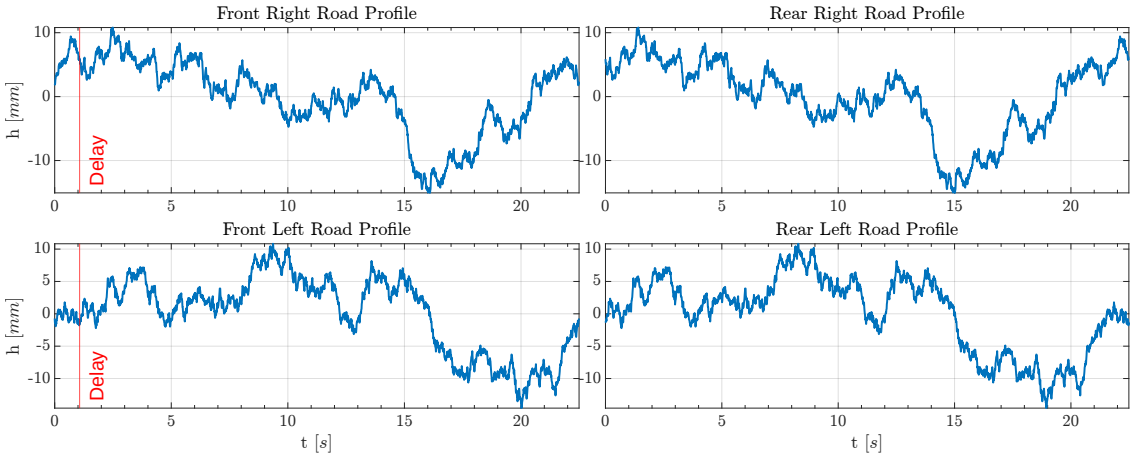

Figure 3.7: Road profile generated according to ISO 8608 for each of the 4 wheels

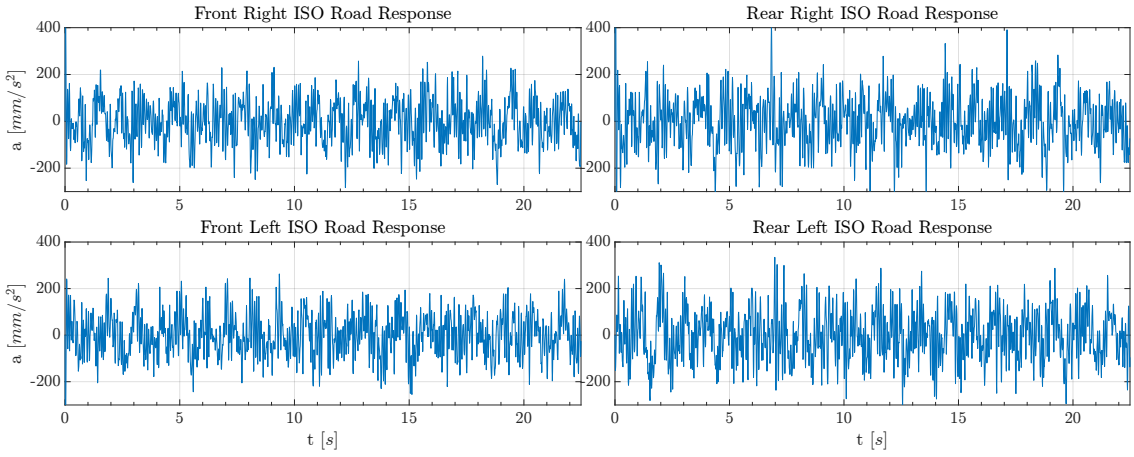

Figure 3.8: Full car model output response using the road profiles as input

4

ILC Theory

The objective of this chapter is to introduce the Iterative Learning Control algorithm. After a brief summary of the history, the main differences with other so-called ”repetitive” algorithms will be introduced and then the fundamental theoretical assumptions a the current application domains will be listed. The theoretical foundations of the algorithm will then be explained, in particular referring to the representation in the so-called lifted domain for the implementation of the algorithm in the time domain. Finally, the two main theorems of Iterative Learning Control concerning stability and monotonic convergence will be proved.

4.1 Introduction

4.1.1 Brief History

The concept of Iterative Learning Control was first used in 1978 with the publication of an article by Uchiyama [11]. However, the text of the article was in Japanese, and this did not allow the publication to spread worldwide

It was only in 1984 with the English article by Arimoto et al. [12] that the algorithm gained more interest. Although the text does not explicitly refer to the term iterative learning control but rather as ”Bettering operation,” this contribution is regarded as the origin of the ILC algorithm.

After more than 30 years of growth since Arimoto et al. proposal, ILC has produced many positive outcomes and grown to be a significant area of control research.

4.1.2 Differences with others learning based algorithms

The ILC algorithm has some similarities with other forms of control that act repetitively with the goal of learning from the previous iteration, including adaptive control and repetitive control (RC). However, in order to best select the type of algorithm to use, it is good to make a distinction with respect to the features that differentiate them.

Adaptive control at each iteration modifies a controller, which is a system, unlike the Iterative Learning Control, which at each iteration modifies the input, which is a signal. Also, adaptive control does not exploit repetitive signals, which instead is, as we will see, a necessary condition for the operation of Iterative Learning Control.

The algorithm that turns out to be most similar is Repetitive Control (RC), which, however, is associated with continuous operation, unlike ILC which is associated with discrete operation. Also, while the ILC requires resetting the initial conditions at each cycle to the previous initial conditions, in the RC algorithm the initial conditions are reset to the final conditions of the previous cycle.

4.1.3 Fundamental assumptions

Iterative Learning Control (ILC) is a control methodology that attempts to improve the transient response of systems that operate repetitively by adjusting the input to the plant during future system operation based on the errors observed during past operation.

The fundamental assumption of ILC algorithm are drawn from the work of Norrlöf et al. in [13].

1. The signals are finite on the time axis, so that each iteration ends with the same duration $t = T_f \in \mathbb{R}^+$.
2. Over the interval $t \in \left[0, T_f\right]$, the reference $r(t)$ is provided a priori.
3. The system's starting state is the same for every iteration. i.e. $x_j(0) = x_0 \in \mathbb{R}^n, \forall j \in [0, j_{max}]$.
4. The dynamics of the system remains unchanged during all iterations.
5. Each system output $y_j(t)$ is measurable so that the tracking error information $e_j(t) = r(t) - y_j(t)$ can be obtained each time $u_{j+1}(t)$ is computed.

where j is the iteration number and j_{max} is the maximum number of iterations.

4.1.4 Applications

Iterative Learning Control has emerged as one of the most successful control strategies when addressing issues with recurrent tracking control and in general when we have periodic disturbance.

The ILC algorithm has been used for several practical industrial applications, including industrial robots [14], computer numerical control (CNC) machine tools [15], aluminum extruders [16], wafer stage motion systems [17] and induction motors [18], but also in the automotive field, for example for autonomous vehicles [19] and antilock braking systems [20].

In the next section, an overview of the algorithm based by the paper of Bristow et al. in [21] is presented.

4.2 Algorithm overview

Consider the discrete-time, linear time-invariant (LTI), single-input single-output (SISO) system

$$y_j(k) = P(\mathrm{q})u_j(k) + d(k) \tag{4.1}$$

where d is an exogenous signal that repeats every iteration, y_j is the output, u_j is the control input, q is the forward time-shift operator $\mathrm{q}x(k) = x(k+1)$, $P(\mathrm{q})$ is a proper rational function of q which has a relative degree of m (i.e. a delay of m samples) and k, j, are the time index and iteration index, respectively.

Assuming that the whole sequence of inputs and outputs is composed of N-sample, and that the plant $P(\mathrm{q})$ is a proper rational function of q and that it has a delay (or equivalently a relative degree) m, we have:

$$\begin{aligned} u_j(k), &\quad k \in \{0, 1, \ldots, N-1\} \\ y_j(k), &\quad k \in \{m, m+1, \ldots, N+m-1\} \\ d(k), &\quad k \in \{m, m+1, \ldots, N+m-1\} \end{aligned}$$

The goal of the algorithm is to follow a desired system output

$$r(k), \quad k \in \{m, m+1, \ldots, N+m-1\}$$

that is, to send the tracking error signal defined as:

$$e_j(k) = r(k) - y_j(k) \tag{4.2}$$

From now on, for simplicity, it will be assumed that the plant delay of the m system is $m = 1$.

Numerous techniques for updating the input signal have been covered in the literature, but still almost all ILC algorithms rely on the following input correction:

$$u_{j+1}(k) = Q(\mathrm{q})\left(u_j(k) + L(\mathrm{q})e_j(k+1)\right) \tag{4.3}$$

where $L(\mathrm{q})$ and $Q(\mathrm{q})$ are design parameters of the algorithm.

We can see that the input signal of the current iteration is calculated as the sum of the previous control action u_j and an update term $L(\mathrm{q})e_j$ that is obtained from the previous error information. Consequently, it will be necessary to keep all the previous sequence of inputs and errors in memory for updating the control, as represented in figure 4.1, where the reference r is represented as y_d and G is the transfer function of the plant model $P(\mathrm{q})$.

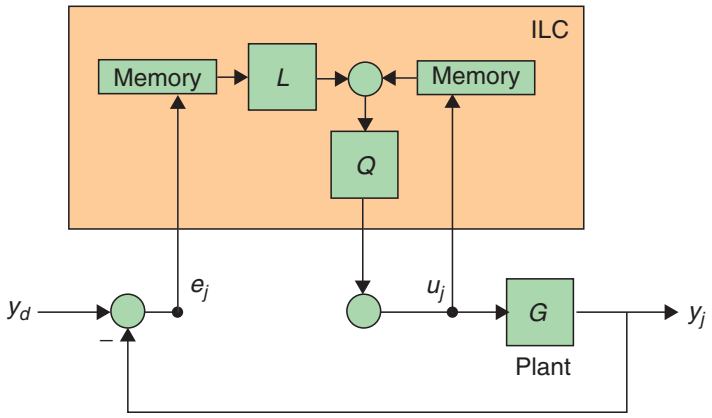

Figure 4.1: ILC architecture scheme

A very important aspect for the operation of the algorithm is that $P(\mathrm{q})$ in the eq. (4.1) is assumed to be asymptotically stable. In case the system is already asymptotically stable, ILC can be applied in the so-called open loop. A feedback controller can be used to stabilize $P(\mathrm{q})$ when it is not asymptotically stable, and the ILC can then be applied to the closed-loop system.

An example of a method to incorporate the feedback controller is through the so-called Current Iteration Iterative Learning Control represented in figure 4.2, where the reference r is represented as y_d and G is the transfer function of the plant model $P(\mathrm{q})$. In this case the update input equation is given by:

$$u_{j+1}(k) = Q(\mathrm{q})\left(u_j(k) + L(\mathrm{q})e_j(k+1)\right) + C(q)e_{j+1}(k) \tag{4.4}$$

in which unlike the general equation eq. (4.3) a learning component $C(q)e_{j+1}(k)$ is added with respect to the current iteration.

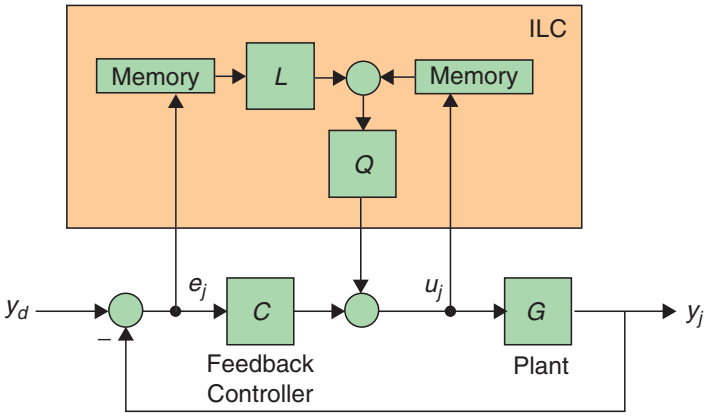

Figure 4.2: ILC architecture scheme using Feedback Controller

A special feature of the ILC algorithm, which differs from controllers such as the traditional feedback and feedforward, is that since the updating eq. (4.3) is carried out offline between trials, it's possible to design the two filters in a noncausal manner. Non-causality is possible in practice because the entire data sequence is available in advance for the duration of the iteration. This makes it possible to anticipate and compensate for any repetitive disturbances or as we will see later, implement the filters in zero-phase lag mode.

There are two methods of representations for implementing an ILC algorithm:

- Time-domain ILC
- Frequency-domain ILC

There are benefits and drawbacks to each option. Although the matrix formulation used by the Time-domain ILC covers a wider range of algorithms, it is generally more computationally

demanding. Conversely, the filter form used by the Frequency-domain ILC needs fewer calculations but it is less general. The following analysis will focus only on lifted time domain-based representation, as this is the method used in this thesis.

4.3 Time-domain ILC

Considering a generic discrete time state space model

$$\begin{aligned} \mathbf{x}_i(k+1) &= \mathbf{A}\mathbf{x}_i(k) + \mathbf{B}u_i(k) \\ y_i(k) &= \mathbf{C}\mathbf{x}_i(k), \end{aligned} \tag{4.5}$$

In order to implement the ILC algorithm in the time-domain, it is necessary to introduce lifted domain representation:

$$P(\mathrm{q}) = \sum_{k=0}^{\infty} p_k \mathrm{q}^{-k} \tag{4.6}$$

Considering a finite sequence of N samples, we obtain:

$$\begin{aligned} P(\mathrm{q}) &= \sum_{k=1}^{N} p_k \mathrm{q}^{-k} \\ &= p_1\mathrm{q}^{-1} + p_2\mathrm{q}^{-2} + p_3\mathrm{q}^{-3} + \ldots + p_N\mathrm{q}^{-N} \end{aligned} \tag{4.7}$$

where the sequence $p_1, p_2, \ldots, p_N$ is the impulse response of the system. For a state space system eq. (4.5), the impulse response is given by:

$$p_k = \mathbf{C}\mathbf{A}^{k-1}\mathbf{B} \tag{4.8}$$

with $k > 0$. In this way the matrix of the system is transformed into an $N \times N$ matrix as follows:

$$\mathbf{P} = \begin{bmatrix} \mathbf{CB} & 0 & \cdots & 0 \\ \mathbf{CAB} & \mathbf{CB} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^{p-1}\mathbf{B} & \mathbf{CA}^{p-2}\mathbf{B} & \cdots & \mathbf{CB} \end{bmatrix} \tag{4.9}$$

which we can rewrite as:

$$\mathbf{P} = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_{N-1} & \cdots & p_1 \end{bmatrix} \tag{4.10}$$

where we can see that the impulse response sequence of the function $P(\mathrm{q})$ is found in the first column of matrix $\mathbf{P}$. The remaining columns are simply obtained by shifting this sequence by one sample at a time until the column N. Hence, now we can rewrite explicitly eq. (4.1) in lifted form as:

$$\underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\mathbf{y}_j} = \underbrace{\begin{bmatrix} p_1 & 0 & \cdots & 0 \\ p_2 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N & p_{N-1} & \cdots & p_1 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} u_j(0) \\ u_j(1) \\ \vdots \\ u_j(N-1) \end{bmatrix}}_{\mathbf{u}_j} + \underbrace{\begin{bmatrix} d(1) \\ d(2) \\ \vdots \\ d(N) \end{bmatrix}}_{\mathbf{d}} \tag{4.11}$$

and

$$\underbrace{\begin{bmatrix} e_j(1) \\ e_j(2) \\ \vdots \\ e_j(N) \end{bmatrix}}_{\mathbf{e}_j} = \underbrace{\begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(N) \end{bmatrix}}_{\mathbf{y}_d} - \underbrace{\begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(N) \end{bmatrix}}_{\mathbf{y}_j} \tag{4.12}$$

and considering the general case in which $\mathbf{L}$ and $\mathbf{Q}$ are noncausal filters, that is, in matrix form:

$$\mathbf{L} = \begin{bmatrix} l_0 & l_{-1} & \cdots & l_{-(N-1)} \\ l_1 & l_0 & \cdots & l_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1} & l_{N-2} & \cdots & l_0 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} q_0 & q_{-1} & \cdots & q_{-(N-1)} \\ q_1 & q_0 & \cdots & q_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N-1} & q_{N-2} & \cdots & q_0 \end{bmatrix} \tag{4.13}$$

the input correction can be written in the lifted form as:

$$\underbrace{\begin{bmatrix} u_{j+1}(0) \\ u_{j+1}(1) \\ \vdots \\ u_{j+1}(N-1) \end{bmatrix}}_{\mathbf{u}_{j+1}} = \underbrace{\begin{bmatrix} q_0 & q_{-1} & \cdots & q_{-(N-1)} \\ q_1 & q_0 & \cdots & q_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N-1} & q_{N-2} & \cdots & q_0 \end{bmatrix}}_{\mathbf{Q}} \left(\underbrace{\begin{bmatrix} u_j(0) \\ u_j(1) \\ \vdots \\ u_j(N-1) \end{bmatrix}}_{\mathbf{u}_j} + \underbrace{\begin{bmatrix} l_0 & l_{-1} & \cdots & l_{-(N-1)} \\ l_1 & l_0 & \cdots & l_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1} & l_{N-2} & \cdots & l_0 \end{bmatrix}}_{\mathbf{L}} \underbrace{\begin{bmatrix} e_j(1) \\ e_j(2) \\ \vdots \\ e_j(N) \end{bmatrix}}_{\mathbf{e}_j} \right) \tag{4.14}$$

In the case, on the other hand, in which they are causal, we will have that

$$\begin{cases} q_{-1} = & q_{-2} = \ldots = 0 \\ l_{-1} = & l_{-2} = \ldots = 0 \end{cases}$$

and consequently **L** and **Q** will be lower triangular. In both cases, since all of the entries along each diagonal of the matrices **P**, **L** and **Q** are identical, the matrices are Toeplitz.

The logic of the lifted domain ILC can be represented according to the block diagram in figure 4.3 proposed by Rotariu et al. in [22].

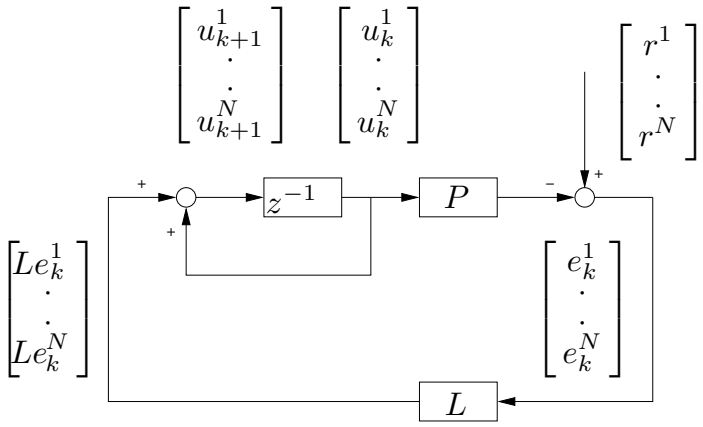

Figure 4.3: ILC architecture scheme represented in lifted domain

In the next two sections, the two main theorems of the ILC algorithm, the stability theorem and the monotonic convergence theorem, will be analyzed, with the reference to the PhD dissertation by Koçan in [23]. In both cases, for simplicity, the lifted domain representation of the ILC algorithm will be used, such that the whole vector sequence is considered. Hence, the the

system dynamics, the tracking error and the input correction become as follows, respectively:

$$\mathbf{y}_j = \mathbf{P}\mathbf{u}_j + \mathbf{d} \tag{4.15}$$

$$\mathbf{e}_j = \mathbf{r} - \mathbf{y}_j \tag{4.16}$$

$$\mathbf{u}_{j+1} = \mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\mathbf{e}_j\right) \tag{4.17}$$

4.3.1 Stability Analysis

In order to study the stability of the ILC algorithm, the error dynamics equation should be analysed. Using the system dynamics and the tracking error as in eq. (4.15) and in eq. (4.16) respectively, and considering the $(j+1)^{\text{th}}$ iteration, one gets:

$$\mathbf{y}_{j+1} = \mathbf{P}\mathbf{u}_{j+1} + \mathbf{d} \tag{4.18}$$

$$\mathbf{e}_{j+1} = \mathbf{r} - \mathbf{y}_{j+1} \tag{4.19}$$

Considering the dynamic of the error in eq. (4.19) and substituting $\mathbf{y}_{j+1}$ as in eq. (4.18) and $\mathbf{u}_{j+1}$ as in eq. (4.17) and finally $\mathbf{y}_j$ as eq. (4.15), we obtain:

$$\begin{aligned}
\mathbf{e}_{j+1} &= \mathbf{r} - \mathbf{y}_{j+1} \\
&= \mathbf{r} - \mathbf{P}\mathbf{u}_{j+1} - \mathbf{d} \\
&= \mathbf{r} - \mathbf{P}\mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\mathbf{e}_j\right) - \mathbf{d} \\
&= \mathbf{e}_j + \mathbf{y}_j - \mathbf{P}\mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\mathbf{e}_j\right) - \mathbf{d} \\
&= \mathbf{e}_j + \mathbf{y}_j - \mathbf{P}\mathbf{Q}\mathbf{u}_j - \mathbf{P}\mathbf{Q}\mathbf{L}\mathbf{e}_j - \mathbf{d} \\
&= \mathbf{e}_j + \mathbf{P}\mathbf{u}_j + \mathbf{d} - \mathbf{P}\mathbf{Q}\mathbf{u}_j - \mathbf{P}\mathbf{Q}\mathbf{L}\mathbf{e}_j - \mathbf{d} \\
&= (\mathbf{I} - \mathbf{P}\mathbf{Q}\mathbf{L})\mathbf{e}_j - \mathbf{P}(\mathbf{I} - \mathbf{Q})\mathbf{u}_j
\end{aligned}$$

$$\mathbf{e}_{j+1} = (\mathbf{I} - \mathbf{P}\mathbf{Q}\mathbf{L})\mathbf{e}_j - \mathbf{P}(\mathbf{I} - \mathbf{Q})\mathbf{u}_j \tag{4.20}$$

It is easy to see in eq. (4.20) that the matrix $(\mathbf{I} - \mathbf{P}\mathbf{Q}\mathbf{L})$ and its eigenvalues play a major role in determining the error convergence of the iterative system. Furthermore, the error at the

subsequent iteration is increasingly influenced by the value of $\mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j$, which is simply the difference between the output of the system without any filter and the output of the system with a Q-filter applied. Hence, in order to diminish this effect and achieve a faster convergence, the traditional choice of $\mathbf{Q}=\mathbf{I}$ is typically used.

Assuming for simplicity that $\mathbf{Q}=\mathbf{I}$, then the error dynamics in eq. (4.20) becomes:

$$\mathbf{e}_{j+1} = (\mathbf{I}-\mathbf{PL})\mathbf{e}_j \tag{4.21}$$

Hence, the ILC algorithm is stable if

$$\boxed{\rho(\mathbf{I}-\mathbf{LP}) < 1} \tag{4.22}$$

where $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$ is the maximum singular value.

Alternatively, the stability of the ILC algorithm can be derived considering the update equation in eq. (4.17). Indeed, if we substitute $\mathbf{e}_j$ as in eq. (4.2) and then $\mathbf{y}_j$ as in eq. (4.15), and finally grouping the terms, we get:

$$\begin{aligned}
\mathbf{u}_{j+1} &= \mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\mathbf{e}_j\right) \\
&= \mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\left(\mathbf{r}-\mathbf{y}_j\right)\right) \\
&= \mathbf{Q}\left(\mathbf{u}_j + \mathbf{L}\left(\mathbf{r}-\mathbf{P}\mathbf{u}_j-\mathbf{d}\right)\right) \\
&= \mathbf{Q}\mathbf{u}_j + \mathbf{QLr} - \mathbf{QLP}\mathbf{u}_j - \mathbf{QLd} \\
&= \mathbf{Q}(\mathbf{I}-\mathbf{LP})\mathbf{u}_j + \mathbf{QL}(\mathbf{r}-\mathbf{d})
\end{aligned} \tag{4.23}$$

Using the same considerations as before, the asymptotic stability is achieved if:

$$\boxed{\rho(\mathbf{Q}(\mathbf{I}-\mathbf{LP})) < 1} \tag{4.24}$$

where $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$.

A very useful tool for stability is to also consider the condition in the frequency domain design case. In this case the condition for the stability of the algorithm becomes:

$$\boxed{\left|1 - L\left(e^{j\omega}\right) G\left(e^{j\omega}\right)\right| < \left|Q^{-1}\left(e^{j\omega}\right)\right|, \quad \forall\omega} \tag{4.25}$$

where G is the transfer function of the plant $\mathbf{P}$ and in particular it is sufficient but not necessary.

This condition can be much more intuitive because it can be represented graphically by a Bode plot. This allows us to do Q-filter design more easily whenever necessary. It also allows

us to understand more easily what happens in case the condition is not verified. Assuming, for example, that:

$$\left|Q\left(e^{j\omega}\right)\left(1-L\left(e^{j\omega}\right)G\left(e^{j\omega}\right)\right)\right|>1 \tag{4.26}$$

meaning that we have a bode plot similar of the one represented in figure 4.4. That is, we will have some errors in frequency that will decay quickly, but others (when the magnitude is greater than 1), that will grow with the number of iterations of the algorithm.

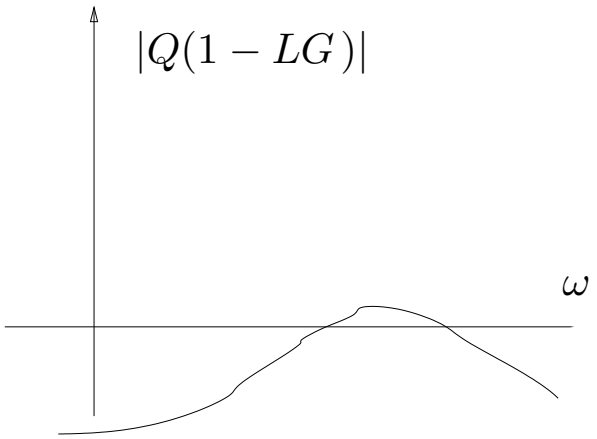

Figure 4.4: Example of frequency domain stability condition not verified

4.3.2 Monotonic Convergence Analysis

The system is monotonically convergent under a given norm if

$$\left\|\mathbf{e}_\infty-\mathbf{e}_{j+1}\right\|\leq\gamma\left\|\mathbf{e}_\infty-\mathbf{e}_j\right\| \tag{4.27}$$

where $\gamma\in[0,1)$ is the rate of convergence and $\mathbf{e}_\infty$ is the asymptotic error. At this point, using the error dynamics in eq. (4.20), and considering that at infinity $\mathbf{e}_\infty=\mathbf{e}_{j+1}=\mathbf{e}_j$ and $\mathbf{u}_\infty=\mathbf{u}_j$, we can obtain:

$$\begin{aligned}\mathbf{e}_\infty&=(\mathbf{I}-\mathbf{PQL})\mathbf{e}_\infty-\mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty\\&=(\mathbf{PQL})^{-1}\mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty\\&=(\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty\end{aligned}$$

$$\mathbf{e}_\infty = (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty \tag{4.28}$$

where we can see that, if the system is stable, if $\mathbf{Q}=\mathbf{I}$ then the asymptotic error $\mathbf{e}_\infty = 0$. Hence, $\mathbf{Q}=\mathbf{I}$ gives the best performance.

At this point, considering the whole vector sequence and using the results found in eq. (4.28) and in eq. (4.20), it is finally possible to obtain the description for γ in eq. (4.27) as follows:

$$\begin{aligned}
\mathbf{e}_\infty - \mathbf{e}_{j+1} =& (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty - (\mathbf{I}-\mathbf{PQL})\mathbf{e}_j - \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
=& (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty - (\mathbf{I}-\mathbf{PQL})\mathbf{e}_j - \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
&+ (\mathbf{I}-\mathbf{PQL})\mathbf{e}_\infty - (\mathbf{I}-\mathbf{PQL})\mathbf{e}_\infty \\
=& (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) - (\mathbf{I}-\mathbf{PQL})\mathbf{e}_\infty \\
&+ (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty - \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
=& (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) + (-\mathbf{I}+\mathbf{PQL})(\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty \\
&+ (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty - \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
=& (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) + (\mathbf{QL})^{-1}(\mathbf{I}-\mathbf{Q})(-\mathbf{I}+\mathbf{PQL}+\mathbf{I})\mathbf{u}_\infty \\
&- \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
=& (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) + \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_\infty - \mathbf{P}(\mathbf{I}-\mathbf{Q})\mathbf{u}_j \\
=& (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) + \mathbf{P}(\mathbf{I}-\mathbf{Q})\left(\mathbf{u}_\infty - \mathbf{u}_j\right)
\end{aligned} \tag{4.29}$$

Assuming once again that $\mathbf{Q}=\mathbf{I}$ for simplicity, we can neglect the second part of the previous equation, getting:

$$\mathbf{e}_\infty - \mathbf{e}_{j+1} = (\mathbf{I}-\mathbf{PQL})\left(\mathbf{e}_\infty - \mathbf{e}_j\right) \tag{4.30}$$

and assuming to consider the Euclidean norm

$$\left\|\mathbf{e}_\infty - \mathbf{e}_{j+1}\right\|_2 \leq \|\mathbf{I}-\mathbf{PQL}\|_2 \left\|\mathbf{e}_\infty - \mathbf{e}_j\right\|_2 \tag{4.31}$$

we can clearly see that γ is:

$$\gamma := \|\mathbf{I}-\mathbf{PQL}\|_2 \tag{4.32}$$

Hence, the ILC system is monotonically convergent if:

$$\begin{aligned}\gamma &:= \|\mathbf{I} - \mathbf{PQL}\|_2 < 1 \\ &= \bar{\sigma}(\mathbf{I} - \mathbf{PQL}) < 1\end{aligned}$$

$$\boxed{\gamma = \bar{\sigma}(\mathbf{I} - \mathbf{PQL}) < 1} \tag{4.33}$$

where $\bar{\sigma}(\cdot)$ is the maximum singular value.

It is important to note that the result for monotonic convergence obtained is independent of the desired reference trajectory.

5

ILC Implementation

In this chapter, the Iterative Learning Control algorithm introduced in the chapter 4 will be implemented in simulation using the MATLAB and Simulink environment. In particular, we will focus on the main design techniques of the **L** and **Q** filters introduced in the update input equations shown in the previous chapter. Each of these techniques will first be introduced from a theoretical point of view, and then the results obtained in simulation exploiting the mathematical model of the system will be shown. The goal will be to follow the reference in acceleration obtained in the chapter 3 following the excitation of the system through the previously generated road profile. Finally, a comparison of the result obtained with Iterative Learning Control versus more traditional controllers, proportional and feedforward control, will be shown.

5.1 MATLAB implementation of the ILC algorithm

The Iterative Learning Control (ILC) algorithm is implemented in MATLAB according to the pseudocode shown in the algorithm 5.1. It is executed cyclically a number of times defined by j_{max}, where j_{max} is the number of iterations.

It can also be seen that the last step is to calculate the performance of the current iteration. To evaluate the performance of the ILC algorithm, one of the most used performance index is

Algorithm 5.1 Iterative Learning Control

1: **for** $j \leftarrow 1$ to j_{max}
2: Compute new input $\mathbf{u}_j = \mathbf{Q}\left[\mathbf{u}_{j-1} + \mathbf{L}\mathbf{e}_{j-1}\right]$
3: Compute system response $\mathbf{y}_j = \mathbf{P}\mathbf{u}_j$
4: Add initial conditions $\mathbf{y}_j = \mathbf{y}_j + \mathbf{y}_0$
5: Compute error $\mathbf{e}_j = \mathbf{y}_d - \mathbf{y}_j$
6: Compute RMS of the iteration $\text{RMS}_i = \sqrt{\frac{1}{N}\mathbf{e}_j^2}$
7: **end for**

the root mean square (RMS) of tracking error, defined as:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{5.1}$$

where e is the tracking error between the reference signal and the current signal and N is the number of samples.

5.2 Verify fundamental assumptions

In order to implement the ILC algorithm, it is necessary to initially verify that we have the assumptions introduced in the previous chapter. Furthermore, the model of the system must be linear, time-invariant and asymptotically stable. Regarding the first two conditions, they turn out to be already verified based on the modeling of the system done in chapter 2. It therefore remains to verify that the system is asymptotically stable. Although we already had a clue about the stability of the system through model simulation using bump road profiles as input, to be certain we can study the eigenvalues location of the system. The eigenvalues of the system turn out to be:

$$\lambda_{1,2} = -14.2952 \pm 68.8743i \qquad \lambda_{3,4} = -0.9340 \pm 4.7651i \tag{5.2}$$

which are represented in figure 5.1. We note that they turn out to be all four in the left part of the plane, and consequently the system turns out to be asymptotically stable. Recall also that if it was not, the algorithm could still be applied by resorting to the addition of a feedback controller capable of stabilizing the system.

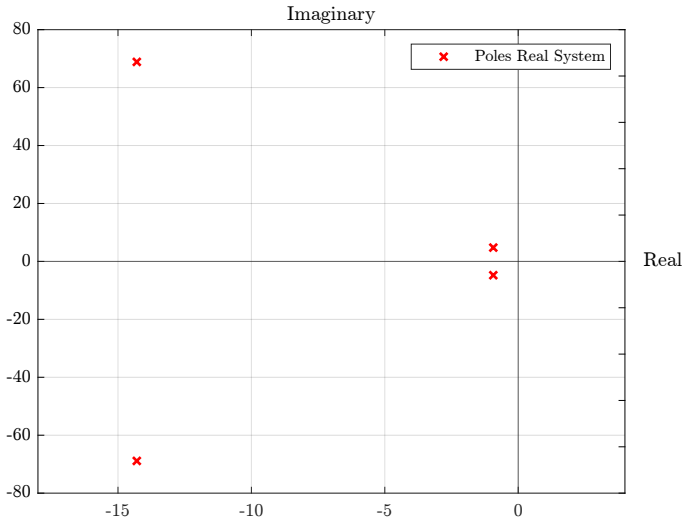

Figure 5.1: Representation of the poles of the continuous-time quarter car model in the complex plane

5.2.1 Discretization

Starting from the continuous time model obtained in chapter 2, we can use the MATLAB command `ss2tf` to obtain the representation of the system by transfer function. We then obtain the continuous time model:

$$G(s) = \frac{10625s^3 + 125000s^2 - 1.3102e^{-10}s - 1.0186e^{-09}}{s^4 + 30.4583s^3 + 5025s^2 + 10625s + 125000} \tag{5.3}$$

As we have seen, it is necessary for the ILC algorithm to obtain the system in discrete time. To do this, it is possible to use discretization by ZOH provided by the MATLAB command `c2d`. In this way we get the following discrete time model:

$$G(z) = \frac{89.5152z^3 - 258.3006z^2 + 248.0464z - 79.261}{z^4 - 3.3173z^3 + 4.3818z^2 - 2.8009z + 0.73743} \tag{5.4}$$

Defined the pseudocode implemented in MATLAB, the index for performance evaluation, and having verified that the fundamental assumptions hold, the next section will analyze the design methods for the two main parameters of the algorithm, the **L** matrix and the **Q** matrix.

5.3 Design of the learning matrix L

As in the case of the system matrix **P**, the learning matrix **L** must also be represented in the lifted domain as:

$$L(\mathrm{q}) = \sum_{k=0}^{\infty} l_k \mathrm{q}^{-k} \tag{5.5}$$

Considering a finite sequence of N samples, we obtain:

$$\begin{aligned} L(\mathrm{q}) &= \sum_{k=-N}^{N} l_k \mathrm{q}^{-k} \\ &= l_{-(N-1)}\mathrm{q}^{N-1} + \ldots + l_{-2}\mathrm{q}^{2} + l_{-1}\mathrm{q}^{1} + l_0 + l_1\mathrm{q}^{-1} + l_2\mathrm{q}^{-2} + \ldots + l_{N-1}\mathrm{q}^{-(N-1)} \end{aligned} \tag{5.6}$$

and then in matrix form:

$$\mathbf{L} = \begin{bmatrix} l_0 & l_{-1} & \cdots & l_{-(N-1)} \\ l_1 & l_0 & \cdots & l_{-(N-2)} \\ \vdots & \vdots & \ddots & \vdots \\ l_{N-1} & l_{N-2} & \cdots & l_0 \end{bmatrix} \tag{5.7}$$

considering the general case in which the learning function L is non-causal.

5.3.1 Heuristic Arimoto type

One of the simplest method for designing the learning filter $L(\mathrm{q})$ is the one proposed by Arimoto et al. [12], in which the matrix consists simply in a gain as follows:

$$L(\mathrm{q}) = \gamma \tag{5.8}$$

In this way, the general ILC input update becomes:

$$\boxed{u_{j+1}(k) = u_j(k) + \gamma e_j(k+1)} \tag{5.9}$$

At this point the previous update criterion in the lifted time domain is analyzed. Starting from the general learning control law:

$$\mathbf{u}_{j+1} = \mathbf{u}_j + \mathbf{L}\mathbf{e}_j$$

in which we assume $\mathbf{Q} = \mathbf{I}$, and considering this case in the lifted domain so that it becomes:

$$\mathbf{L} = \text{diag}(\gamma, \gamma, ..., \gamma) \tag{5.10}$$

or in matricial form:

$$\mathbf{L} = \begin{bmatrix} \gamma & 0 & \cdots & 0 \\ 0 & \gamma & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma \end{bmatrix} \tag{5.11}$$

one gets:

$$\begin{aligned} u_{j+1}(k) &= u_j(k) + \gamma e_j(k+1) \\ &= u_0(k) + \gamma \sum_{\ell=0}^{j} e_\ell(k+1) \end{aligned} \tag{5.12}$$

substituting $u_j(k)$ recursively. One notes from eq. (5.12) that the simplest form of linear iterative learning control is found to be a form of integral control running in repetitions. Consequently, the error corrective action continues to increase until the error goes away (or the process becomes unstable).

Stability

In order to study the stability for the particular case of the Arimoto type approach, we can consider a single input, single output system for simplicity, so that the equation of the plant in the lifted domain in the eq. (4.9) can be rewritten as follows:

$$\mathbf{P} = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix} \tag{5.13}$$

Assuming that the learning matrix $\mathbf{L}$ is a diagonal matrix whose components are given by a generic γ, considering the general formula for stability in eq. (4.24). Assuming we consider

$\mathbf{Q} = \mathbf{I}$ we will then have that $\mathbf{I} - \mathbf{LP}$ is a lower triangular matrix with all components along the diagonal equal to $1 - CB\gamma$, which for a lower triangular matrix will correspond to the eigenvalues.

As a result, the condition of stability:

$$\rho(\mathbf{Q}(\mathbf{I} - \mathbf{LP})) < 1 \tag{5.14}$$

where $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$, can be rewritten as:

$$\max_i |\lambda_i((\mathbf{Q}(\mathbf{I} - \mathbf{LP}))| < 1 \quad \forall i \tag{5.15}$$

and considering $\mathbf{Q} = \mathbf{I}$, it becomes in this case:

$$-1 < 1 - CB\gamma < 1 \tag{5.16}$$

or alternatively:

$$0 < CB\gamma < 2 \tag{5.17}$$

where CB is the response of the system at time step one for a unit pulse input at time step zero. Note that this condition is easily obtained since almost all systems have $CB \neq 0$, and particularly when the system is obtained by discretization from a continuous system, it will not normally be zero.

Particular attention, however, must be paid to the sign of the gain γ.

Study of sign of the controller gain

As we have just seen, to obtain stability we will have to guarantee the condition eq. (5.17).

Consequently we will have to use a positive sign of γ in case CB is itself positive, and a negative sign of γ in case CB is negative.

Normally, the sign of CB is known. For a typical feedback system, the response of the system to a positive step input starts in the positive direction, meaning that CB is positive.

However, a possible exception is in the case of a nonminimum phase system, that is, with at least one zero outside the unit disk. In this case in fact initially the system response goes first toward the opposite direction of the expected response (steady-state value) before moving toward the expected response. Non-minimum-phase systems can be due directly from the system

model, as in the case of the aircraft, or from the combination of model and feedback controller, as in the case of an inverted pendulum.

Consequently, the first step is to figure out whether our system is nonminimum phase, and one way to do this, as we have said, is to see the first step of the unit step response. The step response are shown in figure 5.2. We can see that for the model we are considering, the first step it is in the same direction as the step, consequently our system has no zeros out of the unit disk and so we will have to use a positive gain γ.

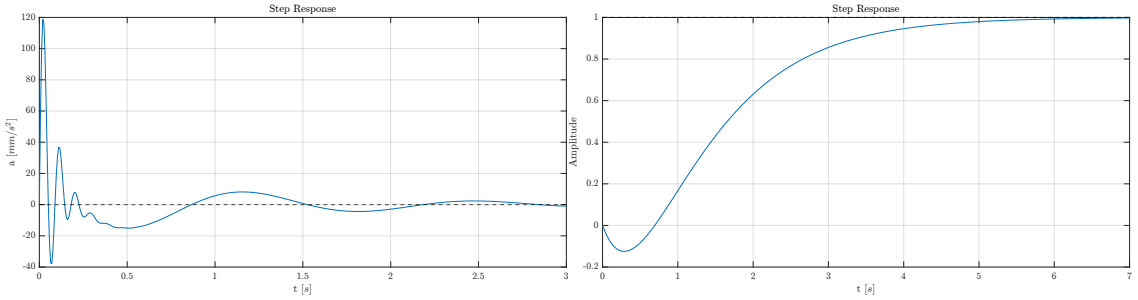

Figure 5.2: On the left: Step response for quarter car model. On the right: Step response for Non-minimum phase system

Once we assure ourselves that the system is nonminimum-phase, meaning that CB is positive, we can test the algorithm by varying the gain γ, making sure to satisfy condition in eq. (5.17).

P-type Simulation results

Before proceeding with the discussion of the results obtained by varying the gain γ, it should be noted that all simulations performed in this chapter use the first 3 seconds of the acceleration response obtained in figure 3.6 in chapter 3 as the reference trajectory, which is shown in figure 5.3. In addition, the initial conditions of the system at the beginning of the algorithm and at each new iteration are always zero. (Recall that one of the fundamental assumptions was that the initial resetting conditions at each iteration must always be the same.)

The results are represented in figure 5.4. It is possible to see that in all the figures we have an initial decrease in the error RMS and then a fast and large increase. This phenomenon is called Learning Transient, and it was already observed during the first implementations of the ILC algorithm with only proportional gain by Longman in [24].

He decided, after failing the experiment on a robot due to very high component oscillations, to try the experiment in simulation. In that case the RMS of the tracking error decreased from

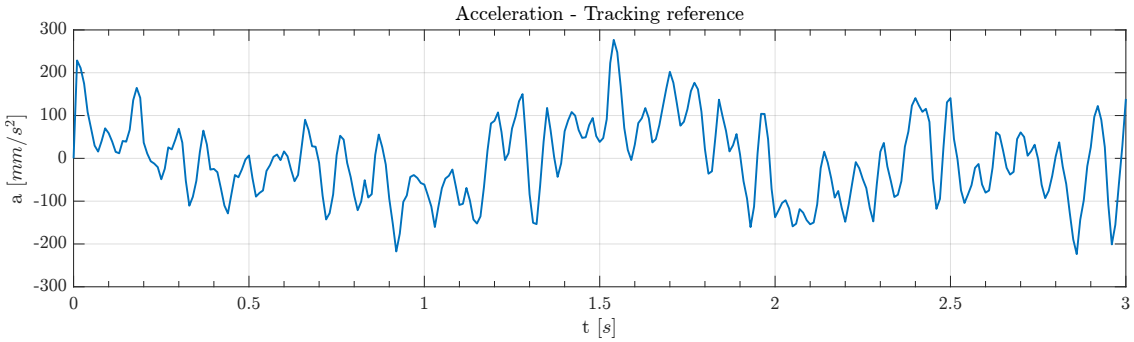

Figure 5.3: Reference trajectory of the ILC algorithm

repetition 0 until repetition 7, and then increased until it reached a maximum of 1.1991×10^{51} at repetition 62123. Thereafter the error decreased and reached to numerical zero.

Assuming for simplicity a single input, single output case, if one considers a diagonal learning matrix $\mathbf{L}$ with a generic gain γ and again considering the dynamics of the system eq. (4.9) in lower triangular Toeplitz form:

$$\mathbf{P} = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix} \tag{5.18}$$

and error evolution as found in eq. (4.21):

$$\mathbf{e}_j = (\mathbf{I} - \mathbf{PL})\mathbf{e}_{j-1} \tag{5.19}$$

one gets:

$$\mathbf{e}_j = \begin{bmatrix} 1 - CB & 0 & \cdots & 0 \\ -CAB & 1 - CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -CA^{p-1}B & -CA^{p-2}B & \cdots & 1 - CB \end{bmatrix} \mathbf{e}_{j-1} \tag{5.20}$$

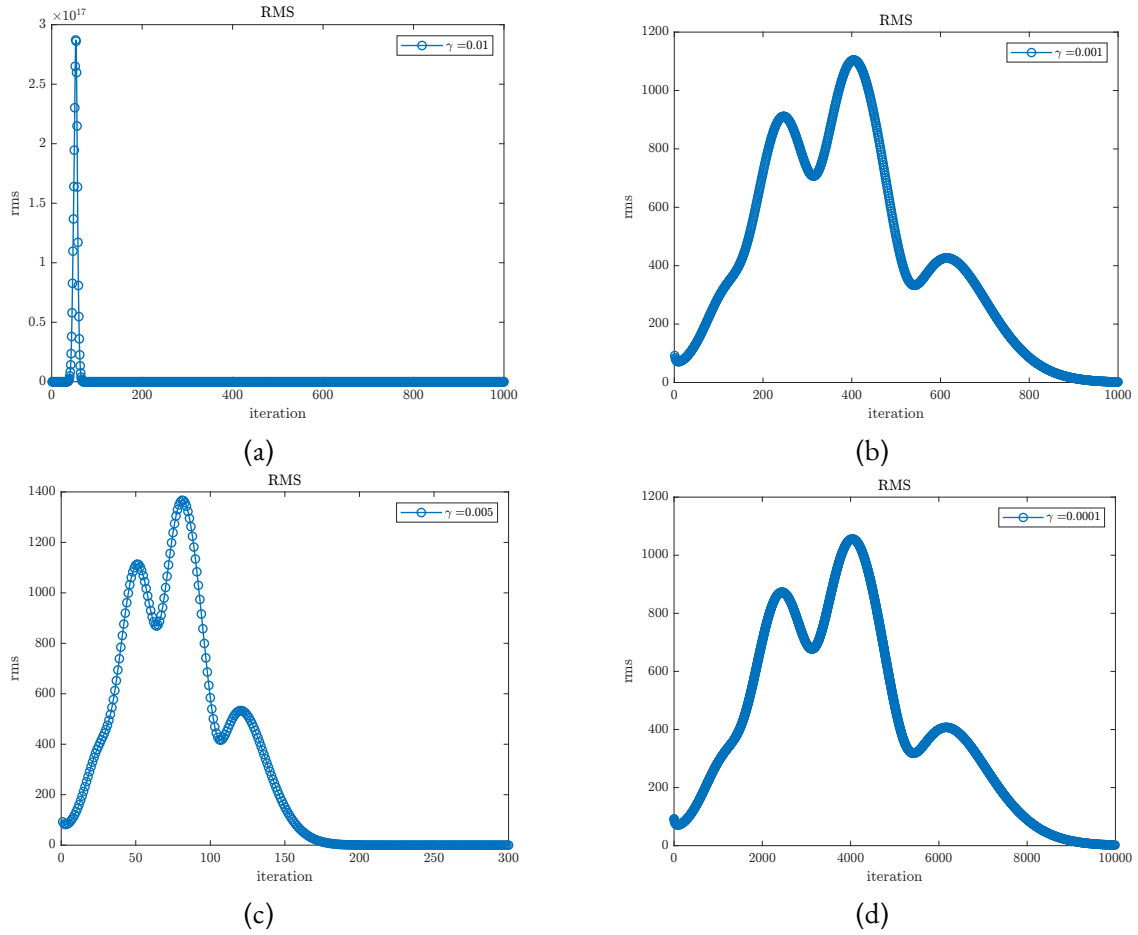

Figure 5.4: RMS using ILC Arimoto approach with: (a) $\gamma = 0.01$ (b) $\gamma = 0.001$ (c) $\gamma = 0.0005$ (d) $\gamma = 0.0001$

By explicitly writing steps for errors at a generic iteration j:

$$\begin{cases} e_j(1) = & (1 - CB\gamma)e_{j-1}(1) \\ e_j(2) = & (1 - CB\gamma)e_{j-1}(2) - CAB\gamma e_{j-1}(1) \\ e_j(3) = & (1 - CB\gamma)e_{j-1}(3) - CAB\gamma e_{j-1}(2) - CA^2B\gamma e_{j-1}(1) \\ \vdots & \\ e_j(p) = & (1 - CB\gamma)e_{j-1}(p) - CAB\gamma e_{j-1}(p-1) - \ldots - CA^{p-1}B\gamma e_{j-1}(1) \end{cases} \tag{5.21}$$

For the first step, one keeps multiplying by $(1 - CB\gamma)$ every repetition, and if this number is less than one in magnitude, as seen in eq. (4.22), then the magnitude of this error decays mono-

tonically with repetitions. Once $e_{j-1}(1)$ has become essentially zero, then the second equation in eq. (5.21) drops its last term, and from then on $e_{j-1}(2)$ will decay monotonically, and so on.

As a result, the convergence begins at the first time step and advances through the step process step by step in a wave.

Consider now the generic error at step p in the equation eq. (5.21)

$$e_j(p) = (1 - CB\gamma)e_{j-1}(p) - CAB\gamma e_{j-1}(p-1) - \ldots - CA^{p-1}B\gamma e_{j-1}(1) \tag{5.22}$$

This can easily be a very large number of terms with a quick sample time. The behavior may deteriorate if the sample rate is increased. There would be 250 terms at a sample time of 1000 Hz and a settling time of 0.5 seconds.

Therefore, it is easy to imagine that, while waiting for the wave of convergence to arrive, the sum of all of these terms would result in a very large error at the end of the trajectory.

Solutions to High Learning Transients

In this case, the first possible solution is to decrease the length of the tracking period. The second solution is to decrease the sample time T_s of the system. Alternatively, one can lower the learning filter gain $L(q)$ until the error and consequently the corrective input falls below the limit allowed by the actuators. However, this will involve more iterations to achieve convergence. The result of each proposed solution is shown in figure 5.5.

We can therefore conclude that in general with a model free approach we will always have a trade-off between the speed of convergence and the applicability of the algorithm.

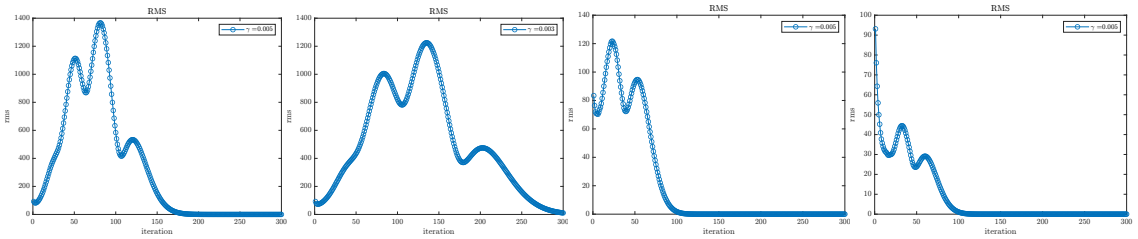

Figure 5.5: Different approaches for dealing with high learning transient. Starting from the left: Original RMS result using P-type with $\gamma = 0.005$, lowering the gain of the learning filter L ($\gamma = 0.003$), lowering the time duration of reference ($T = 1.5s$), lowering the sample time of the signals and the algorithm ($Ts = 0.02$)

In conclusion, it could be observed that with the P-type approach, whose main advantage is its simplicity, stability is easily achieved, but it is difficult to implement on a real system because

Type	Max RMS	Percentage Change
Arimoto type with $\gamma = 0.005$	1366.9	
Arimoto type with $\gamma = 0.003$	1225.0	-10.38 %
Arimoto type with T = 1.5s	121.8	-91.09 %
Arimoto type with Ts = 0.02s	93.1	-93.19 %

Table 5.1: Numerical performance for each of the different approaches for dealing with high learning transient

of the high learning transient issue.

Finally, figure 5.6 shows the tracking error at each iteration using the ILC P-type approach with $l_0 = 0.001$. It can be seen particularly between iteration 200 and 500 that the error increases by propagating like a wave. As a result, the convergence begins at the first time step and advances gradually through the process in a wave.

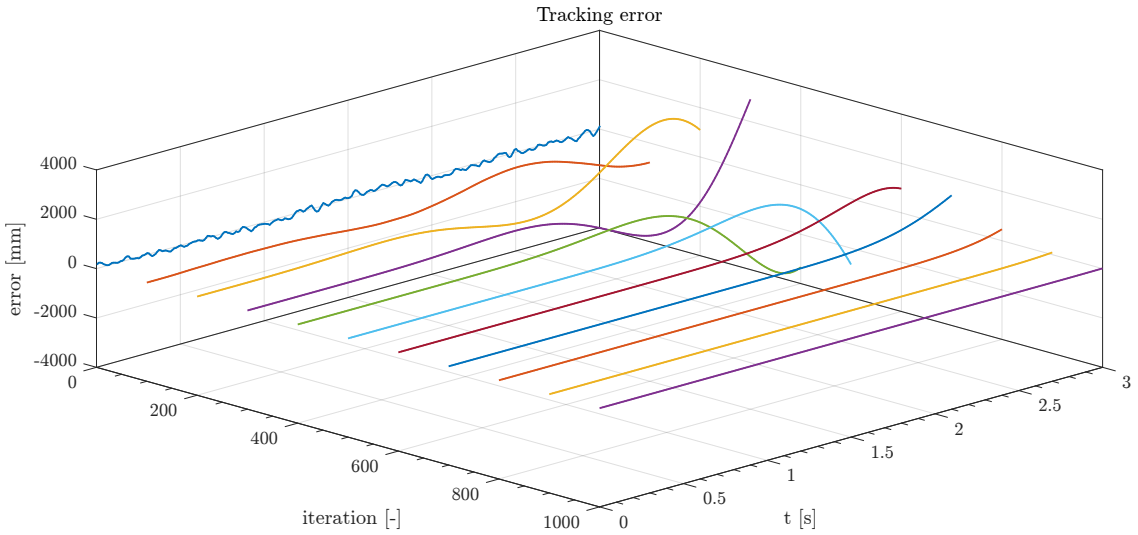

Figure 5.6: Tracking error every 100 iteration using the ILC P-type approach with $l_0 = 0.001$

5.3.2 PD type

Building on Arimoto's proposal, subsequent model free methods for the design of the $L(q)$ learning matrix were proposed. In addition to Arimoto's algorithm considered P-type, further heuristic approaches can be found in the literature: the PD-type and the PID-type.

In the former case the learning function is:

$$L_p(\mathrm{q}) = K_p \qquad L_d(\mathrm{q}) = K_d \tag{5.23}$$

where $L_p(\mathrm{q})$ and $L_d(\mathrm{q})$ are the learning filter for proportional and derivative part, respectively.

In this way the ILC input update becomes:

$$u_{j+1}(k) = u_j(k) + K_p e_j(k) + K_d \dot{e}_j(k) \tag{5.24}$$

By exploiting the formula of the derivative as a limit of the incremental ratio

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{5.25}$$

where $\Delta x = 1$ in our case, one gets:

$$\boxed{u_{j+1}(k) = u_j(k) + K_p e_j(k) + K_d \left[e_j(k+1) - e_j(k) \right]} \tag{5.26}$$

In this case the parameters used for the ILC of PD-type are:

$$K_p = 0.001; \qquad K_d = 0.0004; \tag{5.27}$$

while we keep using a gain $K_p = 0.001$ for the ILC of P-type.

The results obtained are shown in the figure 5.7. One can see that using the same P-type controller gain, the addition of the derivative term allows us to have slightly faster convergence for the same learning transient.

The proportional and derivative gains in the PD controller are comparable to the roles of learning gains K_p and K_d in eq. (5.24). Although a higher gain can speed up error convergence, there are severe error oscillations within a certain range, and additionally, they may cause instability. On the other hand, greater iterations will be necessary to meet the error requirement for smaller learning gains.

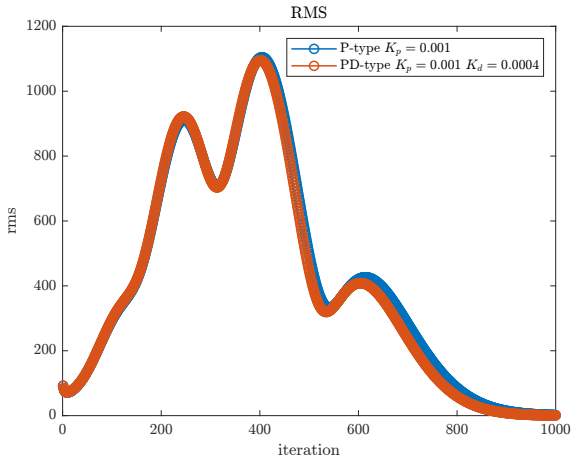

Figure 5.7: Comparison of the RMS of the tracking error at each iteration between ILC P-type and ILC PD-type

5.3.3 PID type

As the last model free method considered, the ILC of PID-type is based on the following learning functions:

$$L_p(\mathrm{q}) = K_p \qquad L_d(\mathrm{q}) = K_d \qquad L_i(\mathrm{q}) = K_i \tag{5.28}$$

where $L_p(\mathrm{q}), L_d(\mathrm{q})$ and $L_i(\mathrm{q})$ are the learning filter for proportional, derivative and integral part, respectively.

In this way the ILC input update becomes:

$$\boxed{u_{j+1}(k) = Q(q)\left[u_j(k) + K_p e_j(k) + K_d \dot{e}_j(k) + K_i \int e_j(k)\right]} \tag{5.29}$$

However, this algorithm is almost never used, since the addition of the integrative part has no considerable effect since as we have already seen in eq. (5.12) the ILC already has a form of integral action of the error that accumulates with each iteration.

5.4 Inverse Model Approach

The idea behind the design by inverse model approach is to use a priori knowledge of our system to build a model that will then be inverted to produce the input that generated the desired

output. Consequently, this approach requires accurate a priori knowledge of the system. The learning filter's design quality determines the ILC performance in terms of convergence rate and converged error.

The learning function $L(\mathrm{q})$ will in this case

$$L(\mathrm{q}) = \hat{P}^{-1}(\mathrm{q}) \tag{5.30}$$

where $\hat{P}$ is the estimate of the true model P. Accordingly, by substituting eq. (5.30) into eq. (4.3) the update input equation becomes:

$$\boxed{u_{j+1}(k) = u_j(k) + \hat{P}^{-1}(\mathrm{q})e_j(k+1)} \tag{5.31}$$

The optimal learning filter is the one that matches the system's inverse, in particular in the case $\mathbf{L}$ coincides with the real inverse of the model, we will have perfect tracking in just one iteration.

As already discussed in the case of the Arimoto type, even with this approach special care must be taken if the model turns out to be nonminimum phase.

In fact since it will have zeros in the right-hand side, upon inversion of the model they will become poles that will make the filter unstable.

In the latter case the literature proposes to use the stable inversion method, discussed in the next section, so as to obtain a stable filter.

5.4.1 Stable inversion

In case the system is nonminimum phase, the inversion will be unstable. However, it is possible to take advantage of the fact that ILC allows acausal filtering to perform a stable inversion as proposed in [25].

In this case the nonminimum phase system $\mathbf{G}(z)$ is factorized as:

$$\mathbf{G}(z) = \mathbf{G}_+(z)\mathbf{G}_-(z)$$

where $\mathbf{G}_+(z)$ includes the minimum phase zeros and $\mathbf{G}_-(z)$ includes the non-minimum phase zeros.

The inversion is then achieved by initially filtering the signal through the causal stable filter

$$\mathbf{w}(t) = \mathbf{G}_+^{-1}\mathbf{y}(t)$$

and then filtering the result by the acausal filter

$$\mathbf{u}(t) = \mathbf{G}_{-}^{-1}\mathbf{w}(t)$$

In particular, the last operation can be achieved by filtering the reversed sequence $\mathbf{p}(t) = \mathbf{w}(N-t)$ through the stable causal filter $\mathbf{v}(t) = \mathbf{G}_{-}^{-1}\mathbf{p}(t)$ and finally reversing the result again $\mathbf{u}(t) = \mathbf{v}(N-t)$.

5.4.2 Model Inverse

We now implement the model-based inversion approach in MATLAB and Simulink and analyze the results later. As noted earlier, the quarter car system model is minimum phase, so it is not necessary to use the stable inversion method.

We assume that the starting model is known at minus parameter value. We also assume that we have three different discrete-time models:

- the Real model
- the Estimated model
- the Limit model

The models with their parameter values are listed in the table 5.2, in particular the Real model have the same already used parameter values in table B.1, while the Estimated model have the value of each parameter with a random error of 10% with respect to the Real model parameters.

Parameter	**Symbol**	**Value**		
		Real	Estimated	Limit
Sprung Mass	M	400 Kg	370 Kg	430 Kg
Unsprung Mass	m	30 Kg	28 Kg	25 Kg
Suspension Spring Stiffness	k_s	10000 N/m	10750 N/m	13000 N/m
Suspension Damper Coefficient	c_s	850 N/(m/s)	914 N/(m/s)	650 N/(m/s)
Tyre Spring Stiffness	k_t	140000 N/m	150500 N/m	190000 N/m

Table 5.2: Parameter values for the three different quarter car models tested

For the real model we use a learning matrix $\mathbf{L} = \mathbf{P}^{-1}$, thus assuming that we know the system exactly, while for the Estimated and Limit models we use a learning matrix $\mathbf{L} = l_0 \hat{\mathbf{P}}^{-1}$ respectively for the two estimated models, where $0 < l_0 < 1$ is a gain that is intended to give greater robustness to the inverse model approach in the case of uncertainty.

Before implementing the models in MATLAB and Simulink to perform the simulation, we analyze the results of the stability theorem in eq. (4.24) and the monotonic convergence in eq. (4.33) represented in table 5.3.

Theorem	**Condition**	**Value**		
		Real	Estimated	Limit
Stability	$\rho(\mathbf{Q}(\mathbf{I} - \mathbf{LP})) < 1$	0	0.3472	0.2688
Monotonic convergence	$\bar{\sigma}(\mathbf{I} - \mathbf{PQL}) < 1$	$2.5994e^{-13}$	0.4453	1.0705

Table 5.3: Stability and monotonic convergence values for the three different quarter car models tested

We can observe that the Estimated model ensures both stability and monotonic convergence, while the Limit model ensures stability but not monotonic convergence.

Finally, we assume that the starting real model has a pure time delay of 0.1s due to the actuators and analyze the case where we assume that it is not taken into account in estimating the model for the inverse basis approach. We call this new analysis by the name Limit case with Delay.

We then use the Limit case model again in the inverse for the learning filter, which in this case will have in addition to the parameters a new incorrectly modeled component, the Delay. Recall also that for any repetitive-type control, such as the Iterative Learning Control, the delay is a critical part. Following discretization, the intrinsic delay due to implementation, which we can assume to be equal to a period k will therefore be added to the already present time delay due to the relative degree of the system equal to 1. Using the theorem for stability eq. (4.24) we obtain in this case that the maximum singular value is not below 1, consequently we expect that the algorithm will be unstable. Moreover since the algorithm will turn out to be unstable, we will not have monotonic convergence either.

Finally, in figure 5.9 the graph related to the stability theorem in the frequency domain is shown, recalling that the equation for stability is

$$\left|1 - L\left(e^{j\omega}\right) G\left(e^{j\omega}\right)\right| < \left|Q^{-1}\left(e^{j\omega}\right)\right|, \quad \forall\omega \tag{5.32}$$

and turns out to be a sufficient but not necessary condition.

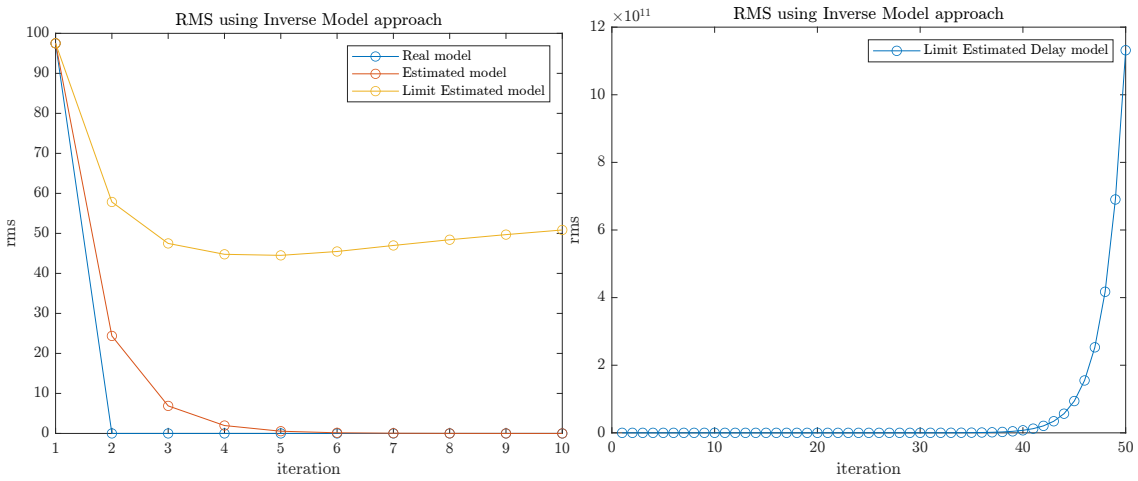

Figure 5.8: On the left: comparison of the RMS at each iteration for the three different quarter car models tested. On the right: RMS at each iteration for the Limit Estimated model with Delay

In particular in the figure it is represented $1 - LG$ for each of the models from a minimum frequency $f_{min} = 10^{-2}$ Hz up to the maximum Nyquist frequency $f_{max} = \frac{2\pi}{Ts}$ since above that there is no information being discrete-time models.

We observe that the Estimated model, being for all frequencies below the stability boundary, verifies the stability condition.

As for the Limit model, it is slightly above the stability boundary at frequencies between 70 Hz and 80 Hz. However, as seen above, we still get stability being an unnecessary condition.

In contrast, the Limit model with delay turns out to be well above the stability boundary for a greater range of frequencies between . This results, as seen above, in the instability of the ILC algorithm with the inverse model approach.

To deal with the uncertainties, two different approaches, Early Stopping and filter design $\mathbf{Q}$, will be introduced in the next two sections.

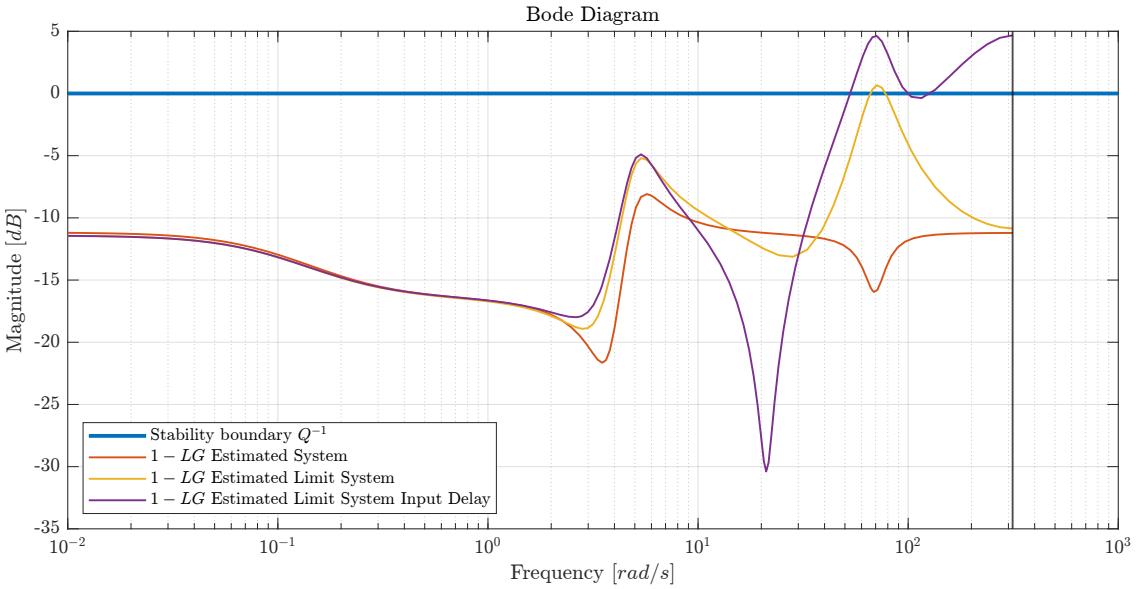

Figure 5.9: Frequency domain representation of the stability theorem

5.4.3 Dealing with Uncertainties with Inverse Model approach

We saw from the plot in figure 5.8 how the Limit model with delay had an error RMS that diverged, and in particular it grew with time.

The simplest approach to achieve robustness to uncertainties, as we mentioned in the previous section, is to exploit the l_0 gain, so that the inversion using the eq. (5.30) becomes

$$L(q) = l_0 \hat{P}^{-1}(q) \tag{5.33}$$

Using again the Limit case model with delay with inverse model approach but with a gain $l_0 = 0.005$ we get the result shown in the figure 5.10. In this case, although using the stability theorem we find that the algorithm still turns out to be unstable, in this case we will no longer have monotonic growth, but initially the RMS decreases.

In this case a widely used method is the so-called Early Stopping.

Early stopping

A widely used solution to this problem is to introduce the early stopping procedure. The starting algorithm 5.1 will then be modified as follows: In this way the algorithm will stop before diverging to the first minimal value of RMS.

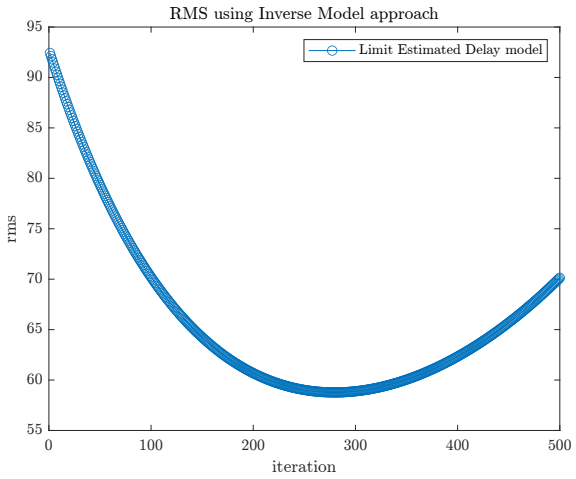

Figure 5.10: RMS at each iteration for the Limit Estimated model with Delay with $l_0 = 0.005$

Algorithm 5.2 Iterative Learning Control with Early Stopping

1: **for** $j \leftarrow 1$ to j_{max}
2: Compute new input $\mathbf{u}_j = \mathbf{Q}(q)\left[\mathbf{u}_{j-1} + \mathbf{L}(q)\mathbf{e}_{j-1}\right]$
3: Compute system response $\mathbf{y}_j = \mathbf{P}\mathbf{u}_j$
4: Add initial conditions $\mathbf{y}_j = \mathbf{y}_j + \mathbf{y}_0$
5: Compute error $\mathbf{e}_j = \mathbf{y}_d - \mathbf{y}_j$
6: Compute RMS of the iteration $\text{RMS}_i = \sqrt{\frac{1}{N}\mathbf{e}_j^2}$
7: **if** $\text{RMS}_i > \text{RMS}_{i-1}$
8: stop the algorithm
9: **end if**
10: **end for**

The main disadvantage of this solution is that the algorithm will generally stop when a relative minimum is present, which does not always coincide with the absolute minimum.

5.4.4 Design of the Q-filter

To give more robustness to our algorithm in the presence of modeling uncertainties, we can exploit the matrix $\mathbf{Q}$ from the formula eq. (4.3) that we have so far assumed as $\mathbf{Q} = \mathbf{I}$.

As seen above for the matrix $\mathbf{P}$ and $\mathbf{L}$, we consider

$$Q(q) = \sum_{k=0}^{\infty} q_k q^{-k} \tag{5.34}$$

Considering a finite sequence of N samples, we obtain:

$$\begin{aligned} Q(\mathrm{q}) &= \sum_{k=-N}^{N} q_k \mathrm{q}^{-k} \\ &= q_{-(N-1)}\mathrm{q}^{N-1} + \ldots + q_{-2}\mathrm{q}^{2} + q_{-1}\mathrm{q}^{1} + q_0 + q_1\mathrm{q}^{-1} + q_2\mathrm{q}^{-2} + \ldots + q_{N-1}\mathrm{q}^{-(N-1)} \end{aligned} \tag{5.35}$$

In this way it is possible to obtain a noncausal filter, a necessary requirement for the implementation of the robustness Q filter as a zero-phase lag filter.

A possible implementation of a zero-phase filter is through the MATLAB command `filtfilt`. As in the case of the system matrix $\mathbf{P}$ and the learning filter $\mathbf{L}$, we need an implementation of the Q-filter as a matrix in the lifted domain in order to analyze the stability theorem and the monotonic convergence theorem.

A possible implementation of a Q-filter in the lifted domain representation is described by Wallen et al. in [26] where once the causal filter $\bar{Q}$ is implemented, the forward-backward filtering technique is used. Considering a finite sequence of N samples, we now obtain the causal filter:

$$\begin{aligned} \bar{Q}(\mathrm{q}) &= \sum_{k=1}^{N} \bar{q}_k \mathrm{q}^{-k} \\ &= \bar{q}_0 + \bar{q}_1\mathrm{q}^{-1} + \bar{q}_2\mathrm{q}^{-2} + \ldots + \bar{q}_{N-1}\mathrm{q}^{-(N-1)} \end{aligned} \tag{5.36}$$

which rewriting in matrix form becomes:

$$\overline{\mathbf{Q}} = \begin{bmatrix} \bar{q}_0 & 0 & \cdots & 0 \\ \bar{q}_1 & \bar{q}_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{q}_{N-1} & \bar{q}_{N-2} & \cdots & 0 \end{bmatrix} \tag{5.37}$$

Filtering a generic signal vector x of length N defines as:

$$x = (x(0), \ldots, x(N-1))^T \tag{5.38}$$

can be interpreted as a matrix-vector multiplication

$$x_{ff} = \mathbf{Q}x \tag{5.39}$$

where x_f is the filtered signal vector and $\mathbf{Q}$ is the generic robustness filter implemented as a zero-phase filter using the procedure described in the algorithm 5.3.

Algorithm 5.3 Zero phase filtering

1: Filter the signal vector through $\overline{\mathbf{Q}}$:	$x_f = \overline{\mathbf{Q}}x$
2: Reverse the order of the filtered data points:	$\tilde{x}_f = \text{flip}(x_f)$
3: Filter the data points once more through: $\overline{Q}$:	$\tilde{x}_{ff} = \overline{\mathbf{Q}}\tilde{x}_f$
4: Reverse the order of the data points again:	$x_{ff} = \text{flip}(\tilde{x}_{ff})$

By taking advantage of matrix operations we can get the equivalent in one formula:

$$\begin{aligned} x_{ff} =& \text{flip}(\tilde{x}_{ff}) \\ =& \text{flip}(\overline{\mathbf{Q}}\tilde{x}_f) \\ =& \text{flip}(\overline{\mathbf{Q}}(\text{flip}(x_f)) \\ =& \text{flip}(\overline{\mathbf{Q}}(\text{flip}(\overline{\mathbf{Q}}x)) \\ =& \text{flip}(\overline{\mathbf{Q}}(\text{flip}(\overline{\mathbf{Q}}))x \\ =& \overline{\mathbf{Q}}^T\overline{\mathbf{Q}}x \end{aligned} \tag{5.40}$$

We then obtain the zero-phase lag robustness filter of noncausal type using the formula:

$$\boxed{\mathbf{Q} = \overline{\mathbf{Q}}^T\overline{\mathbf{Q}}} \tag{5.41}$$

An aspect which has been neglected during the implementation of the zero-phase lag filter is the handling of boundary effects. However, for further details, the paper [26] proposes two alternative methods for handling boundary effects, which consist in extend both the signal vector x and the matrix $\mathbf{Q}$ at the beginning and at the end, in order to better reproduce the MATLAB command `filtfilt` which implements by default a similar technique.

5.4.5 Get stability

Once we have described how to get $Q(\mathrm{q})$ as a zero-phase lag filter, the next step is to figure out how to use it to guarantee more robustness to the model uncertainties or as in our case, to guarantee the stability of the algorithm. To do this, the robustness filter $Q(\mathrm{q})$ is implemented as a low pass filter, since, as we have already observed in figure 5.9, the uncertainties in the model fall into stability problems affecting high frequencies. Once the type of flow pass filter to be

used has been chosen, which in our case fell to a Butterworth filter, it is necessary to determine the order and cut-off frequency such that stability is guaranteed.

Specifically, by exploiting the stability theorem for the frequency domain in order to have a graphical representation and the trial and error technique we can find the order and the cut-off frequency of the filter optimal for our particular case.

Finally, a first-order low pass filter with a cut-off frequency of 50 Hz was used in our case. Specifically, the MATLAB command `butter` was used, which produces the time domain filter represented in figure 5.11 and which is subsequently discretized using the ZOH method.

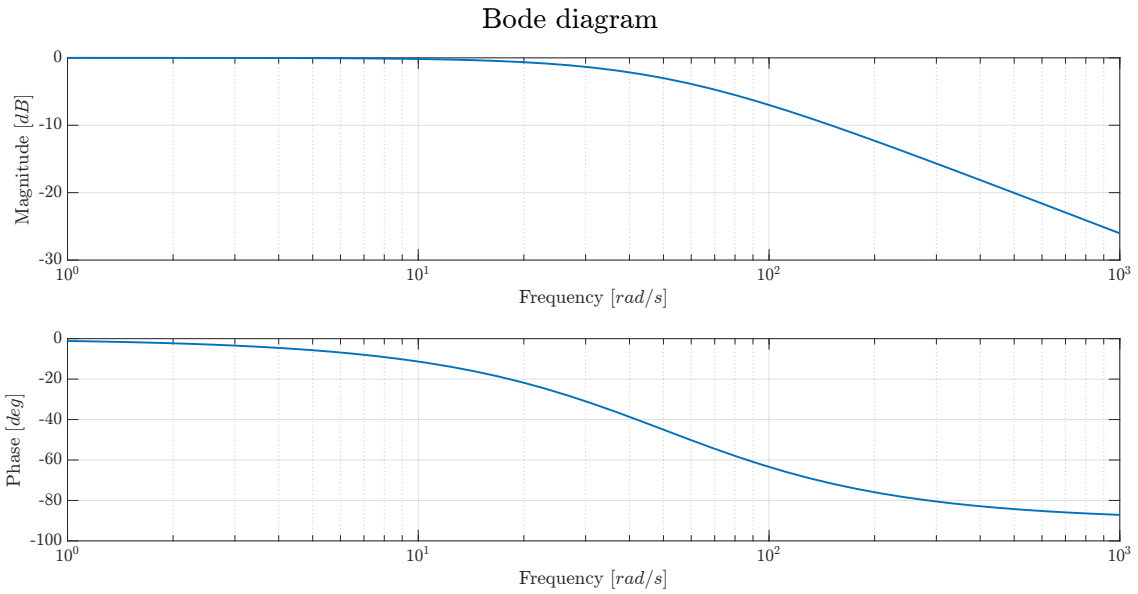

Figure 5.11: Bode plot of the continuous Butterworth filter

Thus by plotting the stability theorem again in frequency in figure 5.12 using the robustness Q-filter just obtained, we can see that the stability boundary in high frequency is broadened and this allows the uncertainties in the model to be included.

Using instead the stability theorem for the lifted time domain representation we obtain the results represented in table 5.4.

Thus it is possible to conclude that in this new case the algorithm will be stable.

In fact doing the simulation of the ILC algorithm based on Inverse model approach again, using the Limit Case delay model and the robustness filter, we obtain the RMS of the error shown in figure 5.13. We can see that, after a rapid decrease and some subsequent initial oscillations, the RMS of the error stabilizes at the value of 55 around 35 iterations. Consequently, we can see that, as expected, we do not obtain perfect tracking, since a Q-filter $\mathbf{Q} \neq \mathbf{I}$ is used.

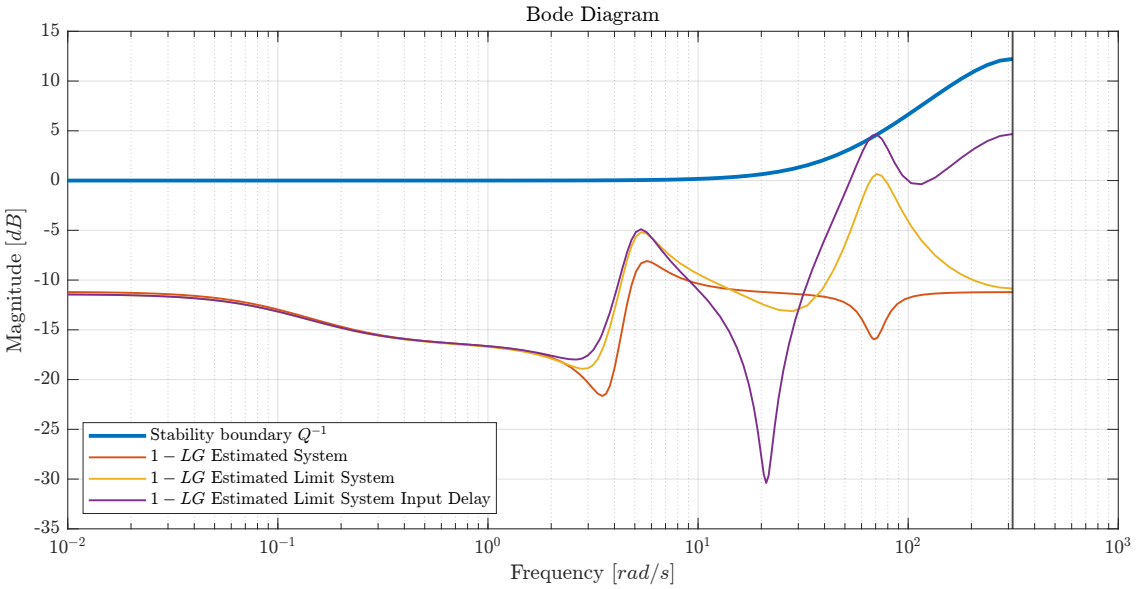

Figure 5.12: Frequency domain representation of the stability theorem using the generated Q filter

Theorem	**Condition**	**Value**
		Limit with Delay
Stability	$\rho(\mathbf{Q}(\mathbf{I} - \mathbf{LP})) < 1$	-0.4236 + 0.1163i
Monotonic convergence	$\bar{\sigma}(\mathbf{I} - \mathbf{PQL}) < 1$	10.7487

Table 5.4: Stability and monotonic convergence values for the Limit Estimated model with Delay using the Q-filter

However, the final RMS value is still quite high and may be unacceptable for certain applications.

Moreover, it can be guessed from the results that most of the uncertainties in modeling a system appear in high frequency. The RMS results are quite high because to remove the uncertainties, the **Q** matrix is filtering the input of the ILC. However, the reference has high dynamics, and the filtering does not allow it to follow the reference perfectly.

5.4.6 Robustness vs Performance trade-off

We saw in the previous section how the implemented Q robustness filter ensured us the stability of the algorithm, however the final performance generated turned out to be poor. We have already seen during the proof of the monotonic convergence theorem in eq. (4.28), that the

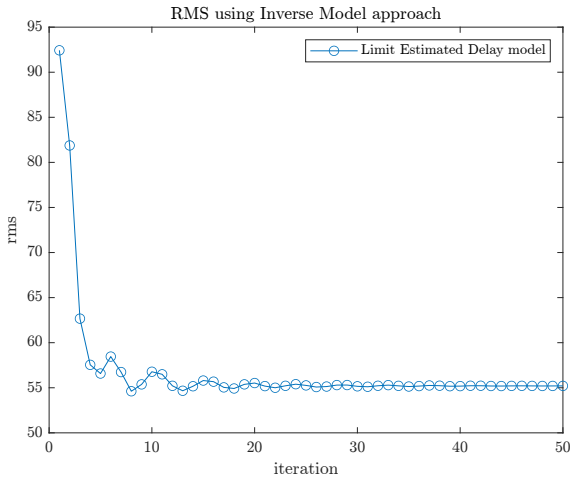

Figure 5.13: RMS using inverse model approach and Q filter

best performance are obtained using a Q-filter $\mathbf{Q} = \mathbf{I}$.

In this new section, the effect of the $Q(q)$ robustness filter applied to the ILC algorithm will be analyzed in more detail. Consider again the implementation of the algorithm by Inverse model approach and using the Estimated model with the use of a Q-filter $\mathbf{Q} = \mathbf{I}$ and three robustness Q filters of zero-phase lag low pass filter type of the second order and with cut-off frequencies of 10 Hz, 100 Hz and 1000 Hz respectively.

The RMS result of the error for each case is shown in figure 5.14. We can notice that in all four cases the stability of the algorithm is obtained. However It is possible to observe that as the cutoff frequency of the low pass filter decreases we get less performance.

From the results obtained, it can be guessed that as filtering by $\mathbf{Q}$ matrix increases to achieve greater robustness, performance decreases. Particularly in our case, the decrease is very evident because the dynamics of the system is high. Therefore, it can be concluded that the less the model is known, the lower the dynamics of the system will have to be chased in order not to lose too much performance, and vice versa.

Hence it is possible to conclude that there is a trade-off between robustness and performance using the ILC algorithm with the Q-filter.

5.5 Repetitive and Non-repetitive disturbances

One of the main problems when dealing with control of a system is the presence of disturbances. In fact, the convergence condition may not be met due to disturbances such as model

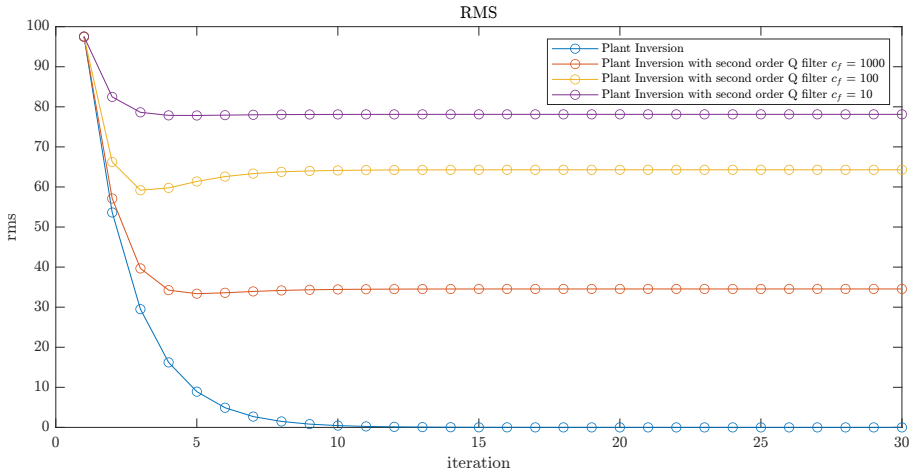

Figure 5.14: Comparison of RMS performance using different types of Q-filter

uncertainties, measurement errors, dynamic/external interactions, actuation delays, or faults.

In this section, the effect of repetitive and non-repetitive disturbances in the Iterative Learning Control algorithm will be analyzed. The first step is to define a disturbance. In [27] the authors distinguish disturbances based on their dependency on the state of the system or on time and as repetitive or non-repetitive based at their occurrences with respect to the iterations. Thus they classify repetitive disturbances as state-repetitive disturbances and time-repetitive disturbances, and non-repetitive disturbances as state-non-repetitive disturbances and time-non-repetitive disturbances.

It also defines state-repetitive disturbances as disturbances that represent an unmodeled part in the dynamics of the system, such as an unmodeled gravity force vector in the dynamics of a robot, while the system nominal model's additive uncertainties can be modeled by time-repetitive disturbances. It is noteworthy to mention that repetitive disturbances occur during every iteration of the entire process.

In contrast, the interaction of a robot with its surroundings or actuator malfunctions or delays might result in state non-repetitive disturbances, while time non-repetitive disturbances are disturbances that have no relation to the state of the system, such as measurements noise. It is important to note that throughout the entire learning process, non-repetitive disturbances only happen for a small number of iterations (or vary with each iteration).

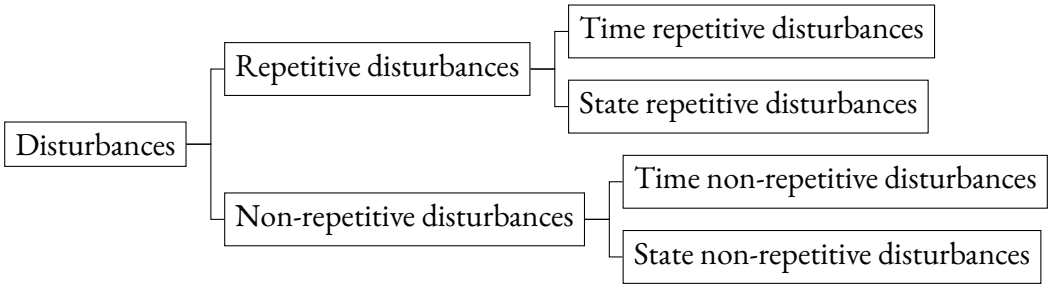

Figure 5.15: Classification of disturbances

5.5.1 Non-repetitive disturbances and Noise

As a first analysis, we assume that we have non repetitive disturbances, due to noise in measurements with the accelerometers, and consequently of the type of time-non repetitive disturbances.

The disturbance is obtained by randomly adding or subtracting to each value obtained after simulation at each iteration according to the formula

$$\mathbf{Y}_j^{\text{noise}} = \mathbf{Y}_j \pm \mathbf{d}_j \tag{5.42}$$

where $\mathbf{d}_j$ is the disturbance at iteration j, which coincides with a value between a maximum e_{max} of the value of $\mathbf{Y}_j$ and a minimum e_{max} of the value of $\mathbf{Y}_j$.

Specifically, in the first test we consider a maximum random error of 10% with respect to the actual value, and in the second test a maximum random error of 20%.

Against the total 20 iterations of the algorithm, the outputs obtained at the last 3 iterations considering a maximum random error e_{max} of 10% and 20% are shown in figure 5.16. It can be easily observed that the trajectory is different at each iteration of the algorithm

In contrast, the RMS obtained in the absence of disturbances and in the presence of the two non-repetitive disturbances is shown in figure 5.17. It can be seen, as might be expected, that while perfect tracking is achieved in the absence of disturbances, the non-repetitive disturbances prevent the achievement of the latter. In particular, it can be observed that the average final RMS value is about 10 in the case of a max random error of 10%, while it is about 20 in the case of a max random error of 20%, demonstrating that the pure Iterative Learning Control succeeds in tracking in the absence of non repetitive disturbances, but is unable to cope with the latter, which, varying with each iteration, do not allow the memorization on which the algorithm is based.

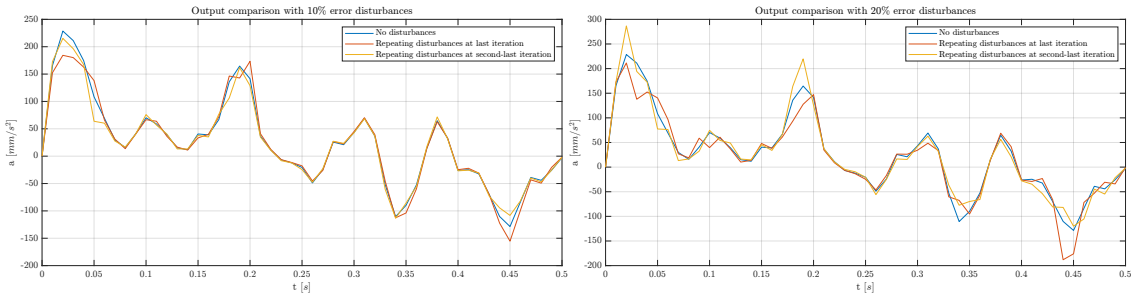

Figure 5.16: Comparison of the output response at the last two iterations of the algorithm in the presence of non-repetitive disturbances. In the left figure: assuming max random error e_{max} of 10%. In the right figure: assuming max random error e_{max} of 20%.

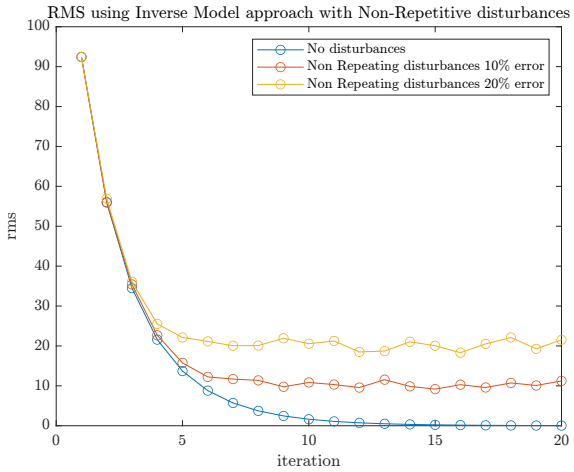

Figure 5.17: Comparison of the RMS of the tracking error every 100 iterations with ILC using inverse model approach in presence of non-repetitive disturbances

Finally the tracking error at each iteration using the ILC Inverse model approach in the case of non-repetitive disturbances is represented in the figure 5.18. It is possible to see in the late iterations the noise still present in the tracking error.

5.5.2 Repetitive disturbances

In this section, the effect of repetitive disturbances in using ILC will be analyzed. Specifically, repetitive disturbances have been implemented, according to the previously discussed classification, as state-repetitive disturbances, i.e., it has been assumed that we have not modeled a part in the dynamics of the system.

Until now, in fact, we had considered the quarter car suspension model as a linear model.

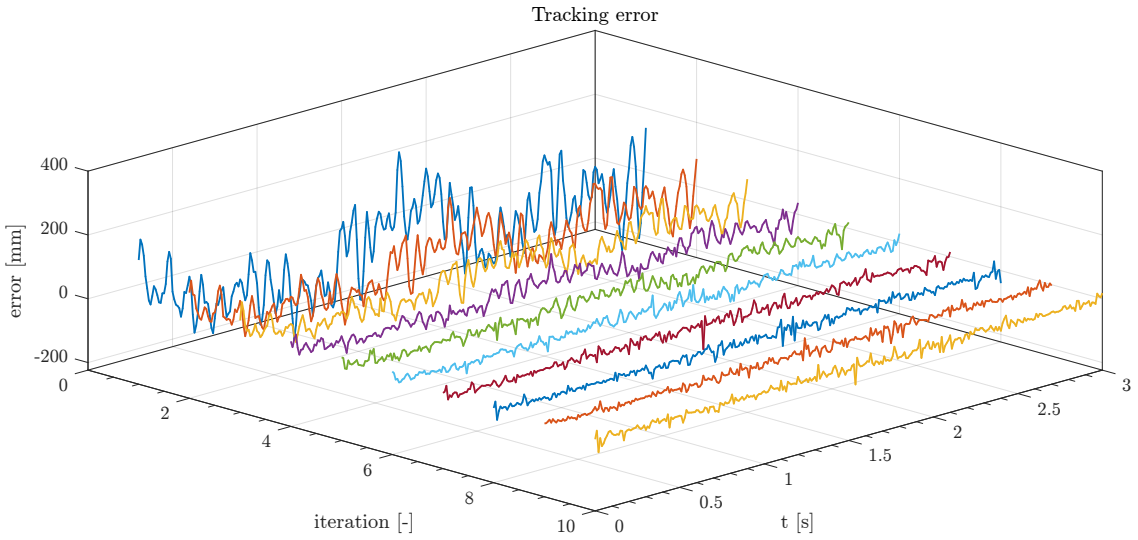

Figure 5.18: Tracking error at each iteration using the ILC inverse model approach in the case of non-repetitive disturbances assuming max random error e_{max} of 10%

However, all real systems exhibit nonlinear characteristics, which in the case of a suspension can include hardening springs, a quadratic damping force and tire "lift-off" phenomenon.

The model implemented for a non-linear quarter car model is proposed in [28]. It also considers the nonlinear component of suspension damping and the nonlinear component of tyre spring stiffness and suspension spring stiffness, thus adding the parameters k_{s2}, c_{s2}, k_{t2}, k_{t3} for the square tyre spring coefficient, square suspension damper coefficient, square suspension tyre coefficient and cubic tyre spring coefficient, respectively. The old parameters k_s, c_s, k_t, for the linear suspension spring coefficient, the linear suspension damping coefficient and the linear tyre suspension coefficient, respectively, are renamed as k_{s1}, c_{s1} and k_{t1}. The equations of motion then are generalized by eq. (2.6) as:

$$\begin{aligned} M_v\ddot{z}_u + c_{s1}\left(\dot{z}_u - \dot{z}_c\right) + \underbrace{c_{s2}\left(\dot{z}_u - \dot{z}_c\right)^2}_{Nonlinear} + k_{t1}(z_u - z_q) + \underbrace{k_{t2}(z_u - z_q)^2 + k_{t3}(z_u - z_q)^3}_{Nonlinear} \\ + k_{s1}\left(z_u - z_c\right) + \underbrace{k_{s2}\left(z_u - z_c\right)^2}_{Nonlinear} = 0 \\ m_c\ddot{z}_c + c_{s1}\left(\dot{z}_c - \dot{z}_u\right) + \underbrace{c_{s2}\left(\dot{z}_c - \dot{z}_u\right)^2}_{Nonlinear} + k_{s1}\left(z_c - z_u\right) + \underbrace{k_{s2}\left(z_c - z_u\right)^3}_{Nonlinear} = 0 \end{aligned} \tag{5.43}$$

where the new non-linear components are highlighted in the dynamics equations using the

underbrace symbol. The parameters of the nonlinear model were adapted from the same paper and are given in Appendix C.

The ILC algorithm is then implemented again using the inverse model approach technique. However, in this case we assume that we only know the linear system with the value of the parameters defined by the Estimated model. Using a learning filter matrix $\mathbf{L} = l_0\hat{\mathbf{P}}^{-1}$, where $l_0 = 0.85$ and a $\mathbf{Q} = \mathbf{I}$ and simulating for a total of 10 iterations, we obtain the graph of the RMS shown in figure 5.19.

Note that when there are unmodeled parts of the system, it is not guaranteed to get perfect tracking, that is, to achieve an RMS of the error equal to zero. This is because in this case having a difference between the dynamics of the system and the $\mathbf{L}$ matrix, the best input obtainable with the latter may not match the real one. That's because there may be some parts that are unreachable due to the lack of dynamics in the estimated model.

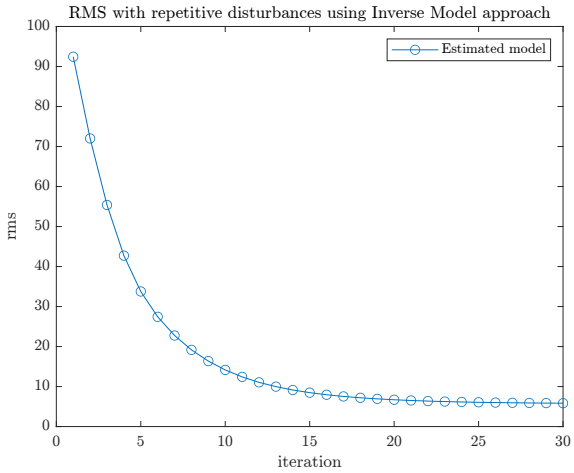

Figure 5.19: RMS with ILC inverse model approach with unmodeled dynamics

5.6 Controllers Comparison

This section depicts the comparison for acceleration reference tracking using three different types of controllers: a conventional feedback controller based on the proportional control

$$\boxed{u(t) = K_p e(t)} \tag{5.44}$$

a feedforward controller based on inverse model

$$\boxed{u(t) = G^{-1} y_d(t)} \tag{5.45}$$

where the input at the current instant is obtained by multiplying the reference at the current instant by the inverse of the estimated model, and the Iterative Learning control, which is a repetitive controller based on the inverse model approach:

$$\boxed{u_{j+1}(k) = Q(\mathrm{q})(u_j(k) + L(\mathrm{q})e_j(k+1))} \tag{5.46}$$

In particular, a proportional feedback controller with gain $K_p = 0.01$, and an Iterative Learning Control based on inverse model approach where $\mathbf{Q} = \mathbf{I}$ and $\mathbf{L} = l_0 \hat{\mathbf{P}}^{-1}$ with $l_0 = 0.75$ and considering 10 iterations, were used for this analysis.

Figure 5.20 shows the comparison result where in all 3 cases it was assumed to be in the absence of disturbances. It is possible to see that the proportional controller struggles to follow the reference because of the fast dynamics of acceleration. A big improvement is obtained by exploiting the feedforward technique, in which the Estimated model is used. However, due to the parameter uncertainties, also this controller can not achieve perfect tracking. The best result is obtained using the ILC controller based on model inversion approach after 5 repetitions. In particular in this last case, since we don't have any disturbances, we are able to achieve perfect tracking.

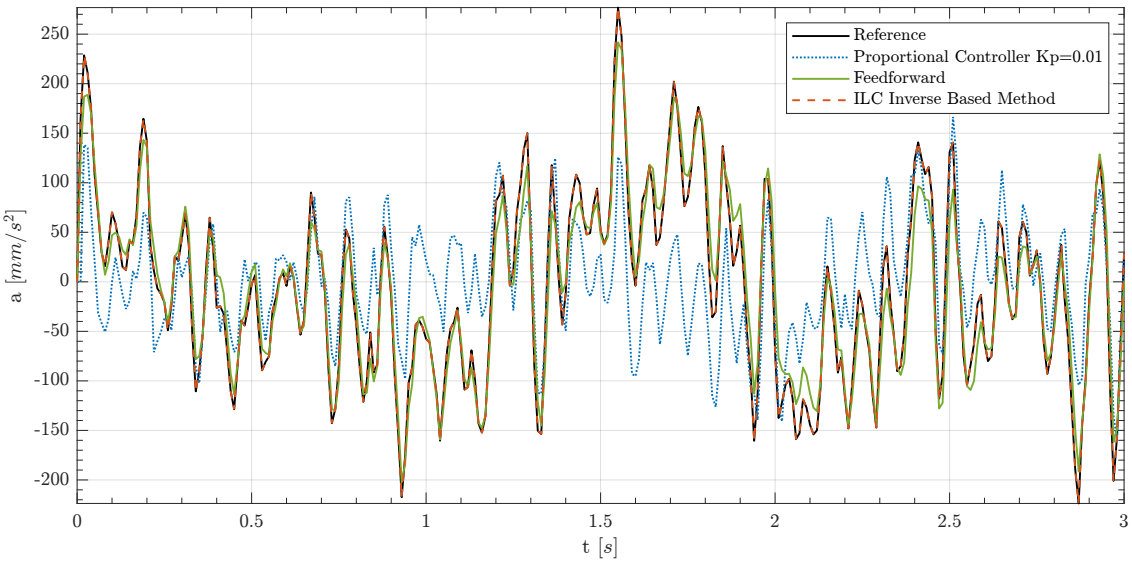

Figure 5.20: Controller results comparison

5.7 Generalization to the Multivariable Case

In this last section of the chapter, the ILC algorithm is analyzed for use in the case of multivariable systems. After a reminder about the theory and generalizing the algorithm to MIMO systems case, the ILC algorithm is implemented by considering the full car suspension model.

Note that the following algorithm can be applied only for square multivariable systems, meaning that the number of inputs n is equivalent to the number of outputs m, hence $m = n$. In the more complex scenario where we have non-square matrices in the multivariable cases, a possible procedure is proposed by Santina et al. [29].

Considering a generic multivariable system with transfer function $\mathbf{P} \in \mathbb{R}^{n\times n}$ with n input and n output:

$$\mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix} \tag{5.47}$$

where $P_{ij} \in \mathbb{R}^{1\times 1}$ is the transfer function from input i to output j.

In order to implement the ILC algorithm in the time-domain, it is necessary to introduce the lifted domain representation:

$$P_{ij}(\mathrm{q}) = \sum_{k=0}^{\infty} p_k^{ij} \mathrm{q}^{-k} \tag{5.48}$$

Considering a finite sequence of N samples, we obtain:

$$\begin{aligned} P_{ij}(\mathrm{q}) =& \sum_{k=1}^{N} p_k^{ij} \mathrm{q}^{-k} \\ =& p_1^{ij}\mathrm{q}^{-1} + p_2^{ij}\mathrm{q}^{-2} + p_3^{ij}\mathrm{q}^{-3} + \ldots + p_N^{ij}\mathrm{q}^{-N} \end{aligned} \tag{5.49}$$

where $p_1^{ij}, p_2^{ij}, \ldots, p_N^{ij}$ is the impulse response sequence of the transfer function from input i to output j.

Hence, each transfer function P_{ij} becomes a lower triangular Toeplitz $(N \times N)$ matrix as

follows:

$$\mathbf{P}_{ij} = \begin{bmatrix} p_1^{ij} & 0 & \cdots & 0 \\ p_2^{ij} & p_1^{ij} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_N^{ij} & p_{N-1}^{ij} & \cdots & p_1^{ij} \end{bmatrix} \tag{5.50}$$

containing the N impulse response of the single transfer function P_{ij} from input i to output j.

Rewriting the system dynamics equation in lifted form as in eq. (4.11), one obtains:

$$\begin{bmatrix} \mathbf{Y}_j(1) \\ \mathbf{Y}_j(2) \\ \vdots \\ \mathbf{Y}_j(N) \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 & 0 & \cdots & 0 \\ \mathbf{P}_2 & \mathbf{P}_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_N & \mathbf{P}_{N-1} & \cdots & \mathbf{P}_1 \end{bmatrix} \begin{bmatrix} \mathbf{U}_j(0) \\ \mathbf{U}_j(1) \\ \vdots \\ \mathbf{U}_j(N-1) \end{bmatrix} + \begin{bmatrix} \mathbf{D}(1) \\ \mathbf{D}(2) \\ \vdots \\ \mathbf{D}(N) \end{bmatrix} \tag{5.51}$$

where $\mathbf{Y}_j(k) \in \mathbb{R}^{1\times n}$ is the output vector at instant k, $\mathbf{U}_j(k) \in \mathbb{R}^{1\times n}$ is the input vector at instant k, $\mathbf{D}(k) \in \mathbb{R}^{1\times n}$ is the disturbance vector at instant k and

$$\mathbf{P}_k = \begin{bmatrix} p_k^{11} & p_k^{12} & \cdots & p_k^{1n} \\ p_k^{21} & p_k^{22} & \cdots & p_k^{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_k^{n1} & p_k^{n2} & \cdots & p_k^{nn} \end{bmatrix} \tag{5.52}$$

is a $\mathbb{R}^{n\times n}$ matrix containing the impulse response of each transfer function $\mathbf{P}_{ij}$ in lower triangular Toeplitz as in eq. (5.50) from input 1 to n and from output 1 to n at the generic instant k.

5.7.1 Design of the ILC filters in the multivariable case

In order to design the filters $L(q)$ and $Q(q)$ in the multivariable case, we can refer to the procedure proposed by Blanken et al. in [30], where P_{ii} corresponds to the real transfer function from input i to output i from eq. (5.47), while $\widehat{P}_{ii}$ is the estimated transfer function from input i to output i, and n is the number of inputs and outputs.

The first step in the algorithm proposed is the interaction analysis, and in particular the pa-

Algorithm 5.4 MIMO-Iterative Learning Control-Design

1: Interaction analysis. Decoupled?
   - Yes: independent SISO design.
   - No: proceed to next step.

2: Decoupling transformations. Decoupled?
   - Yes: independent SISO design.
   - No: proceed to the next step.

3: Robust multi-loop SISO design.
   1. Obtain SISO models $\widehat{P}_{ii}$ of the diagonal elements P_{ii}, $i = 1, \ldots, n$;
   2. Multi-loop SISO design of L using $\widehat{P}_{ii}$;
   3. Robust SISO design of Q.

   Performance not satisfactory? Proceed to next step.

4: Robust decentralized MIMO design.
   1. Obtain SISO models $\widehat{P}_{ii}$ of the diagonal elements P_{ii}, $i = 1, \ldots, n$;
   2. Multi-loop SISO design of L using $\widehat{P}_{ii}$;
   3. Robust decentralized design of Q with respect to ignored interaction and modeling errors.

   Performance not satisfactory? Proceed to next step.

5: Centralized MIMO design.
   1. Obtain MIMO model $\hat{P}$ of P, including interaction;
   2. MIMO design of L using $\hat{P}$ such that $I - L\hat{P}$ is small;
   3. Robust design of Q through decentralized design with respect to modeling errors.

per suggests using as interaction measure the frequency-dependent relative gain array (RGA). However, considering the full car suspension model, even without performing the interaction analysis, we can infer that the road excitation at each wheel will mostly affect the accelerometer located in the corner of the chassis above the corresponding considered wheel. Moreover, although the coupling of the system is present, we can neglect the second step concerning the decoupling transformations since it can be assumed that each road excitation almost exclusively affects the corresponding accelerometer. Thus, our approach will consist of implementing the third step, that is, a multi-loop SISO design without taking into consideration the interactions of the system.

We can design the **L** matrix as suggested by Blanken et al. [31] for the multi-loop SISO ILC

approach as:

$$\mathbf{L} = \text{diag}\left\{L_1, L_2, \ldots, L_n\right\} \tag{5.53}$$

where $L_i = \hat{P}_{ii}^{-1}$. Note that in this case we are not taking into account the interactions of the systems.

5.7.2 Stability and Monotonic convergence Theorems

We shall now see how the main stability and monotonic convergence theorems found in chapter 4 can be generalised in the multivariable case. Note that neglecting the system interactions, these conditions are no longer sufficient and in particular they will depend on how many interactions are present in the system.

Regarding the stability theorem in eq. (4.24) in the lifted domain representation, it becomes:

$$\rho(\mathbf{Q}_i(\mathbf{I} - \mathbf{L}_i\mathbf{P}_{ii})) < 1, \quad \forall i, i = 1, \ldots, n \tag{5.54}$$

while for the stability theorem in the frequency domain found in eq. (4.25), it is generalised as:

$$\left|Q_i\left(e^{j\omega}\right)\left(1 - L_i\left(e^{j\omega}\right)P_{ii}\left(e^{j\omega}\right)\right)\right| < 1, \quad \forall \omega, \forall i, i = 1, \ldots, n \tag{5.55}$$

Regarding the monotonic convergence theorem in eq. (4.27), it becomes as follows:

$$\bar{\sigma}(\mathbf{I} - \mathbf{P}_{ii}\mathbf{Q}_i\mathbf{L}_i) < 1, \quad \forall i, i = 1, \ldots, n \tag{5.56}$$

5.7.3 Road Excitation of all Wheels

In this section, the procedure for generating road profiles and simulation results considering the full car suspension model will be analyzed. The model parameters used for the simulations are given in the Appendix B.

Once we have done the design of the learning filter $L(\mathrm{q})$ and the robustness filter $Q(\mathrm{q})$ we can proceed to the simulation of the algorithm. In the same way as done previously with the quarter car model, we use as input for the four wheels the road excitation generated in the chapter 2 to obtain the accelerations at the four corners of the chassis which will be used as reference signal for the ILC algorithm. In addition, exactly as done with the implementation of the algorithm

in the SISO case, only the first 3 seconds of the trajectories will be considered. In particular, using the sinusoidal approximation method, two different road profiles were generated for the right and left front wheel. They are shown in the upper left and lower left graphs in figure 5.21.

For the rear wheels, the road profile of the corresponding front wheel shifted in time according to a time delay represented by a red vertical line was used, so as to simulate the through of the same two imaginary lines of the longitudinal road profile as defined. The time delay used is the same as that used in chapter 3 and corresponds to $\tau = 1.1$ s. The resulting road profiles for the two rear wheels are shown in the upper right and lower right graphs in figure 5.21.

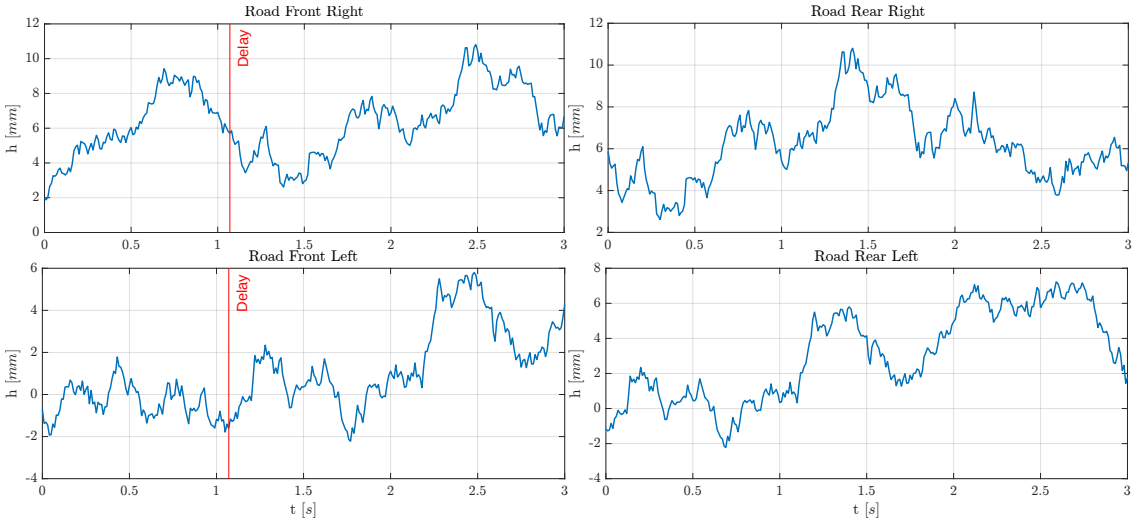

Figure 5.21: Road profiles for the multivariable case

The acceleration reference to track are shown in the figure 5.22.

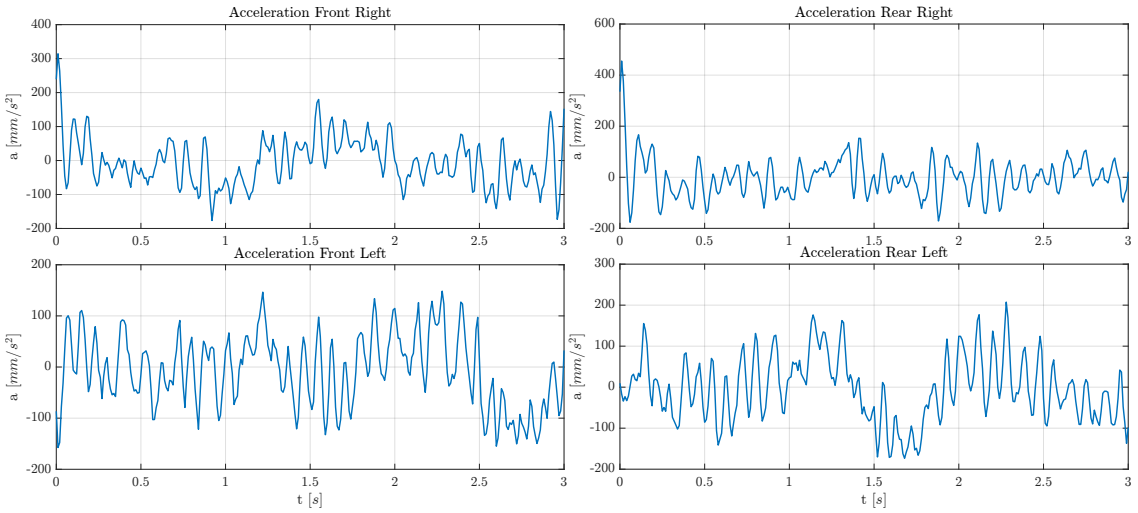

Figure 5.22: Accelerations for the multivariable case

Using the procedure proposed by Blanken et al., we can assume that although our system is not completely decoupled, we can still take advantage of the multi-loop SISO design technique. In the multi-loop SISO design approach, in the case the interactions were greater, to account for them the ILC could be robustified a posteriori using the Q-filter.

The RMS of the simulation results are shown in figure 5.23. In particular, the ILC algorithm with inverse model approach as defined in eq. (5.53) was used. The maximum number of iterations j_{max} was defined to be $j_{max} = 10$ and the sample time used was $T_s = 0.01$ s.

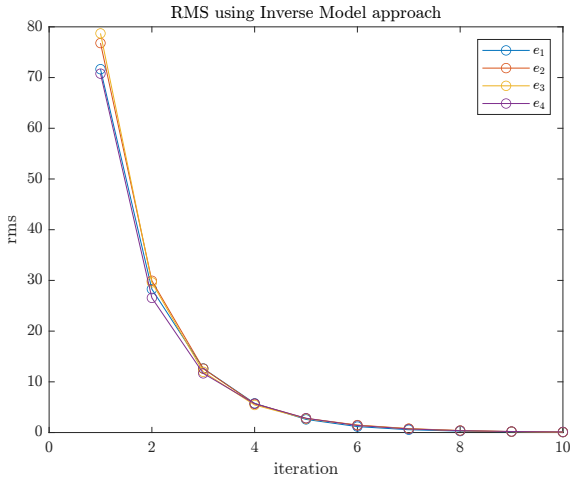

Figure 5.23: RMS obtained with ILC inverse model approach for the multivariable case

It is possible to see that at the last iteration, each of the RMSs referring to each tracking error

goes to 0, thus achieving perfect tracking of the references.

The input at the last iteration obtained using the implemented ILC algorithm are shown in figure 5.24. It can be seen that although we manage to obtain perfect tracking of the reference, the reconstructed inputs do not match the inputs used in the simulation in chapter 2.

There may be multiple reasons why the input found does not exactly match the input used in simulation. Particularly in the multivariable case, one reason is that the same output can be generated by different linear combinations of the inputs.

Moreover, note that as time increases, the error between the actual input and the reconstructed input increases. This result is due to the fact that the more the model prediction is projected forward, the greater the uncertainty (think for example of the Kalman Filter). Consequently, because the ILC considers the whole period at each iteration, an error will have a greater effect as time increases.

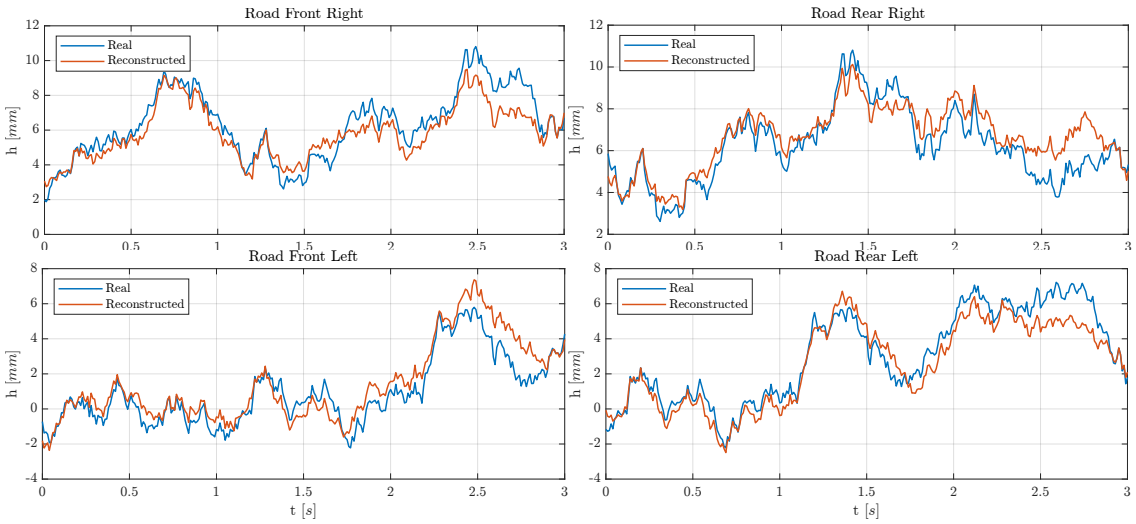

Figure 5.24: Comparison between the input at last ILC iteration and the real input for the multivariable case

6 Experiment

In this chapter, the Iterative Learning Control algorithm will be tested on a real system provided by the STEP Lab company. Initially, a model of the system will be formulated by differential equations and State Space representation to be used with Inverse Model based method of ILC. Then the procedure by workflow scheme that will be used will be introduced. Finally, the results obtained will be analyzed and discussed, particularly comparing them with those found in simulation.

6.1 Introduction

STEP Lab was founded in 2011 as a manufacturer of mechanical testing machines. Today STEP Lab is a company that manufactures and sells systems for static, dynamic and impact mechanical testing. The testing department offers testing services for materials and products, for their development and for quality control. Some of the tests available are for example: fatigue, impact, tensile, torsion and bending. All their machines are based on electrical actuation and are operated by the same Test Center interface controller and software.

Some of the main industries the company works on include for example: automotive, bicycles, sports, biomedical fields, springs, DMA (Dynamic Mechanical Analysis).

6.2 Experiment

The STEP Lab company was able to provide me with the structure of the experiment, which is shown schematically in figure 6.1, the schematic of which was obtained by modifying the schematic in [32].

It consists of a personal computer, an industrial computer, the STEP Lab test manager inside which are the data acquisition card and the PLC Controller. Finally there is the actual machine on which the experiment is carried out consisting of an electrodynamic motor, a suspension and a load cell.

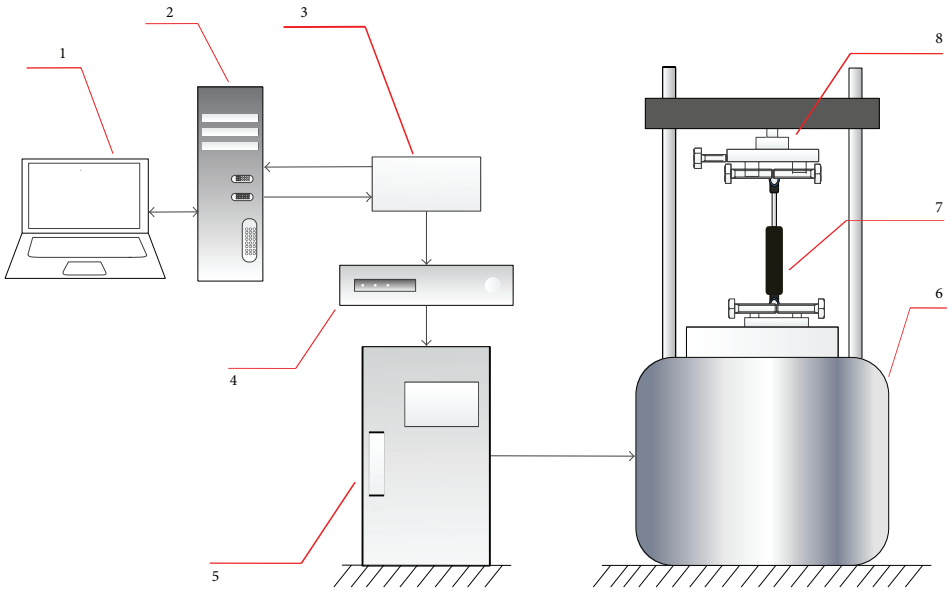

Figure 6.1: Schematic diagram of the HIL experiment platform: 1-Personal Computer (PC), 2-Industrial PC with TestCenter software, 3-Data acquisition card, 4-PLC controller, 5-STEP Lab Test Manager, 6-Electrodynamic system based on linear motor, 7-Suspension, 8-Load cell

Specifically in the machine that was provided to me, the electrodynamic test system was based on a linear motor and the PLC controller was based on a PID type controller. Ethernet cables were used for the connections between the industrial computer and the test manager and between the test manager and the machine.

The photo of the testing system taken at the company is represented in figure 6.2, where it is possible to distinguish all the above components except for the personal computer, placed to the left of the industrial computer.

Figure 6.2: Photo of the testing system provided by the STEP Lab company

Finally, to interact with the machine, the company's proprietary software called Test Center is installed in the industrial computer, the interface of which is shown in figure 6.3. Through this program, it is possible to control the machine actuator either in manual mode or through high-level PLC commands such as positioning to a certain position or following a certain pattern. Also from the same program it is also possible to see the graphs of the experiment in real time.

Note that unlike the simulations done in the previous chapter, in this case the machine had structural limitations, shown in table 6.1. Consequently, these did not allow the heuristic type approach of the ILC algorithm due to the high learning transient phase. Consequently, a model-based approach was used to implement the algorithm, specifically the inversion model-based approach already analyzed in chapter 5.

The use of this approach requires the definition of a linear model that best represents the system to be controlled, which in this case consists of the machine consisting of the electrodynamic actuator, the suspension, and the load cell. The model is defined and described in the next section.

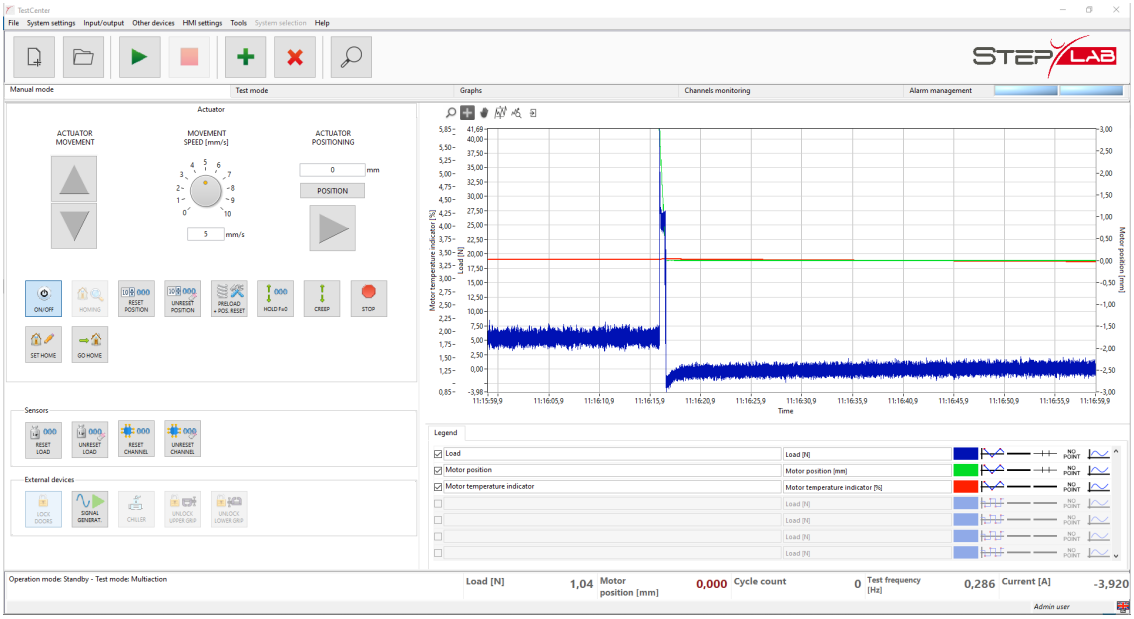

Figure 6.3: STEP Lab Test Center program interface

6.3 Mathematical model

The system can be approximated by means of a mass-spring-damper model according to the scheme depicted in figure 6.4, where m is the mass of the mechanical actuator, k_s is the spring coefficient, c_s is the damping coefficient, x is the displacement, and i is the current that will correspond to the system input.

Since, as described above, the actuator generates a force F by means of a linear motor, it is possible to use the following relationship to calculate the force produced:

$$F = ki \tag{6.1}$$

where k is the motor coefficient and i is the applied current. Using Newton's law:

$$m\ddot{x} = \Sigma F \tag{6.2}$$

as already seen in chapter 2 and thus considering all the forces acting on the system, it is possible to define the equation of motion, which can be written as:

$$m\ddot{x} = -c_s\dot{x} - k_s x + ki \tag{6.3}$$

Parameter	Value
Maximum static load	5000 N
Maximum dynamic load	5100 N
Maximum position in absolute coordinate	220 mm
Minimum position in absolute coordinate	0 mm
Maximum speed	5500 mm/s
Maximum acceleration	500 m/s^2
Maximum jerk	10^9 m/s^3
Maximum test frequency	300 Hz

Table 6.1: Limit parameters

where forces due to friction have been neglected. Note also that the proposed model does not take into account the force of gravity, which as we shall see will be taken into account later in generating the input. The previous relationship can be rewritten as:

$$\ddot{x} = \frac{-c_s\dot{x} - k_s x + ki}{m} \tag{6.4}$$

Taking into consideration that the input of the system is current i and the output of the system is the position of the suspension x, the equation of motion is finally represented by State Space representation in eq. (6.5), in which the choice of states of the system results immediately in x and $\dot{x}$.

$$\begin{aligned} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\frac{k_s}{m} & -\frac{c_s}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{m} \end{bmatrix} i \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \end{aligned} \tag{6.5}$$

The nominal parameters of the model are given directly from the company and they are reported in table 6.2.

Parameter	Symbol	Value
Suspension Mass	m	2 Kg
Suspension Spring Stiffness	k_s	4 N/m
Suspension Damper Coefficient	c_s	2703 N/(m/s)
Current Coefficient	k	50 N/A

Table 6.2: System parameters of STEP Lab suspension model

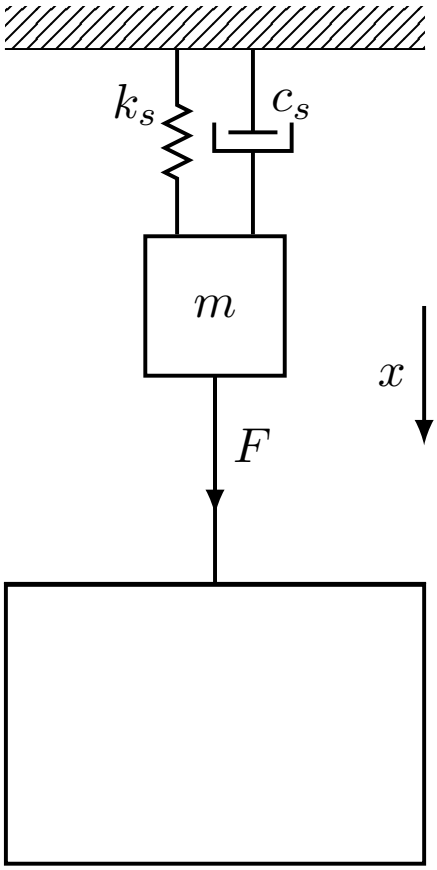

Figure 6.4: Mass-spring-damper model of the testing system

6.4 Experiment

The goal of the experiment is tracking the position-reference signal represented in figure 6.5, which corresponds to a sine wave of amplitude $A = 10$ mm and frequency $f = 1$ Hz.

To do this, the workflow shown in the figure 6.6 was used. Specifically, the first three steps of the workflow correspond to iteration 1 of the algorithm. Then the remaining steps are repeated cyclically a number of times defined by the maximum number of iterations j_{max}. In addition, some considerations are necessary for the implementation of the algorithm.

The first consideration is that the ILC algorithm was executed offline at each iteration by MATLAB. In order to interact with the machine, it was necessary to generate at each iteration

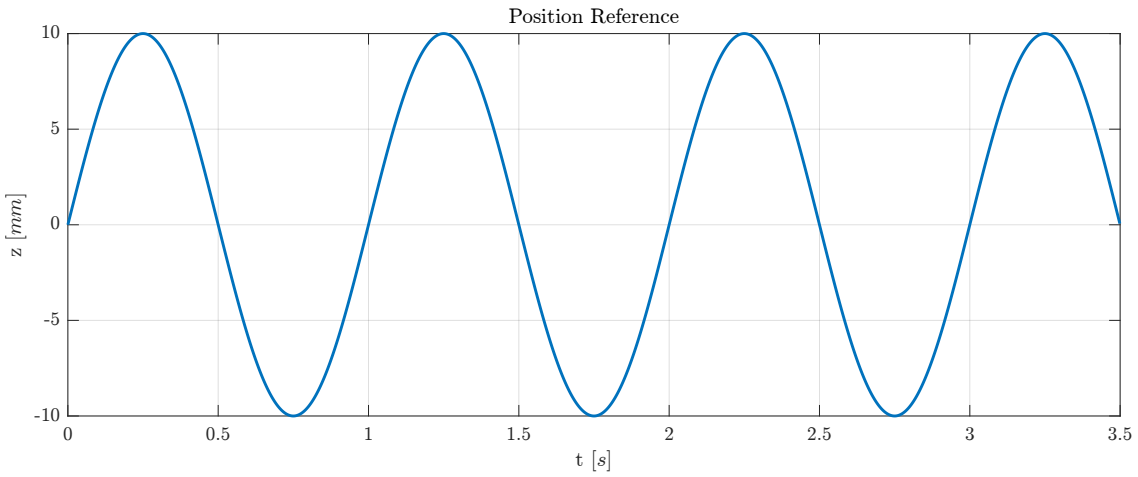

Figure 6.5: STEP Lab experiment: sinusoidal position reference

a csv file containing the input that had to be imported into the Test Center software. Then, once the iteration was completed, the software would generate a tdms file containing all the results of the experiment at the j-th iteration. The csv file and tdms file had to be generated and read by MATLAB at each iteration of the algorithm, respectively.

The second consideration to be taken into account concerns the units of measurement. In fact, while the defined linear model considers the position in meters, the generated sinusoidal reference considers the position in millimeters. Consequently, before calculating the tracking error it is necessary to convert the reference from millimeters to meters. Similarly, it will be necessary to convert the position signal read from the tdms file from millimeters to meters in order to have a tracking error for updating the defined input in meters.

In addition, as seen in chapter 4, one of the fundamental assumptions of the ILC algorithm was that the initial state is invariant over iterations. Consequently, at each iteration of the algorithm, through the Test Center software, a reset of the suspension position to the initial value of $p_0 = 0$ mm had to be performed. In fact, remember that a fundamental assumption of the algorithm was to have the same initial condition between different iterations of the algorithm.

Finally, the algorithm had to take into account the force of gravity, not present in the defined linear model but certainly present in reality. Consequently, once the input for the current iteration is updated, the contribution of the force of gravity is added to the input.

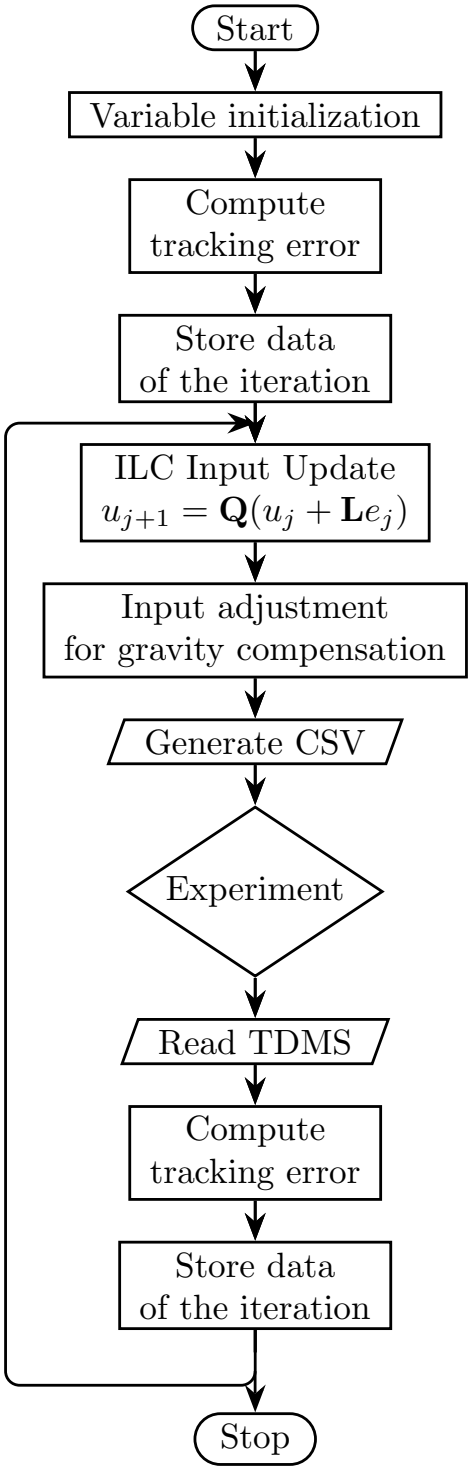

Figure 6.6: Workflow of the STEP Lab experiment

6.4.1 Results

This section will show the results of the experiment obtained in the company, using a sampling time of $T_s = 0.001$ s.

Since as mentioned above, due to the limitations of the machine it was not possible to use a heuristic approach of the ILC algorithm, the algorithm by inversion model based method was used, with a maximum number of iterations $j_{max} = 9$.

The RMS error is represented in figure 6.7. We can see that in this case we fail to achieve perfect tracking, and in particular we settle around an RMS value of 0.81. This result we could expect, since in this case there are definitely non-repetitive disturbances within the system, due for example to sensor noise, which prevent as we have seen perfect tracking. Moreover, in this case not only in the previous mathematical model we did not model any kind of friction, but surely the real system will be nonlinear. Hence, the unmodeled dynamics can be considered the second factor that prevents from the perftect tracking. Finally, as already seen above, the third factor will definitely be the use of the Q-filter, which allows for greater robustness of the algorithm at the expense of loss of performance.

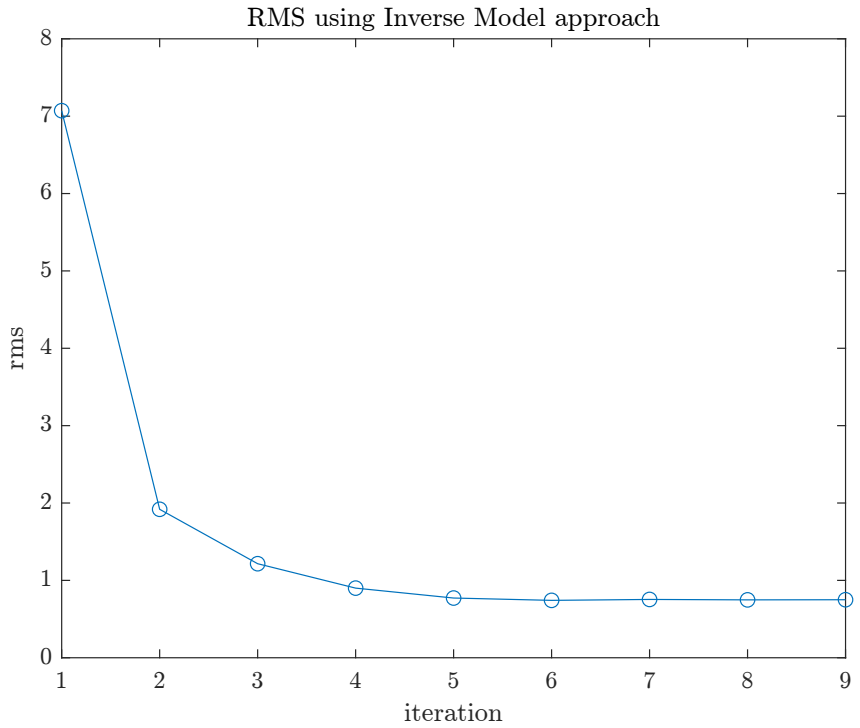

Figure 6.7: STEP Lab experiment: RMS of the tracking error at each iteration

The analysis of the output at each iteration is given in figure 6.8, where the reference corresponds to the sinusoidal reference signal generated in MATLAB in figure 6.5. Looking at the result, we can see that at each iteration it seems to get closer to the reference value, as we could expect from the graph on RMS.

The analysis of the final output is given in figure 6.9. We can see that, as might be expected from any real system, there is a very slight delay between the two signals.

The analysis of the input at each iteration is given in figure 6.10. In this case, the value of the reference current in the figure was obtained by performing an experiment in which the suspension was checked directly in position through the Test Center software. Then using the sinusoidal profile in figure 6.5 as input, following the experiment it was possible to extract from the tdms file the current value used, which is considered the reference.

It can also be seen that the average value of the current in the figure is -3.9 A. This is in fact the value that was added at each iteration to the input calculated by the algorithm to account for the force of gravity. In fact, since the mass of the suspension is 2 Kg, that is, it has a weight of 19.62 Kg, a force F of $19.62 \times 9.81 = 192.5$ N will be required to keep the suspension in equilibrium. In fact, if we multiply the offset current by the value of the motor constant k we get $-3.9 \times 50 = -195$ N, which approximately balances the above force.

The analysis of the final input is given in figure 6.11. It can be observed that unlike the reference input, the input at the last iteration is much smoother, due to the filtering of the input by Q-filter. Also it is possible to note that the greatest difference between the two signals is present in the first instants, where the reference exhibits more nonlinear behavior. Indeed, one can assume a higher input due to the initial presence of static friction, which is greater than the dynamic friction acting later.

As final image, the error at each iteration is represented in figure 6.12. It can be seen that the amplitude decreases with each iteration until it stabilizes about iteration number 7.

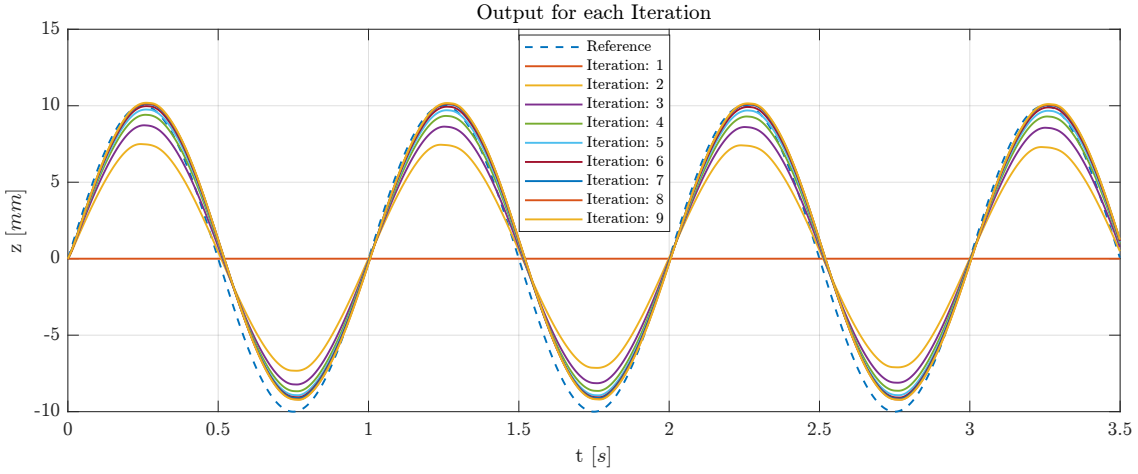

Figure 6.8: STEP Lab experiment: comparison between reference output and ILC output at each iteration

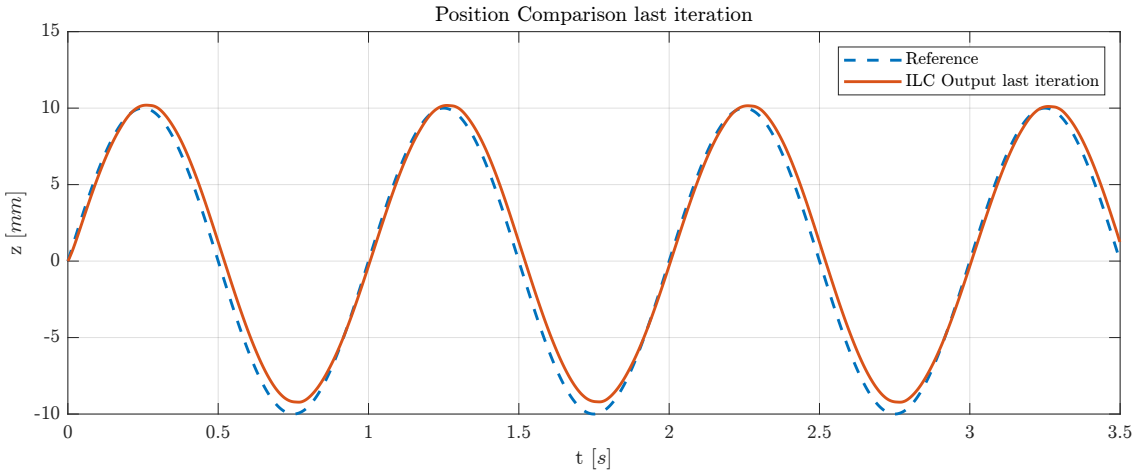

Figure 6.9: STEP Lab experiment: comparison between reference output and ILC output at last iteration

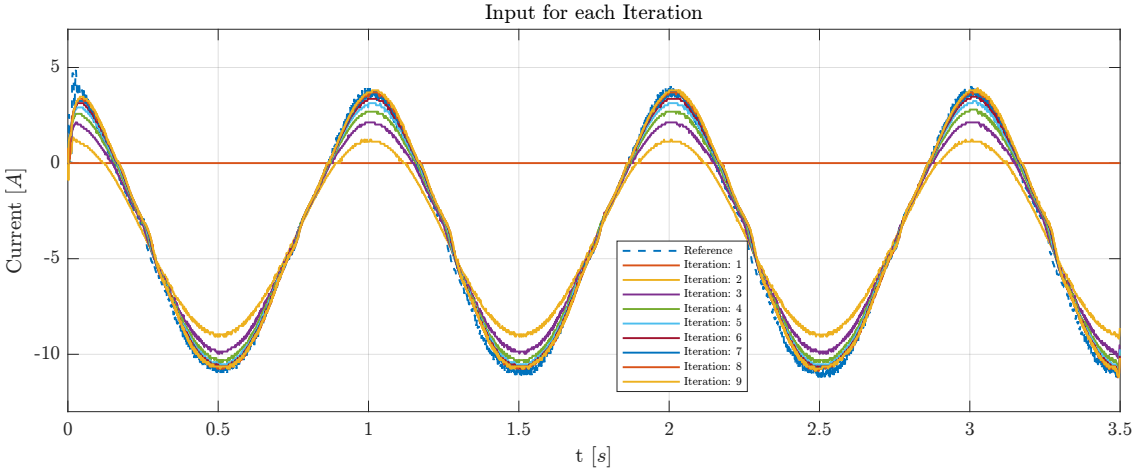

Figure 6.10: STEP Lab experiment: comparison between reference input and ILC input at each iteration

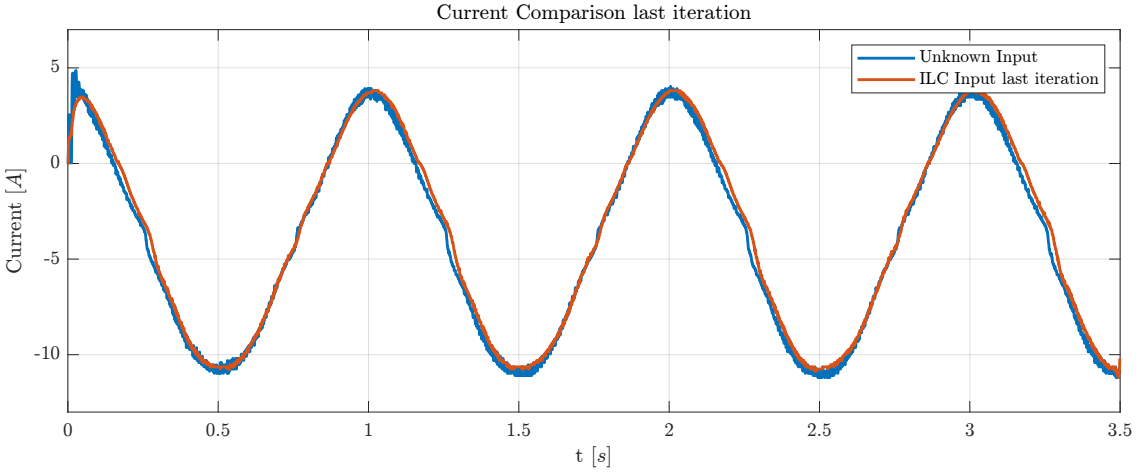

Figure 6.11: STEP Lab experiment: comparison between reference input and ILC input at last iteration

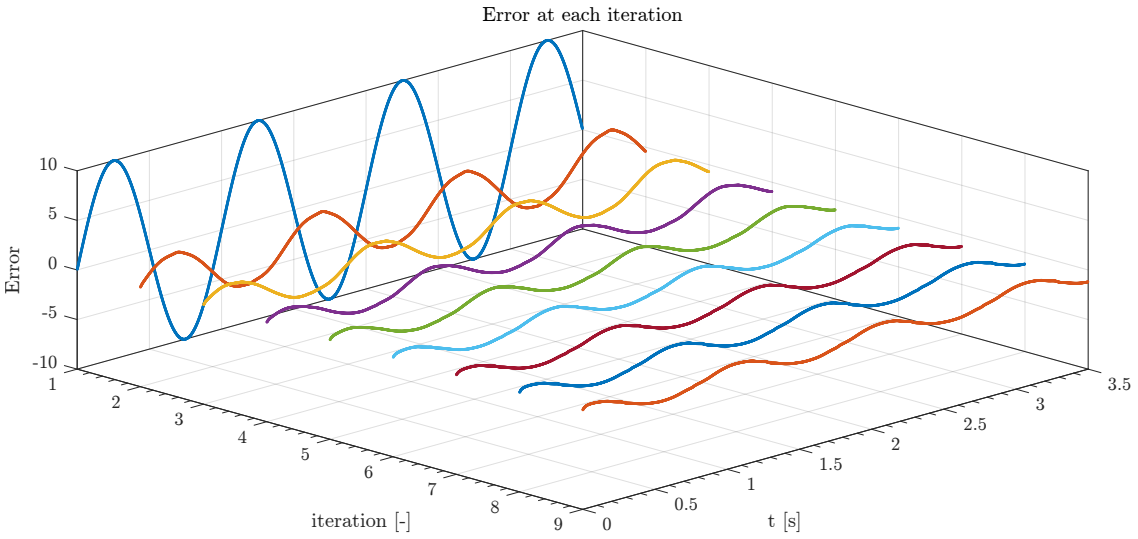

Figure 6.12: STEP Lab experiment: comparison of the tracking error at each ILC iteration

7 Conclusion

The main objective of this thesis was to explore and analyze the application of the Iterative Learning Control algorithm in the automotive domain.

Initially, an accurate literature review was conducted and the algorithm was tested through simulations in MATLAB and Simulink environment. Then, thanks to the equipment provided by the company, it was possible to implement the algorithm in a real environment for validation and comparison of the results. In addition, the various design approaches for iterative control were analyzed, showing the advantages and problems related to each. During the experimentation and analysis, it was found that the iterative control algorithm has been shown to be effective in improving the performance of dynamic systems subject to repeated iterations and to enable better tracking of a reference signal than more traditional controllers. In particular, the use of a heuristic approach, while very intriguing from a theoretical point of view because of its simplicity and guarantee of asymptotic stability, is difficult to apply in practice because of the high learning transient behavior. In contrast, the inversion model-based approach was also found to be applicable in a real system, but it is followed by the need to model the system through a nonlinear model as consistent with reality as possible. It was also possible to see how the uncertainty in the estimated model affects the final performance, and how the use of filtering in updating the input by using the Q-filter results in a trade-off between performance and robustness of the algorithm. Finally, the real-world experiment allowed us to show that the algorithm performs well even in the presence of unmodelled dynamics due to the approximation of a real system in a linear model.

For a more complete validation of what has been shown in simulation, some future work is to conduct the experimental studies on more complex machines, which allow more similarity with the quarter car and full car model analyzed. A further possible future development is to test the algorithm in a real scenario which is known for the presence of some repetitive disturbances.

In conclusion, this thesis has laid the foundation for a comprehensive analysis of the Iterative Learning Control algorithm, showing step-by-step modeling of the system, generation of tracking profiles, and implementation of the algorithm both in simulation and in a real-world environment.

A

Full Car model State Space representation

$$
A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{k_{s1}+k_{t1}}{m_{v1}} & -\frac{c_{s1}}{m_{v1}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{k_{s2}+k_{t2}}{m_{v2}} & -\frac{c_{s2}}{m_{v2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{k_{s3}+k_{t3}}{m_{v3}} & -\frac{c_{s3}}{m_{v3}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{k_{s4}+k_{t4}}{m_{v4}} & -\frac{c_{s4}}{m_{v4}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{Fk_{s1}}{D} & \frac{Fc_{s1}}{D} & \frac{Gk_{s2}}{D} & \frac{Gc_{s2}}{D} & -\frac{Hk_{s3}}{D} & -\frac{Hc_{s3}}{D} & \frac{Lk_{s4}}{D} & \frac{Lc_{s4}}{D} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{Gk_{s1}}{D} & \frac{Gc_{s1}}{D} & \frac{Zk_{s2}}{D} & \frac{Zc_{s2}}{D} & \frac{Pk_{s3}}{D} & \frac{Pc_{s3}}{D} & -\frac{Hk_{s4}}{D} & -\frac{Hc_{s4}}{D} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{Hk_{s1}}{D} & -\frac{Hc_{s1}}{D} & \frac{Pk_{s2}}{D} & \frac{Pc_{s2}}{D} & \frac{Qk_{s3}}{D} & \frac{Qc_{s3}}{D} & \frac{Tk_{s4}}{D} & \frac{Tc_{s4}}{D} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{Lk_{s1}}{D} & \frac{Lc_{s1}}{D} & -\frac{Hk_{s2}}{D} & -\frac{Hc_{s2}}{D} & \frac{Tk_{s3}}{D} & \frac{Tc_{s3}}{D} & \frac{Wk_{s4}}{D} & \frac{Wc_{s4}}{D}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{k_{s1}}{m_{v1}} & \frac{c_{s1}}{m_{v1}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{k_{s2}}{m_{v2}} & \frac{c_{s2}}{m_{v2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{k_{s3}}{m_{v3}} & \frac{c_{s3}}{m_{v3}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{k_{s4}}{m_{v4}} & \frac{c_{s4}}{m_{v4}} \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{Fk_{s1}}{D} & -\frac{Fc_{s1}}{D} & -\frac{Gk_{s2}}{D} & -\frac{Gc_{s2}}{D} & \frac{Hk_{s3}}{D} & \frac{Hc_{s3}}{D} & -\frac{Lk_{s4}}{D} & -\frac{Lc_{s4}}{D} \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-\frac{Gk_{s1}}{D} & -\frac{Gc_{s1}}{D} & -\frac{Zk_{s2}}{D} & -\frac{Zc_{s2}}{D} & -\frac{Pk_{s3}}{D} & -\frac{Pc_{s3}}{D} & \frac{Hk_{s4}}{D} & \frac{Hc_{s4}}{D} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\frac{Hk_{s1}}{D} & \frac{Hc_{s1}}{D} & -\frac{Pk_{s2}}{D} & -\frac{Pc_{s2}}{D} & -\frac{Qk_{s3}}{D} & -\frac{Qc_{s3}}{D} & -\frac{Tk_{s4}}{D} & -\frac{Tc_{s4}}{D} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-\frac{Lk_{s1}}{D} & -\frac{Lc_{s1}}{D} & \frac{Hk_{s2}}{D} & \frac{Hc_{s2}}{D} & -\frac{Tk_{s3}}{D} & -\frac{Tc_{s3}}{D} & -\frac{Wk_{s4}}{D} & -\frac{Wc_{s4}}{D}
\end{bmatrix}
$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{k_{t1}}{m_{v1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{k_{t2}}{m_{v2}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{k_{t3}}{m_{v3}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{k_{t4}}{m_{v4}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} \frac{Fk_{s1}}{D} & \frac{Fc_{s1}}{D} & \frac{Gk_{s2}}{D} & \frac{Gc_{s2}}{D} & -\frac{Hk_{s3}}{D} & -\frac{Hc_{s3}}{D} & \frac{Lk_{s4}}{D} & \frac{Lc_{s4}}{D} \\ \frac{Gk_{s1}}{D} & \frac{Gc_{s1}}{D} & \frac{Zk_{s2}}{D} & \frac{Zc_{s2}}{D} & \frac{Pk_{s3}}{D} & \frac{Pc_{s3}}{D} & -\frac{Hk_{s4}}{D} & -\frac{Hc_{s4}}{D} \\ -\frac{Hk_{s1}}{D} & -\frac{Hc_{s1}}{D} & \frac{Pk_{s2}}{D} & \frac{Pc_{s2}}{D} & \frac{Qk_{s3}}{D} & \frac{Qc_{s3}}{D} & \frac{Tk_{s4}}{D} & \frac{Tc_{s4}}{D} \\ \frac{Lk_{s1}}{D} & \frac{Lc_{s1}}{D} & -\frac{Hk_{s2}}{D} & -\frac{Hc_{s2}}{D} & \frac{Tk_{s3}}{D} & \frac{Tc_{s3}}{D} & \frac{Wk_{s4}}{D} & \frac{Wc_{s4}}{D} \end{bmatrix}$$

$$\begin{bmatrix} -\frac{Fk_{s1}}{D} & -\frac{Fc_{s1}}{D} & -\frac{Gk_{s2}}{D} & -\frac{Gc_{s2}}{D} & \frac{Hk_{s3}}{D} & \frac{Hc_{s3}}{D} & -\frac{Lk_{s4}}{D} & -\frac{Lc_{s4}}{D} \\ -\frac{Gk_{s1}}{D} & -\frac{Gc_{s1}}{D} & -\frac{Zk_{s2}}{D} & -\frac{Zc_{s2}}{D} & -\frac{Pk_{s3}}{D} & -\frac{Pc_{s3}}{D} & \frac{Hk_{s4}}{D} & \frac{Hc_{s4}}{D} \\ \frac{Hk_{s1}}{D} & \frac{Hc_{s1}}{D} & -\frac{Pk_{s2}}{D} & -\frac{Pc_{s2}}{D} & -\frac{Qk_{s3}}{D} & -\frac{Qc_{s3}}{D} & -\frac{Tk_{s4}}{D} & -\frac{Tc_{s4}}{D} \\ -\frac{Lk_{s1}}{D} & -\frac{Lc_{s1}}{D} & \frac{Hk_{s2}}{D} & \frac{Hc_{s2}}{D} & -\frac{Tk_{s3}}{D} & -\frac{Tc_{s3}}{D} & -\frac{Wk_{s4}}{D} & -\frac{Wc_{s4}}{D} \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where

$$F = I_{c_x} m_c l_f^2 + I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y}$$

$$G = I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y} - I_{c_x} l_f l_r m_c$$

$$H = I_{c_x} l_f l_r m_c - I_{c_x} I_{c_y} + I_{c_y} m_c t_l t_r$$

$$L = I_{c_x} m_c l_f^2 + I_{c_x} I_{c_y} - I_{c_y} m_c t_l t_r$$

$$P = I_{c_x} m_c l_r^2 + I_{c_x} I_{c_y} - I_{c_y} m_c t_l t_r$$

$$Q = I_{c_x} m_c l_r^2 + I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y}$$

$$T = I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y} - I_{c_x} l_f l_r m_c$$

$$W = I_{c_x} m_c l_f^2 + I_{c_y} m_c t_l^2 + I_{c_x} I_{c_y}$$

$$Z = I_{c_x} m_c l_r^2 + I_{c_y} m_c t_r^2 + I_{c_x} I_{c_y}$$

$$D = I_{c_x} I_{c_y} m_c$$

B

Car model parameters

B.0.1 Quarter Car model parameters

Parameter	Symbol	Value
Sprung Mass	M	400 Kg
Unsprung Mass	m	30 Kg
Suspension Spring Stiffness	k_s	10000 N/m
Suspension Damper Coefficient	c_s	850 N/(m/s)
Tyre Spring Stiffness	k_t	140000 N/m

Table B.1: System parameters of Quarter Car model

B.0.2 Full Car model parameters

Parameter	Symbol	Value
Sprung Mass		
Sprung Mass	m_c	1600 Kg
Moment of inertia about x axis	I_x	500 Kg m^2
Moment of inertia about y axis	I_y	2400 Kg m^2
Unsprung Mass		
Front Right Unsprung Mass	M_{v1}	30 Kg
Rear Right Unsprung Mass	M_{v2}	30 Kg
Rear Left Unsprung Mass	M_{v3}	30 Kg
Front Left Unsprung Mass	M_{v4}	30 Kg
Suspension Spring Stiffness		
Front Right Suspension Spring Stiffness	k_{s1}	10700 N/m
Rear Right Suspension Spring Stiffness	k_{s2}	10000 N/m
Rear Left Suspension Spring Stiffness	k_{s3}	10000 N/m
Front Left Suspension Spring Stiffness	k_{s4}	10700 N/m
Suspension Damper Coefficient		
Front Right Suspension Damper Coefficient	c_{s1}	850 N/(m/s)
Rear Right Suspension Damper Coefficient	c_{s2}	855 N/(m/s)
Rear Left Suspension Damper Coefficient	c_{s3}	855 N/(m/s)
Front Left Suspension Damper Coefficient	c_{s4}	850 N/(m/s)
Tyre Spring Stiffness		
Front Right Tyre Spring Stiffness	k_{t1}	141200 N/m
Rear Right Tyre Spring Stiffness	k_{t2}	140150 N/m
Rear Left Tyre Spring Stiffness	k_{t3}	140150 N/m
Front Left Tyre Spring Stiffness	k_{t4}	141200 N/m
Distance to the body COM		
Right wheel	t_r	0.7 m
Left wheel	t_l	0.7 m
Front axle	l_f	1.0 m
Rear axle	l_r	1.5 m

Table B.2: System parameters of Full Car model

C

Non linear Quarter Car model parameters

Parameter	Symbol	Value
Sprung Mass	M	400 Kg
Unsprung Mass	m	30 Kg
Suspension Spring Stiffness	k_{s1}	10000 N/m
Non-linear square Suspension Spring Stiffness	k_{s2}	700 N/m^2
Suspension Damper Coefficient	c_{s1}	850 N/(m/s)
Non-linear square Suspension Damper Coefficient	c_{s2}	0 N/(m^2/s)
Tyre Spring Stiffness	k_{t1}	140000 N/m
Non-linear square Tyre Spring Stiffness	k_{t2}	13000 N/m^2
Non-linear cube Tyre Spring Stiffness	k_{t3}	1200 N/m^3

Table C.1: System parameters of Non-linear Quarter Car model

References

[1] S. Sharma, M. Chouksey, V. Pare, and P. Jain, "Modal and frequency response characteristics of vehicle suspension system using full car model," *IOP Conference Series: Materials Science and Engineering*, vol. 810, no. 1, p. 012056, mar 2020. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/810/1/012056

[2] R. Lopes, B. V. Farahani, F. Queirós de Melo, and P. M. G. P. Moreira, "A dynamic response analysis of vehicle suspension system," *Applied Sciences*, vol. 13, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/4/2127

[3] J. E. D. Ekoru, O. A. Dahunsi, and J. O. Pedro, "Pid control of a nonlinear half-car active suspension system via force feedback," in *IEEE Africon '11*, 2011, pp. 1–6.

[4] V. J. S. Leite and P. L. D. Peres, "Pole location control design of an active suspension system with uncertain parameters," *Vehicle System Dynamics*, vol. 43, no. 8, pp. 561–579, 2005. [Online]. Available: https://doi.org/10.1080/00423110420002667O2

[5] R. Desai, A. Guha, and P. Seshu, "A comparison of quarter, half and full car models for predicting vibration attenuation of an occupant in a vehicle," *Journal of Vibration Engineering & Technologies*, vol. 9, no. 5, pp. 983–1001, Jul 2021. [Online]. Available: https://doi.org/10.1007/s42417-020-00278-3

[6] M. W. Sayers and S. M. Karamihas, "The little book of profiling: basic information about measuring and interpreting road profiles," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:109987860

[7] "ISO 8608:2016 available online:," https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/12/71202.html, accessed: 2023-09-30.

[8] C. Gorges, K. Öztürk, and R. Liebich, "Road classification for two-wheeled vehicles," *Vehicle System Dynamics*, vol. 56, no. 8, pp. 1289–1314, 2018. [Online]. Available: https://doi.org/10.1080/00423114.2017.1413197

[9] T. Lenkutis, A. Čerškus, N. Šešok, A. Dzedzickis, and V. Bučinskas, "Road surface profile synthesis: Assessment of suitability for simulation," *Symmetry*, vol. 13, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2073-8994/13/1/68

[10] M. Agostinacchio, D. Ciampa, and S. Olita, "The vibrations induced by surface irregularities in road pavements – a matlab® approach," *European Transport Research Review*, vol. 6, pp. 267–275, 09 2013.

[11] M. Uchiyama, "Formation of high-speed motion pattern of a mechanical arm by trial," *Transactions of the Society of Instrument and Control Engineers*, vol. 14, no. 6, pp. 706–712, 1978.

[12] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *J. Field Robotics*, vol. 1, pp. 123–140, 1984. [Online]. Available: https://api.semanticscholar.org/CorpusID:26645449

[13] M. Norrlöf and S. Gunnarsson, "Time and frequency domain convergence properties in iterative learning control," *International Journal of Control*, vol. 75, no. 14, pp. 1114–1126, 2002. [Online]. Available: https://doi.org/10.1080/00207170210159122

[14] M. Norrlof, "An adaptive iterative learning control algorithm with experiments on an industrial robot," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 245–251, 2002.

[15] D.-I. Kim and S. Kim, "An iterative learning control method with application for cnc machine tools," *IEEE Transactions on Industry Applications*, vol. 32, no. 1, pp. 66–72, 1996.

[16] M. Pandit and K.-H. Buchheit, "Optimizing iterative learning control of cyclic production processes with application to extruders," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 3, pp. 382–390, 1999.

[17] D. D. Roover and O. H. Bosgra, "Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system," *International Journal of Control*, vol. 73, no. 10, pp. 968–979, 2000.

[18] S. S. Saab, "A stochastic iterative learning control algorithm with application to an induction motor," *International Journal of Control*, vol. 77, no. 2, pp. 144–163, 2004.

[19] Y. Q. Chen and K. L. Moore, "A practical iterative learning path-following control of an omni-directional vehicle," *Asian Journal of Control*, vol. 4, no. 1, pp. 90–98, 2002.

[20] C. Mi, H. Lin, and Y. Zhang, "Iterative learning control of antilock braking of electric and hybrid vehicles," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 2, pp. 486–494, 2005.

[21] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[22] I. Rotariu, B. Dijkstra, and M. Steinbuch, "Standard and lifted approaches of iterative and learning control applied on a motion system," 01 2004.

[23] O. Koçan, "Feedback via iterative learning control for repetitive systems," Ph.D. dissertation, Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), 2020.

[24] R. W. Longman, "Iterative learning control and repetitive control for engineering practice," *International Journal of Control*, vol. 73, no. 10, pp. 930–954, 2000. [Online]. Available: https://doi.org/10.1080/002071700405905

[25] O. Markusson, H. Hjalmarsson, and M. Norrlof, "Iterative learning control of nonlinear non-minimum phase systems and its application to system and model inversion," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 5, 2001, pp. 4481–4482 vol.5.

[26] J. Wallen, S. Gunnarsson, and M. Norrlöf, "Some implementation aspects of iterative learning control," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:29649857

[27] M. Pierallini, F. Angelini, R. Mengacci, A. Palleschi, A. Bicchi, and M. Garabini, "A robust iterative learning control for continuous-time nonlinear systems with disturbances," *IEEE Access*, vol. 9, pp. 147 471–147 480, 2021.

[28] A. G. Mohite and A. C. Mitra, "Development of linear and non-linear vehicle suspension model," *Materials Today: Proceedings*, vol. 5, no. 2, Part 1, pp. 4317–4326, 2018, 7th International Conference of Materials Processing and Characterization, March 17-19, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221478531732970X

[29] C. Della Santina and F. Angelini, "Iterative learning in functional space for non-square linear systems," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 5858–5863.

[30] L. Blanken, J. van Zundert, R. de Rozario, N. Strijbosch, and T. Oomen, *Multivariable iterative learning control: analysis and designs for engineering applications*, ser. IET control, robotics and sensors series. United Kingdom: Institution of Engineering and Technology (IET), 2019, pp. 109–143.

[31] L. Blanken, J. Willems, S. Koekebakker, and T. Oomen, "Design techniques for multivariable ilc: Application to an industrial flatbed printer," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 213–221, 2016, 7th IFAC Symposium on Mechatronic Systems MECHATRONICS 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896316321425

[32] H. Zhang, N. Zhang, F. Min, S. Rakheja, C. Su, and E. Wang, "Coupling mechanism and decoupled suspension control model of a half car," *Mathematical Problems in Engineering*, vol. 2016, p. 1932107, May 2016.

Ringraziamenti

Quando alla fine del liceo bisognava scegliere dove proseguire gli studi, non ero certo della scelta di ingegneria, temevo di non farcela, ma decisi comunque di provare. Ora che sono finalmente giunto alla fine, ho capito che la determinazione, la forza di volontà e l'impegno ti permettono di fare qualunque cosa. Tutto ciò, comunque, non sarebbe stato possibile senza una serie di persone che voglio riportare in quest'ultima sezione della mia tesi.

Desidero innanzitutto ringraziare il professor Alberto Dalla Libera che mi ha guidato in quest'ultimo progetto accademico, per la sua disponibilità e comprensione.

Ruggero Baesso, Matteo Mastellaro e Stefano Sartorato di STEP Lab, per aver reso possibile gli esperimenti finali in azienda e Kevin, per la gentilezza e simpatia che ha permesso di rendere un po' meno pesante il tirocinio.

Mohamed e tutti i colleghi di corso che hanno collaborato in qualche progetto con me, per l'impegno e la fiducia nei miei confronti.

Gregorio e Victor, per le lezioni e gli esami passati insieme, i confronti, gli scambi di idee e ovviamente per le grandi serate padovane.

Anna e Luca, per aver condiviso questi ultimi due anni da fuorisede nel motel di via Dorighello 10 che mi ha permesso di crescere molto.

Carlo, per le chiacchierate scherzose e le chiacchierate terapeutiche, ma anche e forse soprattutto, per le pazzie.

gli amici Alessandro, Andrea, Ezio, Luca e mio fratello Alberto, per essere sempre stati al mio fianco, per le risate e per tutte le avventure passate insieme. Alessandro, grazie anche per le intense sessioni di studio in biblioteca. La tua dedizione mi ha motivato molto.

i miei nonni e in particolare il nonno Vittorio per la pazienza, la fiducia e l'immenso sostegno.

i miei genitori, per non avermi mai fatto mancare nulla, per tutti i sacrifici e per il supporto in tutti i miei momenti di difficoltà.

Grazie ♡

Marco