UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

FINAL PROJECT

FIELD AND SERVICE ROBOTICS

AUTONOMOUS CAR LIKE ROBOT CONTROL

Candidato
Matteo Langella
P38/212

Candidato
Raffaele Freschini
P38/226

Anno Accademico 2023/2024

Contents

1 Introduction **3**
1.1 Legend . . . 3

2 Modeling and analysis **4**
2.1 Car-Like Kinematic model . . . 4
2.2 Controllability Analysis . . . 5
2.3 Real model parameters . . . 6
2.3.1 Yuhesen FR-09 . . . 6
2.3.2 Agilex Hunter 2.0 . . . 6
2.3.3 Traxxas XRT . . . 7

3 Path and Trajectory Planning **8**
3.1 Circuits . . . 8
3.2 Path Calculation (Delaunay Triangulation) . . . 10
3.3 Trajectory Planning . . . 12

4 Sensing and Control **14**
4.1 Stabilization of trajectories for a non-constrained point . . . 14
4.2 Odometry . . . 15

5 Results and discussion **16**
5.1 Maximum Steering Angle Problem . . . 16
5.2 IO feedback Linearization Control . . . 17
5.2.1 Full-state Feedback . . . 17
5.2.2 Runge Kutta 2nd order . . . 19
5.2.3 Runge Kutta 4th order . . . 21
5.3 IO feedback Linearization Control with Model Uncertainty . . . 22
5.3.1 Full-state feedback . . . 22
5.3.2 With Odometry . . . 24
5.4 Map-Dependent Controller Gain Analysis . . . 25
5.5 Conclusion . . . 25

Chapter 1

Introduction

The objective of this project is to control a car-like robot for a trajectory tracking task by comparing the performance of different real robot models. The trajectories employed in the project were derived through the application of Delaunay's triangulation algorithm to a different set of circuits delineated by a variable number of landmarks. The circuits on which the triangulation algorithm is applied are representative of those circuits that are actually used in Formula Student racing. The study, therefore, proposes an analysis of the control's ability to execute plausible trajectories in the context of a competition. The control apparatus employs a model-based approach, with the objective of stabilising an unconstrained point in Cartesian space, connected to the robot's steering wheel reference system, around a pre-defined trajectory (I-O linearization). In respect of odometry, the initial assumption will be that the entire state vector is known instant by instant. Subsequently, second and fourth order Runge-Kutta integration method will be employed to estimate the state. Furthermore, in order to assess the robustness of the control, uncertainty will be introduced to the model parameters that will affect the performance and accuracy of the odometry.

1.1 Legend

** Landmarks

--- Reference

--- Real Trajectory

Chapter 2

Modeling and analysis

2.1 Car-Like Kinematic model

Let us consider a robot with the same kinematic characteristics as an automobile. For the sake of simplicity, it is assumed that the two wheels on each axle (front and rear) collapse into a single wheel located at the midpoint of the axle, thus creating a car-like model. The front wheel is steerable, whereas the rear wheel's orientation is fixed. The generalized coordinates are:

$$q = \begin{bmatrix} x \\ y \\ \theta \\ \phi \end{bmatrix}$$

where x and y are the cartesian coordinates of the rear wheel, θ measures the orientation of the car body with respect to the x-axis, and ϕ is the steering angle. The system is subject to two non-holonomic constraints (pure rolling constraint condition), one for each wheel:

$$\begin{cases} \dot{x}_f \sin(\theta + \phi) - \dot{y}_f \sin(\theta + \phi) = 0 \\ \dot{x} \sin\theta - \dot{y} \sin\theta = 0 \end{cases}$$

with x_f, y_f denoting the cartesian coordinates of the front wheel. By using the rigid-body constraint

$$\begin{cases} x_f = x + l \cos\theta \\ y_f = y + l \sin\theta \end{cases} \tag{2.1}$$

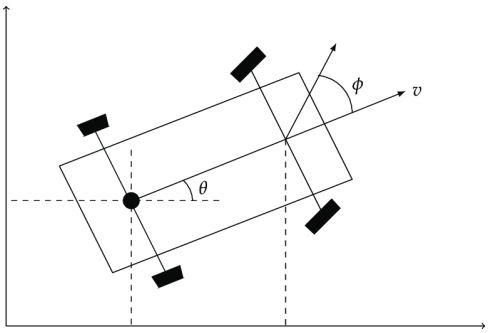


Figure 2.1

where l is the distance between the wheels. The first kinematic constraint becomes:

$$\dot{x}\,\sin(\theta+\phi) - \dot{y}\,\sin(\theta+\phi) - \dot{\theta}\,l\,\cos\phi = 0$$

The Pfaffian constraint matrix is:

$$C = \begin{bmatrix} \sin(\theta+\phi) & -\cos(\theta+\phi) & -l\,\cos\phi & 0 \\ \sin\theta & -\cos\theta & 0 & 0 \end{bmatrix}$$

and has constant rank equal to 2. If the car has rear-wheel drive, the kinematic model is derived as

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{bmatrix} v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v_2 \tag{2.2}$$

In this context, v_1 and v_2 represent the driving and steering velocity input, respectively. A model singularity is observed for the steering angle $\phi = \pi/2$, where the first vector field exhibits a discontinuity. This phenomenon is analogous to the vehicle becoming immobilised when the front wheel is aligned with the longitudinal axis of the body. Nevertheless, the significance of this singularity is constrained by the limited range of the steering angle ϕ in the majority of practical scenarios.

2.2 Controllability Analysis

Equation (2.2) may be rewritten as

$$\dot{q} = g_1(q)\,v_1 + g_2(q)\,v_2 \quad \text{with} \quad g_1 = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{bmatrix}, g_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.3}$$

The above system is non-linear, drift-less, and there are fewer control inputs than generalised coordinates. Although every driver's experience suggests that a car-like robot should be fully controllable, it is not trivial to establish such a property on a mathematical basis. A useful tool for testing the controllability of drift-less nonlinear systems is the is the *accessibility rank condition*. To verify this, it is necessary to establish whether the dimension of the accessibility distribution, denoted by $\Delta_a(q)$, is equal to the number of degrees of freedom, denoted by n, of the system under consideration. In this case, n is equal to 4. Since

$$\Delta_a = \text{span}\{g_1, g_2, g_3, g_4\}$$

where

$$g_3 = [g_1, g_2] = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{l\,\cos^2\phi} \\ 0 \end{bmatrix}$$

$$g_4 = [g_1, g_3] = \begin{bmatrix} -\frac{\sin\theta}{l\,\cos^2\phi} \\ -\frac{\cos\theta}{l\,\cos^2\phi} \\ 0 \\ 0 \end{bmatrix}$$

In accordance with prior statements, the system will be completely non-holonomic if the resulting matrix

$$F = \begin{bmatrix} \cos\theta & 0 & 0 & -\frac{\sin\theta}{l\,\cos^2\phi} \\ \sin\theta & 0 & 0 & \frac{\cos\theta}{l\,\cos^2\phi} \\ \frac{\tan\phi}{l} & 0 & -\frac{1}{l\,\cos^2\phi} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

is full rank. Since $rank(F) = 4 \implies dim(\Delta_a) = 4 \implies$ the system is completely non-holonomic.

2.3 Real model parameters

In order to simulate the behaviour of a real car-like robot, a number of models were selected from which the kinematic parameters were extrapolated. In particular, the parameters of most interest for this application are

- Maximum heading velocity (v_{max})
- Maximum steering velocity (w_{max})
- Maximum steering angle (ϕ_{max})
- Wheelbase length (l)

This section provides a concise overview of the models employed.

2.3.1 Yuhesen FR-09

Characteristics:

- $v_{max} = 5\,m/s$
- $w_{max} = 0.94\,rad/s$
- $\phi_{max} = \pm 0.47\,rad$
- $l = 0.85\,m$

2.3.2 Agilex Hunter 2.0

Characteristics:

- $v_{max} = 1.5\,m/s$
- $w_{max} = 1.16\,rad/s$
- $\phi_{max} = \pm 0.58\,rad$
- $l = 0.65\,m$

2.3.3 Traxxas XRT

Characteristics:

- $v_{max} = 10\,m/s$
- $w_{max} = 5.8\,rad/s$
- $\phi_{max} = \pm 1.4\,rad$
- $l = 0.48\,m$

It is to be expected that different models will perform in a way that is specific to the track on which they are operating. This is due to the fact that the steering angle references that have been assigned to each model may not align with the limits that have been set for it.

Chapter 3

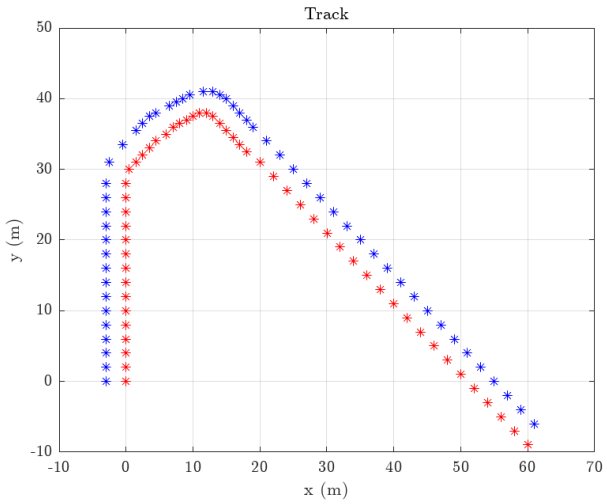
Path and Trajectory Planning

Let us examine the process of obtaining a trajectory from a classical test environment for this specific type of vehicle. We will analyse the steps involved in transforming the position of the landmarks delineating the path to be followed into a central path, and will then apply a time law to execute this path. The present study will not address issues and cases concerning the acquisition and updating of landmarks' positions. It should be noted that these are not fictitious circuits; rather, one of them represents a genuine circuit employed in a Formula Student competition, where the algorithms under discussion are routinely deployed.

3.1 Circuits

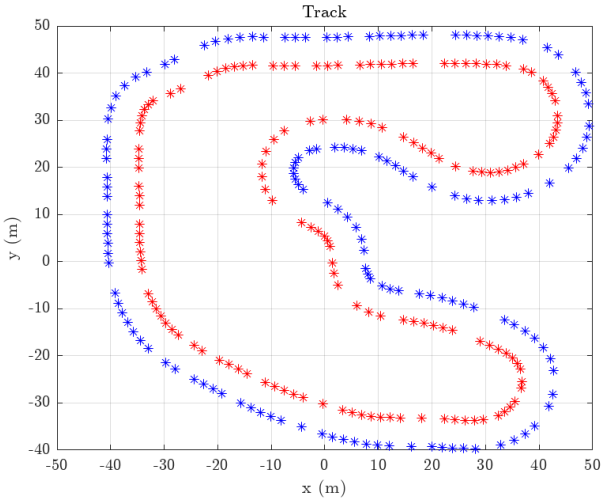
The circuits employed vary in size and shape, with markedly disparate numbers of landmarks, to assess the resilience of the central path calculation algorithm, designated as Delaunay Triangulation.

- **Simple Circuit**



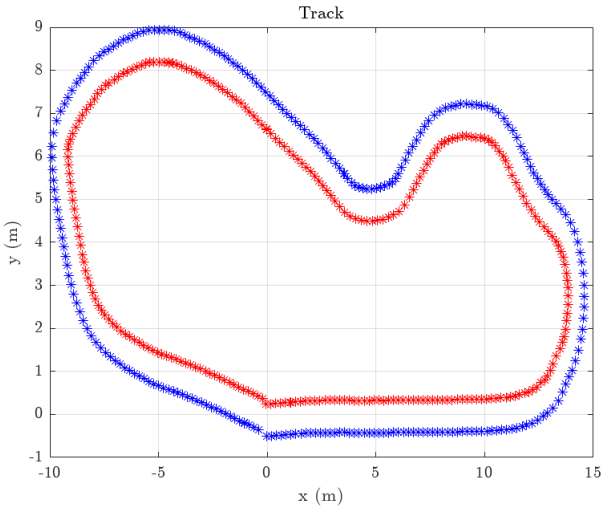
Number of landmarks: 38.
We have constructed a straightforward circuit for the purpose of initiating model testing and attempting to saturate the v variable.

- **Matlab Circuit**



Number of landmarks: 147.
A circuit provided by MATLAB is more complex than the previous one and comprises a greater number of bends.

- **FSI 2022 Autocross circuit**



Number of landmarks: 500.
A genuine circuit devised during the FSI 2022 competition for the Driverless section, comprising a multitude of cones, diminutive in comparison to the preceding one and exhibiting considerably elongated bends.

3.2 Path Calculation (Delaunay Triangulation)

In this type of study, the path planning plays a significant role in understanding the behaviour and performance of the controller. Given the presence of landmarks delineating the track, the Delaunay triangulation emerges as a compelling geometric approach, offering a robust foundation for path planning in this context.

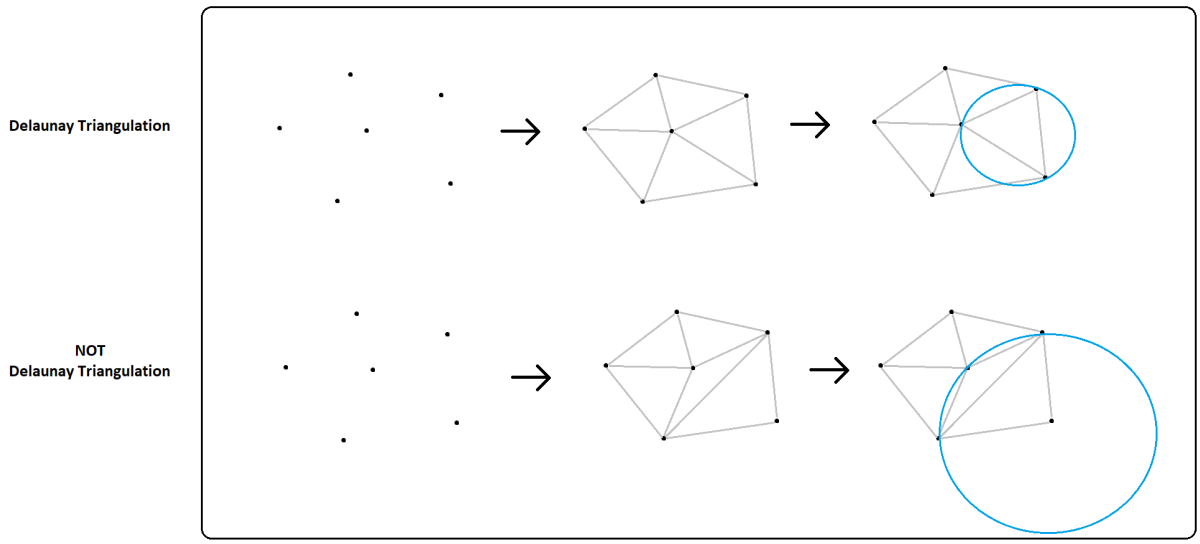


Figure 3.1: Delaunay Triangulation

Let us consider a set of points across a surface. The Delaunay triangulation of these points subdivides the area into triangles in a way that no other points lie inside the empty circle formed by connecting the three corners of a triangle. This results in the maximisation of the minimum angle within the triangulation, leading to the formation of well-shaped triangles with desirable properties for path planning. This approach ensures obstacle avoidance in accordance with the "no point in the circle rule," thereby providing the robot with a navigational roadmap. It is important to note that this approach does not guarantee the absolute shortest path or the optimal path. However, it provides a valuable reference point for achieving these objectives. In the case of a track delimited by landmarks, this approach results in a path that is perfectly centered within the track, which is sufficient for testing the control. In a more complex control scenario, such as an MPC, the central reference can be utilized to obtain an optimal path after the control.

With regard to our case study, the stages of the path planning process are as follows:

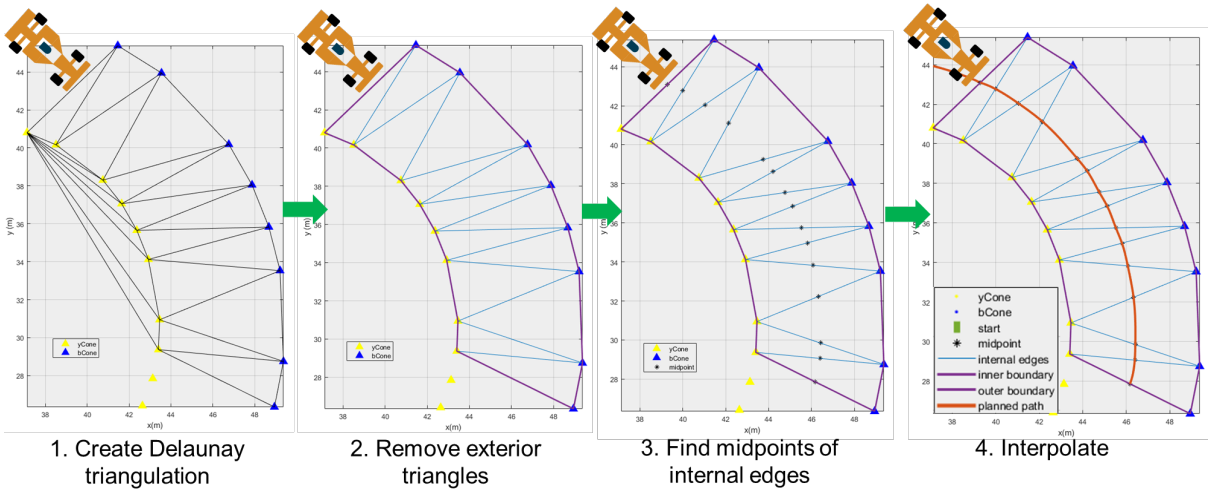


Figure 3.2: Path Planning stages

The second phase is of significant importance, as while performing triangulation, the coordinates of the inner and outer cones are bound to create triangles outside the boundary of the track. As these triangles have the potential to generate an erroneous path, they have been removed by imposing constraints, designated as C. These constraints comprise the vertex IDs of constrained edges, specified as a two-column matrix. Each row of C corresponds to a constrained edge and contains two IDs:

- C(j,1) is the ID of the vertex at the commencement of an edge.
- C(j,2) is the ID of the vertex at the conclusion of the edge.

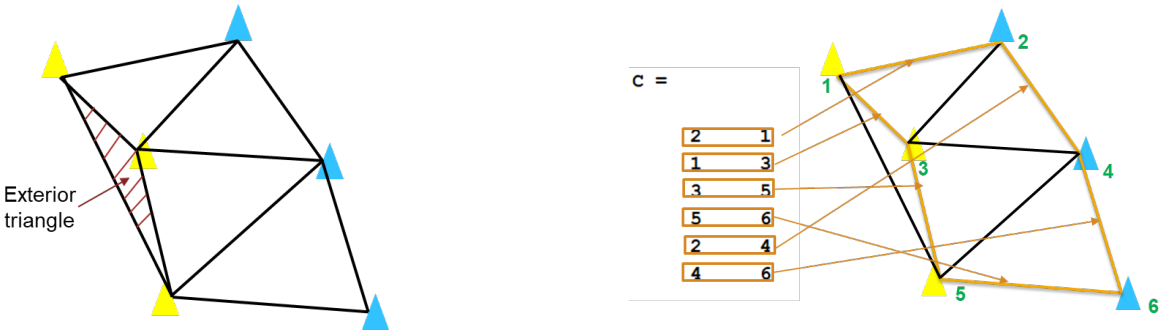


Figure 3.3: *Second phase zoom*

The third and fourth stages are relatively straightforward and intuitive:

- in the third stage, the midpoint between the connection of the landmarks is selected;
- in the fourth stage, the interpolation is performed.

A series of experiments were conducted to assess the impact of varying the number of landmarks considered simultaneously during the execution of the algorithm. To minimise the simulation time and reduce the number of interpolation points, a range of 10–20 landmarks was selected for each iteration, which is a realistic number for a practical application involving a camera and a neural network for image recognition.

3.3 Trajectory Planning

Assume that a feasible and smooth desired output trajectory is given in terms of the cartesian position of the car rear wheel:

$$x_d = x_d(t), \quad y_d = y_d(t), \quad t \geq t_0 \tag{3.1}$$

This natural approach to defining the motion of a car-like robot exhibits an appealing property. Indeed, this information allows us to derive the corresponding time evolution of the remaining coordinates (state trajectory) and the associated input commands (input trajectory). This is feasible due to the fact that, analogous to the unicycle, the Cartesian coordinates are the flat outputs of the system. The desired output trajectory is feasible when it can be obtained from the evolution of a reference car-like robot.

$$\dot{x}_d = \cos\theta_d \, v_{1d} \tag{3.2}$$

$$\dot{y}_d = \sin\theta_d \, v_{1d} \tag{3.3}$$

$$\dot{\theta}_d = \frac{\tan\phi_d}{l} \, v_{1d} \tag{3.4}$$

$$\dot{\phi}_d = v_{2d} \tag{3.5}$$

Solving for v_{1d} from (3.2) and (3.3) we get

$$v_{1d}(t) = \pm\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \tag{3.6}$$

where the $\pm$ sign represents the choice of moving forward or backwards.
Dividing equations (3.5) by (3.4), and keeping the sign of the linear velocity input into account, we compute the desired orientation of the car as

$$\theta_d(t) = ATAN2\left\{\frac{\dot{y}_d(t)}{v_{1d}(t)}, \frac{\dot{x}_d(t)}{v_{1d}(t)}\right\} \tag{3.7}$$

Differentiating eqs. (3.2) and (3.3), and combining the results so as to eliminate v_{1d}, we obtain

$$\dot{\theta}_d(t) = \frac{\ddot{y}_d(t)\,\dot{x}_d(t) - \ddot{x}_d(t)\,\dot{y}_d(t)}{v_{1d}^2(t)} \tag{3.8}$$

Plugging this into eq. (3.4) we get the desired steering angle

$$\dot{\phi}_d(t) = \arctan\left\{\frac{l\,[\ddot{y}_d(t)\,\dot{x}_d(t) - \ddot{x}_d(t)\,\dot{y}_d(t)]}{v_{1d}^3(t)}\right\} \tag{3.9}$$

Finally, differentiating (3.9) and substituting the result in eq. (3.5) yields the second input

$$v_{2d}(t) = l\,v_{1d}\,\frac{(\dddot{y}_d\,\dot{x}_d - \dddot{x}_d\,\dot{y}_d)\,v_{1d}^2 - 3\,(\ddot{y}_d\,\dot{x}_d - \ddot{x}_d\,\dot{y}_d)\,(\ddot{x}_d\,\dot{x}_d - \ddot{y}_d\,\dot{y}_d)}{v_{1d}^6(t) + l^2(\ddot{y}_d\,\dot{x}_d - \ddot{x}_d\,\dot{y}_d)^2} \tag{3.10}$$

Equations (3.6 - 3.10) provide the unique state and input trajectory needed to reproduce the desired output trajectory. These expressions depend only on the values of the output

trajectory (3.1) and its derivatives up to the third order. The only situation in which the reference signals (3.6) - (3.10) are not defined is when $v_{1d} = 0$ for some $\bar{t} \geq t_0$. In this case, it is convenient to use a parameterized trajectory in which the geometric path description is separated from the timing information. This is the methodology that was employed in the course of this research project. The initial step involved deriving the geometric path through Delaunay, followed by parameterising all the references with respect to the curve. Subsequently, a polynomial time law was assigned to the resulting outcome. Subsequently, the trajectory is scaled in time in order to ensure that the maximum speeds that can be developed by the robot are in accordance with the imposed constraints.

Chapter 4

Sensing and Control

4.1 Stabilization of trajectories for a non-constrained point

The objective of the control technique that we present is to stabilise the position of an unconstrained point in Cartesian space around a reference trajectory. Let us consider the position $P = (y_1, y_2)$ of a point that is connected to the car chassis at a certain distance b from the front wheels (2.1).

$$\begin{cases} y_1 = x_f + b\,\cos(\theta + \phi) \\ y_2 = y_f + b\,\sin(\theta + \phi) \end{cases} \tag{4.1}$$

By deriving this relationship with respect to time and substituting $\dot{x}$ and $\dot{y}$ from the model equations (2.2), we obtain

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = T(\theta, \phi) \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{4.2}$$

where

$$T(\theta, \phi) = \begin{bmatrix} \cos\theta - \tan\phi\,\sin\theta - b\,\frac{\tan\phi}{l}\,\sin(\theta+\phi) & -b\,\sin(\theta+\phi) \\ \sin\theta + \tan\phi\,\cos\theta + b\,\frac{\tan\phi}{l}\,\cos(\theta+\phi) & +b\,\cos(\theta+\phi) \end{bmatrix} \tag{4.3}$$

It can be demonstrated that the determinant of the matrix $\det(T) \neq 0$ only when $b \neq 0$. Indeed, T represents the input matrix of the reduced system, comprising solely the coordinates of the unconstrained point.
Consequently, if T is invertible, it can be concluded that the system in question is fully actuated. Given such condition, we can design

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = T^{-1}(\theta, \phi) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.4}$$

where u_1 and u_2 are auxiliary control inputs to be decided later. Substituting (4.4) in (4.2) we obtain

$$\begin{cases} \dot{y}_1 = u_1 \\ \dot{y}_2 = u_2 \end{cases} \tag{4.5}$$

Now we can choose our virtual inputs as

$$\begin{cases} u_1 = \dot{y}_{1d} + k1\,(y_{1d} - y_1) \\ u_2 = \dot{y}_{2d} + k2\,(y_{2d} - y_2) \end{cases} \tag{4.6}$$

with $k1, k2 > 0$. This selection of auxiliary inputs ensures that the two variables will converge on the desired value. Nevertheless, this approach, which deals with stabilisation around an unconstrained point trajectory, leaves the orientation of the robot uncontrolled.

4.2 Odometry

The preliminary objective is to evaluate the overall performance of the controller, operating under the assumption that the system state is fully known and that highly reliable proprioceptive and exteroceptive sensors are available. Subsequently, the Runge-Kutta method was implemented for the odometry component. In a practical setting, the use of Runge-Kutta for extended paths may result in a notable decline in performance and an accumulation of errors. However, due to the nature of simulation, the impact of this loss of control may not be readily discernible. Nevertheless, it might be of interest to evaluate the robustness of the control by introducing an element of uncertainty into the model parameters, which will inevitably be reflected in the calculation of the state.
The second order Runge-Kutta equations for this model are

$$\begin{cases} x_{k+1} = x_k + v_{1k}\,T_s\,\cos[\theta_k + \frac{1}{2}\,v_{1k}\,T_s\,\frac{\tan\phi_k}{l}] \\ y_{k+1} = y_k + v_{1k}\,T_s\,\sin[\theta_k + \frac{1}{2}\,v_{1k}\,T_s\,\frac{\tan\phi_k}{l}] \\ \theta_{k+1} = \theta_k + v_{1k}\,T_s\,\frac{\tan\phi_k}{l} \\ \phi_{k+1} = \phi_k + v_{2k}\,T_s \end{cases} \tag{4.7}$$

where T_s is the sample time and v_{1k} and v_{2k} are the commanded velocities.

The fourth order Runge-Kutta equations for this model are:

$$\begin{cases} x_{k+1} = x_k + v_{1k}\,\frac{T_s}{6}\,[\cos\theta_k + 5\,\cos(\theta_k + \frac{1}{2}\,v_{1k}\,T_s\,\frac{\tan\phi_k}{l})] \\ y_{k+1} = y_k + v_{1k}\,\frac{T_s}{6}\,[\sin\theta_k + 5\,\sin(\theta_k + \frac{1}{2}\,v_{1k}\,T_s\,\frac{\tan\phi_k}{l})] \\ \theta_{k+1} = \theta_k + v_{1k}\,T_s\,\frac{\tan\phi_k}{l} \\ \phi_{k+1} = \phi_k + v_{2k}\,T_s \end{cases} \tag{4.8}$$

Chapter 5

Results and discussion

5.1 Maximum Steering Angle Problem

One of the issues that arose from the tests pertains to the maximum feasible steering angle for the models and the angle required by the circuit itself to ensure optimal trajectory tracking. The control system demonstrates effective tracking capabilities even when the steering angle required by the circuit exceeds the available range. The following section presents cases where this phenomenon occurs and tests the limits of the control's robustness. One of the most extreme cases that we encountered was with the *FR09* in the *FSI* circuit. It proved impossible to allow the robot to perform the track in an accurate way due to the high discrepancy (76%) between the maximum curvature and the curvature limit of the robot.

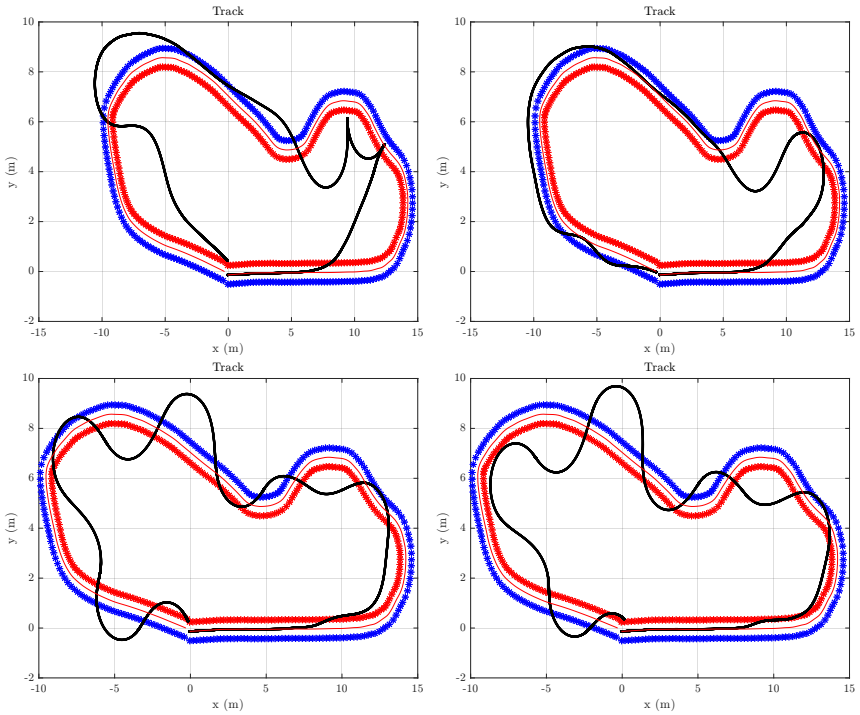


Figure 5.1: *FR09 in FSI Circuit with Full-State Feedback approch. Gains:* $k_1 = 2$, $k_2 = 0.5$*;* $k_1 = 2$, $k_2 = 1.75$*;* $k_1 = 3$, $k_2 = 6.5$*;* $k_1 = 5$, $k_2 = 10.5$ *[from left to right]*

5.2 IO feedback Linearization Control

5.2.1 Full-state Feedback

The following illustration depicts the controller's behaviour in the presence of complete state feedback. The Matlab circuit and the Traxxar model, which features Ackermann steering and is the fastest and has the highest steering capability of the three models, were selected for testing purposes.

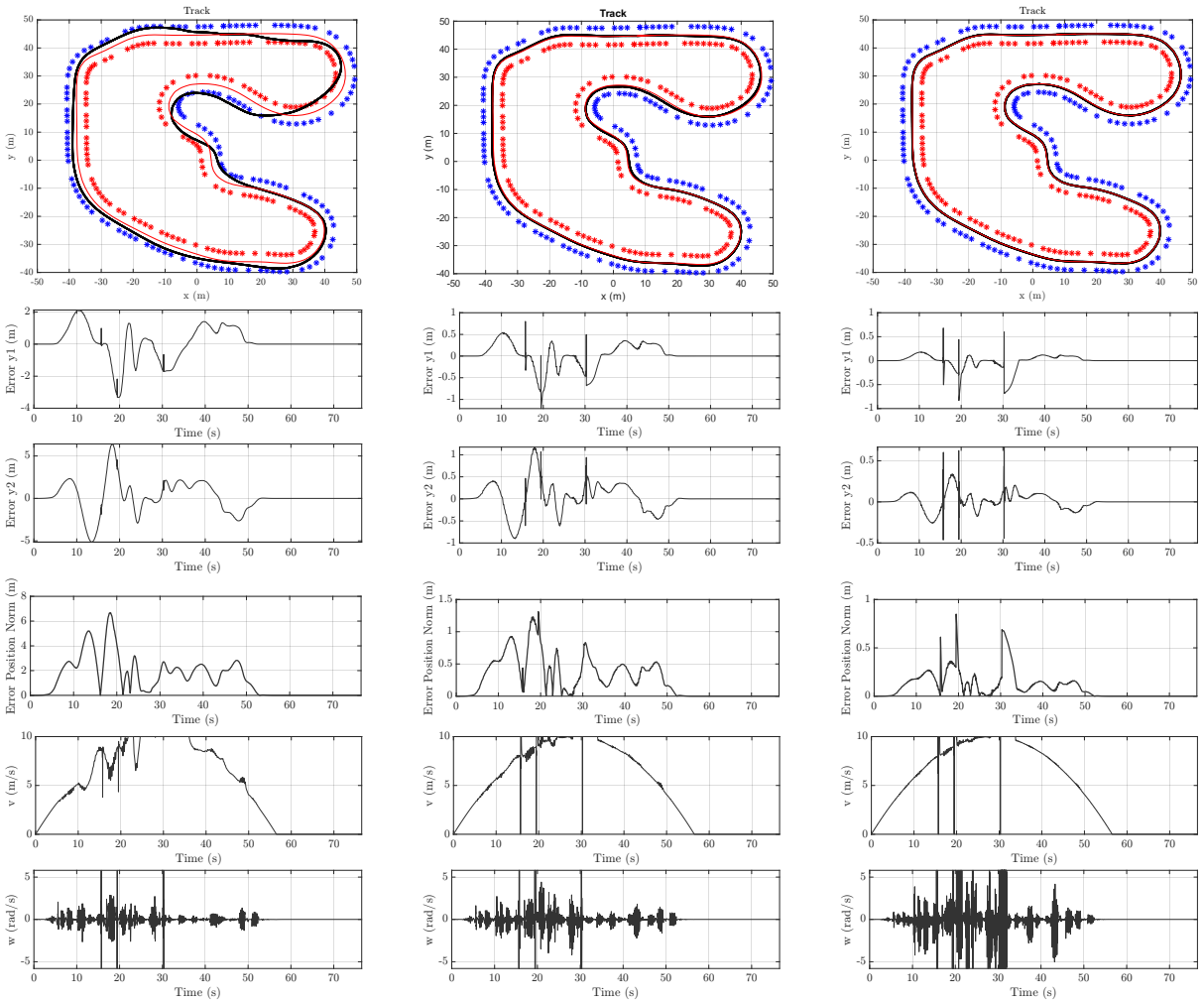


Figure 5.2: *Traxxar in Matlab Circuit with Full-State Feedback approch. Gains:* $k_1 = 5$, $k_2 = 2$*;* $k_1 = 20$, $k_2 = 12$*;* $k_1 = 60$, $k_2 = 42$ *[from left to right]*

As can be observed, as k_1 and k_2 increase, the model demonstrates enhanced capability to follow the trajectory within the confines of the track limits, as evidenced by the graph of the decreasing error norm. Nevertheless, an increase in the gains results in more onerous control actions for the system, as evidenced by the continuous saturations observed in the w graph and the y_1 and y_2 error graphs. The same outcome is observed in three additional simulations, in which the Hunter model is employed on a simple circuit of our own design.

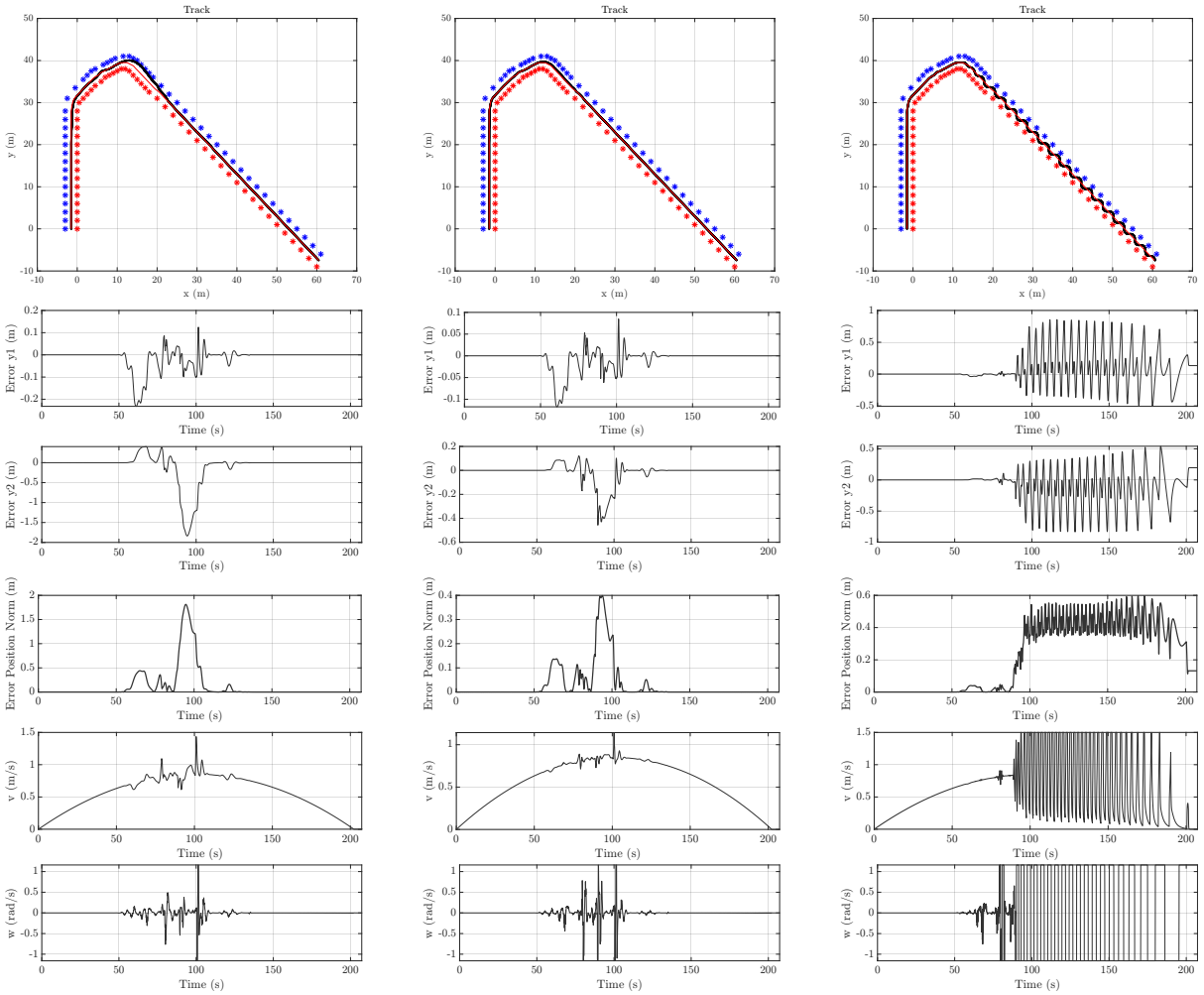


Figure 5.3: *Hunter in Simple Circuit with Full-State Feedback approch. Gains:* $k_1 = 2.5$, $k_2 = 0.5$*;* $k_1 = 5$, $k_2 = 2.5$*;* $k_1 = 15$, $k_2 = 12.5$ *[from left to right]*

In this instance, we may also observe another curious phenomenon: an oscillation around the reference trajectory, accompanied by a complete chattering of the v (v_1) and ω (v_2) actions. This phenomenon is caused by the curve of the track, which is particularly tight (with a maximum curvature of 0.68 and a limiting curvature of 0.58 for the robot), and the high values of k_1 and k_2. Upon reducing these two gains slightly, it becomes evident that while the chattering is less pronounced, it nevertheless commences almost immediately after the curve.
Therefore, even in the event of a maximum curvature exceeding the robot's curvature limit by 10%, the control system is sufficiently robust to enable the robot to complete the track.

5.2.2 Runge Kutta 2nd order

In order to facilitate a meaningful comparison, the entire test will be repeated with the inclusion of Runge-Kutta feedback. To ensure consistency, the same gains were employed in both tests.

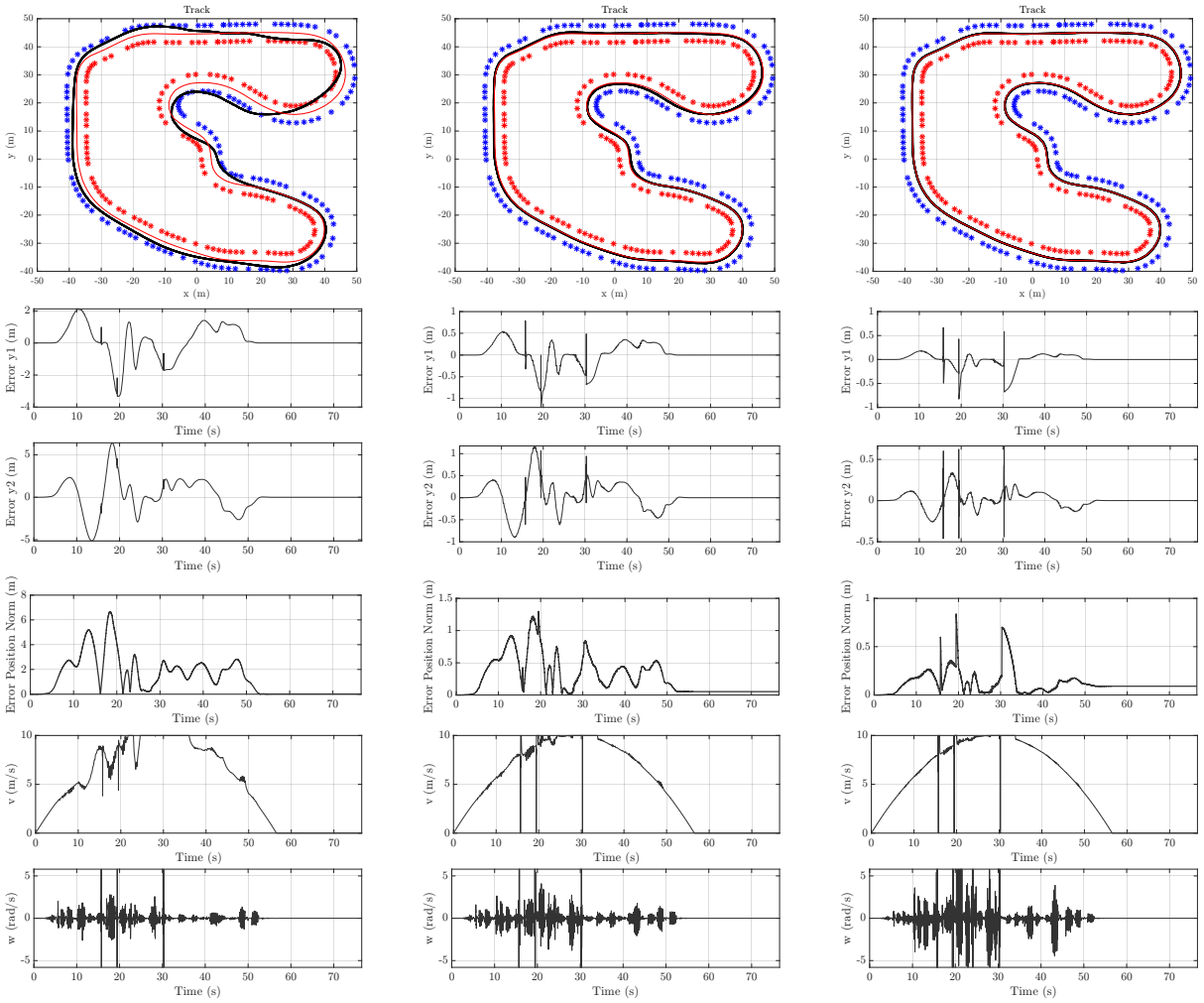


Figure 5.4: *Traxxar in Matlab Circuit with Runge-Kutta integration method. Gains:* $k_1 = 5$, $k_2 = 2$*;* $k_1 = 20$, $k_2 = 12$*;* $k_1 = 60$, $k_2 = 42$ *[from left to right]*

As can be observed, given that we are in a simulation environment, there is not a significant distinction between the plot with the full-state feedback and the Runge-Kutta of the second order. However, there is a notable difference in the plot of ω (v_2) in the case of high values of gains, as there is less chattering and a more typical action.

A direct comparison between the FR09 and the Simple circuit, utilising the same values of gains, reveals significant discrepancies between the full-state error feedback and the Runge-Kutta 2th approach.

The FR09 with full-state feedback is capable of completing the track with the same gains. However, the Runge Kutta 2th approach results in a sudden divergence after the track curve. This discrepancy is reflected in all the plots as a consequence. Nevertheless, the control demonstrates resilience in this instance, as evidenced by the discrepancy between the limit curvature of the robot and the maximum curvature of the track, which amounts to 66%.

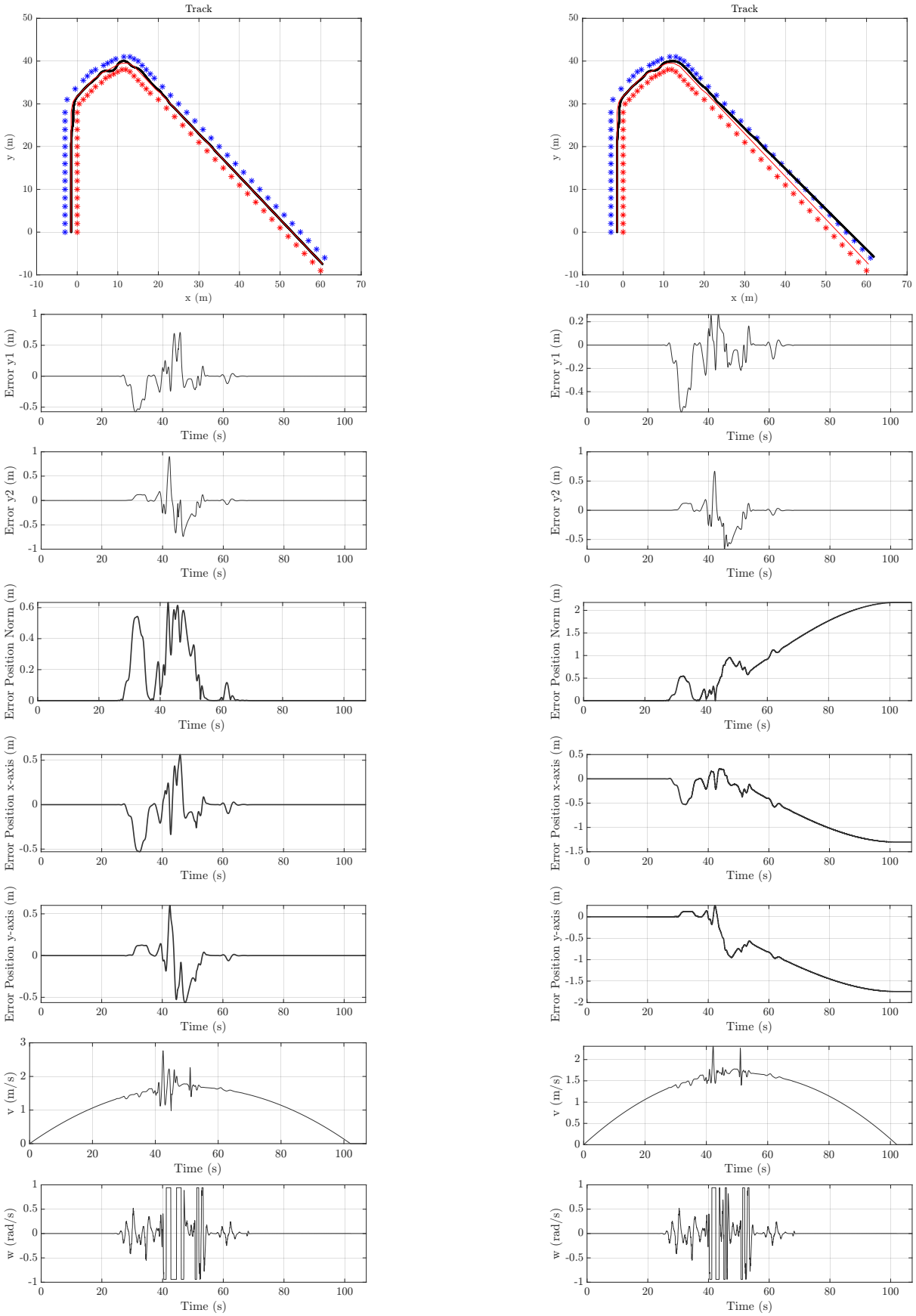


Figure 5.5: *FR09 in Simple Circuit with Full-State Feedback approch [L] and Runge Kutta 2th approch [R] with gains:* $k_1 = 2$, $k_2 = 3.5$

5.2.3 Runge Kutta 4th order

A direct comparison was conducted between the Traxxar in the FSI circuit across the three cases:

- Full-State Feedback,
- Runge Kutta of 2nd order,
- Runge Kutta of 4th order;

which demonstrated that the only discernible difference was in the w action. Given the nature of the simulation, there was no significant distinction in the utilisation of Runge-Kutta of the second or fourth order.

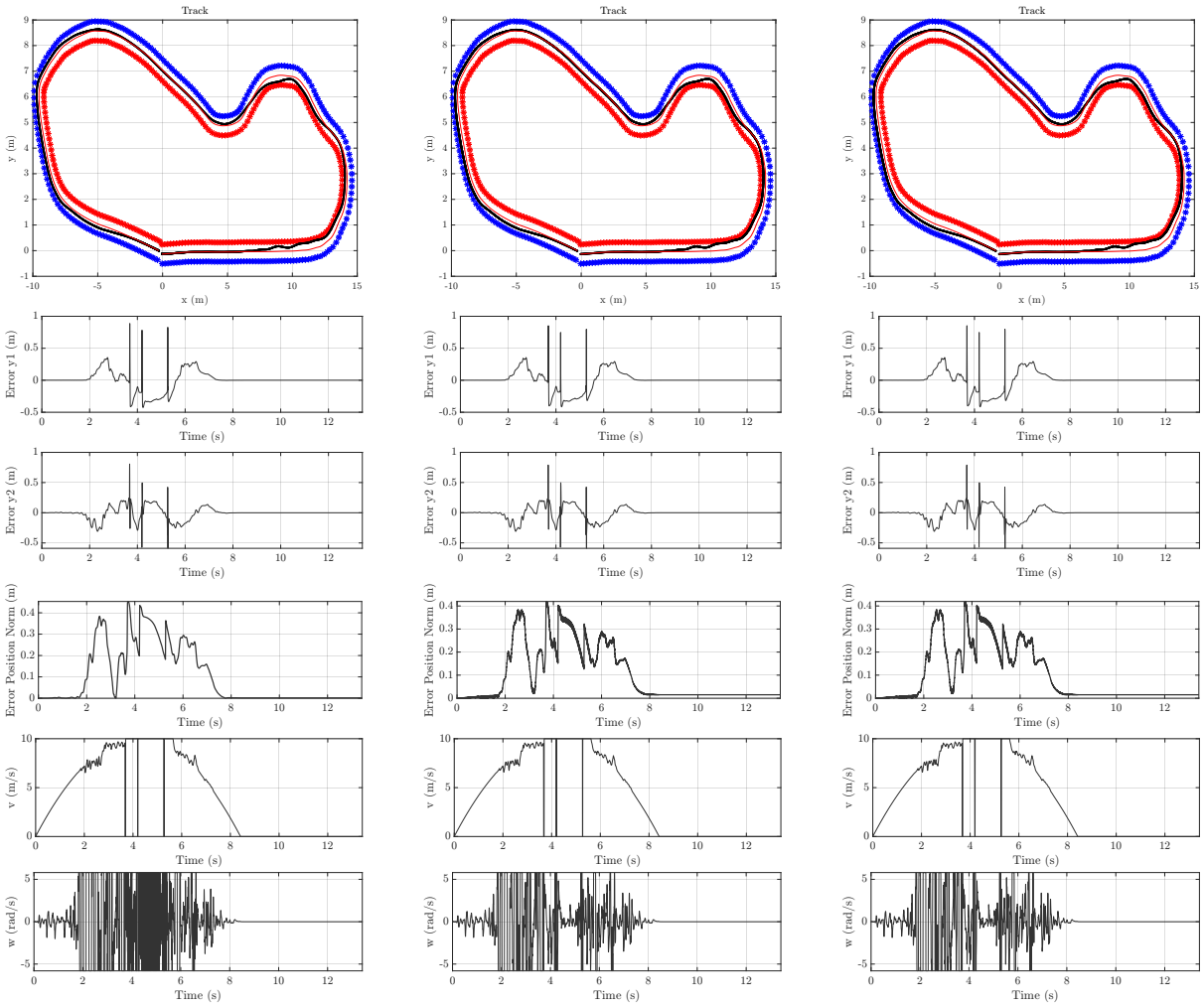


Figure 5.6: *Traxxar in FSI circuit with Full-state Feedback (on the left), with Runge Kutta of 2nd order (in the middle) and with Runge Kutta of 4th order (on the right). Gains:* $k_1 = 40$, $k_2 = 35$

5.3 IO feedback Linearization Control with Model Uncertainty

This paragraph presents an analysis of the robustness of the control in the context of uncertainty in the model. The focus is on the modification of the L value, which represents the wheelbase of the car-like robots. In order to highlight the distinction in the case of ambiguity, we employed the simplest circuit, which is the shortest and has a highly concentrated curve in comparison to the others, which can facilitate more rigorous control.

5.3.1 Full-state feedback

In this instance, a comparison was conducted between the Hunter model in the Simple circuit and a series of wheelbase variations, ranging from -5% to 5%, where there are not significant differences. If the wheelbase value is tested with a major error, there is a slight difference in the behaviour of the robot, but the control remains robust. This is consistent with the hypothesis that, in the case of Full State Feedback, this control is robust.

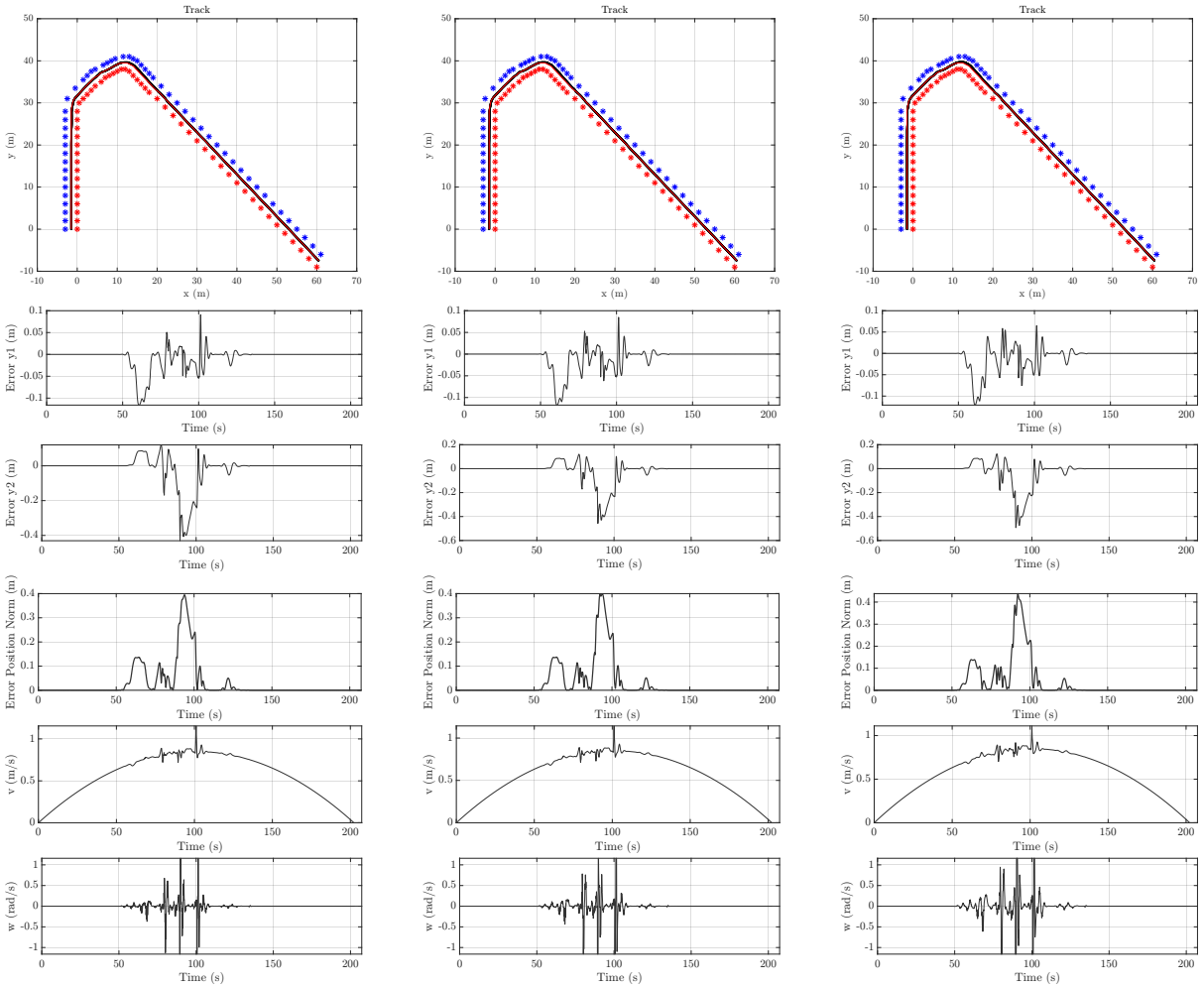


Figure 5.7: *Hunter model in the Simpler Circuit with an uncertainty of -30% ($l = 0.455m$) on the right, uncertainty of 0% ($l = 0.65m$) in the middle and an uncertainty of a 60% ($l = 1.17m$) on the right. Gains: $k_1 = 5$, $k_2 = 2.5$*

A similar outcome is observed in the case of a Traxxar model in the Matlab Circuit. However, given a discrepancy of 35% from the actual wheelbase, there are slight discrepancies.

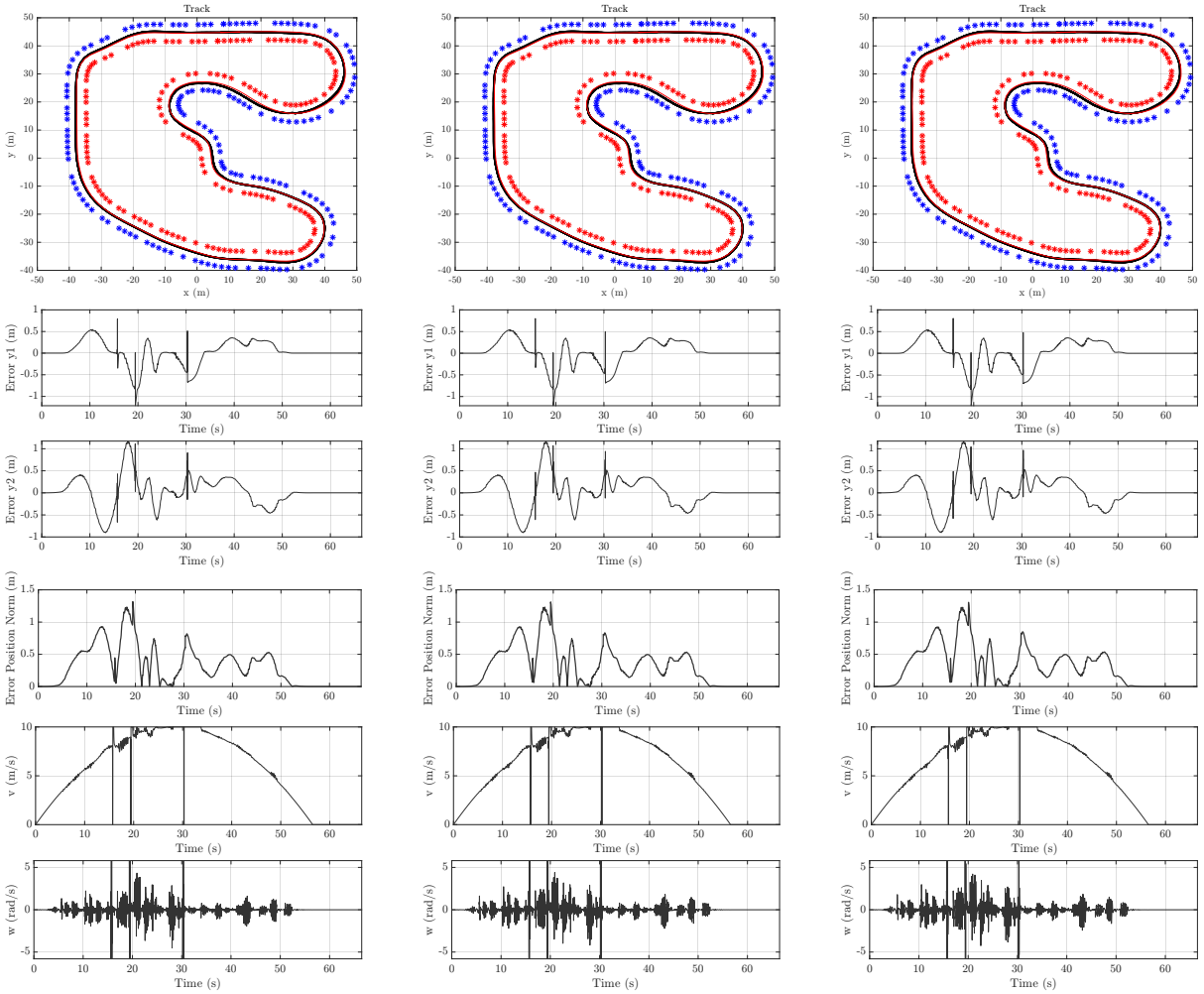


Figure 5.8: *Traxxar model in the Matlab Circuit with an uncertainty of -35% ($l = 0.312m$) on the right, uncertainty of 0% ($l = 0.48m$) in the middle and an uncertainty of a 35% ($l = 0.648m$) on the right. Gains: $k_1 = 20$, $k_2 = 12$*

5.3.2 With Odometry

The experiment was repeated with the application of the Runge–Kutta method of second order to the Hunter model in the Simpler Circuit.

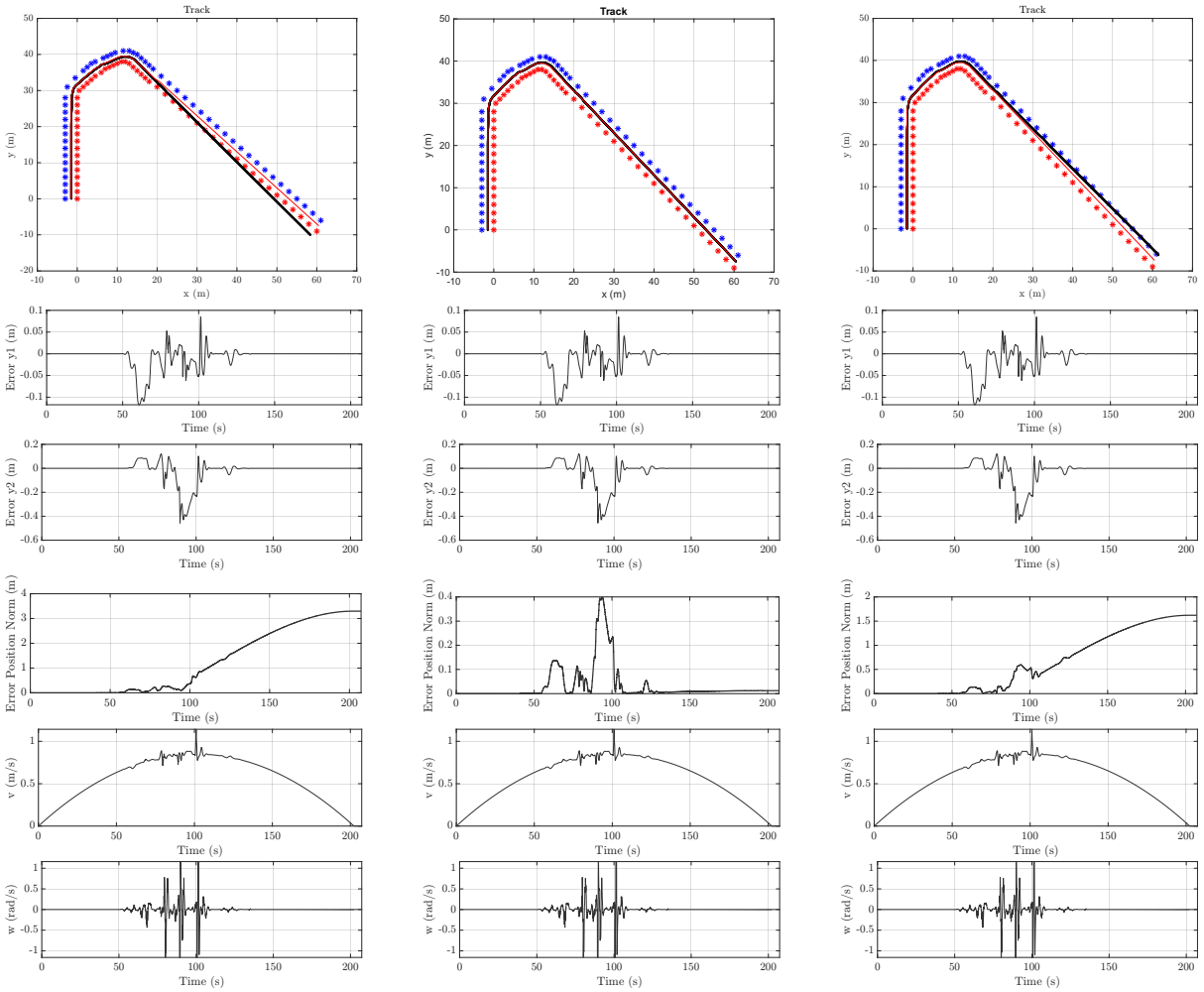


Figure 5.9: *Hunter model in the Simpler Circuit with Runge Kutta of 2th order and an uncertainty of -2% ($l = 0.637m$) on the right, uncertainty of 0% ($l = 0.65m$) in the middle and an uncertainty of a 1% ($l = 0.6565m$) on the right. Gains: $k_1 = 5$, $k_2 = 2.5$*

In accordance with the theory studied in class, Runge-Kutta is highly sensitive to uncertainty in the model. Indeed, with only 1% uncertainty in the wheelbase, the robot exhibited markedly divergent behavioural profile. In the absence of a discernible difference, an uncertainty of a maximum of 0.5% should be permitted. The same results are obtained with the RK4 method.

5.4 Map-Dependent Controller Gain Analysis

A series of analyses were conducted to evaluate the efficacy of the control in the context of map alteration. The objective is to ascertain whether, within the specified modification environment, the robot is capable of completing the circuit with the same gain values of k_1 and k_2. The results of the test demonstrate that the control is sufficiently robust to permit the use of the same gain in all three maps, thereby ensuring optimal behaviour. It is, however, possible to achieve even better behaviour by making minor adjustments to each map, which is in line with the type of controller that has been implemented. In order to achieve complete independence from the environment and the map, it would be necessary to transition from this type of control to a more complex nonlinear model-based control.

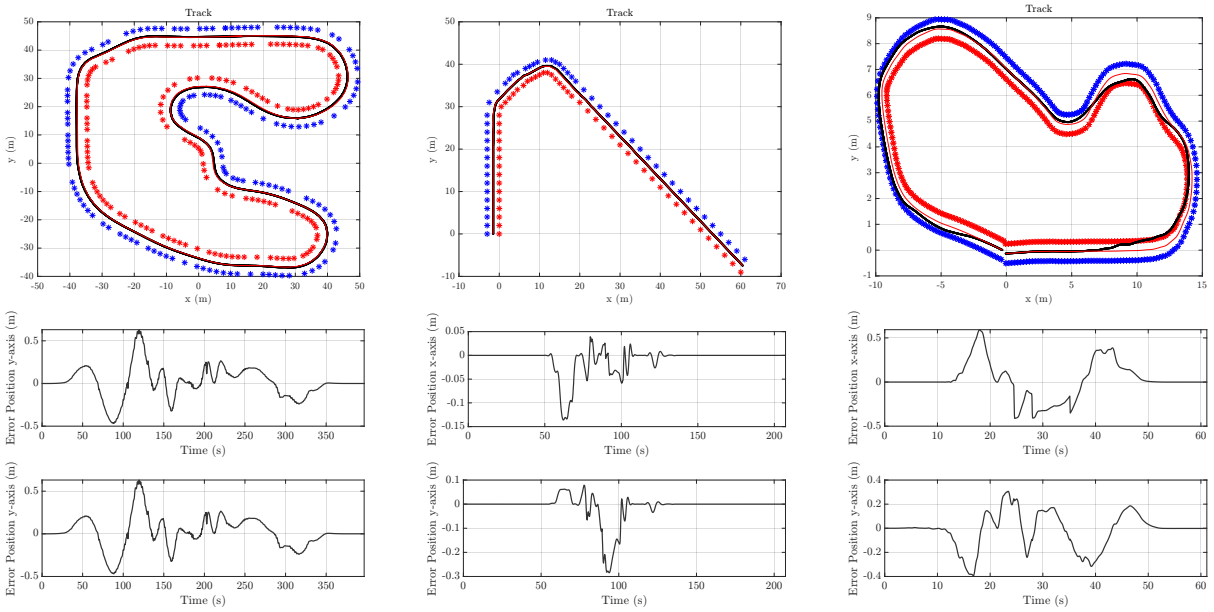


Figure 5.10: *Hunter model in the three circuit (from left to right: Matlab, Simple and FSI) with the same gains:* $k_1 = 5$, $k_2 = 2.5$

5.5 Conclusion

In conclusion, the controller is able to govern the mobile robots in a robust way with a good performance, despite the initial expectation. It is evident that certain limitations are imposed by the robot model (for example, the FR09 in the FSI Circuit) and the nature of the simulations themselves, which prevent us from fully appreciating the advantages and disadvantages of an odometry approach.