SAPIENZA
UNIVERSITÀ DI ROMA

Dipartimento di ingegneria Informatica, Automatica e Gestionale
(DIAG)

Generation of minimum-sensitivity trajectories for differential drive robot

Professor:

Prof. Giuseppe Oriolo

Tutor:

Dott. Tommaso Belvedere
Dott. Francesco D'Orazio

Autori:

Antonio Scardino
Marco Nacca
Niccolò Piraino

Anno accademico 2022/2023

Contents

1 Introduction

When one talks about automatic robot motion, we mainly refer to the conditions under which robots operate and thus follow a certain path. One of the main problems in this context concerns the parametric uncertainty in the real world. In fact, the models used in control actions only approximate the real-world behavior of the system, but, obviously, cannot take into account variations that may occur in the external world. This causes situations in which the performance of the control deviates greatly from the ideal situation, generating unsatisfactory results.

Several approaches have been implemented to deal with this problem, including the use of online parameter estimation while the robot performs its task or using robust control. In the former case, which derives from the adaptive control paradigm [1]), unintended transient behavior may occur and in general it is difficult to guarantee stability to the dynamics of the system with the union of the estimator. Instead, the second case, so the use of a robust control [2], lacks generalization, as the control must be adapted to the system considered in the particular situation, and in general, the increased robustness in control causes less accuracy in the execution of the control task.

The approach used in the referenced paper [3], is to improve the robustness of the robot control concerning the parametric uncertainties, by creating an optimal trajectory by minimizing the sensitivity of the state to parameter uncertainty. This minimization is accomplished, as we shall see later, using a gradient descent algorithm. In this way, this optimization will generate a better tracking performance, going only to modify the trajectory itself so that the effect of the not complete knowledge of the parameters is reduced in the control and thus more accurate even in the real world. This optimization process can be used for any choice of the control action since it does not act directly on the latter but, through trajectory optimization, improves the results obtained from a control task.

Starting from this, what is done in this report is to implement the process and simulate it on Matlab, applying it to two different trajectories, to demonstrate its efficiency in different cases. In our case, the optimization process is used on a tracking control process, where a Differential Drive robot (DDR) is used with the addition of a DFL control. In order to implement these latter, have been taken as a reference both Paper [3] and Paper [4], where the latter was also consulted for further clarification.

2 Problem Development

The problem starts by considering a generic nonlinear system

$$\dot{q} = f(q, u, p) \tag{1}$$

where $q \in \mathbb{R}^{n_q}$ is the state, $u \in \mathbb{R}^{n_u}$ is the input and $p \in \mathbb{R}^{n_p}$ is the parameters vector. It has been considered a desired trajectory $r_d(t) \in \mathbb{R}^{n_r}$, with $t \in [t_0, t_f]$ that should be followed by an n_r-output of the state $r(q)$. The problem is a redundancy task because $n_r \leq n_q$. For our purpose, can be consider the desired trajectory to belong to a group of polynomial curves $r_d \equiv r_d(\boldsymbol{a}, t)$, where $\boldsymbol{a} \in \mathbb{R}^{n_a}$ are the coefficients that represents the polynomial. With this consideration, the trajectory is a function that depends not only on time $\boldsymbol{t}$ but also on the parameters $\boldsymbol{a}$. In fact, by selecting values for these coefficients, a single curve (trajectory) can be defined. In the following sections this has been explained, also how generate these coefficients and thus, our initial unoptimal trajectory.

In this situation, it can be assumed that exists a tracking controller that follows exactly the desired trajectory r_d and namely that it has an error $e(t) = r_d(t) - r(q(t)) = 0$. Consider a generic non-linear control that will then have the following form:

$$\begin{cases} \dot{\xi} = g(\xi, q, r_d(t), p_c) \\ u = h(\xi, q, r_d(t), p_c) \end{cases} \tag{2}$$

where $\xi \in \mathbb{R}^{n_\xi}$ are the internal state of the controller. In this case, p_c are the nominal parameters of the system, which may differ from what are the actual parameters in the real world.
Regarding the latter, two possible cases has been considered:

- $p = p_c$, this is the "ideal case" where the controller knows exactly the real system parameters and so it can perform the tracking task properly, with an error $e(t_0) = 0, \forall t \in [t_0, t_f]$. This is so-called *nominal case*;
- $p \neq p_c$, this is the *perturbed case* in which the controller does not know exactly the real system parameters and so the tracking task cannot be exactly realized. In this situation, the error may not converge to 0 because the nominal values of parameters, which are the one used by the control, are different from the real ones p.

Given this, the purpose is to minimize the tracking error $e(t)$ in the presence of parameter uncertainty, and so in the case in which $p \neq p_c$. To do this,the goal is to find the optimal parameters of the desired trajectory $r_d(a_{opt}, t)$ to achieve the least effect to variation in system parameters.

3 Closed-Loop State Sensitivity

Consider the following quantity, that is the variation of the state of the system concerning changes in parameters evaluated in the nominal parameters. This is called *state sensitivity*, as cited in [3].

$$\Pi(t) = \frac{\partial q(t)}{\partial p}|_{p=p_c} \tag{3}$$

The goal of the reference paper is to minimize some norm of this *sensitivity* $\Pi(t)$, written respect to the coefficients of the trajectory a. In this way, an optimal trajectory $r_d(a_{opt}, t)$ can be found, computed from the coefficients a_{opt} which minimize the sensitivity $\Pi(t)$, in such a way that the state evolution $q(t)$ in the *perturbed case* is close as possible to the evolution in the *nominal case*, where the tracking task is executed perfectly by assumption.

In general, the sensitivity $\Pi(t)$ in closed-form cannot be computed directly, but it's possible to use its dynamics to obtain an estimate through the use of integration. It's known, from the (2) that the control input u is indirectly affected by variation in p, because it is a function of the state q and of the control state ξ. To take into account this, it is considered:

$$\Pi_\xi(t) = \frac{\partial \xi(t)}{\partial p}|_{p=p_c} \tag{4}$$

that represent the sensitivity of the *controller state* write respect to vector of parameters p. From these, a system of differential equations has been extracted:

$$\begin{cases} \dot{\Pi} = \frac{\partial f}{\partial q}\Pi + \frac{\partial f}{\partial p} + \frac{\partial f}{\partial u}(\frac{\partial h}{\partial q}\Pi + \frac{\partial h}{\partial \xi}\Pi_\xi) \\ \dot{\Pi}_\xi = \frac{\partial g}{\partial q}\Pi + \frac{\partial g}{\partial \xi}\Pi_\xi \end{cases} \quad \Pi_\xi(t_0) = 0,\, \Pi(t_0) = 0 \tag{5}$$

where f represents the system dynamic, h is the control inputs and g is the dynamics of control states.

As previously mentioned, the optimization problem consists in looking for the optimized coefficients a_{opt} which will form the optimal trajectory. To find these, as said before, it is used a algorithm of the descending gradient where the aim is to update the vector of coefficients, following the opposite of direction of the gradient of the sensitivity Π respect to the coefficients a, until obtaining the vector of the coefficients optimal a_{opt} in correspondence of the global minimum of sensitivity Π. The trajectory obtained from these coefficient will be so less sensitive to changes in DDR parameters. For this reason, it is necessary to define how to obtain an expression for the gradient of Π write respect each of the optimization coefficients a_i. Thus, it has been obtained:

$$\Pi_{a_i}(t) = \frac{\partial \Pi(t)}{\partial a_i}|_{p=p_c} \tag{6}$$

We also need, in the same way as the sensitivity, the gradient of the controller state written with respect to a_i which is defined as follows:

$$\Pi_{\xi_{a_i}}(t) = \frac{\partial \Pi(t)}{\partial a_i}|_{p=p_c} \tag{7}$$

As before, for the closed-loop situation, Π_{a_i} cannot be computed directly but we will need to define the dynamic of these quantities:

$$\Gamma_i(t) = \frac{\partial q(t)}{\partial a_i}|_{p=p_c} \tag{8}$$

$$\Gamma_{\xi_i}(t) = \frac{\partial \xi(t)}{\partial a_i}|_{p=p_c} \tag{9}$$

These are the gradient of the system/control states written respect to the optimization variables. Exactly as did for the (5), the evolution of $\Gamma_i(t)$, $\Gamma_{\xi_i}(t)$ can be obtained by forward integration of their dynamic:

$$\begin{cases} \dot{\Gamma}_i = \frac{\partial f}{\partial q}\Gamma_i + \frac{\partial f}{\partial u}\left(\frac{\partial h}{\partial q}\Gamma_i + \frac{\partial h}{\partial \xi}\Gamma_{\xi_i} + \frac{\partial h}{\partial a_i}\right) \\ \dot{\Gamma}_{\xi_i} = \frac{\partial g}{\partial q}\Gamma_i + \frac{\partial g}{\partial \xi}\Gamma_{\xi_i} + \frac{\partial g}{\partial a_i} \end{cases} \quad \Gamma_{\xi_i}(t_0) = 0,\, \Gamma_i(t_0) = 0 \tag{10}$$

Once these results are obtained, the vector u_{a_i} has been defined:

$$u_{a_i} = \frac{\partial h}{\partial q}\Gamma_i + \frac{\partial h}{\partial \xi}\Gamma_{\xi_i} + \frac{\partial h}{\partial a_i} \tag{11}$$

Computing the Γ_i and the Γ_{ξ_i} with (10) and taking into account the product between tensors, as explained in [3]:

$$\frac{\partial A}{\partial x} \circ b = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}(\frac{\partial [A(x)]_{i,j}}{\partial x} b)\iota_i k_j^T \in R^{n_1 \times n_2}$$

where $A(x) \in \mathbb{R}^{n_1 \times n_2}$ is a matrix function of a vector quantity $x \in \mathbb{R}^{n_3}$, $b \in \mathbb{R}^{n_3}$ a vector and ι_i, k_i are the $i-th$ canonical base vector of R^{n_1}, R^{n_2} respectively, the dynamics of the gradient can be defined as follow:

$$\begin{cases} \dot{\Pi}_{a_i} = \left(\frac{\partial f_q}{\partial q} \circ \Gamma_i + \frac{\partial f_q}{\partial u} \circ u_{a_i}\right)\Pi + f_q\,\Pi_{a_i} + \frac{\partial f_p}{\partial q} \circ \Gamma_i + \\ \quad \frac{\partial f_p}{\partial u} \circ u_{a_i} + \left(\frac{\partial f_u}{\partial q} \circ \Gamma_i + \frac{\partial f_u}{\partial u} \circ u_{a_i}\right)\left(h_q\,\Pi + h_\xi\,\Pi_\xi\right) + \\ \quad f_u\left(\left(\frac{\partial h_\xi}{\partial q} \circ \Gamma_i + \frac{\partial h_q}{\partial \xi} \circ \Gamma_\xi + \frac{\partial h_q}{\partial a_i}\right)\Pi + \right. \\ \quad \left.\left(\frac{\partial h_\xi}{\partial q} \circ \Gamma_i + \frac{\partial h_\xi}{\partial \xi} \circ \Gamma_\xi + \frac{\partial h_\xi}{\partial a_i}\right)\Pi_\xi + h_q\,\Pi_{a_i} + h_\xi\,\Pi_\xi a_i\right) \\ \dot{\Pi}_{\xi_{a_i}} = \left(\frac{\partial g_q}{\partial q} \circ \Gamma_i + \frac{\partial g_q}{\partial \xi} \circ \Gamma_{\xi_i} + \frac{\partial g_q}{\partial a_i}\right)\Pi + \\ \quad \left(\frac{\partial g_\xi}{\partial q} \circ \Gamma_i + \frac{\partial g_\xi}{\partial \xi} \circ \Gamma_{\xi_i} + \frac{\partial g_\xi}{\partial a_i}\right)\Pi_\xi + g_q\,\Pi_{a_i} + g_\xi \Pi_{\xi_{a_i}} \end{cases} \tag{12}$$

with $\dot{\Pi}_{a_i}(t_0) = 0$, $\dot{\Pi}_{\xi_{a_i}}(t_0) = 0$.

For simplify the notation, in the differential system (12), a_b has been indicated as the derivative of a with respect to b, i.e. $\partial a/\partial b$, where $a \in [f, h, g]$ and $b \in [q, p, \xi]$. So for example, considering this notation, f_q correspond to $\partial f/\partial q$ and so on.

4 Optimization

As done in [3], given the reference trajectory $r_d(a, t)$ defined over a time interval $t \in [t_0, t_f]$ and the controller, the optimal coefficients vector is defined as:

$$a_{opt} = \arg\min_{a \in A} ||\Pi(t_f)|| \tag{13}$$

where $||\Pi(t_f)||$ is the norm of the sensitivity matrix Π and A that is the set of possible coefficients optimization values. In this case, the trajectory is being optimized only at the end time t_f, in such a way as to get the system to the predetermined endpoint with better accuracy, obviously in the perturbed case ($p \neq p_c$). So the purpose is improving the final control error without worrying about the error variation the rest of the time.

As we will elaborate on in the next section, the paper adopts a *gradient-based method* to make the optimization and then to find the trajectory coefficient values a_i corresponding to the minimum sensitivity Π. To do this, it is used as our loss function a trace operator of the 'squared' version of the sensitivity matrix Π, following what was done in [3]:

$$Loss = \frac{1}{2} Tr(\Pi^T(t_f)\Pi(t_f)) \tag{14}$$

In this way, the vector a_{opt} obtained at the end of the algorithm is the one which minimizes the function of *Loss* and so the correspondent trajectory should be the one less influenced by the parameters p variations.

In our situation, the trajectory $r_d\,(a, t)$ is defined with different constraints on the initial and final position and velocity and other continuity constraints on the breakpoints (a better explanation will follow in the next sections). With this in mind, these can be written in the following matrix form $M\,a = d$ where d are the values of the different constraints and M is the matrix constraint, which, as will be specified in the next chapter, is a matrix constructed in such a way as to be no-quadratics, considering fewer constraints than the number of coefficients, so to consider a situation of redundancy in such a way that the optimization process could have some effect. For this, it is used the pseudo-inverse because being the matrix not square, the inversion is not

possible. Starting from this, the trajectory coefficients vector a can be optimized with a null-space approach, while obviously maintaining the same constraints M defined for the starting trajectory, according to the following law:

$$\dot{a} = k_1 \, M^{\dagger} \, (d - Ma) + k_2 \, (I - M^{\dagger} M) \nu \tag{15}$$

where and $k_1 > 0$, $k_2 > 0$ are suitable gains and where

$$\nu_i = -\frac{1}{2} \frac{\partial \, Tr(\Pi^T(t_f) \, \Pi(t_f))}{\partial a_i} = -Tr(\Pi^T(t_f) \, \Pi_{a_i}(t_f)) \tag{16}$$

is the negative gradient of the cost function (13).

5 Problem Setup

The case study concerns the optimization of the trajectory $r_d(a, t)$ of a differential drive robot (DDR) in the presence of parameter perturbations ($p \neq p_c$).

Let $q = [x, y, \theta] \in R^3$ be the DDR state in a world frame $F_W = \{O_W; x_W, y_W\}$, with (x, y) being the planar position in F_W and θ the differential drive robot orientation.
Let also be v, ω the DDR linear/angular velocities and w_r, w_l the right/left wheel spinning velocities. These quantities are related in the following way:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b} & -\frac{r}{2b} \end{pmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = S \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

where r stands for the wheel radius and b for the distance between the wheels. The actual inputs of a differential drive robot are $u = \begin{bmatrix} \omega_R & \omega_L \end{bmatrix}^T$. For our task, parameter variation has been considered, so it has been set a vector containing the same $p = \begin{bmatrix} r & b \end{bmatrix}^T$ as the vector of system parameters w.r.t. which the closed-loop state sensitivity will be evaluated.
The differential drive robot dynamics has the expression:

$$\dot{q} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} Su = GSu = f(q, u, p)$$

The control task was constructed such that the DDR, having as its position $r(q) = \begin{bmatrix} x & y \end{bmatrix}^T$ must follow a reference trajectory $r_d(a, t)$. This tracking control task can be used by implementing a

Dynamic Feedback Linearization controller (with integral action), to guarantee the best possible tracking performance in the nominal case ($p_c = p$). The control strategy can be summarized as follow:

$$\xi = \begin{bmatrix} \xi_v & \xi_x & \xi_y \end{bmatrix}^T \in R^3$$

is the controller states, where ξ_v represents the dynamic extension of the DDR linear velocity v, (ξ_x, ξ_y) the states of the integral action.
Differentiating twice the DDR position $r(q)$, has been obtained this, as seen in [3]:

$$\ddot{r}\,(q) = \begin{pmatrix} \cos(\theta) & -\xi_{\,v}\sin(\theta) \\ \sin(\theta) & \xi_{\,v}\cos(\theta) \end{pmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = A(q,\xi) \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix}$$

$$G = \begin{pmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{pmatrix}$$

$$S = \begin{pmatrix} r/2 & r/2 \\ r/(2b) & -r/(2b) \end{pmatrix}$$

So from this, the dynamic model of our differential drive robot is:

$$\dot{q} = GSu = \begin{pmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r/2 & r/2 \\ r/(2b) & -r/(2b) \end{pmatrix} \begin{bmatrix} \omega_R & \omega_L \end{bmatrix}^T = f\,(q,u,p)$$

Instead, considering the following vectors:

$$\begin{cases} \dot{r_\xi} = [\cos(\theta)\xi_v,\ \sin(\theta)\xi_v] \\ \xi_{xy} = [\xi_x,\ \xi_y]^T \\ \eta = \ddot{r} + k_v(\dot{r_d} - \dot{r_\xi}) + k_p(r_d - r) + k_i\xi_{x,y} \end{cases} \tag{17}$$

Where $k_v > 0$, $k_p > 0$, $k_i \geq 0$ are the control gains, the dynamics of control states can be defined as follow:

$$\dot{\xi} = \begin{pmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} A^{-1}\,\eta \\ r_d - r \end{pmatrix} = g\,(\xi, q, r_d(t)) \tag{18}$$

and the robot control input as:

$$u = S_c^{-1} \begin{pmatrix} \xi_v \\ \begin{bmatrix} 0 & 1 \end{bmatrix} A^{-1}\eta \end{pmatrix} = h\,(\xi, q, r_d(t), p_c) \tag{19}$$

6 Realization

Our work focused on implementing the previous optimization process in Matlab language and simulating it on a *differential drive robot* (DDR). The behavior of the system is represented by the dynamic described before and, to make the situation more realistic, it has been set as nominal value of the parameters $p_c = \begin{bmatrix} 0.1 & 0.25 \end{bmatrix}^T$ m.

The optimization is so applied on a trajectory tracking task executed with the use of a *Dynamic Feedback Linearization* (DFL) control as explained in [4], to reduce the final time control error in the perturbed case $p \neq p_c$. In particular, the simulation time t_f is set at 5 seconds while the control frequency f at 500 Hz (thus with a corresponding time step δ of 0.002 s). These values are chosen in such a way as to come as close as possible to the assumption made by the paper, namely that of having a nominal control whose error tends to zero for each time considered.

For this purpose, two different trajectories have been considered to evaluate the optimization process in two different situations. Of course, as can be seen in the next section, it has been applied to both of these the DFL control in both the nominal and perturbed case, to execute the trajectory tracking task, and then the optimization. Finally, the optimization can be analyzed considering different types of error (that are explained in the statistic analysis section) taking particular attention to the difference between the position errors in the perturbed case with the optimal and unoptimal trajectory. Furthermore, a situation with the integral action and without has been analyzed. This demonstrates that the optimization obtains better results also in a situation with integral action, which in itself already has a beneficial effect in that it reduces constant disturbances at a steady state.

Let us then go on to explain in more detail the various steps used to perform our simulations, starting from the creation of the trajectory on which the task was executed.

6.1 Trajectory Generation

The first step in our realization is the *trajectory generation*, in such a way as to have an initial unoptimized, possibly not robust trajectory, $r_d(a_{nopt}, t)$ for the differential drive robot (DDR) to follow, controlled by the DFL. This is created using 3 polynomial splines of the 15-degree. The coefficients of this trajectory a_{nopt} has been simply obtained by multiplying the matrix of constraints M with the vector d of corresponding values:

$$a_{nopt} = M^{\dagger}\, d \tag{20}$$

In our case, 12 constraints has been imposed:

- The initial p_i and final position p_f which the differential drive robot should assume has been set and the initial v_i and final velocity v_f;
- Other two constraints are used to set the position of the two points of break considered, so p_1 and p_2, in such a way as to make the trajectory assume the desired shape and also to prevent the robot from turning back on its way (thus avoiding negative velocities) ;
- Finally it has been defined three continuity constraints for each of the breakpoints, going to set the continuity in position, velocity, and acceleration. In particular, the last one is used to obtain a more smooth trajectory without too abrupt changes in direction;

With this considerations the matrix M assumed the following form, where $t_b = t_f/3$, which is the final time of each spline:

$$M = \begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t_b^4 & t_b^3 & t_b^2 & t_b & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
4\,t_b^3 & 3\,t_b^2 & 2\,t_b & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
12\,t_b^2 & 6\,t_b & 2 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & t_b^4 & t_b^3 & t_b^2 & t_b & 1 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 4\,t_b^3 & 3\,t_b^2 & 2\,t_b & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 12\,t_b^2 & 6\,t_b & 2 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_b^4 & t_b^3 & t_b^2 & t_b & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4\,t_b^3 & 3\,t_b^2 & 2\,t_b & 1 & 0
\end{pmatrix}$$

Instead the vector d is the following:

$$d = \begin{bmatrix} p_i,\ v_i,\ 0,\ 0,\ 0,\ p_1,\ 0,\ 0,\ p_2,\ 0,\ p_f,\ v_f \end{bmatrix}^T$$

as can be seen, the vector $d \in \mathbb{R}^{n_a}$ and $M \in \mathbb{R}^{n_c \times n_a}$, where n_a is the number of trajectory coefficients (for our case 15) and n_c is the number of the constraints.

The matrix M is voluntarily chosen as no-quadratics, considering fewer constraints than the number of coefficients, so to consider a situation of redundancy in such a way that the optimization process could have some effect. For this in (20) it is used the pseudo-inverse because being the matrix not square, the inversion is not possible.

As mentioned earlier, with this process are generated two trajectories having different curvilinear trends, to study how the optimization works in two different situations.

6.2 Optimization Process

For the optimization process, mean the set of steps taken to perform the simulation level of the optimization explained above, the main goal of this process is to obtain the trajectory coefficients a_{opt} (and thus a new optimal trajectory $r_d\left(a_{opt}, t\right)$) that reduces the influence of DFL control on the variation of parameters p that may occur in the real world, thus achieving a reduction in control error between the case where the original trajectory is used and the case where the optimal one is considered.

As mentioned before, a gradient-based optimization algorithm to exploit what is expressed theoretically in Section (4) can be used. This algorithm was designed by us as a loop that is executed a certain number of times, established by the hyper-parameter *epoch*. The value of this parameter is chosen, as we will see in the next section where the result of our simulations has been exposed, in such a way as to stop the process when a minimum value of the loss function has been obtained, that allows to achieve a significant optimization of the control error in the perturbed case. This process is composed of different steps:

1. At first, it has been computed the new trajectory starts from the coefficient computed in the previous loop. Obviously, for the first iteration of the algorithm, the nominal trajectory $r_d(a, t)$ has been used, generated as seen in the section before;

2. On this trajectory it has been applied the DFL control in the nominal case, i.e. assuming that the nominal values of the parameters, known by the control, are those of the system ($p = p_c$). From this, the state of the system q_{nom}, the input u_{nom}, and the control state ξ_{nom} has been obtained, which are needed to calculate the derivative terms used for the calculation of the sensitivity matrix Π and its gradient concerning the coefficients Π_{a_i};

3. The Π and Π_{a_i} matrix are computed integrating the dynamics described in the Section (3) with the use of Euler integration;

4. The matrices computed in the previous step are used to calculate the negative gradient of the cost function ν_i which is computed for each coefficient of the trajectory, using the equation (16). This ν_i is computed for each of the trajectory coefficient a_i using the value of the sensitivity matrix at the final time t_f, i.e. $\Pi(t_f)$ and its gradient with respect to the considered trajectory coefficient, evaluated always at t_f, i.e. Π_{a_i}. This term makes the update law following the negative direction of the gradient of sensitivity, in such a way

to obtain, at the end of the algorithm, the coefficients a_{opt} which minimize the sensitivity matrix at time t_f;

5. Computed ν_i, the new coefficients a_{opt} has been calculate, going to integrate with Euler the update law (15). The two parameters k_1 and k_2 are chosen case-by-case to obtain a better optimization on the control error in the perturbed case;

6. At each loop the *Loss function* has been computed, using the equation (14). In this way it is possible to verify if the process is working correctly by analyzing if the loss continues to decrease during the various loops. Indeed, in this way we obtain the certainty that the algorithm continues to minimize the sensitivity Π;

At the end of the loop, the coefficient obtained a_{opt} are used to compute the new optimal trajectory $r_d\left(a_{opt}, t\right)$. Furthermore, the sensitivity matrix has been obtained for each of the loops, in such a way as to analyze the different components and the evolution during the loop.

It will be seen, in the simulations present in the next section, how the optimal trajectory generated resembles the unoptimized trajectory in the initial section and takes on a different trend (more curvilinear) when it is about to reach the final position. This happens because, taking into account that the optimization has been applied on the error at the final time $e\left(t_f\right)$, the robot manoeuvres in such a way as to be more accurate in reaching the final position p_f, maybe even increasing the tracking error in the instants preceding the final one.

6.3 Statistical analysis

Once the optimal trajectory is generated, by implementing the DFL controller with the perturbed parameters on both the optimized and the unoptimized trajectory, it can be actually seen the optimization occurred. At first, it is considered a particular situation where the radius of the wheels r by an 80% of the nominal value and the distance of the wheel b by a 120%, $p_{our} = [0.8 * r_c,\, 1.2 * b_c]^T$ has been perturbed. Therefore, the errors that will be presented in the 'Perturbed Control' sections (Section (7.3), Section (7.6), Section (8.3) and Section (8.6)) are calculated considering this variations.

In addition, a statistical analysis on the optimization that has occurred has been considered by going to create a vector of **r** and **b** parameters with values that vary randomly in a range between 80% and 120% of the nominal one, and then, by running the DFL controller with these parameters perturbed for $N = 20$ times. In this way, for each of the simulations, a different perturbation on the two parameters can be obtained and therefore can be seen on how many occasions an optimization (how many times the final error of the perturbed control on the Optimal trajectory

is smaller than the one on the not robust trajectory) has been obtained.

Moreover, by considering these simulations, a statistical results by calculating the means μ and standard deviations σ on the different errors has been given, as said in [3]. To do this, one must consider the different cases involved, i.e., the various states of the robot obtained by following the Optimal or the not robust trajectory, using the nominal or perturbed parameters within the controller:

- $q_{nom}^{nopt}(t)$ represents the state evolution in the nominal case when tracking the unoptimal reference trajectory. This is constant for each simulation;
- $q_{per,k}^{nopt}(t)$ represent the state evolution in the perturbed case of the k-simulation, when tracking the non robust reference trajectory;
- $q_{nom}^{opt}(t)$ represents the state evolution in the nominal case when tracking the Optimal trajectory. This is constant for each simulation;
- $q_{per,k}^{opt}(t)$ represent the state evolution in the perturbed case of the k-simulation, when tracking the Optimal trajectory;

Defined the various states, the two types of errors used to compute the two statistical functions can be considered. The states obtained with the perturbed parameters are compared with the ones in the nominal case, following both the optimal and unoptimal trajectories. For each simulation k, it has been considered:

- The difference between the state of the robot in the nominal and perturbed case, following the unoptimal trajectory. We named this *error in unoptimal case* ${e_k}^{nopt} = q_{nom}^{nopt}(t) - q_{per,k}^{nopt}(t)$;
- The difference between the state of the robot in the nominal and perturbed case, following the optimal trajectory. We named this *error in optimal case* ${e_k}^{opt} = q_{nom}^{opt}(t) - q_{per,k}^{opt}(t)$;

These two errors were calculated for all $N = 20$ simulations, resulting in the end in twenty ${e_k}^{opt}$ and twenty ${e_k}^{nopt}$. As mentioned above, therefore, using these errors, the mean and standard deviation has been calculated, obtaining two means, one for the optimal case μ_{opt} and one for the unoptimal μ_{nopt}, and two standard deviations, one for the optimal case σ_{opt} and one for the unoptimal one σ_{nopt}. These two statistical components were used to make considerations on the optimization process and allowed us to analyze in more detail the behavior of this in the different cases analyzed, i.e. with the two different trajectories and with the presence or absence of the integral action. All the results are presented and analyzed in the next section.

In the latter situation, we deviated slightly from what was done in the reference document. The reference document [3] considers the error calculated through the components of the state, i.e. position and orientation. In our case instead, the mean and the standard deviations of the errors on the positions and, separately, that on the orientations were considered. This way, instead of having two means and two standard deviations, as mentioned above, we will have four for the optimal situation $(\mu_{tot}^{opt}, \mu_{\theta}^{opt}, \sigma_{tot}^{opt}, \sigma_{\theta}^{opt})$ and four for the non-optimal situation $(\mu_{tot}^{nopt}, \mu_{\theta}^{nopt}, \sigma_{tot}^{nopt}, \sigma_{\theta}^{nopt})$.

7 First Case

In this section the results obtained with different simulations of this optimization process have been presented. As previously mentioned, two different trajectories were used on which the entire optimal process was applied, in order to analyze the behavior of the latter in two different situations. We present the first of the two trajectories and will conclude with the statistics on this result, in order to give more objectivity to what we found.

7.1 First Trajectory with No Integral Action

The following figure shows the first trajectory that was generated. It has a beveled shape that was specifically chosen to obtain a more visible optimization than we would have obtained by constructing a straight trajectory.

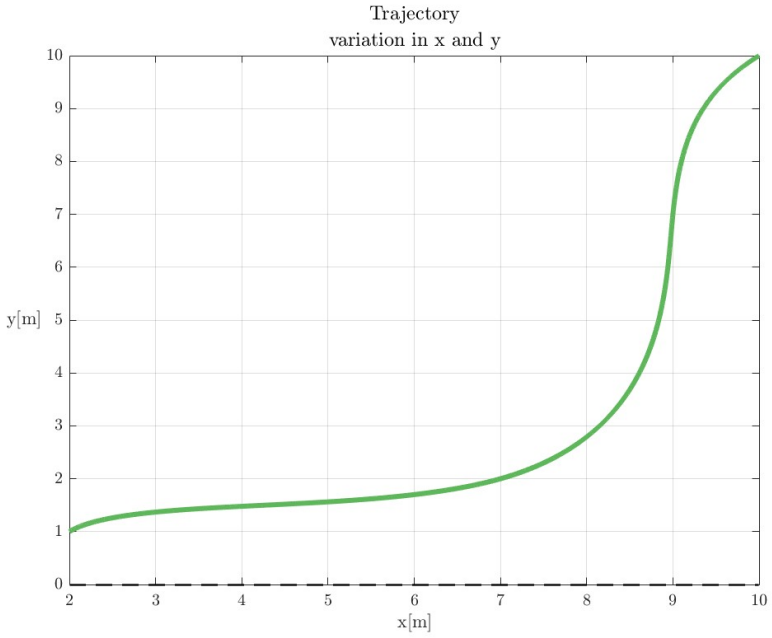

Figure 1: First Trajectory

To create this trajectory, the initial point equal to $p_i = [2\ 1]$ has been set, respectively, and the breakpoints are placed at $p_1 = [7\ 2]$ and $p_2 = [9\ 7]$. Finally, the endpoint is set at $p_f = [10\ 10]$. Moreover, an initial and final velocity has been set not too small, to avoid problems with the DFL control, as said in [4]; in fact, in fact, they have been assigned to values $v_i = v_f = [1\ 1]$.

These velocities are set in this way to reduce the steady-state error of the controller.

In fact, analyzing the Figure (2), the desired linear speed is always much higher than zero just to avoid an increase in error by the DFL control. In particular, an attempt has been made to avoid excessive variations of the latter, introducing, as mentioned above, continuity constraints on accelerations at break points. In addition, this graph is useful for predicting the desired behaviour of the DDR on the trajectory. Actually, the first spline and the second spline are the longest and the robot will need to accelerate to reach the end of the two sections at the set time $2\,t_f/3$. The last spline is the shorter one and, as can be seen, there is a large deceleration of the robot with a consequent decrease in speed.

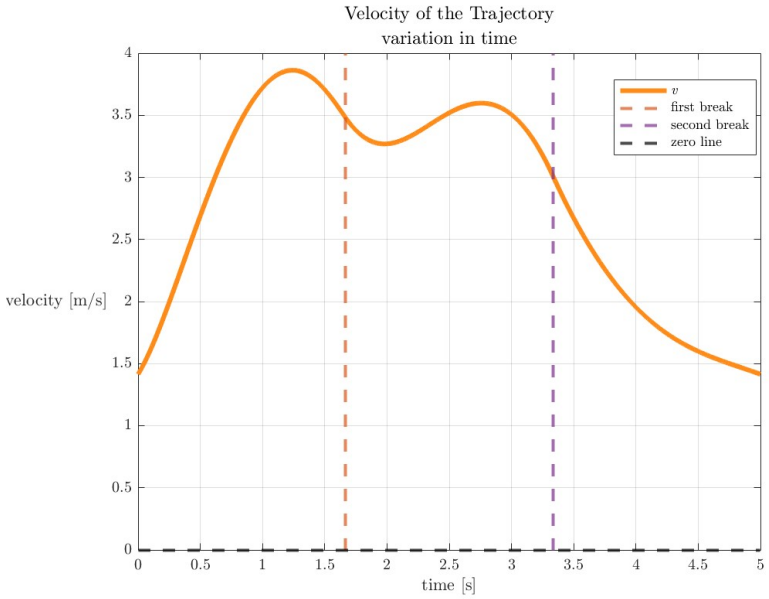

Figure 2: Velocity of the First Trajectory

On this trajectory, a DFL control has been applied to compute trajectory tracking. At first, as explained in the section before, the nominal case ($p = p_c$) has been considered and the gains are chosen in such a way that the error, in this case, is as low as possible (almost zero), in such a manner as to follow the hypothesis made by the paper. They are so set at the following value:

$$k_p = 42 \qquad k_d = 4 \qquad k_i = 0$$

As can be seen, in this first situation there is no integral action. As said before, this will be introduced in the next simulation to finally compare the result obtained in this situation and with the presence of this action.

It has been reported in Fig.(3) the position and orientation error made by the control in the tracking process. It can be seen how the total error e_{tot}, which is the one computed in position, after a transient of $1.8\,mm$ decrease to finally assume a value $0.12\,mm$ at steady state; also the orientation error e_θ is small enough. This demonstrates how the DFL control is almost perfect for the nominal case $p = p_c$.

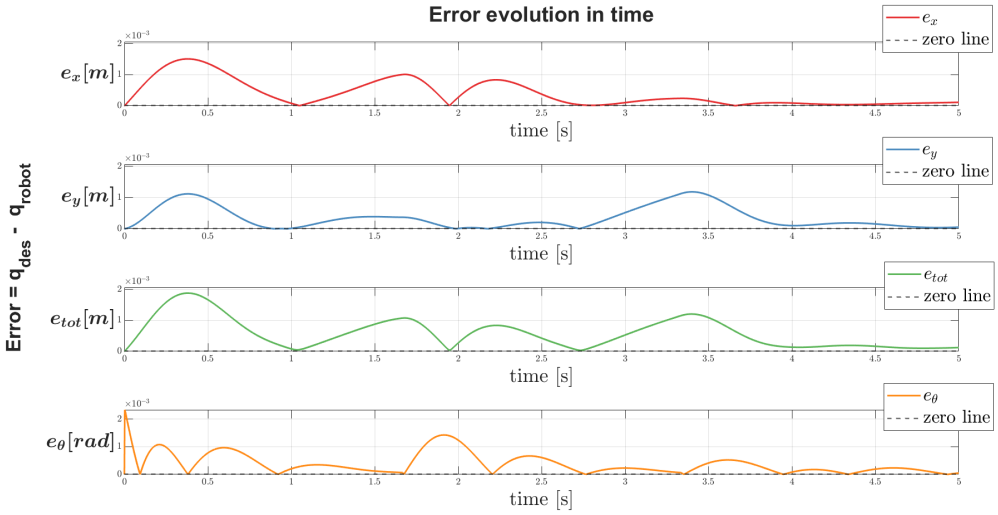

Figure 3: Error Nominal Control on First Trajectory

Finally, the behavior of the DFL-controlled robot in the nominal case in tracking the trajectory and the input velocity generated by the control itself is reported below. The first one confirms the excellent performance mentioned above, as the difference between the robot's course and the trajectory to be followed are almost invisible (as there are millimeter errors). For the second, the input generated by the controller which corresponds to how the robot turns during the simulation can be seen. Remember that has been considered, for our simulations, a differential drive robot, where so the angular velocity of the two wheels is differently related to the orientation the robot must take to follow the predetermined trajectory, through the property of flatness.

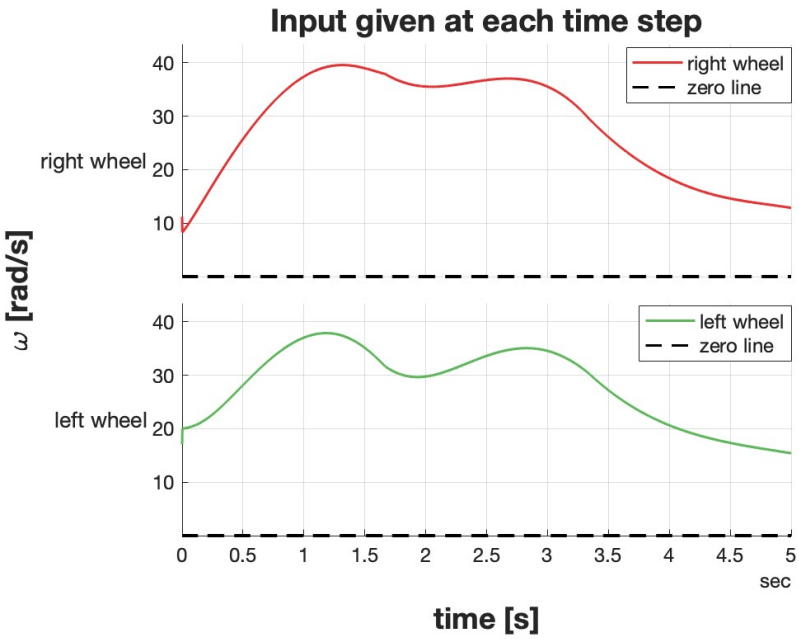

Figure 4: Input Nominal Controller First Trajectory

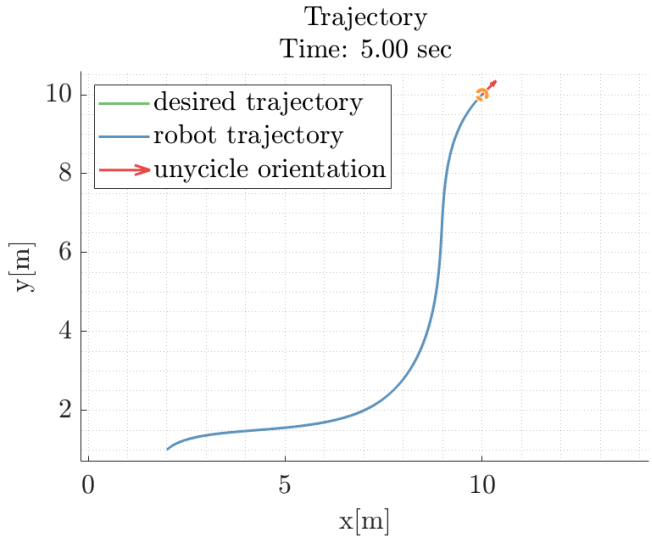

Figure 5: Ideal Control on First Trajectory

Computed the ideal control, as said in the previous section, the necessary data to calculate the sensitivity matrix Π and the gradient Π_{a_i} for performing the optimization process can be determined.

7.2 Optimization on the First Trajectory

For the optimization part, as mentioned earlier, a loop is run in which gains, used in the Formula (15), and the number of epochs is set. In particular, in this situation, the values of these parameters are:

- $k_1 = 3$;
- $k_2 = 0.5$;
- $epochs = 55$;

The values of the two gains, k_1 and $k2$, and the number of *epochs* are chosen by running various simulations with the main objective of having a large difference between the perturbed error obtained by following the unoptimized and the optimized trajectory. In this way, a considerable optimization of the position error e_{tot} is ensured. By setting these parameters, the loss function takes the trend presented in the following Figure (6).

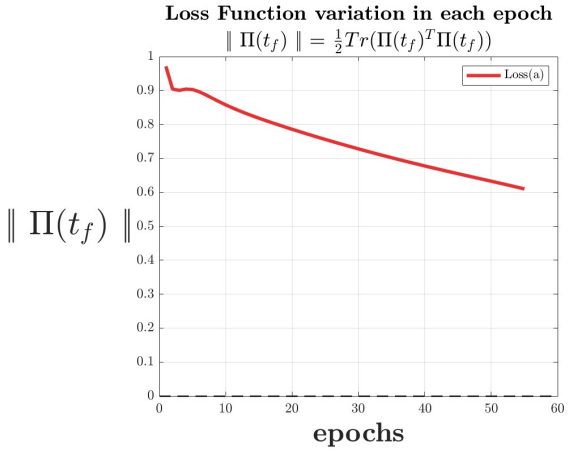

Figure 6: Loss function for First Trajectory

As mentioned in previous sections, the optimization process is stopped at a number of epochs where the minimum value of the loss function is obtained and this allows a significant optimization of the control error in the perturbed case. In fact, after this epoch, the loss tends to increase its value, thus leading to a worse optimization.

In addition to the loss function, the performance of the optimized trajectories at each epoch can be monitored. For each loop and so for each vector of trajectory coefficient a generated by the process, the corresponding trajectory has been printed, in such a way as to find the Figure (7),

where each color corresponds to the trajectory calculated in that specific process loop (or epoch). In particular, a range of colors are used, from purple (trajectory generated by the first loop) to red (trajectory generated by the last loop, as well as optimal trajectory).

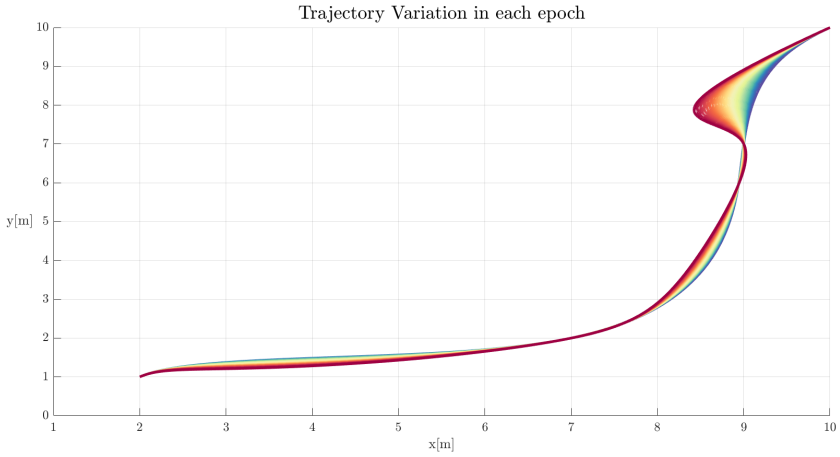

Figure 7: First Trajectory Variation during the Optimization phase
(Color change from purple to red, following color spectrum, indicates increase of epochs)

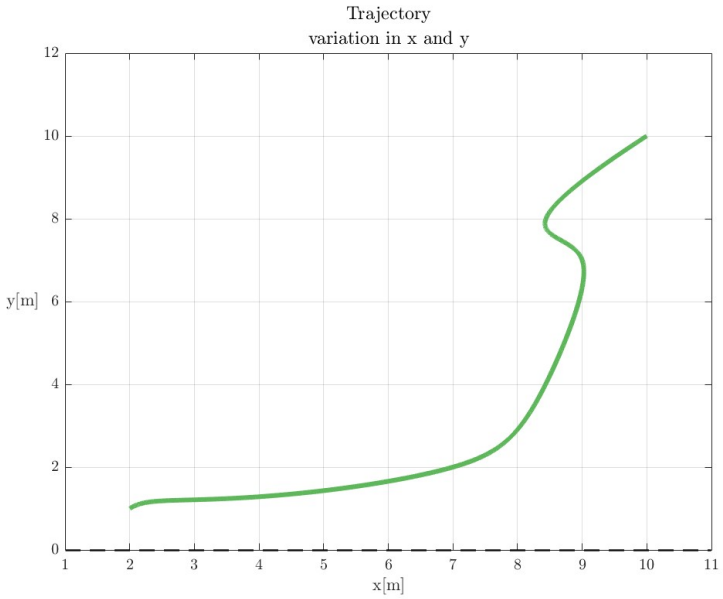

Figure 8: Optimal Trajectory

From what can be seen, at each epoch, an increasingly smooth trajectory is obtained towards the final part t_f, where the optimization is performed. Considering this last aspect, the optimization is free to 'maneuver' in such a way that the robot reaches the final position p_f more precisely. This 'adjustment' causes the optimized trajectory in the last section to arrive more straight at the final point, perhaps increasing the error in this area, but decreasing it at the final time t_f.

As mentioned above, the calculation of the optimized coefficients took place through the calculation, at each epoch, of the sensitivity value and its gradient. Sensitivity is the derivative of the state of the robot with respect to the parameters, therefore relating to the radius r and the distance between the wheels b. In this case, all the sensitivity components for each era have been printed. What has been obtained is an optimization of the variation of the wheel radius since in the derivatives for this, what has been obtained is a decreasing trend. As regards the derivatives with respect to the distance between the wheels, they have assumed an opposite trend for the case discussed above, with the exception of the case θ, where the derivative relating to the radius in the epochs is subject to increase, while with compared to the distance of the wheels undergoes a decrease.

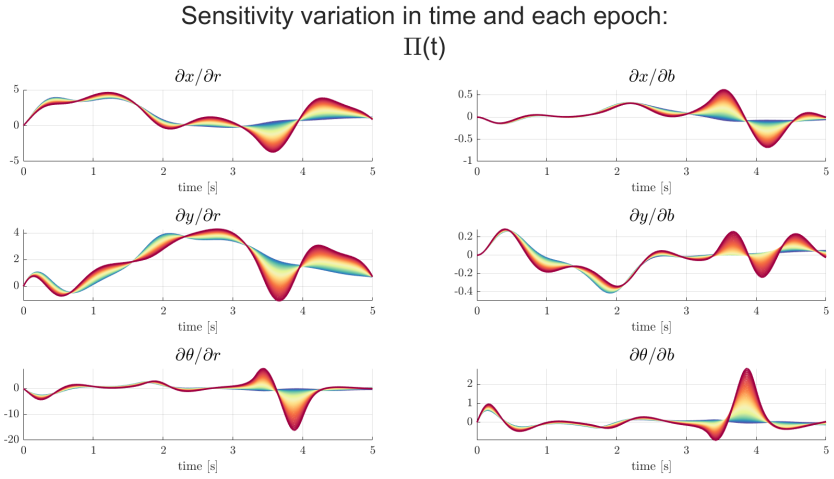

Figure 9: Sensitivity Variation (Color change from purple to red, following color spectrum, indicates increase of epochs)

7.3 Perturbed Control on First Trajectory

In the following section has been shown the errors obtained by performing perturbed control on the optimal trajectory and unoptimal trajectory, to verify the optimization obtained. In this

case a perturbation $p_{our} = [0.8 * r_c,\ \ 1.2 * b_c]^T$ has been considered.

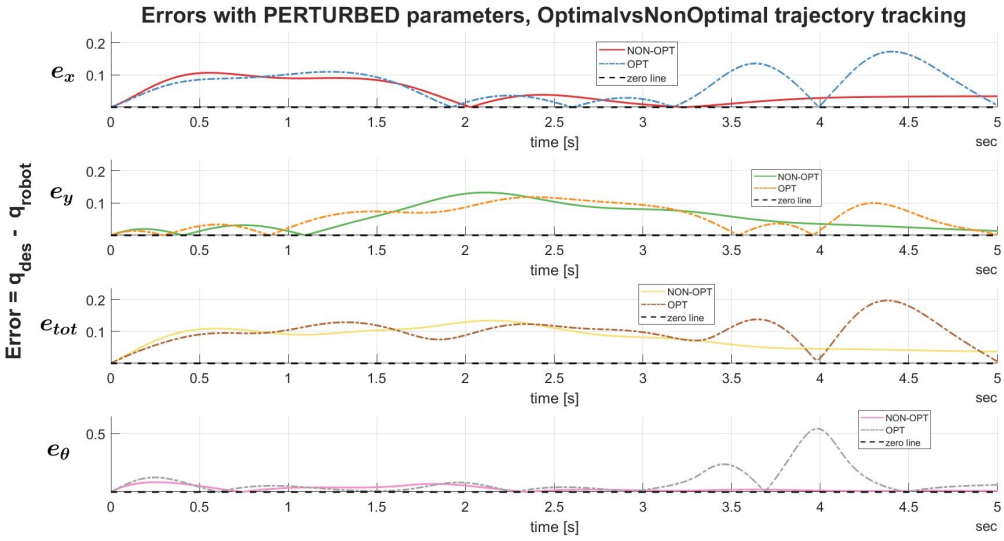

Figure 10: Error in the Perturbed Case

In this case, the error obtained on the non-optimal trajectory is equal to $3.66\,cm$ whereas the one on the optimal trajectory is $0.55\,cm$. So the difference between the two errors, as well as the optimization obtained, is about $3.11\,cm$. Using this specific trajectory, as seen from the graph, a greater optimization is obtained regarding the position of the robot rather than its orientation. A different situation will be found for the second trajectory, in which we have a trajectory that optimizes a lot also on the orientation, unlike the first trajectory in which we had little optimization on the orientation, but we will talk about this later.

From the error just analysed, we will then proceed to calculate the total optimization ratio of the errors on the individual components x, y, θ, to show how much the error is smaller in the optimal case than in the non-optimal case. Considering the error as the difference between the perturbed and the desidered state of the robot, in the optimal and non-optimal case (as considered in the article [3]):

- $e^{nopt}_{pert-nom} = q^{nopt}_{nom}(t) - q^{nopt}_{per}(t)$
- $e^{opt}_{pert-nom} = q^{opt}_{nom}(t) - q^{opt}_{per}(t)$

The difference between the individual components of the states of the optimal case for the unoptimal case is calculated in this way:

$$performance = \frac{\sqrt{e^2_{nopt,\,x} + e^2_{nopt,\,y} + e^2_{nopt,\,\theta}}}{\sqrt{e^2_{opt,\,x} + e^2_{opt,\,y} + e^2_{opt,\,\theta}}} \tag{21}$$

Thus, in the case where the integral action is not considered, a difference is obtained, over the whole state of the robot, of 0.6 smaller than in the unoptimal case. It has been obtained the same value even if it has been considered, in the calculation, only the position.
While, if it has been considered the error calculated by us:

- $e^{nopt}_{pert} = q^{nopt}_{des} - q^{nopt}_{pert}$;
- $e^{opt}_{pert} = q^{opt}_{des} - q^{opt}_{pert}$;

it has been obtained a difference, over the whole state of the robot, of 0.6 smaller than in the unoptimal case. Instead, it has been obtained a difference of 6.7 times smaller for the unoptimal case if has been considered, in the calculation, only the position.

7.4 First Trajectory with Integral Action

In this section,the results obtained has been presented from the optimization process starting from a control that also considers within it the integral action, which from the control point of view takes into account the history and thus how the measurements have varied since the beginning of tracking. So, with the presence of this term, a continuous sum of the control error up has been obtained to the current time and the consequence is the reduction of the steady-state error.

The trajectory used is the one seen above, shown in the Figure (1). The constraints on position and speed are the same and are consequently not shown here to avoid unnecessary repetition. As done before, it has been considered the nominal case ($p = p_c$) with the purpose of computing all that is necessary to calculate the sensitivity matrix and the gradient with respect to the coefficients of the trajectory. Compared with the case previously, given the presence of the integral action, the gains should be set so that the error on the unoptimal trajectory is as low as possible. They take the following values:

$$k_p = 100 \qquad k_d = 3 \qquad k_i = 2$$

As can be seen, in this situation the contribution of integral action is present because the integral gain k_i is different from 0. The position and the orientation error committed by the control in

the tracking process have been reported in Fig (11). It can be seen how the total error e_{tot}, which is the one calculated in position, after a transient of $1.11\,mm$ decreases to finally assume a value $0.8\,mm$ at steady state; the orientation error e_θ is also quite small. This confirms the assumption that the DFL control works perfectly under nominal conditions $p = p_c$.

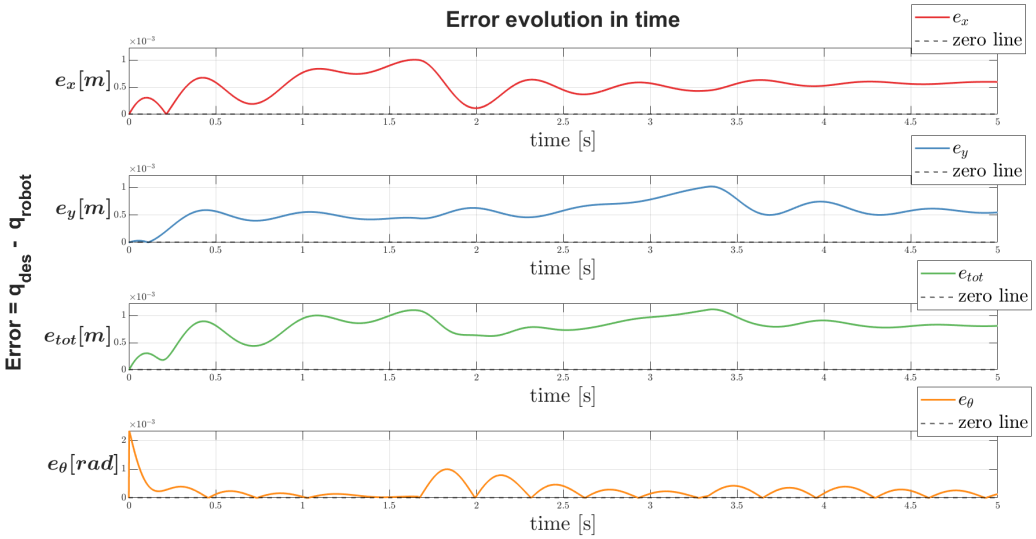

Figure 11: Error Nominal Control with Integral Action on First Trajectory

Another proof of this fact is also given here by the trajectory taken by the controlled differential drive robot with respect to the desired trajectory, shown in the figure below. It can be also seen that clearly the inputs given by the control in this case are almost the same as those given without the integral action. This fact is taken for granted as the trajectory to be followed by the robot is exactly the same.

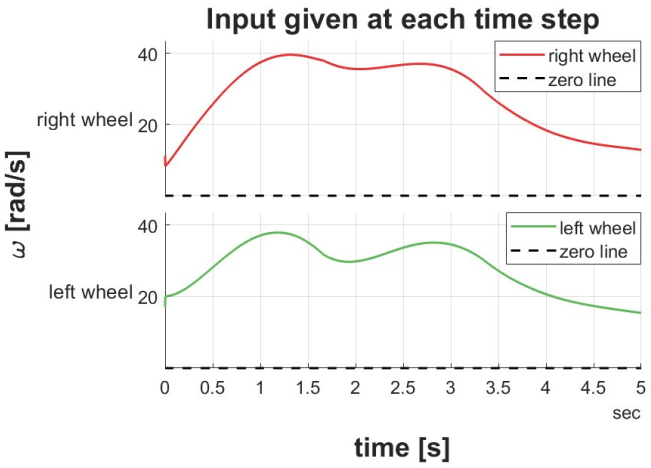

Figure 12: Input Nominal Controller Integral action

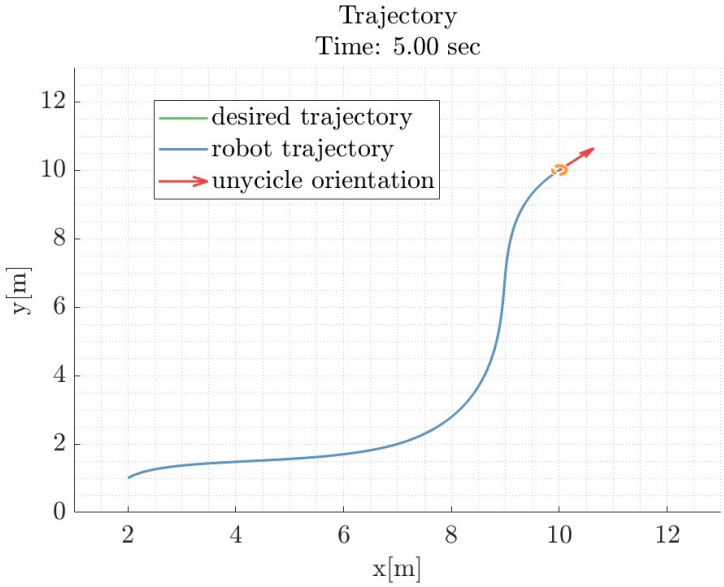

Figure 13: Behaviour of the DDR following the Trajectory Nominal Case Integral action

7.5 Optimization on First Trajectory with Integral Action

As discussed in the previous case, once the ideal control values are obtained, it can proceed by performing the optimization cycle. In this case the integral action in the DFL controller can be taken into account, the gains of the optimization step to prevent the descending gradient algorithm from failing to reach the desired global minimum are set.

For this situation the gains values, chosen empirically, are the following one:

- $k_1 = 3$;
- $k_2 = 5$;
- $epochs = 55$.

As can be seen, the first gain k_1 is chosen of the same value as before but, instead, the second k_2 has a value 10 times higher than that without integral action. The motivation for this choice is linked to the fact that, considering the integral action, the obtainable optimization is generally of the order of millimetres, as the integral contribution leads the error in the final position to be very low. The same also happens for the non-optimal trajectory. The optimization then improves this error but the order of improvement is low and difficult to see visually: considering these aspects, a higher value is chosen value for k_2 in order to obtain a greater and more visible optimization of the error in a shorter time.

As can be seen below, the loss function in this situation assumes a more curvilinear form reaching a minimum around the 55-th epoch, where our simulation is stopped. Obviously, as mentioned above, this allows significant optimization of the control error in the perturbed case.

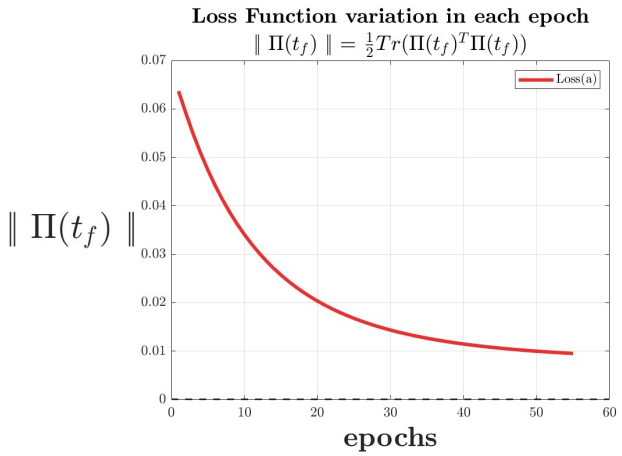

Figure 14: Loss for First Trajectory case with Integral Action

Looking to the Figure (15), the performance of the optimized trajectories at each epoch can be monitored. As can be noted, an increasingly smoothed trajectory at the end point where the optimization occurred has been obtained.

It's important to underline that connecting to what was said earlier, that this variation is not as visible as that without the presence of the integral action. This is because, already having a very low error in the unoptimal trajectory, due to the presence of the integral action, the optimization

process does not need to perform a large 'maneuver' to be more accurate at the end point p_f. In fact, the optimization achieved with the process are in the order of millimeters, but are anyway important as the control error itself on the unoptimal trajectory is in the same order.

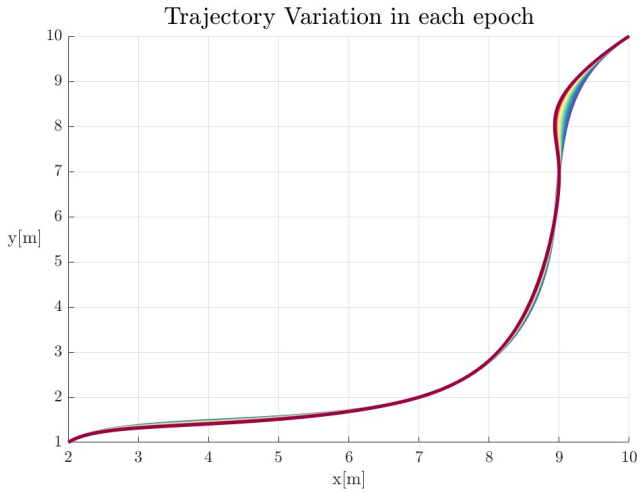

Figure 15: First Trajectory Variation during Optimization phase case with Integral Action (Color change from purple to red, following color spectrum, indicates increase of epochs)

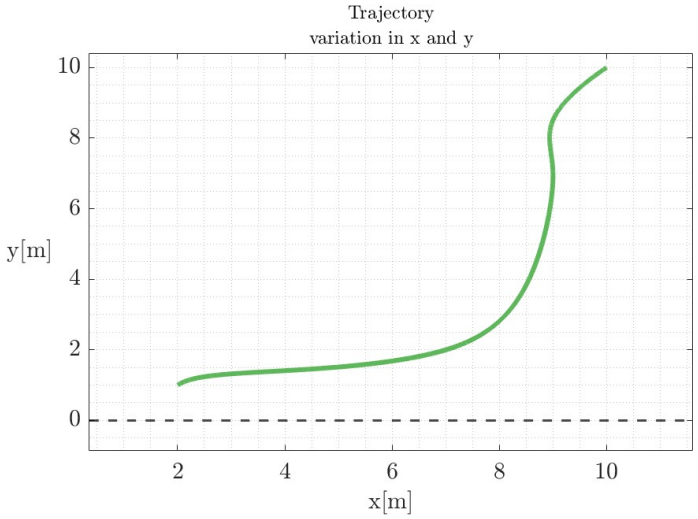

Figure 16: Optimal Trajectory case with Integral Action

As seen above, the trend in sensitivity is the same, where can be seen that a more optimization with respect to one parameter (the wheel radius r) than another is obtained. Also, what can be seen is that, thanks to the properties of the action integral, the sensitivity takes smaller values than in the case considered previously. Also with the integration action, the θ is not very optimized but a slightly better situation respect to the precedent one is obtained. In fact, this component there grows slightly or remains constant over the epochs, showing how the integral action thus slightly improves optimization even on the DDR orientation.

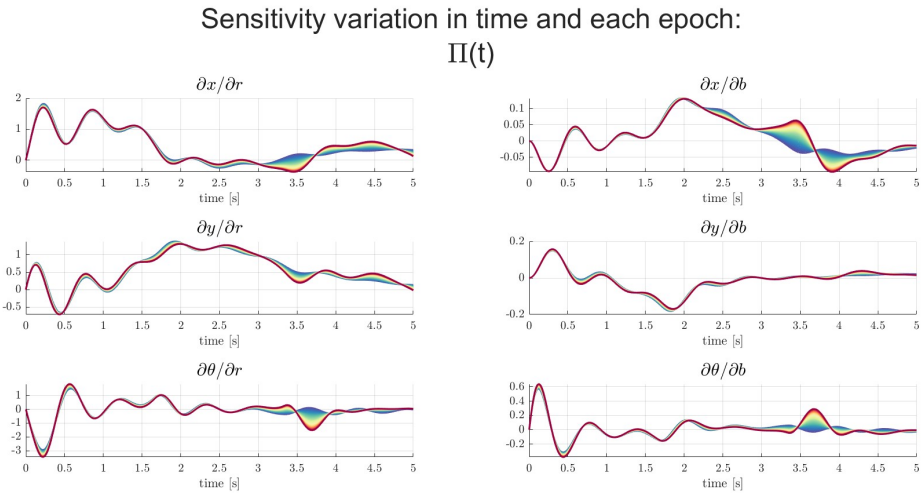

Figure 17: Sensitivity case with Integral Action (Color change from purple to red, following color spectrum, indicates increase of epochs)

7.6 Perturbed Control on First Trajectory with Integral Action

In the following section, the errors obtained by carrying out the perturbed control on the optimal and unoptimal trajectory with the presence of an integral term are shown and analysed, in order to verify the obtained optimization. Obviously also in this situation, the parameter vector p_{our} has been considered for the perturbed case, as mentioned several times above. The two errors in the perturbed case are reported below in the Figure (18).

How can be seen, the error following the unoptimal trajectory is around $0.98\,cm$, instead the one on the optimal trajectory is $0.69\,cm$. Therefore, an optimization in position of $2.9\,mm$ is obtained. In fact, the integral action reduces a lot the position error also in the unoptimal trajectory, passing from an error of $3\,cm$ to one of not even a centimetre. But also in this situation, the optimization is adequate because it is able of reducing the error even more.

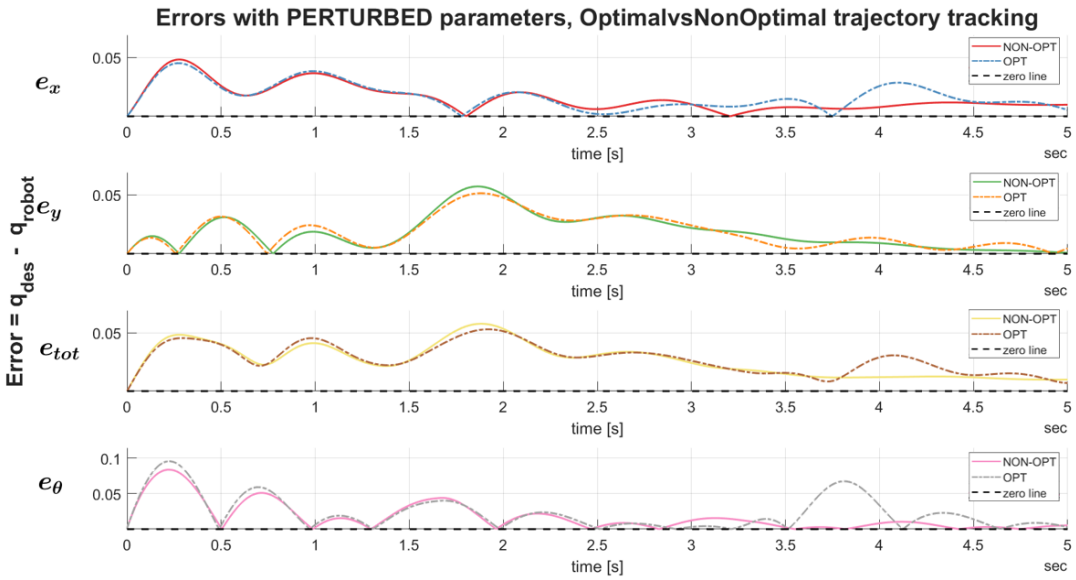

Figure 18: Error in the Perturbed Case

It should also be emphasised that in this case there is also an optimization in the orientation θ of the DDR. It goes from an error on θ on the unoptimal trajectory of $0.0052\,rad$ to an error in the optimal trajectory of $0.0028\,rad$, having so a slightly optimization of $0.0024\,rad$. This confirm what was said earlier during the sensitivity analysis, with the presence of the integral action, the optimization process is able to slightly optimize the orientation error e_θ, unlike what happens when this action is not considered.

Finally, also for this case, the total optimization ratio has been calculate, using the calculations seen in the Section (7.3). It has there that the two error ratios, the one based on the paper's error and the one based on ours, coincide. In particular, the error based only on position is 1.5 times smaller in the optimal case respect to the unoptimal case. Instead considering also the position error e_θ, we have that this ratio decreased slightly, assuming a value of 1.4. As can be seen, this is fairly larger than without integral action, as in this case, also some optimization on orientation are obtained.

8 Second Case

8.1 Second Case with No Integral Action

The following section shows a trajectory constructed differently from that shown in the previous cases, which presents a more geometric trend than the previous case.

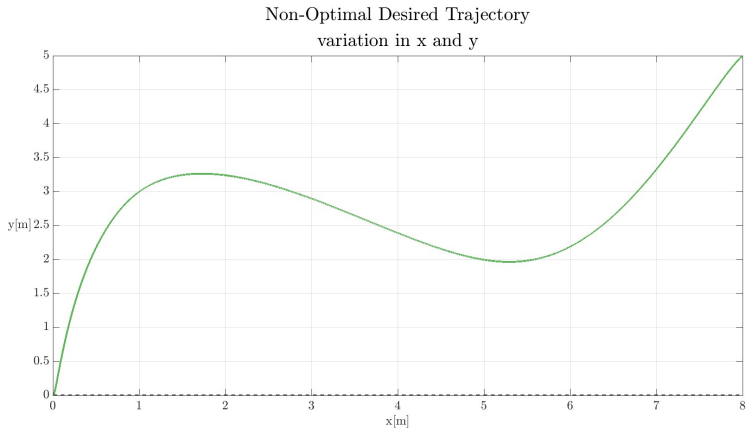

Figure 19: Second Trajectory

To create this trajectory, it can be set the initial point equal to $p_i = [0,\ 0]$, respectively, and the break points are placed at $p_1 = [1,\ 3]$ and $p_2 = [5,\ 2]$. Finally, the end point is set at $p_f = [8,\ 5]$. It has been considered as initial velocity $v_i = [0.1,\ 0.1]$ and as final velocity $v_f = [1,\ 1]$ that are set in this way to reduce the steady state error of the controller, as mention in [4].

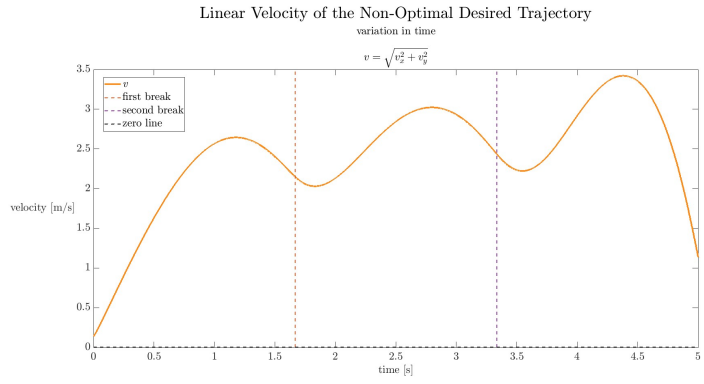

Figure 20: Velocity of the Second Trajectory

As can be seen from the Figure (20), having applied the constraints corresponding to the continuity of the speed at the break points in the matrix, the speeds were set appropriately, so that they never touched the value 0 otherwise there could have been problems relating to the DFL control. In addition, this figure provides us with information regarding the progress of the robot in following the trajectory; in particular, we see that the robot, to reach the first two break points, needs a strong acceleration which will then be followed by a deceleration to reach the interpolation point. Finally, to reach the final point, the robot accelerates and then, given the maneuver to reach the final point, the latter leads to an immediate deceleration.

On this trajectory, a DFL control has been applied to compute trajectory tracking. At first, as explained in the section before, the nominal case ($p = p_c$) has been considered and the gains are chosen in such a way that the error, in this case, is as low as possible (almost zero), in such a manner as to follow the hypothesis made by the paper. They are so set at the following value:

$$k_p = 34 \qquad k_d = 3 \qquad k_i = 0$$

As can be seen, in this first situation there is no integral action. As said before, this will be introduced in the next simulation to finally compare the result obtained in this situation and with the presence of this action.

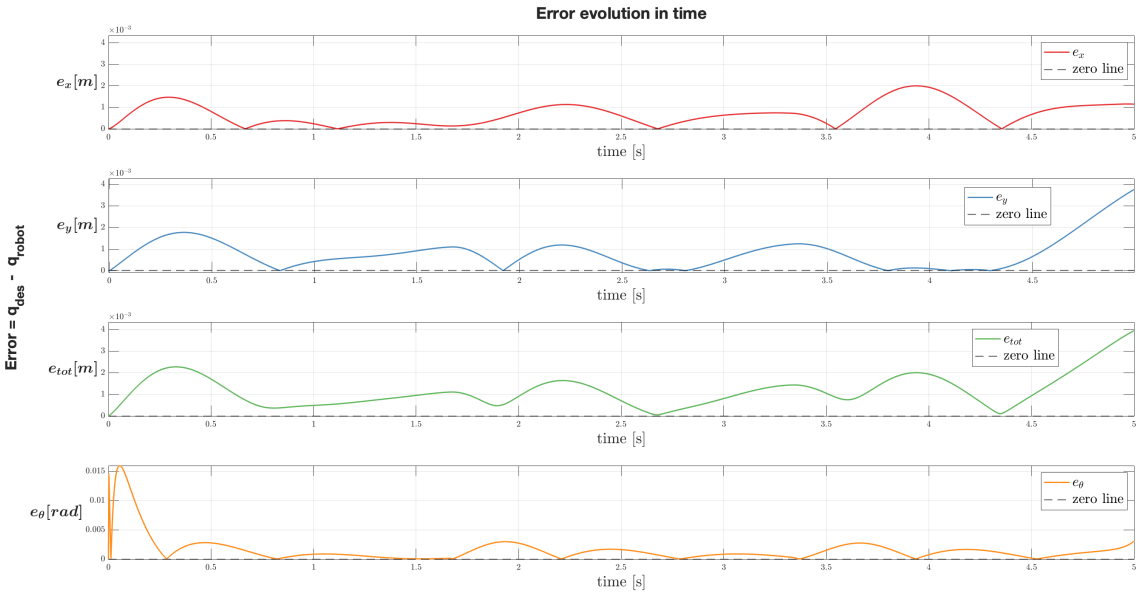

Figure 21: Error Nominal Control on Second Trajectory

In the previous figure is reported the position and orientation error made by the control in the

tracking process. It can be seen how the total error e_{tot}, which is the one computed in position, decrease to finally assume a value $3.95\,mm$ at steady state; also the orientation error e_θ is small enough. This demonstrates how the DFL control is almost perfect for the nominal case $p = p_c$.

Finally, the behavior of the DFL-controlled robot in the nominal case in tracking the unoptimized trajectory and the input velocity generated by the control itself are reported below.

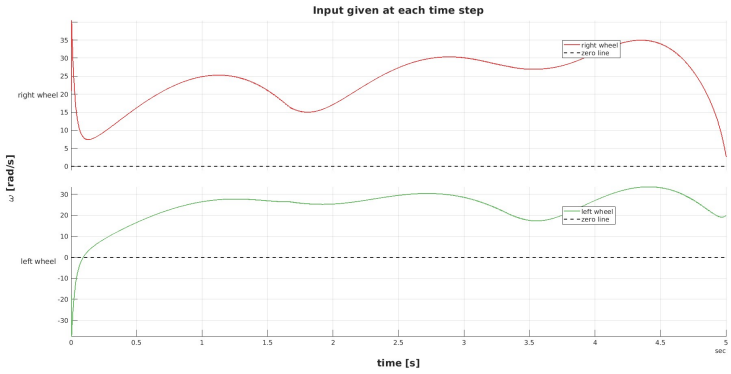

Figure 22: Input Nominal Controller Second Trajectory

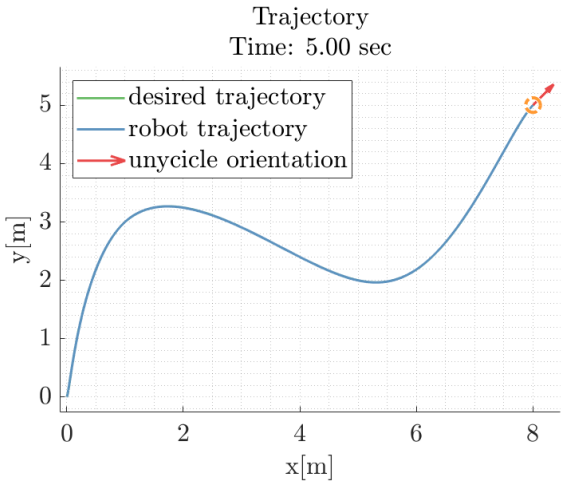

Figure 23: Behaviour DDR following the Second Trajectory Nominal Case

The DFL-controlled robot behaviour confirms the excellent performance mentioned above, as the difference between the robot's course and the trajectory to be followed are almost invisible (as there are millimeter errors). Instead, from the first image, Figure (22), can be seen the input generated by the controller which corresponds to how the robot turns during the simulation.

Remember that has been considered, for our simulations, a differential drive robot, where so the angular velocity of the two wheels is differently related to the orientation the robot must take to follow the predetermined trajectory, through the property of flatness.

Computed the ideal control, as said in the previous section, the necessary data to calculate the sensitivity matrix Π and the gradient Π_{a_i} for performing the optimization process can be determined.

8.2 Optimization on the Second Trajectory

For the optimization part, as mentioned earlier, a loop is run in which gains, used in the Formula (15), and the number of epochs is set. In particular, in this situation, the values of these parameters are:

- $k_1 = 1$;
- $k_2 = 0.18$;
- $epochs = 100$;

The values are chosen empirically, to increase the difference between the error obtained from the Perturbed Control, following the unoptimized trajectory and this new optimal trajectory.
By setting these parameters, the loss function takes the trend presented in the following Figure (24).

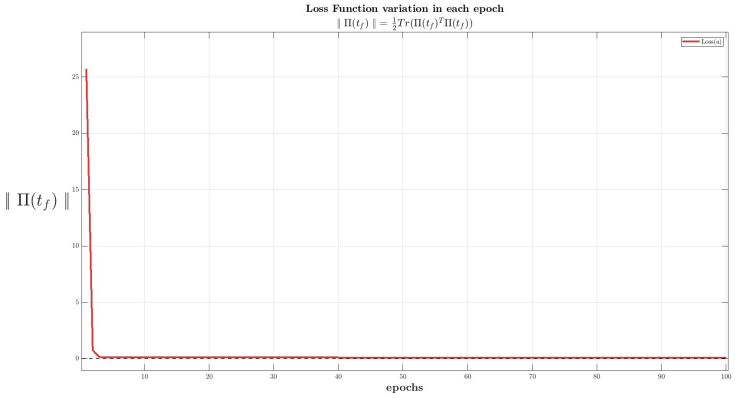

Figure 24: Loss function for Second Trajectory

As mentioned in the previous sections, the optimization process is empirically interrupted in a certain number of epochs in which the minimum value of the loss function is obtained and this allows a significant optimization. As can be seen from the figure, the loss decreases a lot in the first epochs, varying little in the remaining epochs around a very small value close to 0, finally reaching its lowest value in the epoch we are considering.

In addition to the loss function, the performance of the optimized trajectories at each epoch can be monitored. For each loop and so for each vector of trajectory coefficient a generated by the process, the corresponding trajectory has been printed, in such a way as to find the Figure (25), where each color corresponds to the trajectory calculated in that specific process loop (or epoch).

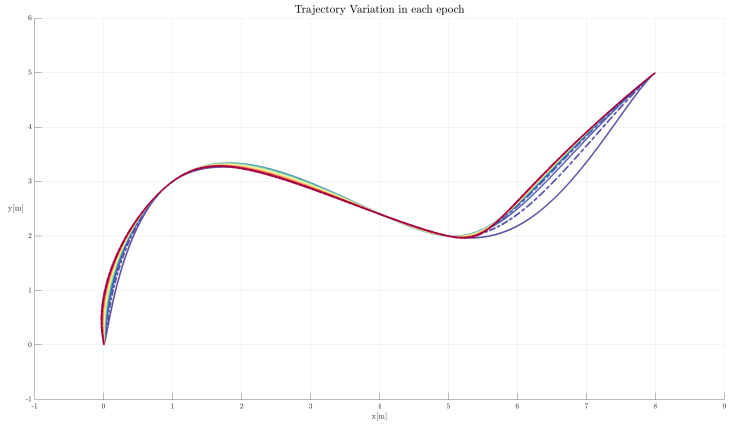

Figure 25: Second Trajectory Variation during the Optimization phase
(Color change from purple to red, following color spectrum, indicates increase of epochs)

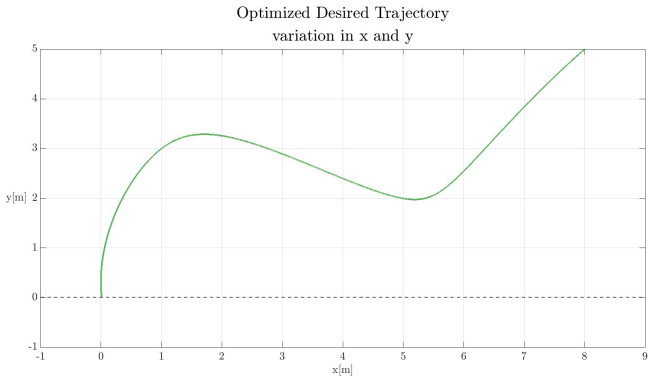

Figure 26: Optimal Trajectory

As mentioned above, also in this case, the optimization is free to modify the trajectory to make a 'maneuver' in such a way that the robot reaches the final position p_f more precisely. This 'adjustment' causes the optimized trajectory in the last section to arrive more straight at the final point, perhaps increasing the position error e_{tot} in this area, but decreasing it at the final time $e_{tot}\,(t_f)$. It is interesting to note that the optimization process, in this case, causes the shape of the trajectory to change strongly as early as the first few epochs. While, for the many remaining epochs, the variation is not excessive, as the trajectory tends to make small changes around an already defined shape.

As can be seen from the Figure (27), the trend of the sensitivity is always the same, thus an opposite trend between the components. The situation to be highlighted in this case is a slight improvement on θ. In particular, the component $\frac{\partial \theta}{\partial b}$ seems to follow a downward trend; this is probably connected with the fact that, with this trajectory, the optimization goes also to the orientation error e_θ. This may depend on the fact that the robot arrives at the final point with a different orientation and therefore the optimization process focuses more on this aspect than on the final positioning.

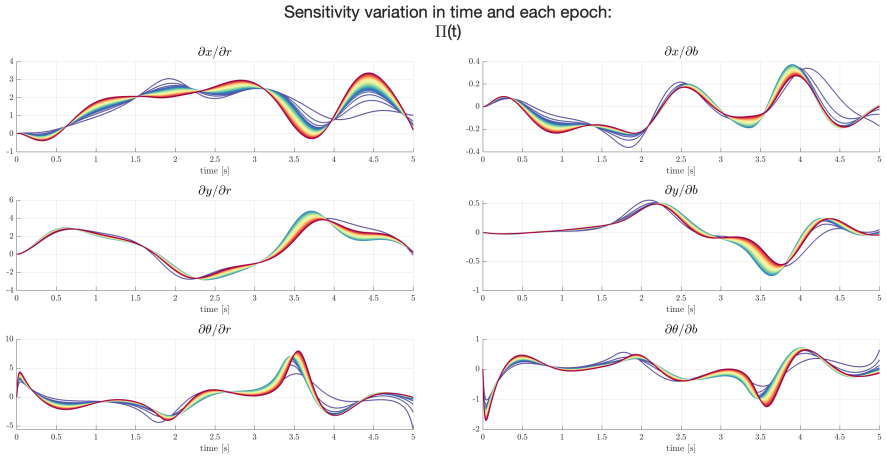

Figure 27: Sensitivity Variation (Color change from purple to red, following color spectrum, indicates increase of epochs)

8.3 Perturbed Control on Second Trajectory

In the following section, the errors obtained by carrying out the perturbed control on the optimal and unoptimal trajectory with the absence of an integral term are shown and analysed, in order to verify the obtained optimization. Obviously also in this situation, the parameter vector p_{our} has been considered for the perturbed case, as mentioned several times above. Let's report below the two errors in the perturbed case.

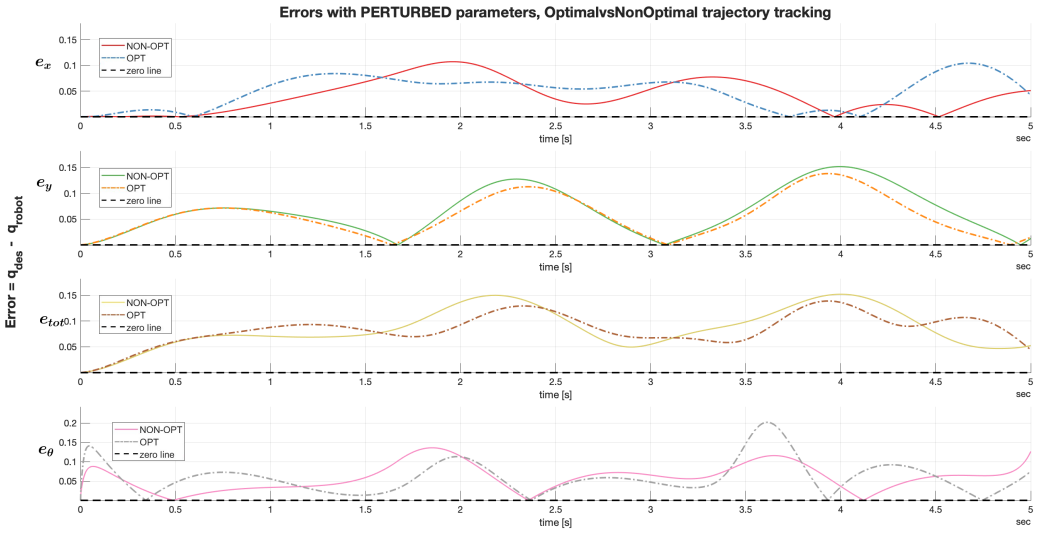

Figure 28: Error in the Perturbed Case

How can be seen, the error following the unoptimal trajectory is around $5.25\,cm$, instead the one on the optimal trajectory is $4.44\,cm$. Therefore, an optimization in position of $0.81\,cm$ is obtained.

It should also be emphasised that in this case there is also an optimization in the orientation θ of the DDR. It goes from an error on θ on the unoptimal trajectory of $0.12\,rad$ to an error in the optimal trajectory of $0.075\,rad$, having so a slightly optimization of $0.045\,rad$. This confirm that, in this case, the optimization process is able to slightly optimize the orientation error e_θ, unlike what happens in the case seen before.

Finally, also for this case, the total optimization ratio has been calculate, using the calculations seen in the Section (7.3). It has there that the two error ratios, the one based on the paper's error and the one based on ours, coincide. In particular, the error based only on position is 1.2

times smaller in the optimal case respect to the unoptimal case. Instead considering also the orientation error e_θ, we have that this ratio increase slightly, assuming a value of 1.6. The latter increase is clearly closely linked to the fact that in this case there is also an optimization on the orientation which therefore influences the overall trend of the error with respect to the optimal trajectory.

8.4 Second Trajectory with Integral Action

In this section, as done previously, the integral action is introduced into the DFL control, which is responsible for tracking the above-mentioned trajectory Figure (19). In this way, it is possible to verify how the optimization is useful even with the presence of this action, which, as is well known, reduces constant disturbances at steady state (such as perturbations on the parameters considered in this work).

The values of the selected DFL control gains are shown below. As can be seen, it is presented also the integral action because the integral gain k_i is different from zero. Recall that the criterion for the choice of gains in the situation under consideration (i.e. the nominal $p = p_c$) is to have the position error e_{tot} as low as possible (close to 0).

$$k_p = 34 \qquad k_d = 3 \qquad k_i = 2$$

It has been reported in the Figure (29) the position and orientation error made by the control in the tracking process. It can be seen how the total error e_{tot}, which is the one computed in position, reach a value of $5.7\,mm$ at steady state; also the orientation error e_θ is small enough. In general, even for this case, the DFL control in the nominal case can be considered 'almost perfect' and the small error present can be explained by the integration of the system dynamics, which in fact introduces some error on the calculation of the system state itself. In this way, also for this case, the paper's assumption concerning control in the nominal situation is fulfilled. In general, it can be seen that, the introduction of integral action does not significantly alter the error at steady state. In fact, since the nominal case is being considered, there are no constant variations on the parameters and therefore the action itself does not provide great changes with respect to the case without integral action (as will happen in the perturbed case).

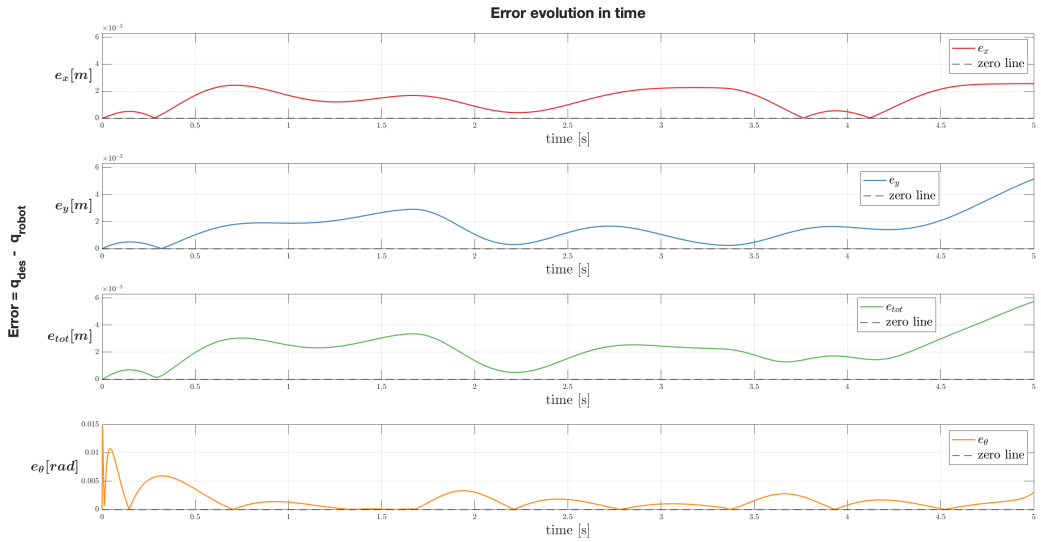

Figure 29: Error Nominal Control on Second Trajectory case with Integral Action

Finally, the behaviour of the DFL-controlled robot in the nominal case in tracking the trajectory and the input velocity generated by the control itself has been reported.

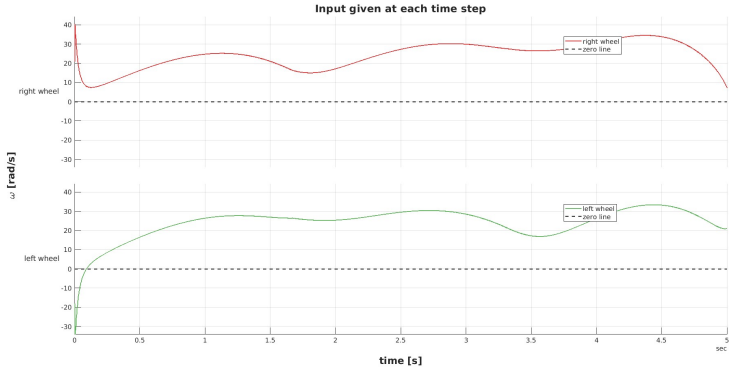

Figure 30: Input Variation Nominal Controller Second Trajectory case with Integral Action

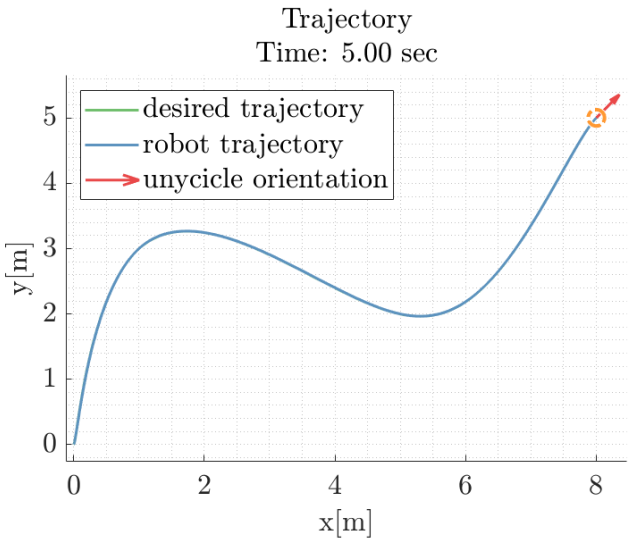

Figure 31: Behaviour of the DDR following the Second Trajectory Nominal Case with Integral Action

Once the ideal control has been performed, it is then possible to move on to the analysis of the optimization performed for this situation as well.

8.5 Optimization on Second Trajectory with Integral Action

For the optimization part, a loop has been performed in which gains and the number of epochs are set. In particular, the values chosen for these parameters are:

- $k_1 = 1$;
- $k_2 = 0.18$;
- $epochs = 100$;

As before, also these values are chosen to maximize the difference between the error of the perturbed control in the optimized and unoptimized case, so that the former is much smaller than the latter.

By setting these parameters, the loss function takes the trend presented in the following figure:

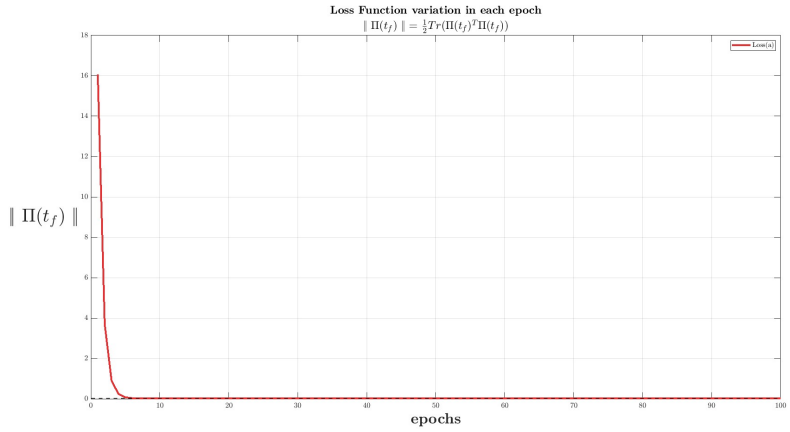

Figure 32: Loss function on Second Trajectory case with Integral Action

In this case, the optimization process is stopped at the 100 epochs in which the minimum value of the loss function has been reached in order to have a consistent optimization. After this epoch, the loss tends to increase in value, thus leading to worse optimization.

Looking at the figure (33), it is possible to monitor the performance of the optimized trajectories in each epoch. In this case it can be noted that the change of trajectory in the last stretch is not immediate (as it was for the case without integral action where it occurred after the first epochs). Here, several rounds of simulations are needed to get a relevant change in the trajectory.

Furthermore, the optimization process causes the last part of the trajectory to be straighter towards the end point and not have a smoother trend as seen in the previous test, in order to reduce the error on the robot's final position.

Furthermore, it is important to underline how the optimized trajectory obtained in this case and the previous one Figure (26) do not differ much from each other, unlike the first trajectory, analyzed in detail in Section (7). The reason for this is probably connected to the fact that, in this trajectory, the error we obtain by considering also the contribution of the integral action does not differ much from what we obtain when we do not consider the latter. Therefore, since the errors are also very low, the optimization probably does not need to vary the shape of the trajectory much to improve the error at the final point p_f.

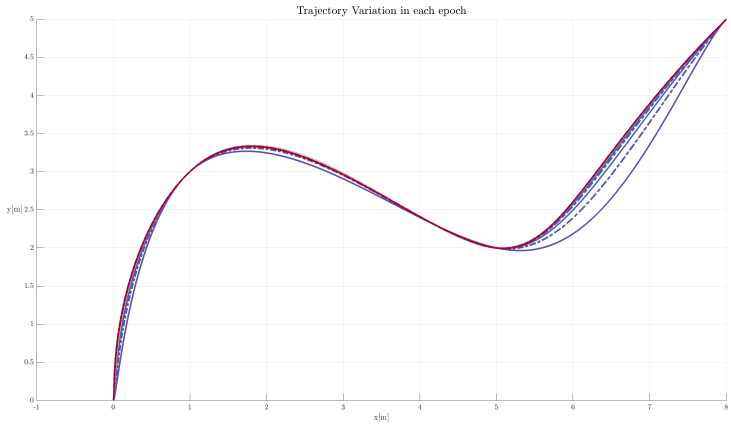

Figure 33: Second Trajectory Variation during the Optimization phase
(Color change from purple to red, following color spectrum, indicates increase of epochs)

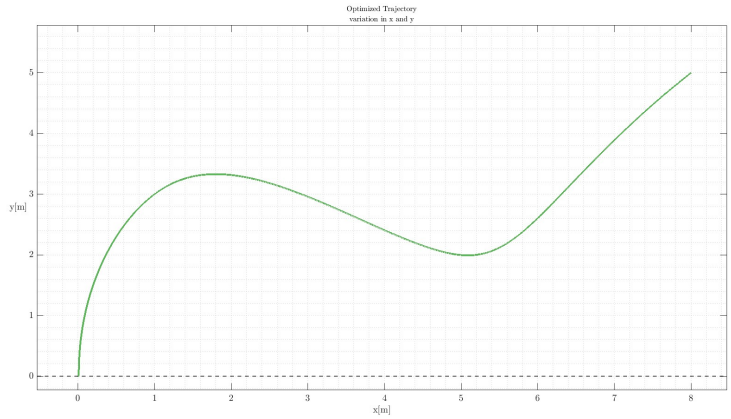

Figure 34: Second Optimal Trajectory with Integral Action

As mentioned above, the next step that has been done is the sensitivity calculation. From the Figure (35) it is possible to note that the trend is always the same, therefore an opposite trend between the components. The situation to highlight is that also in this case there is a slight improvement regarding orientation. Moreover, also sensitivity has a similar behaviour to trajectory variation. In fact, the various components vary greatly in the first few epochs, and then settle around a precise trend, undergoing small changes for the numerous remains. Finally, even for this magnitude, the integral action does not seem to have brought big variations with respect to the previous situation, where this action was not present.

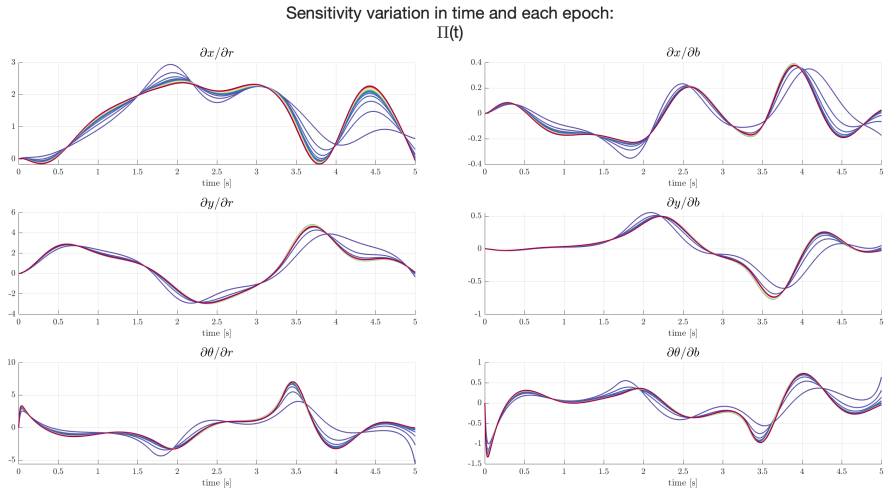

Figure 35: Sensitivity on Second Trajectory (Color change from purple to red, following color spectrum, indicates increase of epochs)

8.6 Perturbed Control on Second Trajectory with Integral Action

In the following section has been shown the errors obtained by performing perturbed control on the optimal trajectory and unoptimal trajectory, with the purpose of verifying the optimization obtained. Also in this case, it has been considered a perturbation $p_{our} = [0.8 * r_c, \ \ 1.2 * b_c]^T$.

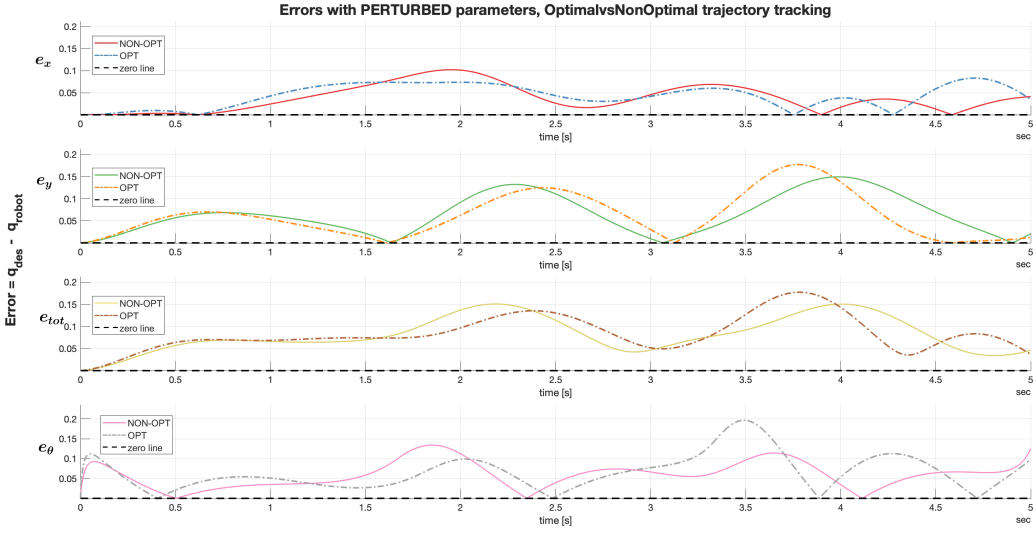

Figure 36: Error in the Perturbed Case

In this case the error obtained on the non-optimal trajectory is equal to $4.64\,cm$ while that on the optimal trajectory is $3.70\,cm$. So the difference of the two errors, as well as the optimization obtained, is about $0.94\,cm$. As in the cases seen previously, we also obtained a good optimization on θ; in fact, in the perturbed case $p \neq p_c$, we have that the orientation error considering the optimized trajectory is $0.09\,rad$, while that relating to the non-optimized trajectory is $0.13\,rad$, thus obtaining an optimization of $0.04\,rad$.

Finally, the total optimization ratio was calculated, using the calculations seen in Section (7.3). In this situation, taking into account the error of the article, the one in the optimal case is 1.3 times smaller than the one calculated in the non-optimal case. Instead, as far as our report is concerned, the error based only on the position is 1.3 times smaller in the optimal case than in the other case.

9 Statistical Analysis

The following section deals with how the statistical analysis on the optimal and unoptimal trajectories was carried out, and therefore how our system was perturbed so as to compare the errors in the optimal case and to be able to estimate how many times an optimization has been achieved. Finally, the mean and the variance of the errors obtained having a perturbed control following both the optimal and the unoptimal trajectory was calculated. Furthermore, as said before, a small modification was made on our part respect to the paper. In fact, in our case, as mention above, the calculations of the mean and standard deviation for the position, in the optimal and unoptimal case has been calculate; then, the same calculations are performed separately for the orientation. As regards what has been done in the paper, the calculation of the mean and standard deviation is considered, in the optimal and unoptimal case, for the state of the robot in its entirety, i.e. considering both position and orientation.

9.1 Statistical Analysis on First Trajectory

Once the optimized trajectory is obtained, the states of the robot in following the optimal trajectory can be observed when the system parameters are not known. The same is done for the unoptimal trajectory to compare the value of the error at the end point, in such a way to study how many times the optimization is obtained. As said before, $N = 20$ simulations with this purpose has been carried out, where, for each of these, the perturbation percentage of the two coefficients r and b has been changed, choosing them randomly in a uniform distribution of [80%, 120%].

The following table shows the information about the percentage of change in the parameters (thus the magnitude of our perturbation) and also the errors obtained at the endpoint t_f on the optimal trajectory, e_{opt}, and the unoptimal trajectory, e_{nopt}, in the case where the first trajectory is considered without the contribution of the integral action:

First Trajectory with No Integral Action			
% Radius Variation	% Distance Variation	Error Optimal [m]	Error Unoptimal [m]
91%	107%	0.0024	0.0140
94%	82%	0.0016	0.0084
84%	100%	0.0126	0.0262
85%	88%	0.0159	0.0234
86%	87%	0.0144	0.0215
81%	106%	0.0132	0.0329
91%	102%	0.0035	0.0136
81%	87%	0.0241	0.0314
91%	106%	0.0026	0.0139
107%	129%	0.0105	0.0090
116%	119%	0.0171	0.0183
97%	88%	0.0034	0.0041
86%	90%	0.0136	0.0216
97%	92%	0.0032	0.0040
118%	83%	0.0228	0.0212
84%	85%	0.0188	0.0252
86%	105%	0.0079	0.0228
103%	82%	0.0116	0.0056
118%	109%	0.0199	0.0204
110%	82%	0.0184	0.0132

Table 1: Table results $N = 20$ simulations on First Trajectory No Integral action

As can be seen from the table, especially from the error section in the optimal and unoptimal cases, 16 times out of 20 an optimization has been obtained, therefore in 80% of the cases. In the remaining 20% of the cases, however, errors that are very low has been obtained, but the combination of the percentages of variation of the parameters considered means that, following the optimal trajectory, an error is obtained that is slightly greater than in the unoptimized case. This could also be caused by the smoother structure of the optimal trajectory in the final stretch which causes the robot to widen by a few millimeters when reaching the final point. These results demonstrate once again how the optimization process performs very well in obtain a optimized trajectory $r_d(a_{opt}, t)$ from perturbed parameters $p \neq p_c$.

Furthermore, the variation of the differential drive robot states in the various simulations has been reported below, thus considering for each case the perturbations shown in the table, both for the cases in which the robots follow the optimal trajectory and for those in which they track the unoptimized one. In all the images, the purple line represents the optimal or unoptimal trajectory (depending on the case), instead each line corresponds to one of the $N = 20$ simulations considered. Obviously, the lines with the same color and with the same path in both sets of images go to indicate a situation with the same perturbations, so as to compare the performance of the differential drive robot with perturbed control in the two cases.

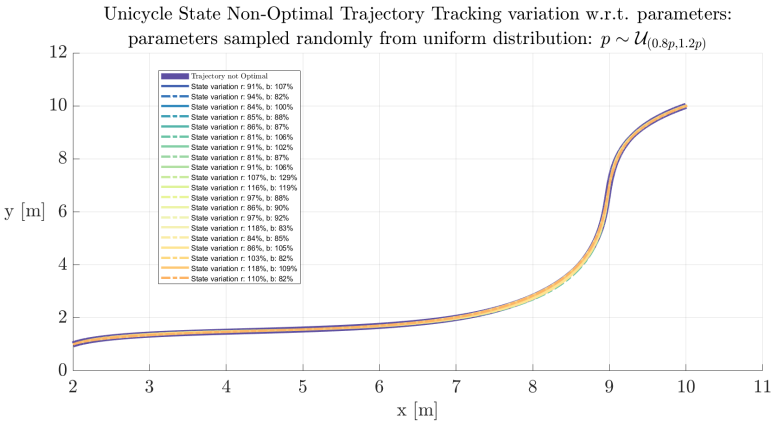

Figure 37: Complete state variation in the Unoptimal Case

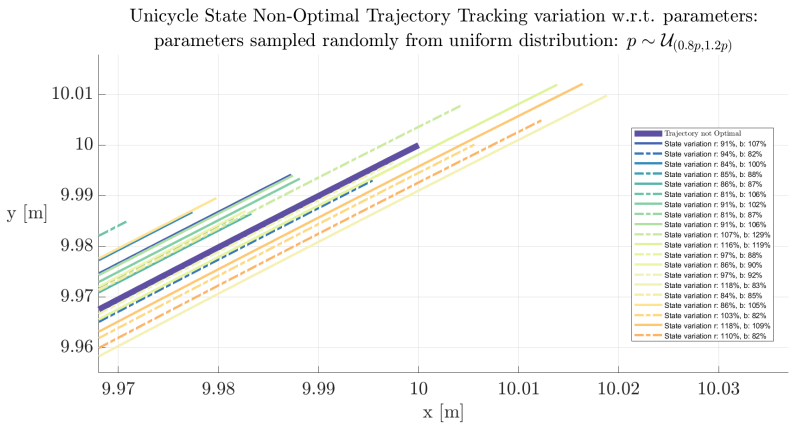

Figure 38: Particular state variation in the Unoptimal Case

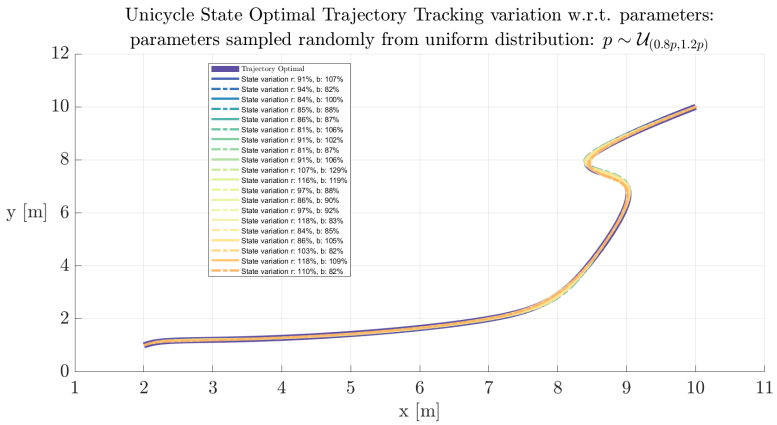

Figure 39: Complete state variation in the Optimal Case

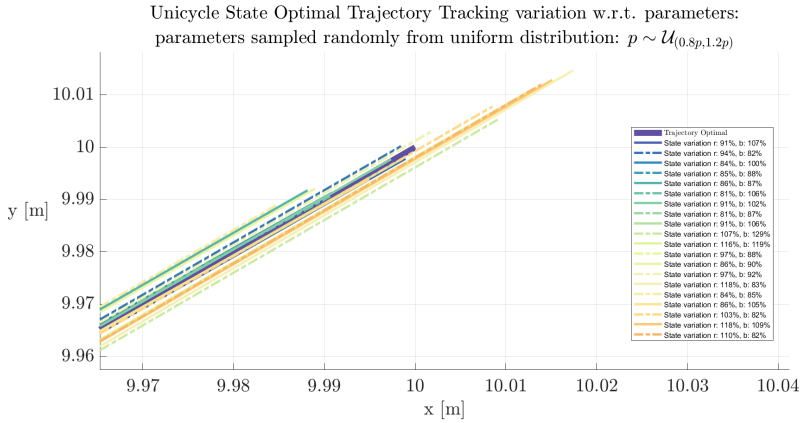

Figure 40: Particular state variation in the Optimal Case

The 'Particular images' (so Figure (38) and Figure (40)) correspond to the final section, at the time t_f, of the other two images (Figure (37) and Figure (39)).

Thank to these, can be seen that the state variations of the differential drive robots following the optimized trajectory are more closely matched to the desired trajectory (i.e. the optimal trajectory) than the situation with the unoptimal trajectory. This is a visual element that further confirms what has been said above, namely that the optimised trajectory $r_d(a_{opt}, t)$ is easier to follow for differential drive robots guided by perturbed DFL controls, which therefore make a smaller position error e_{tot} at the final time t_f. The same analyses were of course also carried out in the situation where the integral action is present. The $N = 20$ simulations, as done before, has been considered. The results of these and the perturbations applied on the two parameters r and b for each simulations in the following table has been reported.

First Trajectory with Integral Action			
% Radius Variation	% Distance Variation	Error Optimal [m]	Error UnOptimal [m]
95%	118%	0.0022	0.0026
110%	104%	0.0025	0.0039
86%	86%	0.0012	0.0047
82%	115%	0.0053	0.0083
104%	109%	0.0019	0.0021
80%	119%	0.0068	0.0098
114%	88%	0.0029	0.0054
87%	87%	0.0011	0.0042
92%	101%	0.0016	0.0026
97%	91%	0.0014	0.0003
105%	85%	0.0023	0.0029
91%	95%	0.0013	0.0027
98%	112%	0.0018	0.0013
88%	101%	0.0019	0.0044
104%	81%	0.0023	0.0028
104%	86%	0.0022	0.0026
82%	118%	0.0058	0.0086
119%	113%	0.0030	0.0062
92%	84%	0.0010	0.0020
108%	98%	0.0024	0.0034

Table 2: Table results $N = 20$ simulations on First Trajectory with Integral action

As can be seen from the table, considering also the integral action, 18 times on 20 simulations an optimization in the position error has been achieved. So, in 90% of cases, the process can optimize without problems. It can be seen, as stated earlier, that the errors in the unoptimal case e_{nopt} are quite small, all in the millimeter range.
As before, the variation of the differential drive robot states in the various simulations both in the optimal and unoptimal cases has been reported below.

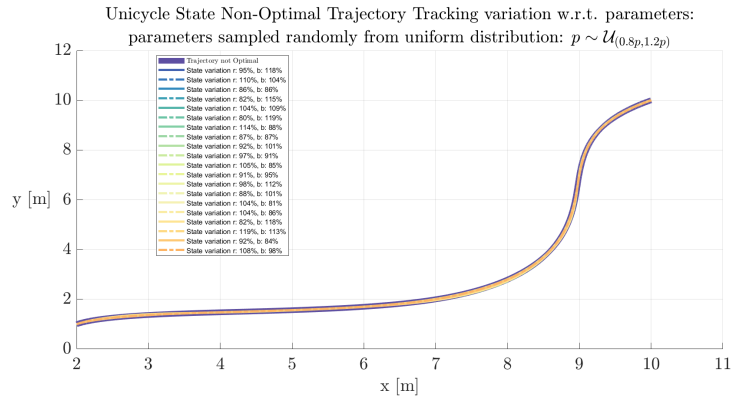

Figure 41: Complete state variation in the Unoptimal Case with Integral action

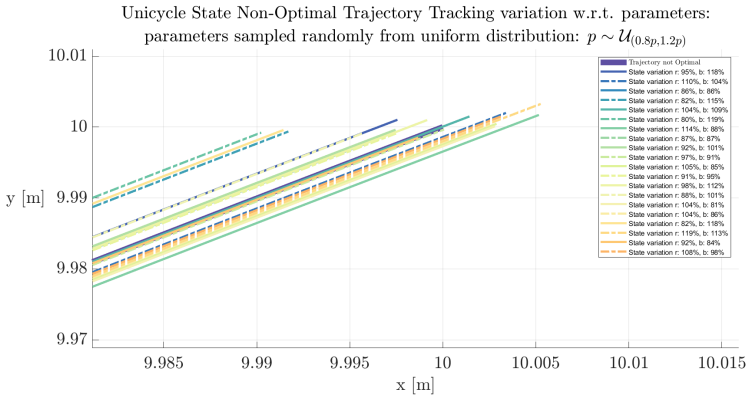

Figure 42: Particular state variation in the Unoptimal Case with Integral action

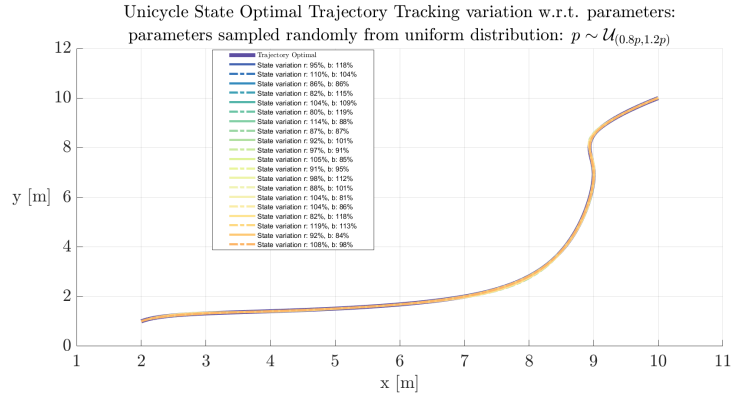

Figure 43: Complete state variation in the Optimal Case with Integral action

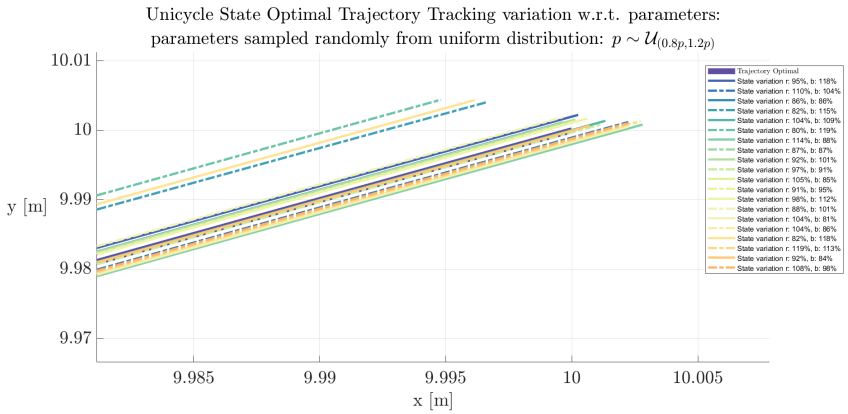

Figure 44: Particular state variation in the Optimal Case with Integral action

What was said in the previous part also turns out to be further true here. In fact, as can be seen from Figure (42) and Figure (44), the coloured lines, representing the evolution of the robot states in the various perturbation simulations, are closer together around the optimal trajectory (the purple line) in the optimal case than in the unoptimal case. This is to symbolise how the perturbed DFL control manages to follow the optimal trajectory better than the unoptimal one. Moreover, in this case with integral action, this closeness between the lines is even more pronounced than in the previous situation, as here the optimization percentage is higher.

Finally the mean μ and the standard deviation σ computed in the different situation has been reported, as explained in the Section (6.3). This allows us to compare, in a more statistical way, the results obtained and to adequately compare optimization with and without integral action. Remember that in this case three different sets of means and standard deviations can be considered:

- the set based on the paper error, then considering the difference between the nominal q_{nom} and perturbed state q_{pert} in both the optimal and unoptimal case, which then compares the error on all three components indiscriminately;
- the set based on our consideration, where the error on position has been divided from that on theta, thus calculating two sets of statistical functions, namely $(\mu_{tot}^{opt}, \mu_{\theta}^{opt}, \sigma_{tot}^{opt}, \sigma_{\theta}^{opt})$ and $(\mu_{tot}^{nopt}, \mu_{\theta}^{bopt}, \sigma_{tot}^{nopt}, \sigma_{\theta}^{nopt})$;

Let analyze these results starting with the mean and the variance considering the errors reported in the paper:

	μ^{Opt}	μ^{Nopt}	σ^{Opt}	σ^{Nopt}
NI	0.01437	0.01730	0.00941	0.01005
I	0.00217	0.00437	0.00206	0.00293

Then, can be seen the results obtained from the calculation of the mean and the variance considering the position errors :

	μ_{tot}^{Opt}	μ_{tot}^{Nopt}	σ_{tot}^{Opt}	σ_{tot}^{Nopt}
NI	0.01290	0.01661	0.00798	0.00973
I	0.00188	0.00415	0.00191	0.00270

and the mean and standard deviation for θ in our case:

	μ_{θ}^{Opt}	μ_{θ}^{Nopt}	σ_{θ}^{Opt}	σ_{θ}^{Nopt}
NI	0.00432	0.00473	0.00690	0.00273
I	0.00092	0.00126	0.00094	0.00125

From these tables can be seen that, in *all* cases, the values for the optimal situation are lower than for the unoptimal situation, both for mean μ and variance σ. This demonstrates, once again, how the optimized trajectory $r_d\left(a_{opt}, t\right)$, for each case considered here, allows the error of the tracking control with respect to parameter perturbation $p \neq p_c$ to be reduced significantly.

It should be noted that optimization is also effective and relevant in the integral situation (I). In fact, in this situation, the μ^{opt} is much smaller than μ^{Nopt} demonstrating how the optimization considered achieves better results even than the integral action alone, which, as mentioned, in itself reduces the steady state error. Obviously, the lower values of both mean and variance in *all* cases where the integral action is present, turn out to be completely usual, precisely because of the effect this action has on the steady-state disturbances, which leads to lower errors in the millimetre range.

9.2 Statistical Analysis on Second Trajectory

To give greater rigor to our study also for the second trajectory is carried out a statistical analysis. That said earlier is also valid for this case. So $N = 20$ simulations with this purpose has been carried out, where, for each of these, the perturbation percentage of the two coefficients r and b has been changed, in the same way done before. The results obtained are reported in the Table (3), with the same methodology as seen in the previous section.

Second Trajectory with No Integral Action			
% Radius Variation	% Distance Variation	Error Optimal [m]	Error Unoptimal [m]
95%	118%	0.0023	0.0161
110%	104%	0.0057	0.0087
86%	86 %	0.0025	0.0086
82%	115 %	0.0283	0.0416
104%	109 %	0.0053	0.0046
80%	119%	0.0429	0.0514
114%	88 %	0.0064	0.0189
87%	87%	0.0019	0.0078
92%	101%	0.0017	0.0096
97%	91%	0.0052	0.0035
105%	85%	0.0065	0.0126
91%	95%	0.0018	0.0073
98%	112%	0.0034	0.0086
88%	101%	0.0034	0.0154
104%	81%	0.0067	0.0138
104%	86%	0.0064	0.0112
82%	118%	0.0323	0.0448
119%	113%	0.0052	0.0119
92%	84%	0.0041	0.0015
108%	98%	0.0059	0.0094

Table 3: Table results $N = 20$ simulations on Second Trajectory No Integral action

As can be seen, above all from the errors section in the optimal and non-optimal cases, an optimization was obtained 17 times out of 20, therefore in 85% of the cases. In the remaining 15% of the cases, anyway, very low errors were obtained in the optimal case. However, in these cases, the combination of the variation in the parameters considered causes the unoptimal error to be less than the optimal one, thus not having an effective optimization. These results demonstrate once again how the optimization process works very well in obtaining an optimized trajectory $r_d(a_{opt}, t)$ from perturbed parameters.

Furthermore, the following are the changes in the DDR states considering the perturbations exhibited in the table, both in the optimized and non-optimized case. We then have $N = 20$ tra-

jectories for each image in which each color is associated with a specific parameter perturbation, from among those in the table. This makes it easy to compare the performance of the DDR, subjected to a perturbed control, in following the unoptimized trajectory and the optimized trajectory, obtained by the process described above.

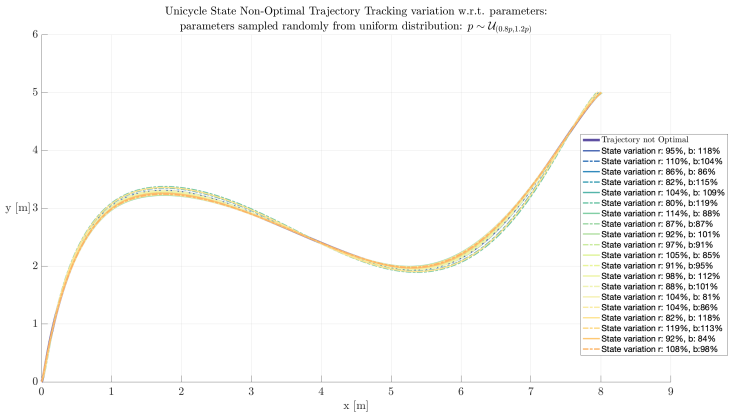

Figure 45: Complete state variation in the Unoptimal Case

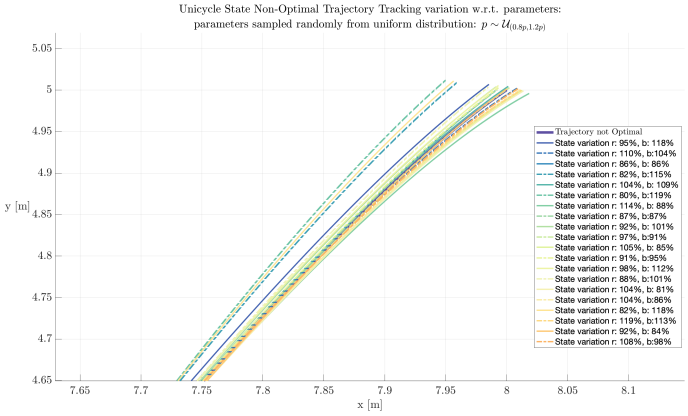

Figure 46: Particular state variation in the Unoptimal Case

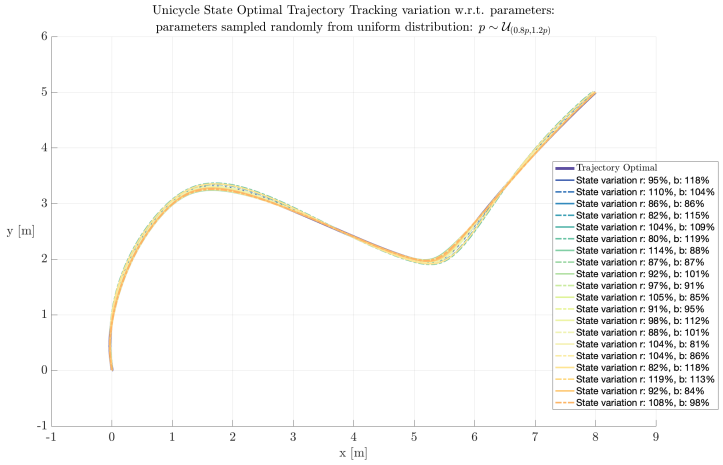

Figure 47: Complete state variation in the Optimal Case

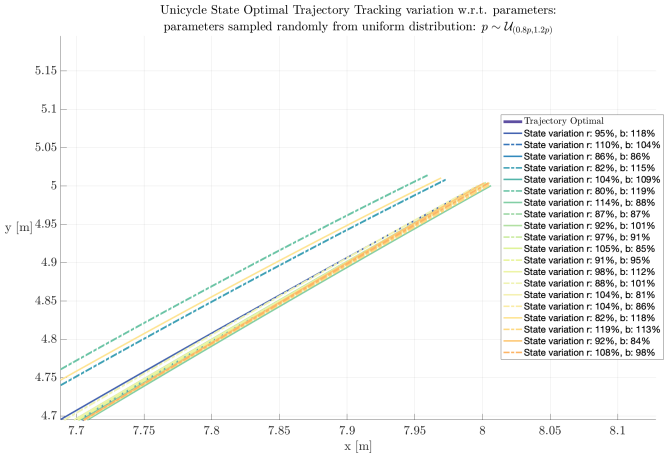

Figure 48: Particular state variation in the Optimal Case

Again, analyzing the 'Particular images', so Figure (46) and Figure (48), it can be notice how the different lines, so the different behaviour of the DDR in the different perturbation case, in the optimal case are all focused around the trajectory to follow (the one in purple). This is in contrast to the unoptimized situation, where these are slightly more spread out, indicating how the positional error e_{tot} in these different cases turns out to be greater than that calculated in the optimal case. So, as already stated several times in this paper, also for this specific situation, the optimized trajectory $r_d(a_{opt}, t)$ is easier to follow for differential drive robots guided by a DFL controls with unknown parameters value, which therefore make a smaller position error e_{tot} at the final time t_f. The same analyses were of course also carried out in the situation where the

integral action is present. Also there $N = 20$ simulations has been considered. The results of these and the perturbations applied on the two parameters are reported in the following table.

Second Trajectory with Integral Action			
% Radius Variation	% Distance Variation	Error Optimal [m]	Error UnOptimal [m]
95%	118%	0.0035	0.0148
110%	104%	0.0039	0.0064
86%	86 %	0.0022	0.0059
82%	115 %	0.0205	0.0358
104%	109 %	0.0050	0.0049
80%	119%	0.0353	0.0453
114%	88 %	0.0038	0.0162
87%	87%	0.0025	0.0060
92%	101%	0.0033	0.0085
97%	91%	0.0049	0.0069
105%	85%	0.0040	0.0121
91%	95%	0.0033	0.0070
98%	112%	0.0047	0.0086
88%	101%	0.0010	0.0120
104%	81%	0.0042	0.0136
104%	86%	0.0042	0.0111
82%	118%	0.0245	0.0393
119%	113%	0.0027	0.0078
92%	84%	0.0047	0.0210
108%	98%	0.0039	0.0481

Table 4: Table results $N = 20$ simulations on First Trajectory with Integral action

As can be seen from the table, considering also the integral action, 19 times on 20 simulations an optimization in the position error has been achieved. So, in 95% of cases, the process can optimize without problems. Moreover, in the only situation where there is no optimization, we have an optimal error almost identical to the unoptimal one, with a difference of $0.1\,mm$. It is therefore evident how the optimization process can be consider almost perfect for this case, thereby achieving excellent results.

It is possible to see how, for this second trajectory, the same parameter variations were used both in the case with integral action and without. This is to provide a more specific analysis of the behavior of the optimization process in the two cases by going to evaluate its performance in situations of identical perturbations. In fact, the introduction of the integral action increases a lot the optimization, going from 85% optimization cases, for the situation without integral action, to 95% for this case.

As before, the variation of the differential drive robot states in the various simulations both in the optimal and unoptimal cases has been reported below.

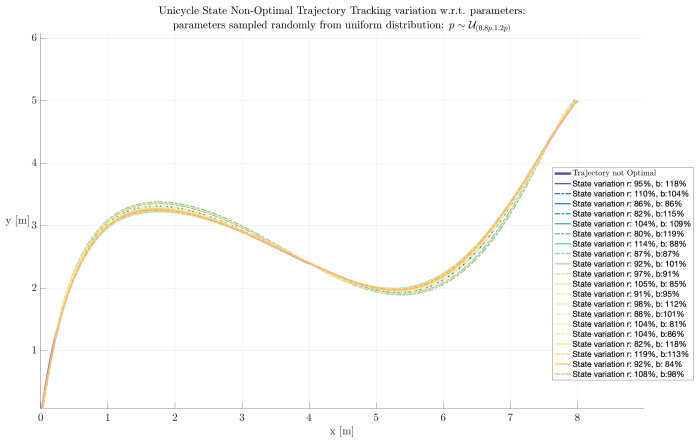

Figure 49: Complete state variation in the Unoptimal Case with Integral action

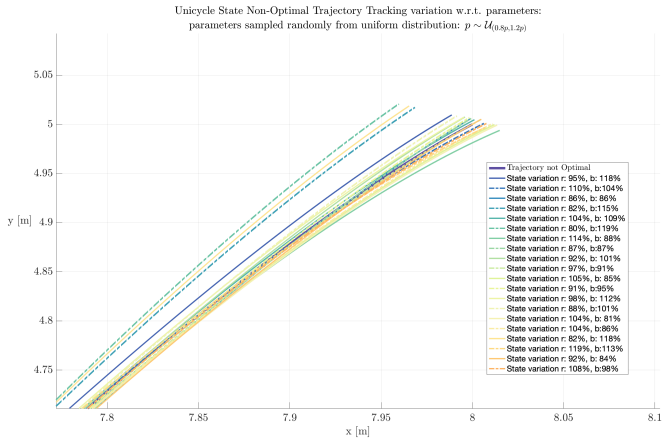

Figure 50: Particular state variation in the Unoptimal Case with Integral action

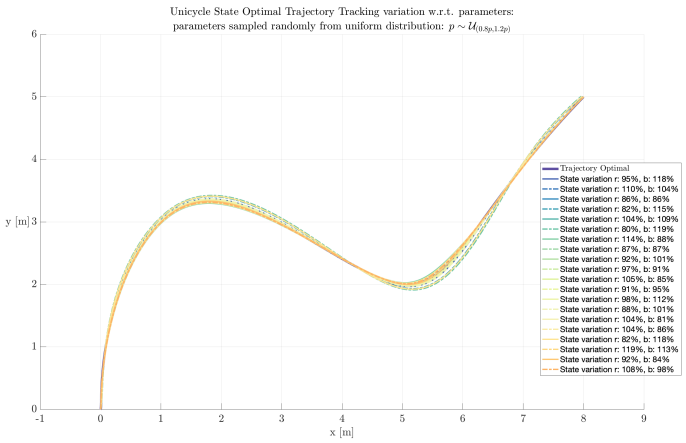

Figure 51: Complete state variation in the Optimal Case with Integral action

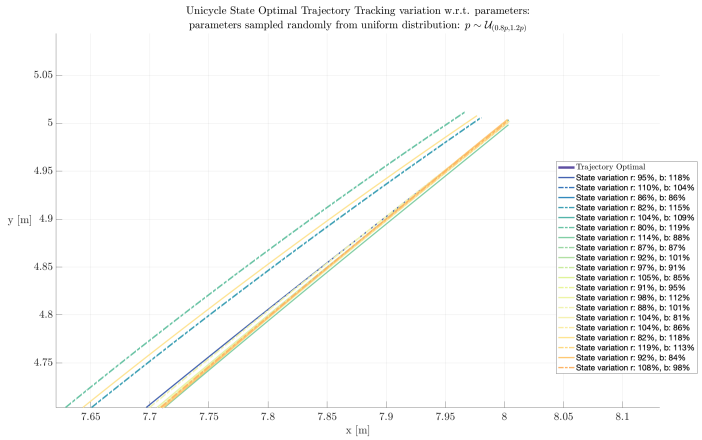

Figure 52: Particular state variation in the Optimal Case with Integral action

The excellent situation described earlier can be seen graphically in the figures above. In tracking the optimal trajectory, the perturbed DFL control, in most of the 20 cases , turns out to be much more accurate than in following the unoptimized trajectory.

Here, too, there is visual confirmation of the above by analyzing the data. In fact, the introduction of the integral action benefits the optimization because, considering the same parameter changes as in the situation without this action, here the various DDRs seem to arrive more precisely at the desired final position p_f.

Finally the mean μ and the standard deviation σ computed in the different situation has been reported, as explained in the Section (6.3). In this way, it is also possible to confirm with statis-

tical data what was said earlier, going to compare the optimization with or without the presence of the integral action. Also for this case, as for all the previous cases, 3 different groups of means and standard deviation have been used, as amply explained previously.

Let analyze these results starting with the mean and the variance considering the errors reported in the paper:

	μ^{Opt}	μ^{Nopt}	σ^{Opt}	σ^{Nopt}
NI	0.01887	0.05977	0.02656	0.03606
I	0.01929	0.05899	0.03253	0.03518

Then, can be seen the results obtained from the calculation of the mean and the variance considering the position errors :

	μ_{tot}^{Opt}	μ_{tot}^{Nopt}	σ_{tot}^{Opt}	σ_{tot}^{Nopt}
NI	0.00820	0.01522	0.01261	0.01428
I	0.00611	0.01254	0.01022	0.01281

and the mean and standard deviation for θ in our case:

	μ_{θ}^{Opt}	μ_{θ}^{Nopt}	σ_{θ}^{Opt}	σ_{θ}^{Nopt}
NI	0.01652	0.05744	0.02373	0.03376
I	0.01806	0.05726	0.03103	0.03347

As expected, statistical analysis also shows how the optimized trajectory $r_d\left(a_{opt}, t\right)$, for each case considered here, allows the error of the tracking control with respect to parameter perturbation $p \neq p_c$ to be reduced significantly. In fact, the values for the optimal situation are lower than for the unoptimal one, both for mean μ and variance σ.

Finally, it can be seen how, comparing the averages between the optimal and not optimal case in the situation with the integral action there is a lesser optimization on the orientation error e_θ than the situation without integral action. In contrast, an opposite situation occurs for e_{tot}, with better optimization in the presence of the integral action.

10 Conclusion

In this work, has been implemented a trajectory generation for minimum closed-loop state sensitivity considering a Differential Driven Robot. Here two trajectories with quite different shape have been considered in order to evaluate the behavior of the optimizations in different cases. The aim of this optimization was to generate, for each case, an optimal trajectory that would decrease the tracking error of the DFL control at the final time t_f in a perturbed situation $p \neq p_c$, in which the robot parameters take different values than the nominal one. In particular, to assess the effectiveness of optimization, it was considered both a situation in which the DFL control was devoid of integral action and a situation with the presence of this action. This is to show how the optimization obtained excellent results even in the presence of the integral action (which, as is well known, reduces the influence of constant disturbances at steady state).

After having carried out various tests we have seen that the optimization process obtains the expected results. Therefore, what we have noticed is that with this process we obtain that the effect of the parameter variation respect to the nominal value (situation that may occur in the external world) is reduced within the control, obtaining an optimized solution in the final point with a smaller error. Furthermore, in the various simulations considered in which, in each of them, a different perturbation of the parameters is generated , a high optimization percentage is always obtained.

For a further improvement of what has been implemented, we could think of a different way of creating the initial trajectory; therefore, instead of using a spline, one could use the Bézier curve or a unique polynomial of degree n. Furthermore, a stop criterion could have been used in the descending gradient algorithm to find the global minimum of the function, avoiding obtaining the loss function empirically, in order to further improve the optimization process.

References

[1] K. J. Astrom and B. Wittenmark, Adaptive Control, 2nd ed. Prentice Hall, 1994.

[2] K. Zhou and J. C. Doyle, Essentials of Robust Control, 1st ed. Prentice Hall, 1997.

[3] Trajectory Generation for Minimum Closed-Loop State Sensitivity, ICRA 2018, Paolo Robuffo Giordano; Quentin Delamare; Antonio Franchi.

[4] WMR control via dynamic feedback linearization: design, implementation, and experimental validation (2002), G. Oriolo; A. De Luca; M. Vendittelli.