

UNIVERSITÀ
DI PAVIA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

MASTER'S DEGREE IN INDUSTRIAL AUTOMATION ENGINEERING

MASTER THESIS

Microscopic Traffic Simulation and Patient-in-the-Loop Longitudinal Control in a Driving Rehabilitation Context

SUPERVISOR

Prof. Antonella Ferrara

CANDIDATE

Davide Faiola

ID: 563969

CO-SUPERVISOR

Nikolas Sacchi, PhD

A.Y. 2025/2026

To my family, always by my side.

Contents

Introduction **5**

Motivation and context 5

Objectives of the thesis 6

Thesis organisation 7

1 Background and state of the art **9**

1.1 Clinical background and driving rehabilitation 9

1.2 Control-oriented driver modelling and assisted driving 11

1.3 Research gap and positioning of the thesis 14

2 Traffic-level simulation framework **16**

2.1 Introduction 16

2.2 Python–SUMO simulation framework 17

2.2.1 Software organisation 17

2.2.2 SUMO files and simulation configuration 19

2.2.3 Interaction between Python and SUMO through TraCI 20

2.3 Graph-based network representation and route computation 22

2.3.1 Road network representation with NetworkX 22

2.3.2 Route computation and time-driven traffic-aware rerouting 24

2.4 Traffic data and simulation outputs 27

2.5 Example of a complete traffic simulation 28

2.5.1 Simulation setup 28

2.5.2 Background traffic simulation results 31

2.5.3 EGO vehicle simulation results 35

3 Microscopic longitudinal control **39**

3.1 Introduction 39

3.2 Software architecture 39

3.2.1 SUMO simulation environment 41

3.2.2 High-level simulation flow 41

3.3 Discrete-time state-space model 43

3.3.1 Longitudinal plant model 44

3.3.2 Delay as an augmented state 47

3.4 Patient model and assist control 49

3.4.1 Patient controller 49

3.4.2 Patient profiles 49

3.4.3 Questionnaire-based patient characterisation 50

3.4.4 Driving test-based patient parameter identification 53

3.4.5 Assist controller 55

3.4.6 Threshold-based assistance activation 57

3.4.7 Parallel control block diagram 59

3.5 Reference velocity profiles generation 61

3.6 Simulation scenarios 62

3.6.1 Base scenario 62

3.6.2 Lead-brake scenario 62

3.7 Simulation loop and data logging 66

3.7.1 CSV output 67

3.7.2 Simulation summary 68

4 Simulation scenarios and performance evaluation **70**

4.1 Base scenario with trapezoidal reference 71

4.1.1 All patient profiles - assist OFF 71

4.1.2 Focus on delayed profile - assist ON/OFF 73

4.2 Base scenario with sinusoidal reference 76
4.2.1 All patient profiles - assist OFF 76
4.2.2 Focus on cautious profile - assist ON/OFF 79
4.3 Base scenario with step reference 81
4.3.1 All patient profiles - assist OFF 81
4.3.2 Focus on normal profile - assist ON/OFF 84
4.4 Base scenario with exponential stop-and-go reference 86
4.4.1 All patient profiles - assist OFF 86
4.4.2 Focus on reactive profile - assist ON/OFF 88
4.5 Lead-brake scenario with trapezoidal reference 91
4.5.1 All patient profiles - assist OFF 91
4.5.2 Focus on delayed profile - assist ON 94
4.6 Discussion of simulation results 97
4.6.1 Different interpretations of the patient profiles 97
4.6.2 Interpretation of the reference speed profiles 98
4.6.3 Effectiveness of the assist controller 98
4.6.4 Analysis of the lead-brake scenario 99
4.6.5 Strengths, limitations and possible extensions 100

Conclusions **102**

Bibliography **104**

List of figures **108**

List of tables **111**

Acknowledgements **113**

Introduction

Motivation and context

Driving represents an important component of personal mobility and independent participation in social and community activities. The ability to resume driving after the onset of a neurological, motor or sensory impairment may therefore have consequences that extend beyond transportation itself. In particular, returning to driving after stroke has been associated with improved community reintegration, although the decision to resume driving must take into account the cognitive, perceptual and motor abilities required to operate a vehicle safely [1, 2].

Driving rehabilitation includes different forms of assessment and training aimed at determining whether an individual can safely return to driving or improve specific driving-related abilities. These interventions may involve clinical assessments, on-road evaluations, cognitive and perceptual training, vehicle adaptations and simulator-based exercises [3, 4]. Driving simulators are particularly relevant during the preliminary stages of rehabilitation, since they allow repeatable driving situations and potentially hazardous events to be reproduced without exposing the patient or other road users to real risk.

Several studies have investigated simulator-based training for individuals with neurological impairments. Improvements in simulated or on-road driving performance have been reported in some cases, particularly after stroke [5, 6]. Nevertheless, systematic reviews have highlighted the limited number and heterogeneity of the available studies, and the current evidence is not sufficient to establish conclusively the effectiveness of driving rehabilitation in improving on-road driving ability [7]. Simulator-based approaches should therefore be regarded as promising tools for assessment, training and preliminary validation rather than as clinically established substitutes for conventional rehabilitation.

At the same time, developments in Advanced Driver Assistance Systems (ADAS) and partially automated vehicles have introduced new forms of interaction between the driver and the vehicle. In shared-control architectures, the human driver and the automated system both contribute to the vehicle control task, with the level of automation authority varying according to the system design, the driver's behaviour and the current operating

conditions [8, 9]. This differs from fully autonomous driving, since the human remains an active component of the control loop rather than being entirely replaced by the automation.

The preservation of active human participation is also a central issue in robot-assisted rehabilitation. Early rehabilitation systems capable of moving the patient along predetermined trajectories were found to potentially discourage voluntary effort. More recent *assist-as-needed* strategies instead aim to provide only the support required to complete the task, thereby promoting the patient's active contribution [10, 11]. Although these principles originate mainly from motor rehabilitation robotics, they provide a relevant conceptual basis for the development of adaptive driving-assistance systems intended for patients with different capabilities.

Assisted and semi-autonomous vehicle technologies may therefore offer a new research direction for driving rehabilitation. Preliminary clinical evidence has shown that existing driver-assistance functions can increase mobility and driving confidence for individuals with specific impairments, but such results are currently based on limited case studies and cannot yet be generalised [12]. Further investigation is required to understand how automated assistance should interact with the patient, how its intervention should be adapted to individual abilities, and how the balance between performance, safety and active participation can be assessed.

Within this context, the present thesis investigates a patient-in-the-loop longitudinal control architecture for a semi-autonomous vehicle. The work considers the patient as an active element of the control system and studies how an additional assistance contribution can support the longitudinal driving task without continuously replacing the action generated by the patient. A microscopic traffic simulation environment is adopted to enable the preliminary evaluation of this interaction under controlled and repeatable driving conditions.

Objectives of the thesis

The main objective of this thesis is to develop and evaluate a patient-in-the-loop microscopic longitudinal control architecture for semi-autonomous driving applications. The proposed approach is intended to support a patient during the execution of a longitudinal driving task while preserving their active contribution to the control of the vehicle.

The work is carried out within a broader traffic simulation framework developed collaboratively, which includes road-network representation, route computation, traffic management and the generation of the velocity reference assigned to the ego vehicle (EGO), namely the vehicle whose motion is studied and controlled. Within this overall architecture, the present thesis focuses on the vehicle-level control problem and, in particular, on the interaction between the patient, the assistance system and the longitudinal vehicle dynamics.

The patient's behaviour is represented through a parameterised discrete-time control model. Different values of the control gains, reaction delay and physical constraints are used to reproduce several patient profiles with different longitudinal driving capabilities. In addition to these predefined profiles, a questionnaire-based procedure is introduced to generate a personalised patient model from the user's subjective responses.

The action produced by the patient is combined with an additional control contribution generated by an assistance controller. Since the purpose of the system is not to replace the patient completely, the assistance is activated only when the actual velocity-tracking error exceeds a predefined threshold. The resulting architecture therefore follows an assist-as-needed principle, providing additional support only when the patient's response is not sufficient to maintain an acceptable vehicle behaviour.

The proposed control system is implemented in Python and integrated with the SUMO microscopic traffic simulator through an external simulation loop. Its behaviour is investigated under different patient profiles, reference velocity trajectories and traffic conditions, including both nominal scenarios and critical events such as the sudden braking of a preceding vehicle.

The evaluation aims to determine whether the proposed architecture can improve the longitudinal response of the ego vehicle while limiting unnecessary automated interventions. For this purpose, the analysis considers the velocity tracking performance, the acceleration and jerk evolution, the preservation of safe operating conditions and the amount of assistance required by the different patient profiles.

Thesis organisation

After this introduction, the thesis is structured as follows.

Chapter 1 presents the background and the state of the art related to driving rehabilitation, human and patient factors, shared-control architectures and longitudinal vehicle control. Particular attention is devoted to the engineering and control aspects that support the development of patient-in-the-loop assistance systems.

Chapter 2 presents the traffic-level simulation framework used as a common basis for the project. It describes the integration between Python and SUMO through TraCI, the graph-based representation of the road network, route computation and time-driven traffic-aware rerouting. The chapter also introduces the traffic and vehicle data collected during the simulation and illustrates the main outputs through a complete simulation example.

Chapter 3 presents the patient-in-the-loop microscopic longitudinal control architecture, which represents the main focus of this work. The patient model, the predefined and

questionnaire-based profiles, the assistance controller, the activation logic and the physical constraints applied to the longitudinal command are described in detail.

Chapter 4 introduces the simulation scenarios and discusses the performance of the proposed control architecture. Different patient profiles, reference velocity trajectories and traffic conditions are analysed, including nominal operating conditions and critical events involving a preceding vehicle. Moreover, the discussion particularly addresses the limitations and possible extensions, including integration with the traffic-level MPC developed by A. Galassi [13].

The Conclusions summarise the main findings and the contribution of the thesis.

1. Background and state of the art

1.1 Clinical background and driving rehabilitation

Driving is a complex activity that requires the continuous integration of visual, perceptual, cognitive and motor functions. An acquired neurological impairment may affect these abilities to different degrees, leading to difficulties in specific aspects of driving. However, the clinical diagnosis alone is generally not sufficient to determine whether an individual can drive safely, since patients with similar diagnoses may exhibit substantially different functional profiles [2].

Visual and perceptual functions allow the driver to identify road boundaries, traffic signs, obstacles and other road users. Relevant impairments may include visual-field loss, unilateral spatial neglect, reduced visual scanning and deficits in visuospatial processing. These limitations can affect lane positioning, hazard detection and the interpretation of complex traffic situations. For example, patients with right-hemisphere stroke have been found to experience greater difficulties in visual scanning, visuoconstructive abilities and divided attention than patients with left-hemisphere stroke, suggesting that the location of the lesion may influence the type of driving impairment observed [14].

Cognitive functions are particularly important when several sources of information must be processed simultaneously. Attention is required to select relevant stimuli, while processing speed influences how rapidly the driver can interpret the surrounding environment. Executive functions support planning, decision-making and adaptation to unexpected events. Attention, processing speed, visuospatial abilities and executive functions are therefore among the domains most frequently associated with driving performance after acquired brain injury [2]. However, these associations are not equally strong across studies, partly because of differences in patient populations, injury severity and assessment procedures.

Motor functions are required to translate the driver's decisions into appropriate steering, acceleration and braking actions. Reduced strength, coordination or range of motion may limit the operation of the vehicle controls. However, movement accuracy and execution time may remain relevant even when sufficient muscle strength has been recovered. This is especially important for longitudinal control, where the driver must release the accelerator,

move the foot towards the brake pedal and apply an appropriate braking action within a limited time.

Lodha et al. investigated the combined influence of cognitive and motor impairments on braking performance in chronic stroke survivors [15]. In their simulator-based experiment, the stroke group showed a longer braking time than the control group. Selective-attention performance and ankle-movement accuracy jointly contributed to the observed response. These results indicate that delayed braking cannot always be represented as a purely cognitive reaction delay, since the final response also depends on the execution of the required motor action.

Driving limitations should therefore be considered as the result of several interacting functional impairments rather than a single isolated deficit. A driver may correctly detect a hazard but execute the required action slowly or inaccurately. Conversely, adequate physical capabilities may not compensate for delayed perception, reduced attention or inappropriate decision-making.

For this reason, fitness-to-drive assessment usually combines different sources of information. An initial off-road evaluation may include visual examinations, motor tests and cognitive screening focused on attention, processing speed, visuospatial abilities, executive functions and reaction time. These tests are relatively safe and inexpensive and can help identify patients who require a more detailed evaluation.

Several screening instruments have been investigated as possible predictors of driving performance after stroke. Selected components of the Stroke Drivers' Screening Assessment and the Trail Making Test Part B have shown consistent associations with the outcome of on-road evaluations [16, 17]. Aslaksen et al. investigated the relationship between neuropsychological performance and on-road driving outcomes in patients with stroke or traumatic brain injury [18]. The significant predictors included measures of simple reaction time, visual search and processing speed, and fine motor dexterity.

However, no individual clinical test can reproduce the complexity of real driving. Clinical measures generally examine specific functions in isolation, whereas driving requires several tasks to be performed simultaneously. Therefore, when the preliminary assessment does not reveal clear contraindications, the patient may undergo an on-road evaluation. This allows the observation of speed adaptation, lane changes, junction management and interaction with other road users in a realistic environment. At the same time, on-road assessments depend on traffic conditions, route difficulty and examiner judgement, and potentially hazardous situations cannot be reproduced in a controlled and repeatable manner [2].

Driving simulators provide an intermediate environment between isolated clinical tests and on-road assessment. They allow driving tasks to be performed under controlled and repeatable conditions without exposing the patient or other road users to actual risk. The same traffic event can be presented to different patients, while variables such as

speed, acceleration, braking time, lane position and collision occurrence can be recorded automatically.

In addition to their role in assessment, driving simulators can also support rehabilitation and training. Driving rehabilitation may involve different approaches, including off-road skill-specific training, simulator-based interventions, educational programmes and vehicle adaptations [3]. Simulator-based training has produced encouraging results in selected studies. Akinwuntan et al. reported benefits following simulator-based training in stroke patients [5], while Devos et al. found that contextual simulator training produced greater improvements in selected on-road abilities than a non-contextual cognitive training programme [6].

These findings should nevertheless be interpreted cautiously. The available studies remain limited and heterogeneous in terms of patient characteristics, training protocols, assessment methods and outcome measures. As a result, the overall effectiveness of driving rehabilitation after stroke has not yet been established with certainty [7].

Despite these limitations, simulation remains valuable for both assessment and training because it enables the patient's response to specific events to be observed safely and repeatedly. From an engineering perspective, observable driving behaviour can be represented through parameters related to response delay, control accuracy and physical limitations. These quantities should not be interpreted as direct clinical measurements or diagnostic indicators, but as simplified representations of behavioural characteristics relevant to vehicle control. This control-oriented interpretation provides the connection between the clinical background and the driver models considered in the following section.

1.2 Control-oriented driver modelling and assisted driving

From a control perspective, the human driver can be regarded as an active element of the driver-vehicle system. The driver perceives information from the vehicle and the surrounding environment and uses it to generate steering, acceleration or braking commands. The resulting vehicle motion modifies the information subsequently available to the driver, thereby forming a closed-loop interaction. MacAdam discusses this representation for both lateral and longitudinal driving tasks, including path following, obstacle avoidance and headway control, while also highlighting the influence of physical and behavioural limitations on human control performance [19].

Human driving behaviour depends on several interacting perceptual, cognitive and motor processes. For control and simulation purposes, however, reproducing all of them is neither practical nor necessary. Human sensory dynamics are particularly important in this context.

Nash et al. review the role of sensory dynamics, perception thresholds and time delays in driver models for steering and speed control, showing that the driver response cannot generally be regarded as instantaneous [20]. Simplified control-oriented representations can therefore capture selected aspects of human behaviour without attempting to reproduce the complete underlying physiological process.

Longitudinal driver behaviour can be modelled according to different objectives. In free-driving conditions, the driver may primarily regulate the vehicle velocity, whereas in car-following situations the response also depends on the motion of a preceding vehicle. Microscopic traffic models provide a well-established example of this second approach. The Intelligent Driver Model introduced by Treiber et al. describes longitudinal vehicle behaviour through a continuous microscopic car-following model and reproduces different traffic conditions through a limited set of behavioural parameters [21]. These models show that longitudinal driving behaviour can be reproduced with relatively simple mathematical relationships while still accounting for interactions with surrounding traffic.

The problem changes when an automated system also contributes to vehicle control. Driver-assistance systems may introduce an automated action while preserving a certain level of human involvement. This leads to shared control, where the driver and the automated system both contribute to the vehicle response.

Griffiths and Gillespie investigated this principle through a haptic interface in a driving simulator, where an automated system assisted a lane-following task by applying torque to a motorised steering wheel. The driver remained able to perceive the automated action and could either follow or override it [22]. Although the considered task was lateral rather than longitudinal control, the study provides a clear example of simultaneous human and automated contributions to vehicle control.

A related concept is cooperative vehicle control, in which the interaction between the driver and the automation is explicitly considered in the design of the assistance system. Flemisch et al. describe cooperative control as a close coupling between the driver and the automated system and present active interfaces as a means of maintaining the driver within the control process [23]. In this view, assistance does not necessarily imply a complete transfer of the driving task to the automation.

An important aspect of this interaction is the distribution of control authority. The contribution of the automated system can range from limited guidance to a dominant control action. Abbink et al. discuss haptic shared control as an approach that allows a continuous interaction between human and automation and permits the level of control authority to vary according to the situation [24]. Excessively weak assistance may have little effect on the vehicle response, whereas excessive authority may reduce the effective contribution of the human operator.

This balance becomes especially important in rehabilitation, where the patient should

remain actively involved in the task. In this context, the objective is not only to obtain an accurate system response but also to preserve the active participation of the patient. Similar considerations have been extensively investigated in robot-assisted rehabilitation through the assist-as-needed principle.

Emken et al. formulated robotic movement training as an optimisation problem in which the controller limits movement errors while systematically reducing the amount of assistance provided to the user [25]. The underlying principle is that assistance should be sufficient to support successful task execution without unnecessarily replacing the voluntary contribution of the patient.

This idea was further investigated by Wolbrecht et al. in an adaptive controller for post-stroke upper-limb rehabilitation. Their approach was designed to provide only the minimum assistance necessary to complete the desired movement. The experiments also highlighted the phenomenon of "slacking", whereby users may reduce their voluntary contribution when the robot provides excessive support. Reducing assistance when task errors are small was shown to increase participation of the patient's motor system [26].

The same principle is relevant to rehabilitation-oriented driving assistance. Although robotic movement rehabilitation and assisted driving involve different physical tasks, both require a compromise between task performance and active human participation. A rehabilitation-oriented driving system may therefore benefit from an architecture in which automation supports the user when necessary without systematically replacing the user's control action.

Longitudinal control must also account for vehicle dynamics, comfort and safety. Automated longitudinal-control systems are often organised hierarchically. Rajamani et al., for example, developed a longitudinal controller for automated vehicle platoons in which an upper-level controller determines the desired acceleration while a lower-level controller generates the throttle and braking actions required to track it [27]. This separation between a desired longitudinal response and its implementation is common in automated vehicle-control architectures.

Good tracking performance alone is not sufficient. Acceleration and its rate of change influence the feasibility and comfort of vehicle manoeuvres. In an adaptive cruise control framework, Guo et al. explicitly considered inter-vehicle distance, relative velocity, acceleration and jerk within an MPC formulation designed to address car-following, safety and comfort requirements [28]. Constraints on longitudinal acceleration and jerk are therefore relevant when the commanded response must remain both physically feasible and acceptable to the vehicle occupants.

Safety becomes particularly important when the controlled vehicle interacts with a preceding vehicle. Car-following models have traditionally represented this interaction by relating the response of the following vehicle to the behaviour of the leader. The behavioural model

proposed by Gipps assumes that drivers impose limits on their desired acceleration and braking rates and uses these characteristics to determine the response of the following vehicle [29]. This type of formulation highlights the importance of considering both vehicle motion and braking capability when evaluating longitudinal safety.

Taken together, these research areas provide the main control concepts relevant to rehabilitation-oriented assisted driving. Driver modelling makes it possible to represent human behaviour within the vehicle control loop, shared control provides a framework for combining human and automated actions, and assist-as-needed strategies introduce the additional objective of preserving active user participation. Longitudinal-control and car-following methods add the performance, comfort and safety requirements associated with interaction between vehicles. However, these concepts have largely been developed for different purposes and are not generally integrated within a single patient-oriented microscopic traffic-control framework. This aspect motivates the research gap discussed in the following section.

1.3 Research gap and positioning of the thesis

The literature on post-injury driving primarily focuses on the assessment of fitness to drive, the identification of functional impairments and the effectiveness of rehabilitation programmes. Clinical tests, on-road evaluations and driving simulators provide complementary information about the patient's abilities. However, the resulting impairments are not always translated into explicit control parameters that can be used to analyse the closed-loop interaction between the patient and the vehicle.

By contrast, the control and automotive literature includes numerous driver models, shared-control architectures and longitudinal-assistance systems. These approaches are generally designed for representative or generic drivers and are mainly evaluated in terms of tracking performance, comfort, traffic efficiency or collision avoidance. Patient-specific limitations and rehabilitation objectives are less frequently considered within the same framework.

A further limitation is that driving rehabilitation, human-driver modelling, assist-as-needed control and microscopic traffic simulation are often examined separately. Simulator-based clinical studies may observe the patient's behaviour without introducing an assistive control contribution, while automotive assistance systems may improve performance and safety without explicitly considering rehabilitation objectives or preserving the driver's active participation.

The present thesis addresses this gap by developing a patient-in-the-loop longitudinal-control framework within a microscopic traffic simulation environment. Different patient behaviours are represented through control-oriented parameters and combined with an

assistive contribution aimed at supporting the driving task without completely replacing the patient's action. This makes it possible to compare different patient profiles and assistance conditions under controlled and repeatable traffic scenarios.

The following chapter presents the traffic-level simulation framework used to generate the road network, vehicle routes, traffic events and reference trajectories required by the microscopic longitudinal-control layer.

2. Traffic-level simulation framework

2.1 Introduction

The preceding introduction and background chapter introduced the clinical and control context of the work. Before focusing on the patient-in-the-loop longitudinal control problem, it is useful to describe the traffic simulation framework used as a common basis for the project.

The framework is developed using the Simulation of Urban MObility (SUMO), an open-source microscopic traffic simulator that reproduces the motion of individual vehicles and their interactions with the road network [30].

Within the overall project, two functional levels are distinguished. The traffic-level layer considers the road network, traffic conditions, route selection and higher-level speed-reference planning. The vehicle-level layer focuses on the longitudinal motion of the individual EGO vehicle and on the interaction between the patient model and the assistance controller. These layers are referred to as macroscopic and microscopic, respectively, according to the scope of the planning and control tasks. SUMO provides a microscopic simulation environment in both cases: the distinction between the two layers concerns the level at which decisions are made, while individual vehicles are simulated throughout the framework.

The simulation can be accessed and modified during execution through the Traffic Control Interface (TraCI). In this work, TraCI is used from Python to start and advance the simulation, introduce vehicles, retrieve their state and collect the quantities required for subsequent traffic analysis.

The SUMO road network is also represented in Python through the NetworkX library. This provides a graph-based description of the road infrastructure and makes it possible to analyse the connectivity of the network and compute routes between different locations. The combination of SUMO, TraCI and NetworkX therefore provides a flexible environment in which traffic simulation, route computation, data acquisition and post-processing can be handled within the same Python framework.

The purpose of this chapter is not to develop an advanced macroscopic traffic-control

strategy. Instead, it presents the fundamental components required to build and analyse a traffic simulation and illustrates the main outputs that can be obtained from it. Particular attention is devoted to the interaction between SUMO and Python, the graph representation of the road network, vehicle routing and the extraction of traffic-related quantities such as speed, flow and density.

The framework described here represents the common simulation basis developed during the project. More advanced macroscopic traffic-management and control aspects were subsequently investigated in the parallel work carried out by A. Galassi [13]. For this reason, the discussion in this chapter is intentionally limited to the elements required to understand the simulation environment and its main analysis capabilities. The following chapter will then move from this traffic-level description to the main focus of the present thesis, namely the microscopic longitudinal control of the individual EGO vehicle and its interaction with the patient model.

2.2 Python–SUMO simulation framework

After introducing the general role of SUMO in the previous section, the present section describes how the simulator is integrated with Python in the developed framework. The objective is to illustrate the basic structure required to configure a SUMO simulation, start it externally, exchange information during its execution and collect the resulting data.

The integration is based on the Traffic Control Interface (TraCI), which provides online access to a running SUMO simulation [31]. Through the Python TraCI library, the external program can retrieve information from the simulator and, at the same time, modify the state or behaviour of simulated objects. The resulting architecture can therefore be interpreted as an interaction between two main components: SUMO is responsible for the evolution of the traffic simulation, while Python manages the higher-level logic, data acquisition and post-processing.

2.2.1 Software organisation

The implemented framework is divided into different Python modules in order to separate simulation configuration, execution, routing, network analysis and post-processing. The main entry point is `main_traci3.py`, which coordinates the different phases of the program.

The role of the main Python files is summarised in Table 2.1.

Python file	Role in the framework
`main_traci3.py`	Main entry point of the program. It coordinates the SUMO setup, execution of the TraCI simulation, data processing, graphical output generation and network analysis.
`config.py`	Defines the main configuration parameters, file paths and settings required by the simulation.
`sumo_setup.py`	Prepares the SUMO working environment and generates the configuration files required before starting the simulation. It also manages the selection of the SUMO executable.
`simulation.py`	Contains the main TraCI simulation loop. It starts SUMO, advances the simulation, introduces vehicles, retrieves simulation variables and sends commands back to the simulator when required.
`routing.py`	Contains the functions used to identify suitable road segments and compute valid routes through the network.
`graph_analysis.py`	Converts the SUMO road network into a NetworkX graph and performs graph-based network analyses.
`plots.py`	Processes the data collected during the simulation and generates the graphical outputs used for traffic analysis.
`state.py`	Stores variables that have to be preserved and updated during the execution of the simulation.
`utils.py`	Contains auxiliary functions shared by the different modules.

Table 2.1: Role of the main Python files in the SUMO–TraCI simulation framework.

This modular organisation is not strictly required to interact with SUMO, since a complete TraCI simulation could also be implemented within a single Python script. However, separating the different operations makes the program easier to interpret and allows simulation, routing, network analysis and post-processing functions to be modified independently.

The overall execution sequence implemented in `main_traci3.py` can be summarised as

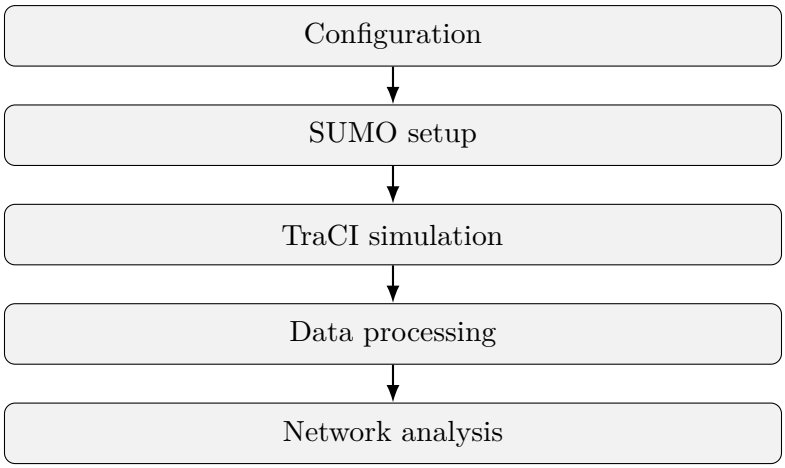


Figure 2.1: Execution sequence of the Python–SUMO framework.

Therefore, before interacting with the traffic simulation itself, Python first prepares the environment and generates the configuration files required by SUMO.

2.2.2 SUMO files and simulation configuration

A SUMO simulation requires a road network together with a configuration defining the files and settings that have to be loaded. The road network is stored in a `.net.xml` file, which contains the topology and geometry of the road infrastructure, including junctions, edges, lanes and their connections.

The central simulation configuration is instead contained in a `.sumocfg` file. In the developed framework, this file is generated automatically from Python before the simulation starts. It specifies the road network to be loaded, the additional files used during the simulation and the main temporal and output settings.

Additional elements can be introduced through `.add.xml` files. In the implemented framework, an additional file is generated to request aggregated traffic measurements at edge level. In this way, the desired outputs can be configured without modifying the original road-network description.

A conventional SUMO simulation may also use a `.rou.xml` file to define the traffic demand and the routes followed by the vehicles before the simulation starts. In the present framework, however, traffic demand can also be generated dynamically through TraCI. Routes and vehicles can be introduced after establishing the connection with SUMO using commands such as `traci.route.add()` and `traci.vehicle.add()`.

This represents one of the advantages of integrating SUMO with Python: the complete traffic demand does not necessarily have to be defined before execution, but can instead be generated or modified according to the logic implemented in the external program.

The framework also produces different output files. Some quantities are generated directly by SUMO, such as trip information and aggregated edge measurements, while other vehicle-level variables are collected directly through TraCI and stored by Python. The specific traffic quantities extracted from these files and their graphical representation are discussed later in this chapter.

2.2.3 Interaction between Python and SUMO through TraCI

The central element of the framework is the communication between the Python program and a running SUMO instance. TraCI allows an external client to access and modify the simulation while it is being executed [31].

Python first selects the SUMO executable. Depending on the configuration, either `sumo` or `sumo-gui` can be used. The graphical version is particularly useful during development because it allows the traffic evolution to be inspected visually, while the command-line version can be used when graphical visualisation is not required.

The selected executable and the previously generated `.sumocfg` configuration are then passed to

`traci.start()`.

This instruction launches SUMO and establishes the connection between the simulator and the Python program.

Once the connection has been established, the simulation is advanced through repeated calls to

`traci.simulationStep()`.

Each call advances SUMO by one simulation step and then returns control to Python. At this point, the external program can retrieve the new state of the traffic system, process the available information and, if necessary, send new commands before advancing the simulation again.

The interaction between Python and SUMO can be summarised through the sequence shown below:

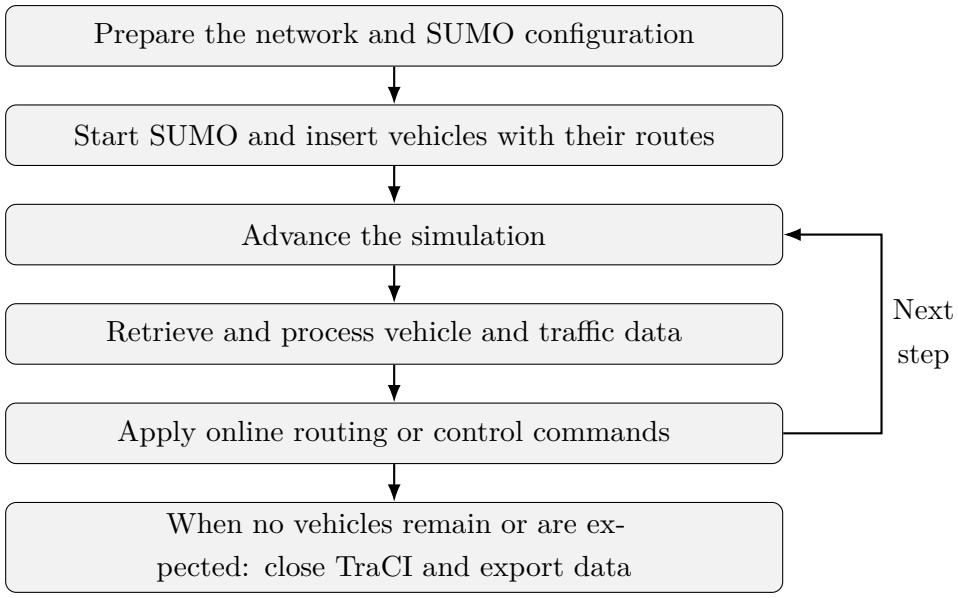


Figure 2.2: TraCI execution flow, including the repeated exchange of measurements and commands.

The execution starts with the preparation of the SUMO configuration and the selection of the network to be loaded. Python then launches SUMO through `traci.start()`, establishing the communication between the external program and the simulator.

Once the connection has been created, the framework defines the vehicle types and introduces both the background traffic and the EGO vehicle. Routes can be assigned dynamically through TraCI, which makes it possible to generate the traffic demand directly from Python rather than relying exclusively on a predefined route file.

The core of the framework is the simulation loop. At each iteration, the instruction `traci.simulationStep()` advances SUMO by one step. After the update, Python retrieves the current simulation time and the list of active vehicles. For each vehicle, quantities such as position, speed, lane, longitudinal position, slope and vehicle type can be collected.

The retrieved information is then processed outside SUMO. In the implemented framework, vehicle speeds are also grouped according to the corresponding road edge in order to obtain an estimate of the current mean speed on the occupied road segments.

The same loop can also be used to apply online modifications to the simulation. In the present implementation, examples include the possibility of updating the route of the EGO vehicle according to the current traffic conditions and applying a speed-control action. These operations demonstrate the bidirectional nature of the TraCI interface: Python can both observe the simulation and send new commands back to SUMO.

The loop continues until no additional vehicles are expected to remain in or enter the network. At that point, the TraCI connection is closed and the collected data are exported

for subsequent analysis.

TraCI organises the available commands according to different simulation domains. For example, the `traci.vehicle` domain provides access to vehicle-related quantities such as speed, position, lane and route, while `traci.edge` can be used to retrieve information associated with road segments. Global simulation variables, including the current simulation time, are available through the `traci.simulation` domain.

Getter functions retrieve vehicle, edge and simulation variables; setter functions apply the routing and control decisions within the loop described above.

From a practical point of view, executing the main Python script is therefore sufficient to initialise the environment, generate the required SUMO configuration, launch the simulator, establish the TraCI connection and execute the complete simulation loop.

The particular road network, traffic demand and numerical parameters adopted in this work represent a specific application of this general procedure and are therefore described separately in the following section.

2.3 Graph-based network representation and route computation

2.3.1 Road network representation with NetworkX

In addition to the representation used internally by SUMO, the road network is also converted into a graph structure in Python through the NetworkX library [32]. This representation is useful for network analysis and for the route-computation procedures introduced in the following subsection.

The conversion is performed starting directly from the same SUMO network file used by the simulator. The network is first loaded through `sumolib`, and a NetworkX `MultiDiGraph` is then created. This graph structure is directed and allows multiple edges to connect the same pair of nodes, which is suitable for representing a road network containing different directed road segments between the same junctions.

The graph can be described as

$$G = (V, E), \tag{2.1}$$

where V represents the set of network nodes and E the set of directed road edges. For each SUMO node, a corresponding NetworkX node is created using the same identifier. The two-dimensional coordinates (x_i, y_i) and the SUMO node type are then stored as

additional attributes of the NetworkX node.

For each SUMO edge, a corresponding directed NetworkX edge is added between the original start and end nodes. In the implemented code, the SUMO edge identifier is used as the edge key and is also stored as an attribute. Additional information associated with the road segment is preserved in the graph.

The main quantities transferred from SUMO to NetworkX are summarised in Table 2.2.

Element	**Information**	**Representation in NetworkX**
Node	Identifier	Used directly as the NetworkX node key.
Node	Position	Stored in the `pos` attribute as the coordinates (x, y).
Node	Type	Stored in the `type` attribute.
Edge	Identifier	Used as the NetworkX edge key and stored in the `id` attribute.
Edge	Length	Stored in the `length` attribute.
Edge	Geometry	Stored in the `shape` attribute as the sequence of coordinates describing the road segment.
Edge	Allowed speed	Stored in the `allowed_speed` attribute.

Table 2.2: Information transferred from the SUMO network to the NetworkX graph.

The geometrical information contained in the edge shape is also used for network visualisation. In this way, the graphical representation does not rely only on straight connections between nodes, but can preserve the geometry of the original road segments defined in SUMO.

After the graph has been generated, the implementation performs a consistency check between the SUMO and NetworkX representations. First, the total number of nodes and edges is compared. A second verification is then performed using the original identifiers of the network elements.

If V_{SUMO} and V_{NX} denote the sets of node identifiers contained in the SUMO network and in the NetworkX graph, respectively, the node conversion is considered complete when

$$V_{\text{SUMO}} = V_{\text{NX}}. \tag{2.2}$$

Similarly, for the edge identifiers,

$$E_{\text{SUMO}} = E_{\text{NX}}. \tag{2.3}$$

The results of these checks are exported by the program, allowing missing or additional graph elements to be identified if the conversion is not consistent with the original SUMO network.

The complete graph used for this verification also includes the internal SUMO edges associated with junction movements. However, not every edge in this complete representation is subsequently considered for vehicle routing. Before the route-search procedure is performed, the framework selects a subset of candidate road segments. Internal edges are excluded, together with edges that do not allow passenger vehicles and road segments shorter than a predefined minimum length.

The resulting subset therefore contains the road segments considered suitable for the subsequent route computation, while the complete NetworkX graph remains available for network-level analysis and graphical representation.

Starting from this graph representation and from the selected candidate edges, the framework can determine a route between an initial and a destination road segment. The route-search procedure implemented in this work is based on the A* algorithm and is described in the following subsection.

2.3.2 Route computation and time-driven traffic-aware rerouting

After the selection of the road segments suitable for passenger vehicles, the framework provides the possibility to compute a route between an initial and a destination edge. The initial route-search procedure is implemented in `routing.py` through a custom A* algorithm operating directly on SUMO edges.

Before the search is performed, the network is filtered in order to exclude internal SUMO edges, road segments that do not allow passenger vehicles and edges shorter than 20 m. An adjacency structure is then constructed by associating each candidate edge with the candidate outgoing edges that can be reached from it. The search is therefore performed at edge level rather than directly on the network nodes.

For a generic edge e, the A* algorithm evaluates the priority function

$$f(e) = g(e) + h(e), \tag{2.4}$$

where $g(e)$ represents the accumulated cost from the initial edge to the current one, while $h(e)$ estimates the remaining cost to the destination.

In the basic formulation, the cost assigned to an edge corresponds to its length,

$$c(e) = L_e, \tag{2.5}$$

and the accumulated cost is consequently updated as

$$g(e_j) = g(e_i) + c(e_j), \tag{2.6}$$

when edge e_j is reached from the current edge e_i.

The heuristic term is based on the Euclidean distance between a representative point of the current edge and the corresponding point of the destination edge. The reference point is obtained from the geometrical shape stored in the SUMO network. If the edge geometry consists of N_e points, its representative coordinates are computed as

$$p_e = \left(\frac{1}{N_e} \sum_{j=1}^{N_e} x_j, \frac{1}{N_e} \sum_{j=1}^{N_e} y_j \right). \tag{2.7}$$

If no shape information is available, the midpoint between the origin and destination nodes of the edge is used instead.

The heuristic can therefore be expressed as

$$h(e) = \sqrt{(x_e - x_g)^2 + (y_e - y_g)^2}, \tag{2.8}$$

where (x_e, y_e) and (x_g, y_g) represent the reference points of the current and destination edges, respectively.

Candidate edges are stored in a priority queue according to their value of $f(e)$. At each iteration, the edge with the lowest priority is extracted and its admissible outgoing edges are evaluated. Whenever a lower accumulated cost is found for one of these edges, its cost and predecessor are updated. Once the destination edge is reached, the route is reconstructed by following the stored predecessor relationships backwards and reversing the resulting sequence. The output is an ordered list of SUMO edge identifiers that can subsequently be assigned to a vehicle.

Time-driven traffic-aware rerouting During the execution of the simulation, the speeds of all active vehicles are retrieved through TraCI at each simulation step. For every non-internal edge occupied by one or more vehicles, the framework computes the arithmetic mean of the measured vehicle speeds,

$$\bar{v}_e(k) = \frac{1}{N_e(k)} \sum_{i=1}^{N_e(k)} v_i(k), \tag{2.9}$$

where $N_e(k)$ is the number of vehicles observed on edge e at simulation step k.

These measurements are used to adapt the route of the EGO vehicle according to the current traffic conditions. In the implementation presented here, the rerouting logic is *time-driven*: the possibility of recomputing the route is evaluated at fixed intervals of

$$T_{\text{reroute}} = 5\,\text{s}. \tag{2.10}$$

With the simulation step used in the example, this corresponds to one rerouting evaluation every 10 simulation steps.

At each rerouting instant, road segments characterised by a mean speed below the predefined threshold are penalised by modifying the travel time used for route computation.

For an uncongested edge, the reference free-flow travel time is estimated as

$$T_e^{\text{free}} = \frac{L_e}{v_e^{\max}}, \tag{2.11}$$

where L_e is the edge length and $v_e^{\max}$ is its maximum allowed speed.

When the measured mean speed $\bar{v}_e$ falls below the low-speed threshold, the routing travel time is instead increased according to

$$T_e^{\text{pen}} = P\frac{L_e}{\max(\bar{v}_e, v_{\min})}, \tag{2.12}$$

where P is a penalty factor and $v_{\min}$ is a small positive lower bound introduced to avoid numerical problems when the measured speed approaches zero. In the implemented configuration,

$$P = 4, \qquad v_{\min} = 0.1\,\text{m/s}. \tag{2.13}$$

The penalised value is passed to SUMO through `traci.edge.adaptTraveltime()`. As a consequence, low-speed road segments become less attractive during the subsequent route computation. When an edge is no longer classified as slow, its travel time is restored to the corresponding free-flow estimate.

After the edge travel times have been updated, the route from the current EGO edge to the predefined destination is recomputed using `traci.simulation.findRoute()`.

If the newly computed sequence differs from the remaining route currently assigned to the EGO vehicle, the route is updated through `traci.vehicle.setRoute()`.

The complete online procedure is summarised in Figure 2.3.

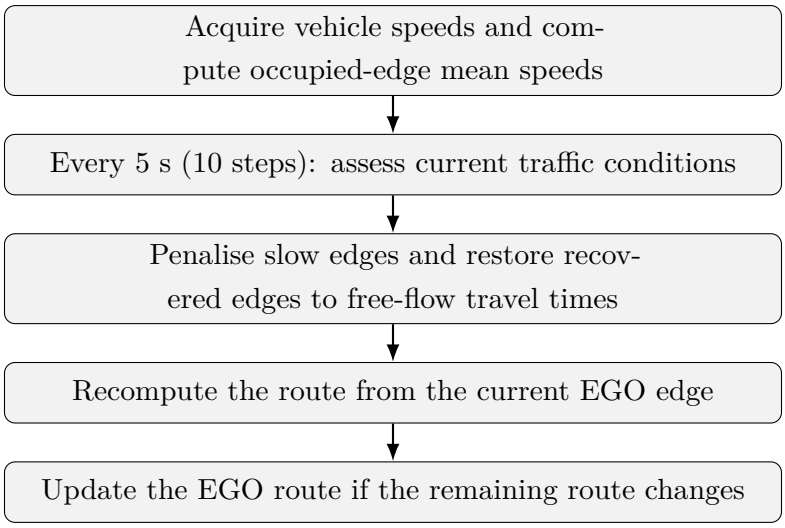


Figure 2.3: Time-driven traffic-aware rerouting procedure.

The distinction between the two routing operations is therefore important. The custom A* implementation provides the graph-based route-search capability and can itself incorporate traffic-dependent edge costs. During the complete online simulation considered here, however, periodic traffic-aware rerouting is performed by adapting the travel times perceived by SUMO and subsequently requesting a new route through TraCI.

The implementation presented in this thesis adopts a time-driven approach, since the routing condition is evaluated periodically every 5 s, regardless of whether a significant traffic event has occurred since the previous evaluation. In subsequent versions of the common framework, an event-driven strategy was introduced so that route recalculation could be triggered only when relevant changes in traffic conditions occurred. The comparison between time-driven and event-driven approaches, with particular attention to their computational requirements, as well as the further development of the traffic-aware routing layer, is investigated in detail in the parallel work of A. Galassi [13].

2.4 Traffic data and simulation outputs

The combination of SUMO, TraCI and Python makes it possible to analyse the simulation at both vehicle and traffic-aggregate level. Vehicle-level information is collected through TraCI and stored as Floating Car Data (FCD), including time, vehicle position, speed, lane and vehicle type. In parallel, SUMO is configured to provide aggregated measurements at edge level over predefined time intervals.

The main traffic quantities considered in this work are speed, density and flow, which represent the fundamental variables commonly used for the macroscopic description of road traffic [33]. In particular, the relation between flow q, density k and mean speed $\bar{v}$ can be expressed as

$$q = k\bar{v}. \tag{2.14}$$

When density is expressed in veh/km and speed in km/h, the resulting flow is expressed in veh/h. In the implemented post-processing procedure, SUMO provides the edge density and mean speed, while the corresponding flow is subsequently computed in Python.

Following the standard traffic representations discussed in [33], the collected data are used to generate several graphical outputs. These include time series of flow, density and mean speed, a space–time diagram representing individual vehicle trajectories along a selected lane, and the flow–density and speed–density diagrams. The framework also provides vehicle-specific outputs, such as the speed and acceleration history of the EGO vehicle, together with graphical representations of the road network and the travelled route.

These outputs provide different views of the same simulation. Vehicle trajectories describe the microscopic evolution of individual vehicles, whereas edge-based quantities allow the corresponding traffic conditions to be observed from an aggregated perspective. The following section illustrates these capabilities through a specific simulation example.

2.5 Example of a complete traffic simulation

The procedure described in the previous sections is applied to a complete traffic simulation in order to illustrate the interaction between SUMO, TraCI and the Python analysis framework. The objective of this example is not to investigate a specific traffic-control strategy, but to show how a road network can be simulated, how individual and aggregated traffic information can be collected, and how the resulting data can be represented graphically.

2.5.1 Simulation setup

The simulation uses the road network contained in `nave_cars.net.xml`, obtained from OpenStreetMap data and converted into the SUMO network format. Figure 2.4 shows the resulting network as visualised in SUMO NetEdit. The network includes roads with different characteristics and speed limits and provides a sufficiently extended environment to generate background traffic and analyse the behaviour of an individual EGO vehicle.



Figure 2.4: Road network used for the example simulation, visualised in SUMO NetEdit.

The same `.net.xml` file is used by the different components of the framework. SUMO uses it as the road-network description during the traffic simulation, while Python loads it through `sumolib` for routing and network-processing operations. The network is subsequently converted into the NetworkX representation described in Section 2.3.1, allowing the consistency between the original SUMO topology and the graph representation to be verified.

The main simulation parameters are summarised in Table 2.3.

Parameter	**Value**
SUMO network	`nave_cars.net.xml`
Simulation step T_s	0.5 s
Background traffic generation interval	$0 - 360$ s
Background vehicle departure period	1 s
Background traffic random seed	123
Edge-data aggregation interval	10 s
EGO departure time	350 s
EGO initial speed	80 km/h
Regular-road target speed	50 km/h
Fast-road target speed	100 km/h

Table 2.3: Main parameters of the example traffic simulation.

Background traffic is generated directly from Python. One vehicle is scheduled every second during the traffic-generation interval, resulting in 360 background vehicles. A fixed

random seed is used in order to make the generated traffic demand reproducible. For each vehicle, the framework selects suitable origin and destination edges and verifies the existence of a valid route before inserting the vehicle into the simulation through TraCI.

The EGO vehicle is introduced later in the simulation, at $t = 350\,\mathrm{s}$, so that it enters a road network in which background traffic is already present. Its initial route is computed using the A* procedure described in Section 2.3.2. In the considered execution, the initial route consists of 28 road edges.

The complete SUMO network is also converted into the corresponding NetworkX graph. The consistency check described previously gives

$$|V_{\mathrm{SUMO}}| = |V_{\mathrm{NX}}| = 979, \tag{2.15}$$

and

$$|E_{\mathrm{SUMO}}| = |E_{\mathrm{NX}}| = 1825. \tag{2.16}$$

The identifier-based comparison also confirms the correspondence between the two representations. Figure 2.5 shows the resulting graph representation together with the route associated with the EGO vehicle.

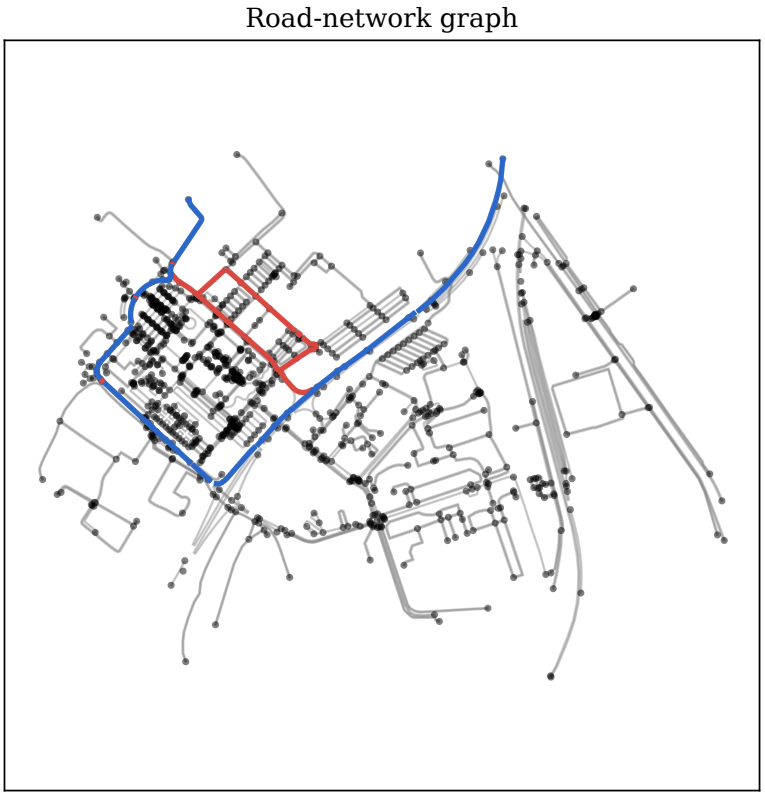


Figure 2.5: NetworkX representation of the SUMO road network and EGO route in the example simulation.

Figure 2.5 shows the NetworkX representation obtained from the SUMO road network. The NetworkX graph preserves the nodes and edges of the original SUMO network, retaining their identifiers. This correspondence allows network elements to be consistently identified across the two representations.

The geometrical shape associated with each SUMO edge is retained in the NetworkX representation and is used to reconstruct the road layout shown in the figure. Black markers represent network nodes, while the road edges not involved in the EGO trajectory are shown in grey. Internal SUMO edges are included in the consistency check but omitted from the graphical representation for clarity.

The figure also provides an overview of the evolution of the EGO route during the simulation. Blue road segments correspond to edges that were effectively traversed by the EGO vehicle, as reconstructed from the collected FCD data. Red segments correspond instead to edges that belonged to at least one route assigned to the EGO during the simulation but were never actually traversed. These segments may therefore originate from routes that were subsequently modified by the time-driven traffic-aware rerouting procedure.

As a result, the figure provides both a validation of the one-to-one SUMO–NetworkX network conversion and a compact representation of the difference between the routes planned during the simulation and the path effectively travelled by the EGO vehicle.

During the simulation, vehicle-level information is recorded every simulation step in the FCD dataset. The complete execution considered here contains 361 different vehicles and more than 9.3×10^4 vehicle-state observations. In parallel, aggregated edge measurements are collected every 10 s. These two sources are subsequently processed to obtain the graphical results presented in the following subsection.

2.5.2 Background traffic simulation results

The first set of results concerns the background traffic generated within the road network. The objective is to illustrate the main traffic quantities that can be extracted from the SUMO simulation and subsequently analysed through Python.

The analysis focuses on the main macroscopic traffic variables introduced in Section 2.4, namely density, mean speed and flow, together with their fundamental relationships [33]. The results are evaluated on lane `26599252_0`, which contains the largest number of vehicle observations in the collected FCD dataset. Aggregate measurements are computed over intervals of 10 s.

Traffic density. Figure 2.6 shows the evolution of traffic density on the selected road segment. During the first part of the simulation, density remains relatively low, while

a significant increase can be observed after approximately 3.5 minutes. The maximum values exceed 130 veh/km, indicating a strong accumulation of vehicles on the considered edge. Density subsequently decreases as the queue progressively dissipates.

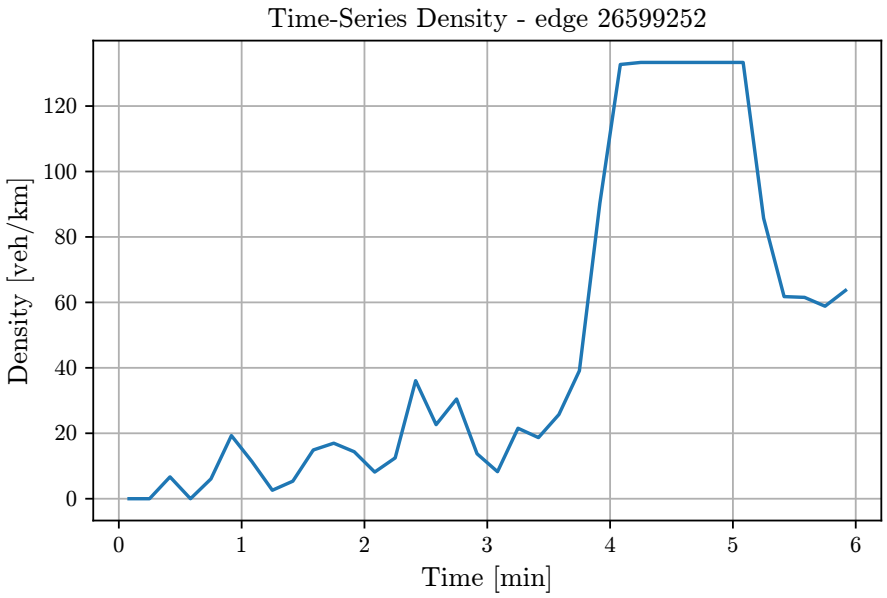


Figure 2.6: Evolution of traffic density on the selected road segment.

Mean traffic speed. The corresponding evolution of mean traffic speed is shown in Figure 2.7. Under low-density conditions, vehicles travel at approximately 40–50 km/h. As the number of vehicles on the edge increases, the mean speed progressively decreases and eventually approaches zero during the most congested phase. When the congestion dissipates, vehicle speed increases again.

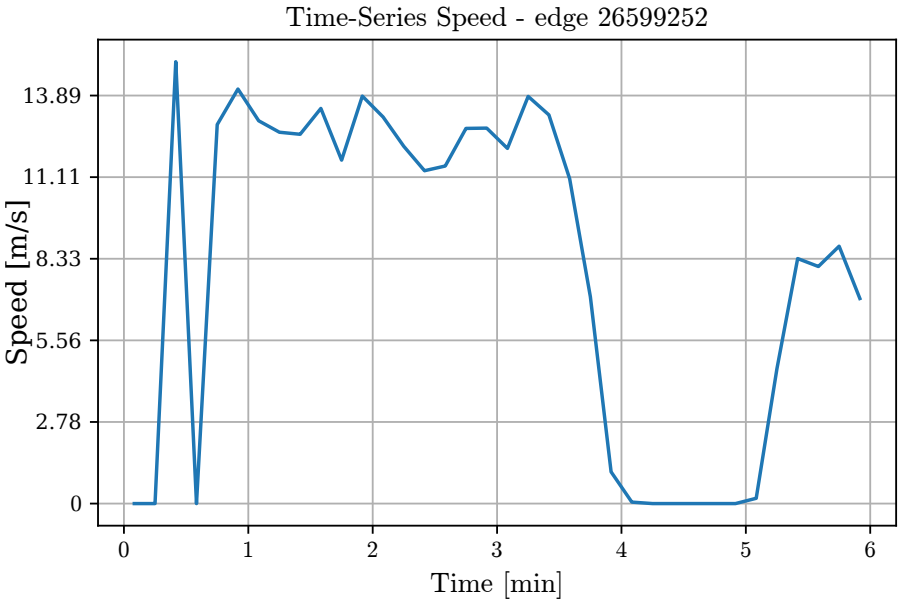


Figure 2.7: Evolution of mean traffic speed on the selected road segment.

Traffic flow. Figure 2.8 shows the traffic flow obtained from the measured density and mean speed. At the beginning of the simulation, flow generally increases as more vehicles reach the considered road segment. During the congested phase, however, the strong reduction in vehicle speed causes the flow to decrease despite the high traffic density. The flow subsequently recovers as the congestion is progressively reduced.

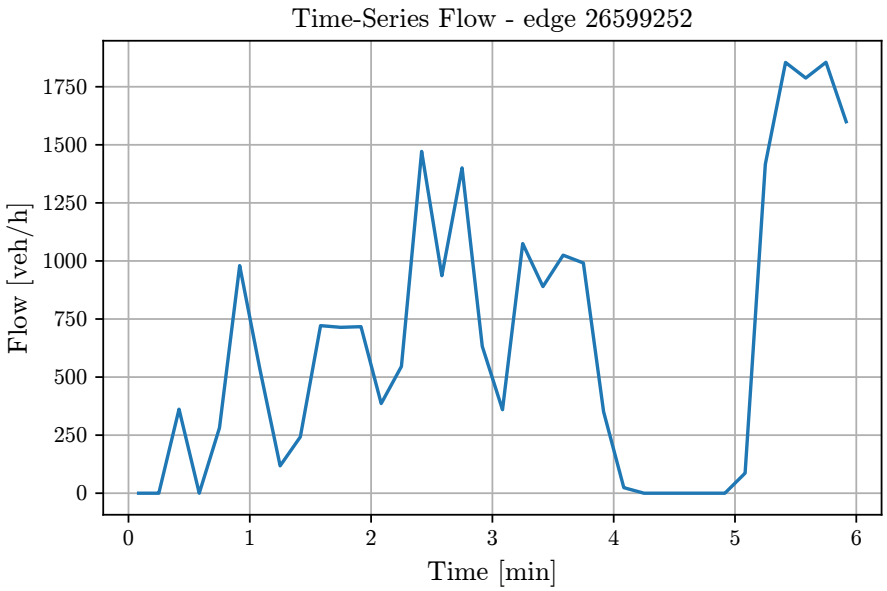


Figure 2.8: Evolution of traffic flow on the selected road segment.

Flow–density relationship. The relation between traffic flow and density is shown in Figure 2.9. As expected from the fundamental traffic relationships discussed in [33], flow initially increases with density. At higher density values, the reduction in vehicle speed becomes dominant and the observed flow decreases, corresponding to congested traffic conditions.

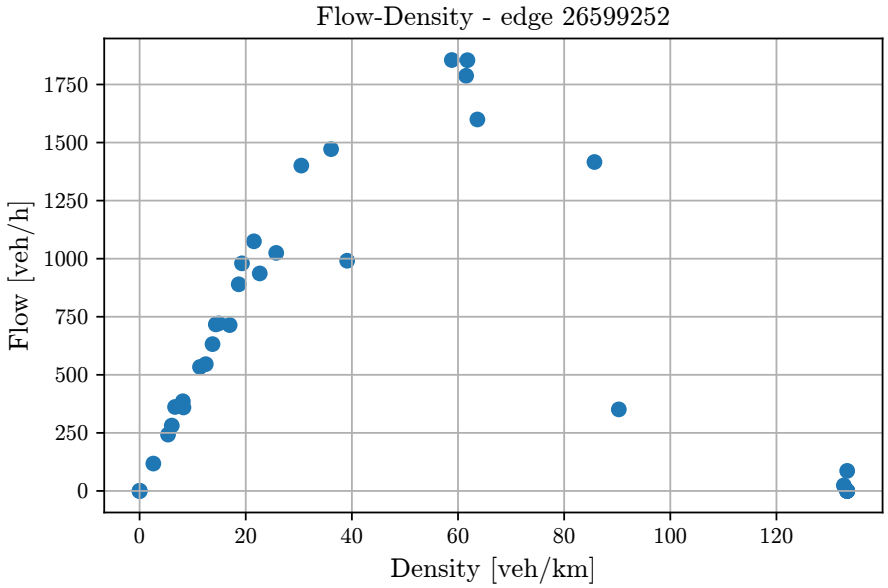


Figure 2.9: Flow–density relationship obtained from the simulated background traffic.

Speed–density relationship. Figure 2.10 shows the corresponding relationship between mean speed and traffic density. Low-density conditions are associated with higher vehicle speeds, whereas increasing density produces a progressive reduction in mean speed. The highest-density observations correspond to the strongly congested phase identified in the previous time-series results.

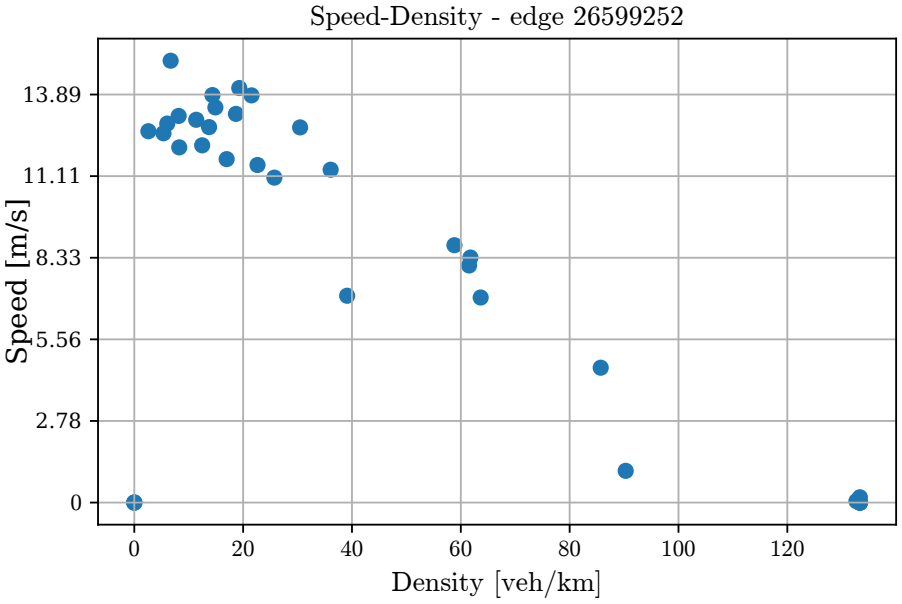


Figure 2.10: Speed–density relationship obtained from the simulated background traffic.

Space–time vehicle trajectories. A complementary representation of the traffic evolution is provided by the space–time diagram in Figure 2.11. Each trajectory represents

the longitudinal position of an individual vehicle along the selected lane as a function of time. During free-moving traffic, vehicles cross the road segment rapidly. In the congested phase, several trajectories become nearly horizontal, indicating vehicles remaining at approximately the same position and therefore the formation of a queue.

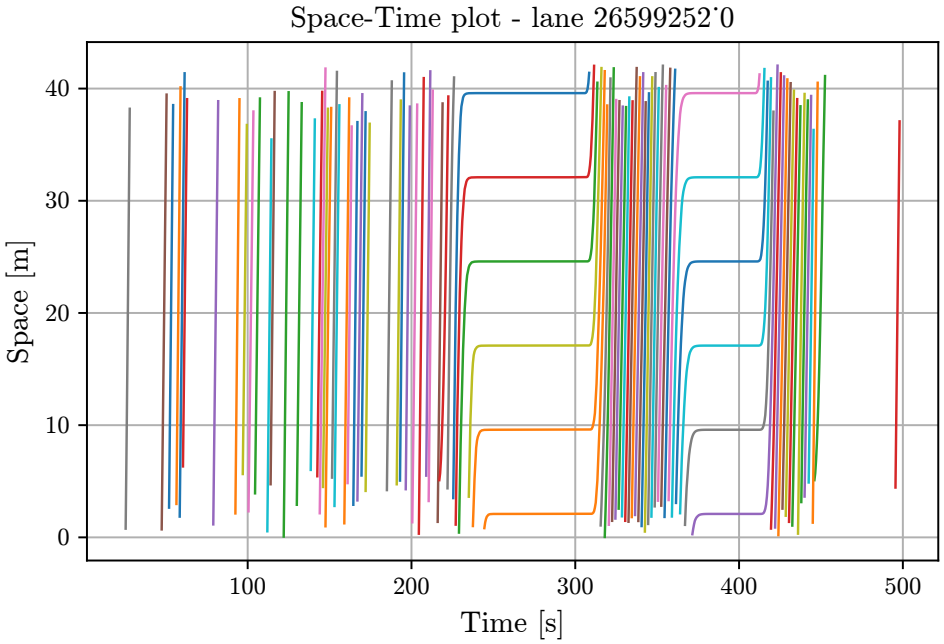
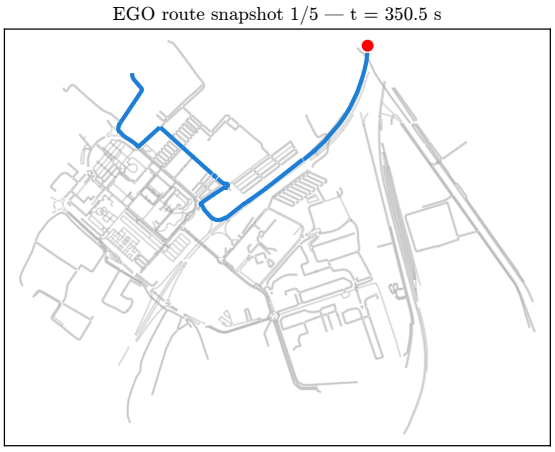


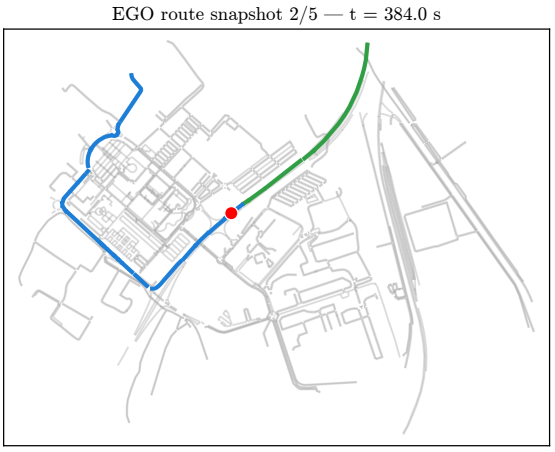
Figure 2.11: Space–time trajectories of vehicles on lane 26599252_0.

2.5.3 EGO vehicle simulation results

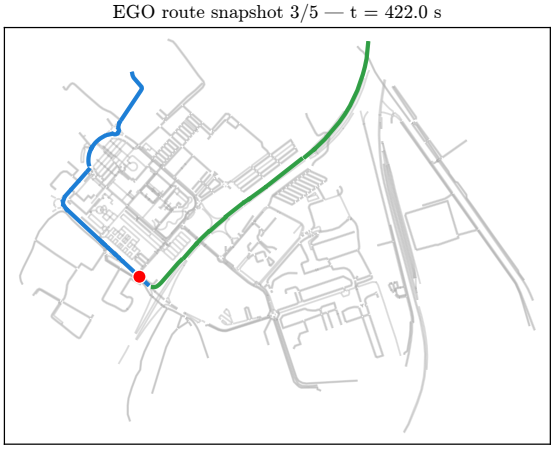
The second part of the analysis focuses on the behaviour of the EGO vehicle during the traffic simulation. In particular, the evolution of its route is first illustrated through a sequence of simulation snapshots, while the longitudinal behaviour of the vehicle is subsequently analysed in terms of speed and acceleration.



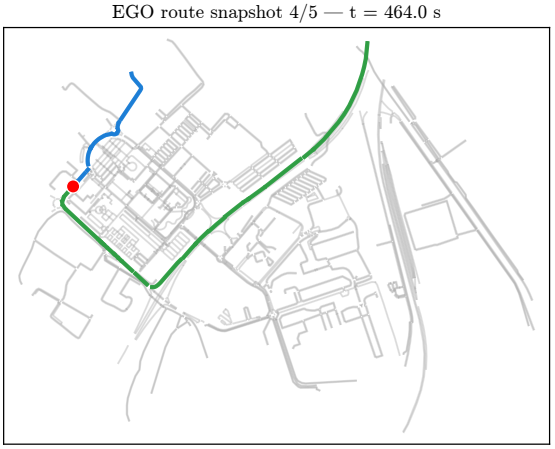
(a) $t = 350.5$ s



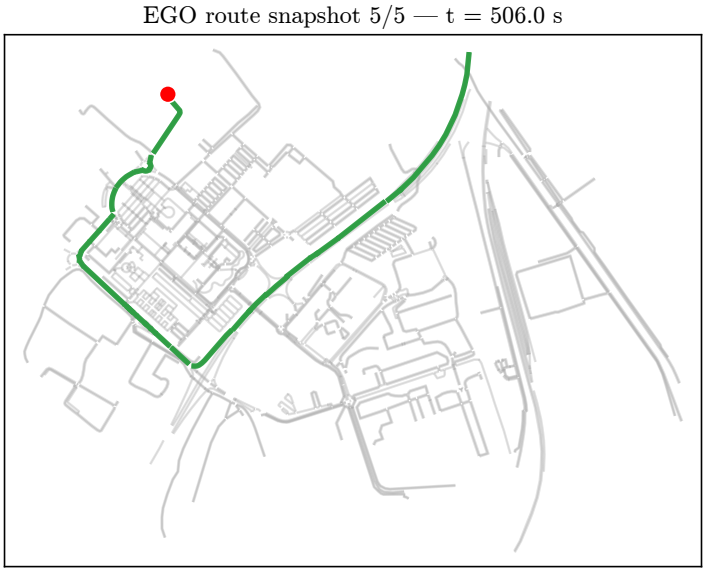
(b) $t = 384.0$ s



(c) $t = 422.0$ s



(d) $t = 464.0$ s



(e) $t = 506.0$ s

Figure 2.12: Evolution of the EGO vehicle during the traffic simulation at different simulation times.

Figure 2.12 shows the evolution of the EGO vehicle at different stages of the simulation. The snapshots provide a qualitative view of the interaction between the vehicle and the surrounding traffic and make it possible to observe the evolution of its assigned route during the simulation. As described in Section 2.3.2, the initial route is computed before departure, while traffic conditions are subsequently evaluated every 5 s and may lead to an online route update.

EGO speed. The speed profile of the EGO vehicle is shown in Figure 2.13. The profile reflects the motion of the vehicle along the selected route and the interaction with the surrounding traffic. Variations in speed are therefore determined by the different road conditions encountered during the trip, including the presence of slower traffic along some sections of the route.

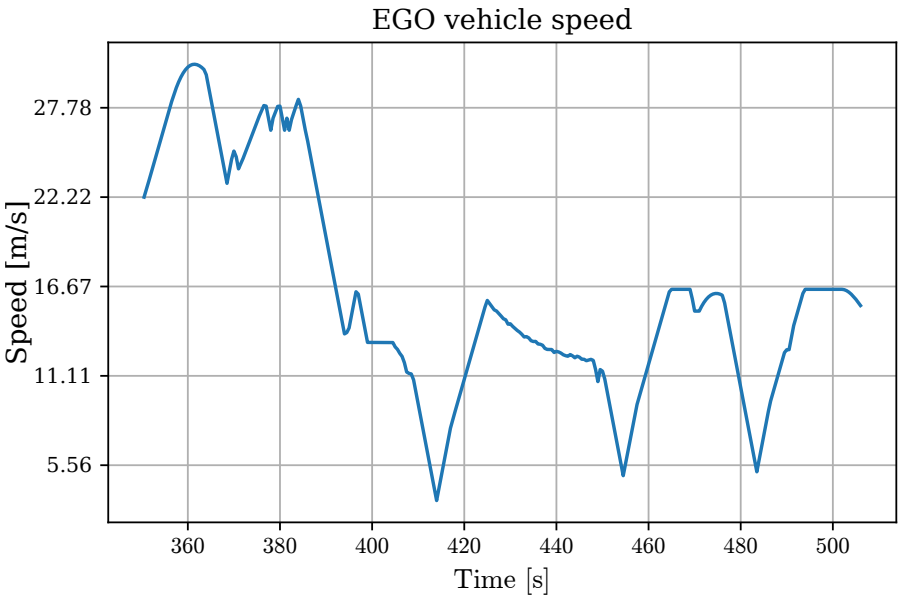


Figure 2.13: Speed profile of the EGO vehicle during the simulation.

EGO acceleration. Figure 2.14 shows the corresponding acceleration profile. Positive and negative acceleration phases are associated with the changes in vehicle speed required while travelling through the network and interacting with the background traffic. The profile therefore provides an additional vehicle-level description of the trajectory shown in the previous results.

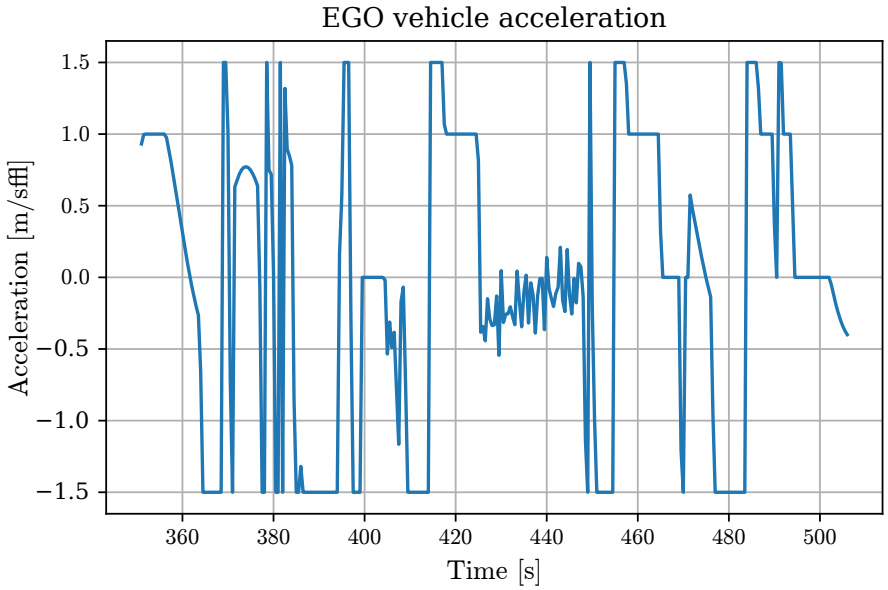


Figure 2.14: Acceleration profile of the EGO vehicle during the simulation.

Overall, these results show how the traffic-level framework combines route planning, online traffic information and vehicle-level data collection within the same SUMO–Python simulation. The objective here is limited to illustrating the behaviour of the EGO vehicle within the traffic-level simulation environment; the longitudinal control problem is analysed separately in the following chapter.

3. Microscopic longitudinal control

3.1 Introduction

Whereas the previous part focused on the behaviour of the EGO vehicle at the traffic-planning level, this chapter shifts the analysis to the microscopic longitudinal control of the individual vehicle.

The central aspect of this section is the introduction of patient profiles, which are examples of different approaches to driving. In the control framework, the patient is considered as an imperfect controller characterised by some limitations, such as reaction delay, reduced promptness and responsiveness, acceleration and deceleration constraints, as well as comfort-related limits.

The EGO vehicle is therefore required to track a specific speed trajectory generated by a higher-level module. In this part, how the trajectory is generated is not the main concern; the objective is instead to determine whether the patient can track it and, when the response is inadequate, whether an assistance controller can provide support to the patient. In this sense, the assistance should not necessarily replace the patient action continuously, but it can be designed to intervene only when the tracking error becomes significant.

It is important to stress that the overall aim of the work is to support driving rehabilitation, therefore providing assistance to patients in order for them to regain confidence in controlling the vehicle.

The chapter first describes the software architecture and the discrete-time plant, including the patient perception delay. It then presents the patient controller, predefined profiles, questionnaire-based personalisation and assistance logic, followed by the reference profiles, scenarios and simulation outputs.

3.2 Software architecture

The project is organised into separate Python files, whose roles are reported in Table 3.1.

File	Role in the program
`config.py`	Defines the global simulation parameters, including SUMO settings, vehicle properties, reference speed profiles, patient profiles, assist-controller gains, assistance activation threshold and lead-vehicle parameters.
`control_models.py`	Implements the discrete-time longitudinal plant, the delayed patient controller and the PID assist controller. This file contains the equations that correspond most directly to the discrete state-space formulation.
`simulation.py`	Contains the main TraCI loop, including SUMO start-up, EGO and lead vehicle insertion, reference generation, delayed perception, acceleration computation, threshold-based assistance activation, speed update and CSV logging.
`microscopic_control.py`	Minimal entry point; calls the `main()` function to run the microscopic simulation.
`main.py`	It starts the SUMO simulation, loads the required settings, initialises the reference speed profile, the patient model and the assistance controller.
`patient_questionnaire.py`	Generates a custom patient profile from an initial questionnaire and saves the corresponding discomfort index and control parameters.
`patient_questionnaire_gui.py`	Provides a graphical user-friendly interface for the questionnaire.
`sumo_setup.py`	Prepares the SUMO working directory, route file, network copy, GUI settings and `.sumocfg` file.
`plots.py`	Reads the CSV outputs and generates plots for speed tracking, tracking error, acceleration and lead-vehicle gap.
`utils.py`	Contains utility functions, such as saturation and conversion between lane identifiers and edge identifiers.

Table 3.1: Role of the main Python files.

The minimal entry point is `microscopic_control.py`, which calls the `main()` function. In a typical execution, the program prepares the folder `micro_run`, builds the SUMO configuration files, starts the simulation through TraCI, applies the microscopic controller

at each sampling instant and stores the results in the `results` folder. The interaction between the Python controller and SUMO is performed through TraCI, which allows retrieving simulation variables and modifying vehicle behaviour online [31].

3.2.1 SUMO simulation environment

The microscopic simulations are performed using SUMO also in this phase [30]. A simple environment is selected: a straight road (NetEdit ID `E0`) with two lanes, a length of 1500 m and a speed limit of 13.89 m/s, equivalent to 50 km/h. These simulations focus on the longitudinal response of the EGO vehicle under a selected patient model; shortest-path computation and route replanning are therefore excluded.

The simulation starts at 0 s and ends at 300 s, with sampling time $T_s = 0.1$ s. The EGO vehicle departs at the beginning of the simulation from the initial position of the road, with zero initial speed. It is represented as a passenger car with a length of 4.8 m, a width of 1.8 m and a height of 1.5 m. In the graphical interface, the EGO vehicle is displayed in red to make it easy to recognise during the simulation.

Two main scenarios are considered in the simulation framework:

- in the base scenario, only the EGO vehicle is present and follows the imposed reference speed profile;
- in the lead-brake scenario, a second vehicle is introduced ahead of the EGO vehicle and performs an abrupt brake. This second case is useful to test the robustness of the control framework in a more critical condition, where the patient and the assistance controller have to react to an external disturbance.

3.2.2 High-level simulation flow

Figure 3.1 summarises in a compact way the overall simulation logic.

First, by means of `config.py` all the simulation parameters, including SUMO settings, reference speed profile, patient profile and scenario are selected. Then, the ideal reference speed is generated from the selected profile. If a lead vehicle is present, the reference can be reduced by a safety layer. In case of a base scenario, i.e. without lead vehicle, $v_{ref,k}$ coincides with $v_{ref,k}^{base}$. In the next step, the imperfect patient model tries to regulate the vehicle speed. If the assistance mode is enabled, the assist controller computes a corrective acceleration in parallel. However, this contribution is actually applied only when the real speed-tracking error exceeds a predefined activation threshold. The patient acceleration and the effective assistance acceleration are then combined, saturated according to the physical and comfort-related constraints, and applied to the discrete longitudinal plant.

Finally, velocity v_k, position s_k and distance from the lead vehicle gap_k (if present), are measured at every step k and given again to the controller (or at the safety block if necessary).

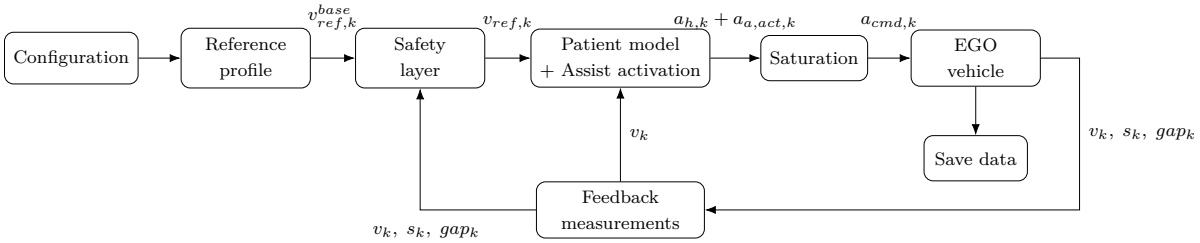


Figure 3.1: Compact architecture of the microscopic longitudinal control framework.

3.3 Discrete-time state-space model

In this section, the discrete-time state-space model is described.

The model is intentionally simplified to study speed tracking and assistance at vehicle level. A detailed dynamic model would require vehicle parameters and environmental conditions that are beyond the scope of this work. The formulation below describes the kinematic plant, patient delay and control actions; the main variables are listed in Table 3.2.

Symbol	**Unit**	**Meaning**
k	–	discrete-time index, with $t_k = kT_s$
T_s	s	simulation sampling time
v_k	m/s	actual EGO speed read from SUMO at step k
$\tilde{v}_k$	m/s	patient-perceived speed
$v_{ref,k}^{base}$	m/s	ideal reference speed generated by the selected profile
$v_{ref,k}$	m/s	effective reference speed, possibly reduced by the safety layer (lead vehicle scenario)
$e_{h,k}$	m/s	patient-based tracking error
$e_{a,k}$	m/s	real speed-tracking error used by the assist controller and by the assistance activation logic.
$a_{h,k}$	m/s^2	patient's acceleration contribution
$a_{a,k}$	m/s^2	raw acceleration contribution generated by the assist controller before activation
$a_{cmd,k}$	m/s^2	total acceleration applied after saturation and jerk limitation
e_{th}	m/s	activation threshold of the assistance controller
$\lambda_{assist,k}$	–	binary variable equal to 1 when the assistance is active and 0 otherwise
$a_{a,act,k}$	m/s^2	effective assistance acceleration after threshold-based activation

Table 3.2: Main variables used in the discrete-time formulation.

3.3.1 Longitudinal plant model

The longitudinal motion of the EGO vehicle is modelled through a discrete-time kinematic plant. The modelling approach adopted in this section follows standard feedback-control and state-space representation concepts, as presented in classical control literature [34, 35]. Since the objective of this work is focused on speed tracking and assistance design, the adopted model does not include a detailed vehicle dynamics representation. Instead, the vehicle speed is propagated starting from the basic continuous-time relation between longitudinal acceleration and speed:

$$\dot{v}(t) = a_{cmd}(t), \tag{3.1}$$

where $v(t)$ is the longitudinal speed of the EGO vehicle and $a_{cmd}(t)$ is the acceleration command effectively applied to the vehicle.

The simulation is executed in discrete time, with sampling instants defined as

$$t_k = kT_s, \tag{3.2}$$

where T_s is the simulation step. In the current implementation, $T_s = 0.1\,\mathrm{s}$. At each time step, the controllers compute an acceleration command that is applied to the vehicle during the following sampling interval. Therefore, between two consecutive sampling instants, the acceleration is assumed to remain constant. This corresponds to a zero-order hold assumption on the input signal:

$$a_{cmd}(t) = a_{cmd,k}, \qquad t \in [kT_s, (k+1)T_s). \tag{3.3}$$

Integrating Equation 3.1 over one interval gives

$$v_{k+1} = v_k + \int_{kT_s}^{(k+1)T_s} a_{cmd}(t)\, dt. \tag{3.4}$$

Under the zero-order hold assumption, this reduces exactly to

$$v_{k+1} = v_k + T_s a_{cmd,k}. \tag{3.5}$$

For this integrator, the update also coincides with forward Euler discretisation.

The resulting speed is then limited to the physically feasible interval. The vehicle cannot move with negative speed and cannot exceed the imposed speed limit. Therefore, the discrete-time speed update is written as:

$$v_{k+1} = \mathrm{sat}_{[0,\, v_{\max}]}\left(v_k + T_s a_{cmd,k}\right), \tag{3.6}$$

where $v_{\max} = 13.89\,\mathrm{m/s}$, corresponding to $50\,\mathrm{km/h}$ in the current simulation configuration. In particular, the saturation of the velocity is defined as:

$$\mathrm{sat}_{[0,\, v_{\max}]}(v) = \begin{cases} 0, & v < 0 \\ v & 0 \le v \le v_{\max}, \\ v_{\max} & v > v_{\max}. \end{cases} \tag{3.7}$$

Patient-specific limits are denoted by $a_{h,\min}$, $a_{h,\max}$ and $j_{h,\max}$; the corresponding global limits on the combined command use the subscript tot. These two sets of constraints serve different roles and must be distinguished.

The acceleration command generated by the control architecture is also subject to physical and comfort-related constraints. Let u_k be the acceleration requested by the patient and assistance controllers before the application of global acceleration and jerk limitations. This signal is not directly applied to the plant. First, it is constrained in order to avoid excessive acceleration values and abrupt acceleration variations.

The jerk limitation requires the current acceleration command to remain close to the previously applied one:

$$a_{cmd,k} \in \left[a_{cmd,k-1} - j_{\mathrm{tot,max}} T_s,\ a_{cmd,k-1} + j_{\mathrm{tot,max}} T_s\right], \tag{3.8}$$

where $j_{\mathrm{tot,max}}$ is the maximum admissible jerk. In addition, the acceleration must remain inside the admissible interval:

$$a_{cmd,k} \in \left[a_{\mathrm{tot,min}},\ a_{\mathrm{tot,max}}\right]. \tag{3.9}$$

The effective lower and upper bounds at time step k are therefore obtained by combining the acceleration and jerk constraints:

$$\underline{a}_k = \max\left(a_{\mathrm{tot,min}},\ a_{cmd,k-1} - j_{\mathrm{tot,max}} T_s\right), \tag{3.10}$$

$$\overline{a}_k = \min\left(a_{\mathrm{tot,max}},\ a_{cmd,k-1} + j_{\mathrm{tot,max}} T_s\right). \tag{3.11}$$

The acceleration effectively applied to the vehicle is then

$$a_{cmd,k} = \mathrm{sat}_{[\underline{a}_k, \overline{a}_k]}(u_k). \tag{3.12}$$

This formulation shows that the previously applied acceleration $a_{cmd,k-1}$ has to be stored, since it is needed to impose the jerk constraint. For this reason, the plant state is defined as

$$x_k = \begin{bmatrix} v_k \\ a_{cmd,k-1} \end{bmatrix}. \tag{3.13}$$

Using this state definition, the complete nonlinear discrete-time plant can be written as

$$x_{k+1} = f(x_k, u_k) = \begin{bmatrix} \mathrm{sat}_{[0,v_{\max}]}\left(v_k + T_s a_{cmd,k}\right) \\ a_{cmd,k} \end{bmatrix}, \tag{3.14}$$

where $a_{cmd,k}$ is computed according to Equation 3.12. The plant is nonlinear because of the saturation functions. These nonlinearities are essential in the simulation framework, since they reproduce both physical limits and comfort-related constraints.

In order to highlight the basic dynamic behaviour of the plant, it is useful to temporarily neglect speed saturation, acceleration saturation and jerk limitation. Under this simplifying assumption, the acceleration requested by the controllers is directly applied to the vehicle:

$$a_{cmd,k} = u_k. \tag{3.15}$$

The corresponding linear discrete-time state-space model is

$$\begin{bmatrix} v_{k+1} \\ a_{cmd,k} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} v_k \\ a_{cmd,k-1} \end{bmatrix} + \underbrace{\begin{bmatrix} T_s \\ 1 \end{bmatrix}}_{B} u_k, \qquad y_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} x_k. \tag{3.16}$$

This representation makes clear that, from the point of view of speed control, the EGO vehicle behaves as a discrete-time integrator driven by acceleration. Indeed, the acceleration command modifies the speed according to the sampling time T_s.

For zero initial conditions, the z-transform of the linear speed update is $(z-1)V(z) = T_s A_{cmd}(z)$. Hence the transfer function is

$$G(z) = \frac{V(z)}{A_{cmd}(z)} = \frac{T_s}{z-1} = \frac{T_s z^{-1}}{1 - z^{-1}}. \tag{3.17}$$

The term z^{-1} makes explicit the one-step propagation introduced by the discrete-time update: the acceleration applied during the interval $[kT_s, (k+1)T_s)$ affects the speed value at the next sampling instant.

The transfer function in Equation 3.17 represents the ideal linear plant used to interpret the speed-control problem. In the actual simulation, however, the complete nonlinear model in Equation 3.14 is used, since the applied acceleration is always constrained by acceleration, jerk and speed limits.

3.3.2 Delay as an augmented state

The patient does not react to the current speed of the EGO vehicle instantaneously. In a real driving situation, a human driver needs a certain amount of time to perceive the vehicle motion, process the information and react. For this reason, the patient model includes a perception delay.

Let τ_h be the reaction delay of the patient. Since the simulation is performed in discrete time with sampling time T_s, this delay can be approximated by an integer number of samples:

$$d_h = \text{round}\left(\frac{\tau_h}{T_s}\right). \tag{3.18}$$

Therefore, the speed perceived by the patient at time step k is not the current speed v_k, but an older value of the speed:

$$\tilde{v}_k \approx v_{k-d_h}. \tag{3.19}$$

This means that the patient computes the tracking error using delayed information. For example,if $T_s = 0.1\,\text{s}$ and the patient delay is $\tau_h = 0.8\,\text{s}$, then $d_h = 8$. In this case, the patient reacts approximately to the speed measured eight simulation steps before.

A state-space interpretation can be obtained by introducing a delay line. The idea is to store the current speed together with a finite number of previous speed values:

$$\mathbf{x}_k^d = \begin{bmatrix} v_k & v_{k-1} & \cdots & v_{k-d_h} \end{bmatrix}^{\mathrm{T}}. \tag{3.20}$$

This vector can be interpreted as a memory of the past speeds. The first element is the most recent speed, while the last element is the oldest speed needed to represent the patient delay.

At each simulation step, a new speed value is computed and inserted at the beginning of the vector. All the previous values are shifted by one position. This behaviour can be

written as

$$x_{k+1}^d = A_d x_k^d + B_d v_{k+1}, \tag{3.21}$$

$$A_d = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \qquad B_d = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.22}$$

The matrix A_d shifts the stored speed values by one position, while the vector B_d inserts the new speed v_{k+1} in the first position of the delay line.

The delayed speed used by the patient is obtained by selecting the last element of the delay state:

$$\tilde{v}_k = C_d x_k^d, \qquad C_d = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{3.23}$$

In this way, the delay is represented as an augmented state. The word "augmented" means that the original vehicle state is extended by adding some memory variables. In other words, these additional variables do not represent new physical dynamics of the vehicle, but only the past speed values required to model the delayed perception of the patient.

In the Python implementation, the same concept is realised by storing the speed history and interpolating the value corresponding to $t_k - \tau_h$. This implementation is more flexible because it does not require the delay to be exactly an integer multiple of the sampling time.

3.4 Patient model and assist control

3.4.1 Patient controller

The patient tracking error is computed from the effective reference and the delayed perceived speed:

$$e_{h,k} = v_{ref,k} - \tilde{v}_k. \tag{3.24}$$

The raw patient acceleration consists of a proportional action:

$$a^{\star}_{h,k} = K_{p,h} e_{h,k} \tag{3.25}$$

The raw command is first saturated:

$$\bar{a}_{h,k} = \mathrm{sat}_{[a_{h,\min},\, a_{h,\max}]} \left(a^{\star}_{h,k} \right), \tag{3.26}$$

and then limited in jerk:

$$a_{h,k} = \mathrm{sat}_{[a_{h,k-1} - j_{h,\max} T_s,\, a_{h,k-1} + j_{h,\max} T_s]} \left(\bar{a}_{h,k} \right). \tag{3.27}$$

3.4.2 Patient profiles

In this part, the predefined patient profiles used for the simulations are analysed in detail. These profiles are useful to test the microscopic control framework under different driving behaviours without running the questionnaire every time.

The patient profiles are defined through a small number of interpretable parameters: proportional gain, perception delay, maximum acceleration, maximum deceleration and maximum jerk. By changing these parameters, the same controller structure can reproduce different driving behaviours.

Profile	$K_{p,h}$	τ_h [s]	$a_{h,\max}$ [m/s^2]	$a_{h,\min}$ [m/s^2]	$j_{h,\max}$ [m/s^3]
reactive	1.15	0.6	2.30	-4.30	3.80
normal	0.45	0.8	1.15	-2.20	1.20
cautious	0.25	1.1	0.65	-1.60	0.55
delayed	0.12	1.6	0.45	-1.20	0.35

Table 3.3: Patient profiles available in the current configuration.

For example, the `delayed` profile has a low gain, a large delay and restrictive acceleration

and jerk limits. This profile is useful to stress the controller, because the EGO vehicle reacts slowly both when accelerating and when braking.

The perception–response delays assigned to the predefined patient profiles were selected by considering reaction-time values reported in driving-simulator literature.

In their literature review, Poliak et al. cite a previous simulator study in which driver response times ranged from 0.32 s to 1.64 s. In that study, the 33rd, 50th, and 67th percentile values, equal to 0.52 s, 0.82 s, and 1.10 s, respectively, were selected to represent different driver response levels in simulation [36], as reported in the review by Poliak et al. [37].

Poliak et al. also conducted their own simulator experiment involving 116 professional drivers. The measured reaction times had a mean value of 1.10 s, a median value of 1.06 s, and ranged from 0.66 s to 1.73 s. Their statistical analysis showed that the mean reaction time was significantly higher than the conventional value of 1.0 s, while it could be considered compatible with a reference value of 1.1 s [37].

Based on these results, delays of 0.6 s, 0.8 s, 1.1 s, and 1.6 s were assigned to the `reactive`, `normal`, `cautious`, and `delayed` profiles, respectively.

The first three values are close to the percentile-based values reported in the simulator literature, whereas the value of 1.6 s represents a substantially slower response while remaining within the experimental ranges discussed by Poliak et al. These values are not intended to define clinically validated patient categories, but rather to provide an engineering parameterisation of progressively slower perception–response behaviours.

Although the experimental task considered by Poliak et al. was based on an emergency braking response, the reported values provide a plausible reference range for modelling the delayed perception and response of a human driver.

3.4.3 Questionnaire-based patient characterisation

The patient parameters can be selected either by choosing one of the predefined profiles, namely `reactive`, `normal`, `cautious`, and `delayed`, or by generating a custom profile through an initial questionnaire.

The questionnaire is not intended as a clinical assessment. It is used as a heuristic interface to translate subjective information about comfort, confidence, and perceived responsiveness into numerical parameters for the patient controller.

The patient is asked to evaluate the following seven aspects:

1. discomfort during acceleration and braking manoeuvres;
2. nausea or discomfort while travelling by car;

3. stress caused by traffic and sudden braking events;
4. perceived slowness in reacting to speed variations;
5. fatigue or lack of concentration;
6. uncertainty while controlling the vehicle;
7. preference for smooth driving without abrupt variations.

Each answer is expressed through an integer score $s_i \in \{1, 2, 3, 4, 5\}$, where 1 represents the least critical condition and 5 the most critical one. The scores are normalised as

$$q_i = \frac{s_i - 1}{4}, \qquad q_i \in [0, 1]. \tag{3.28}$$

A global discomfort index is then computed as

$$D = 100 \sum_{i=1}^{7} w_i q_i, \qquad \sum_{i=1}^{7} w_i = 1, \tag{3.29}$$

with weights

$$\begin{aligned} w_{\text{motion}} = 0.18, \quad w_{\text{nausea}} = 0.14, \quad & w_{\text{traffic}} = 0.18, \\ w_{\text{reaction}} = 0.20, \quad w_{\text{fatigue}} = 0.12, \quad & w_{\text{uncertainty}} = 0.10, \\ w_{\text{smoothness}} = 0.08. & \end{aligned} \tag{3.30}$$

Therefore, $D \in [0, 100]$, where higher values indicate greater discomfort or reduced responsiveness.

Two additional indicators are introduced. The delay score is defined as

$$S_{\text{delay}} = 0.60 q_{\text{reaction}} + 0.25 q_{\text{fatigue}} + 0.15 q_{\text{traffic}}, \tag{3.31}$$

while the comfort score used for classification is

$$S_{\text{comfort}} = 0.35 q_{\text{motion}} + 0.25 q_{\text{traffic}} + 0.25 q_{\text{smoothness}} + 0.15 q_{\text{nausea}}. \tag{3.32}$$

The classification rules are evaluated sequentially. The patient is assigned to the `delayed` class if

$$S_{\text{delay}} \geq 0.68 \qquad \text{or} \qquad q_{\text{reaction}} \geq 0.80. \tag{3.33}$$

Otherwise, the `cautious` class is assigned if

$$S_{\text{comfort}} \geq 0.62 \qquad \text{or} \qquad D \geq 65. \tag{3.34}$$

The `reactive` class is assigned if

$$D \leq 28, \qquad S_{\text{delay}} \leq 0.35, \qquad q_{\text{uncertainty}} \leq 0.40. \tag{3.35}$$

In all remaining cases, the patient is classified as `normal`.

The assigned class provides an interpretable description of the questionnaire result. However, the controller parameters are generated continuously from the answers and are not copied directly from the corresponding predefined profile.

A comfort-related indicator is first defined as

$$S_{\text{need}} = 0.30q_{\text{motion}} + 0.20q_{\text{nausea}} + 0.25q_{\text{traffic}} + 0.25q_{\text{smoothness}}. \tag{3.36}$$

The two comfort indicators serve different heuristic roles: S_{comfort} selects a descriptive class, whereas S_{need} continuously adjusts acceleration, braking and jerk limits. The latter assigns slightly more weight to nausea and less to motion discomfort. This weighting is a design choice, not an experimentally established distinction.

The proportional gain is then computed as

$$K_{p,h} = \text{sat}_{[0.12,\,1.15]}\left(1.05 - 0.60\frac{D}{100} - 0.55q_{\text{reaction}}\right). \tag{3.37}$$

The perception–response delay is constrained within the range defined by the predefined `reactive` and `delayed` profiles:

$$\tau_{\min} = \tau_{\text{reactive}}, \qquad \tau_{\max} = \tau_{\text{delayed}}. \tag{3.38}$$

It is calculated as

$$\tau_h = \tau_{\min} + (\tau_{\max} - \tau_{\min})\left(0.60S_{\text{delay}} + 0.40S_{\text{delay}}^2\right). \tag{3.39}$$

In the current configuration,

$$\tau_{\min} = 0.6\,\text{s}, \qquad \tau_{\max} = 1.6\,\text{s}. \tag{3.40}$$

The maximum acceleration is obtained as

$$a_{h,\max} = \mathrm{sat}_{[0.45,\,2.30]}\left(2.30 - 1.85S_{\mathrm{need}} - 0.45q_{\mathrm{fatigue}}\right). \tag{3.41}$$

The braking capacity is computed as

$$b_h = \mathrm{sat}_{[1.20,\,4.30]}\left(4.30 - 2.75S_{\mathrm{need}} - 0.85q_{\mathrm{traffic}} - 0.45q_{\mathrm{reaction}}\right), \tag{3.42}$$

with

$$a_{h,\min} = -b_h. \tag{3.43}$$

Finally, the maximum jerk is defined as

$$j_{h,\max} = \mathrm{sat}_{[0.35,\,3.80]}\left(3.80 - 4.50S_{\mathrm{need}} - 0.70q_{\mathrm{fatigue}}\right). \tag{3.44}$$

Higher discomfort, fatigue, reaction slowness, and preference for smooth driving therefore produce a lower proportional gain, a larger perception–response delay, and more restrictive acceleration and jerk limits.

The weights, thresholds, and interpolation laws adopted in this procedure are heuristic design choices introduced for the present simulation framework. They are not clinically validated and may be recalibrated when experimental patient data become available.

The questionnaire is implemented in the `patient_questionnaire.py` module, while the graphical interface is provided by `patient_questionnaire_gui.py`. The original answers, assigned class, discomfort index, and generated controller parameters are saved in a CSV file to ensure traceability and reproducibility.

3.4.4 Driving test-based patient parameter identification

This section outlines a possible future extension of the framework, in which patient model parameters would be identified from preliminary driving tests. The procedure is presented at a conceptual level and has not been implemented in the current version of the framework.

The predefined patient profiles introduced above represent reference parameterisations of different driving behaviours. However, in a real application, it would be difficult to assign a proportional gain, a perception delay or a comfort-related constraint directly to a specific patient.

For this reason, a possible extension of the proposed framework consists in identifying the patient parameters from a preliminary driving test. During the test, the patient is required to follow one or more prescribed reference velocity profiles, while the resulting

vehicle speed and acceleration are recorded.

The patient model is described by the parameter vector

$$\boldsymbol{\theta}_h = \begin{bmatrix} K_{p,h} & \tau_h & a_{h,\max} & a_{h,\min} & j_{h,\max} \end{bmatrix}^T. \tag{3.45}$$

These parameters should not be interpreted as direct clinical measurements. In particular, the proportional gain $K_{p,h}$ represents an equivalent control parameter describing how strongly the patient reacts to the perceived tracking error.

A possible identification procedure consists in finding the parameter vector that minimises the difference between the measured patient response and the response generated by the simulation model. This can be expressed as

$$\hat{\boldsymbol{\theta}}_h = \arg\min_{\boldsymbol{\theta}_h} J(\boldsymbol{\theta}_h), \tag{3.46}$$

where the cost function may be defined as

$$J(\boldsymbol{\theta}_h) = w_v \sum_{k=0}^{N} \left(v_k^{\text{meas}} - v_k^{\text{sim}}(\boldsymbol{\theta}_h) \right)^2 + w_a \sum_{k=0}^{N} \left(a_k^{\text{meas}} - a_k^{\text{sim}}(\boldsymbol{\theta}_h) \right)^2. \tag{3.47}$$

Here, v_k^{meas} and a_k^{meas} denote the speed and acceleration measured during the preliminary test, while v_k^{sim} and a_k^{sim} are the corresponding signals generated by the patient model. The weights w_v and w_a determine the relative importance assigned to speed tracking and acceleration behaviour.

The reaction delay can be estimated from the temporal shift between a variation of the reference speed and the beginning of the patient response. The acceleration, deceleration and jerk limits can instead be inferred from the maximum values observed during the test. The proportional gain is then identified as the value that best reproduces the measured relationship between tracking error and longitudinal action.

Once the parameter vector has been estimated, the patient can either be associated with the closest predefined profile or represented through a custom profile. Therefore, the predefined profiles can be interpreted as representative classes rather than as fixed clinical categories assigned a priori.

The complete automatic identification procedure is not implemented in the current version of the framework. Nevertheless, the proposed formulation shows how the same simulation architecture could be used in the future for objective patient characterisation and controller personalisation. Figure 3.2 summarises the conceptual procedure.

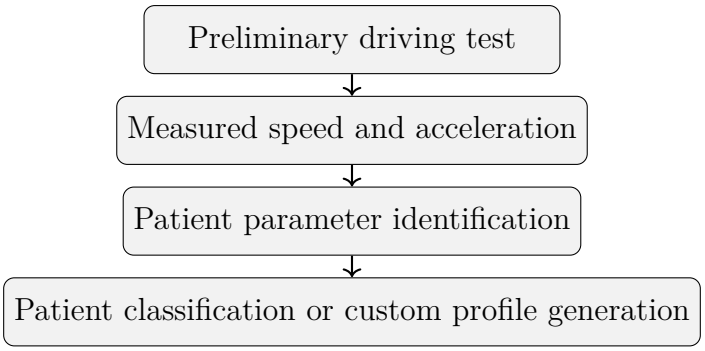


Figure 3.2: Conceptual procedure for test-based patient characterisation.

3.4.5 Assist controller

The assist controller is introduced to support the patient during the tracking task. Its objective is not to replace the patient completely, but to provide an additional acceleration contribution when the patient response is not sufficient to follow the desired speed profile. The assist controller is designed according to a standard feedback-control structure, where the tracking error is used to compute a corrective control action [34].

The assist controller operates in parallel with the patient controller. The patient model computes its action using the delayed perceived speed, while the assist controller uses the current EGO speed measured from the simulation. Therefore, the assist tracking error is defined as

$$e_{a,k} = v_{ref,k} - v_k. \tag{3.48}$$

This difference is important: the patient reacts to delayed information, because the human reaction time is part of the model. On the contrary, the assist controller has access to the current vehicle speed and can therefore react more promptly. For this reason, the assist controller can partially compensate for the delay and the limited responsiveness of the patient.

The assist action is implemented as a discrete-time PID controller. The integral state is updated as:

$$\bar{I}_{k-1} = \begin{cases} 0, & e_{a,k}e_{a,k-1} < 0, \\ I_{k-1}, & \text{otherwise}, \end{cases} \qquad I_k = \mathrm{sat}_{[-I_{\max},\, I_{\max}]}\left(\bar{I}_{k-1} + T_s e_{a,k}\right). \tag{3.49}$$

The saturation of the integral state is used to avoid wind-up. Without this limitation, the integral term could grow excessively when the tracking error remains large for a long time, producing an aggressive acceleration command. In addition, the integral term is reset when the tracking error changes sign. This avoids excessive correction after the EGO

vehicle speed crosses the reference value.

The raw acceleration command generated by the assist controller is

$$a^{\star}_{a,k} = K_{p,a}e_{a,k} + K_{i,a}I_k + K_{d,a}\frac{e_{a,k} - e_{a,k-1}}{T_s}. \tag{3.50}$$

The three terms have different roles. The proportional term reacts directly to the current tracking error. If the EGO vehicle is slower than the reference, the proportional action increases the acceleration request; if the EGO vehicle is faster than the reference, it contributes to braking. The integral term compensates persistent tracking errors that may remain due to the limited action of the patient or to the presence of constraints. The derivative term reacts to fast variations of the error and helps to reduce overshoot and oscillatory behaviour.

In the current configuration, the assist-controller gains are

$$K_{p,a} = 0.75, \qquad K_{i,a} = 0.025, \qquad K_{d,a} = 0.35. \tag{3.51}$$

These gains are not chosen to obtain the most aggressive tracking response. Instead, they are selected to obtain a compromise between tracking accuracy, smoothness and comfort. Since the simulation includes nonlinear effects such as acceleration saturation, jerk limitation, speed limits and patient delay, a purely analytical PID tuning is not sufficient. For this reason, the gains are selected through a simulation-based tuning procedure.

The tuning procedure can be summarised as follows. First, the integral and derivative gains are set to zero, and only the proportional gain is increased. This allows the main responsiveness of the assist controller to be adjusted. A larger value of $K_{p,a}$ reduces the tracking error, but it can also generate excessive acceleration or oscillations. Therefore, $K_{p,a}$ is increased until the response becomes sufficiently fast without becoming too aggressive.

Then, the derivative gain $K_{d,a}$ is introduced. The derivative term is useful to damp the response, especially when the reference speed changes rapidly or when the patient model introduces a significant delay. Increasing $K_{d,a}$ helps to reduce overshoot and oscillations, but too large a derivative action can make the command sensitive to rapid error variations.

Finally, a small integral gain $K_{i,a}$ is added. The integral term is useful to reduce residual steady-state errors. However, in this application it must remain limited, because an excessive integral action may produce wind-up and uncomfortable acceleration after the error changes sign. For this reason, the integral gain is chosen smaller than the proportional and derivative contributions, and the integral state is saturated.

The final values are therefore obtained by comparing different simulations and checking both tracking and comfort indicators. In particular, the tuning is evaluated by observing

the speed tracking error, the maximum overshoot, the acceleration profile and the jerk limitation. The selected gains are those that improve the patient response without producing unrealistic or uncomfortable acceleration commands.

3.4.6 Threshold-based assistance activation

In the first implementation, when the assistance mode was enabled, the assistance contribution was continuously added to the patient command. Although this solution improves the tracking performance, it may also make the assistance too intrusive, since the controller acts even when the patient is able to follow the reference speed with an acceptable error.

A threshold-based activation logic uses the real speed-tracking error $e_{a,k}$ defined in Equation 3.48, evaluated from the current EGO speed.

The assistance is activated only when the absolute value of this error exceeds a predefined threshold:

$$|e_{a,k}| \geq e_{th}. \tag{3.52}$$

In the current implementation, the threshold is set to

$$e_{th} = 1.2\,\mathrm{m/s} = 4.3\,\mathrm{km/h} \tag{3.53}$$

This threshold defines a dead zone around the reference speed. Inside this region, the tracking error is considered acceptable and the patient is allowed to control the vehicle without external support.

The activation variable of the assistance system is defined as:

$$\lambda_{assist,k} = \begin{cases} 0, & |e_{a,k}| < e_{th}, \\ 1, & |e_{a,k}| \geq e_{th}. \end{cases} \tag{3.54}$$

After the PID action is computed, the assist contribution is saturated with its own acceleration and jerk limits. This ensures that the raw assistance command remains smooth and physically reasonable. However, the saturated assist contribution is actually applied to the vehicle only if the activation condition is satisfied. Therefore, the effective assistance acceleration is

$$a_{a,act,k} = \lambda_{assist,k} a_{a,k}. \tag{3.55}$$

The patient and effective assistance contributions are then combined as

$$u_k = a_{h,k} + a_{a,act,k}. \tag{3.56}$$

The combined request u_k is processed by the global acceleration and jerk limiter defined in Equation 3.12. The resulting $a_{cmd,k}$, rather than the unconstrained sum, is applied to the plant. This uses the same final-command definition throughout the model.

3.4.7 Parallel control block diagram

The patient controller and the assist controller operate in parallel, but the assistance contribution is not directly added to the patient action. Instead, it is first processed by an activation logic based on the real speed-tracking error. Therefore, the assist contribution is effectively applied only when the tracking error exceeds the predefined threshold. Moreover, it can be noticed that the upper feedback includes the human delay, which represents the patient's delayed perception of the vehicle speed.

The resulting parallel control architecture is shown in Figure 3.3.

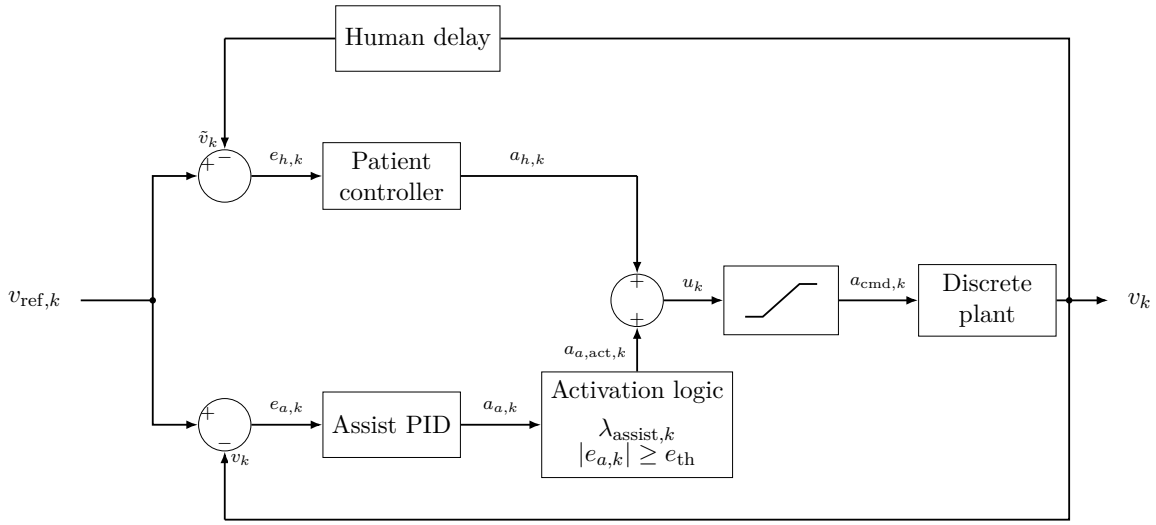


Figure 3.3: Parallel patient–assist longitudinal control architecture with threshold-based assistance activation and final command saturation.

A closed-loop transfer function can be derived by considering a simplified linear representation of the control architecture. The following assumptions are introduced:

- the acceleration, speed and jerk saturations are neglected;
- the patient delay is approximated as an integer discrete-time delay;
- the assistance activation variable is assumed constant, $\lambda_{\mathrm{assist}} \in \{0, 1\}$;
- zero initial conditions are considered.

Under these assumptions, the human perception delay can be represented as

$$D_h(z) = z^{-d_h}, \tag{3.57}$$

where d_h is the number of delayed samples. The patient and assist control actions are therefore written as

$$U_h(z) = C_h(z)\left[V_{\text{ref}}(z) - D_h(z)V(z)\right], \tag{3.58}$$

$$U_a(z) = C_a(z)\left[V_{\text{ref}}(z) - V(z)\right]. \tag{3.59}$$

The total control input is

$$U(z) = U_h(z) + \lambda_{\text{assist}} U_a(z), \tag{3.60}$$

while the linearised vehicle model satisfies

$$V(z) = G(z)U(z), \qquad G(z) = \frac{T_s}{z-1}. \tag{3.61}$$

Consequently, the local closed-loop transfer function from the reference velocity to the vehicle velocity is

$$\boxed{\frac{V(z)}{V_{\text{ref}}(z)} = \frac{G(z)\left[C_h(z) + \lambda_{\text{assist}} C_a(z)\right]}{1 + G(z)\left[C_h(z)D_h(z) + \lambda_{\text{assist}} C_a(z)\right]}} \tag{3.62}$$

For $\lambda_{\text{assist}} = 0$, Equation (3.62) describes the patient-only closed loop, whereas $\lambda_{\text{assist}} = 1$ represents the system when the assistance is active.

Since the complete implementation includes threshold-based switching, saturations and jerk limitations, Equation (3.62) should be interpreted as a local linear representation rather than as a global transfer function of the complete nonlinear system.

3.5 Reference velocity profiles generation

In this section, the reference velocity profiles used for the simulations are introduced.

These profiles represent the output of the higher-level planning or macroscopic control layer. For the purposes of this study, it is not crucial for them to be optimal, since the aim is to observe how the patients react to them. On the contrary, different types of non-ideal profiles are provided in order to simulate some critical situations. Actually, in real driving conditions, drivers are not required to follow a perfectly predefined trajectory; instead, they must respond to traffic conditions and unexpected events.

The selected velocity profiles are gathered in the Table 3.4

Reference profile	**Description**
Trapezoidal	The trapezoidal profile is composed of an acceleration phase, a constant-speed cruise phase and a final deceleration phase. It is used as a basic reference to test the ability of the microscopic controller to follow smooth acceleration and braking manoeuvres.
Sinusoidal	The sinusoidal profile imposes a continuously varying reference speed, saturated between zero and the road speed limit. It is useful to analyse phase lag, tracking attenuation and oscillatory behaviour.
Step	The step profile introduces abrupt changes in the desired speed. It is intentionally demanding and is used to evaluate maximum tracking error, overshoot and settling time.
Exponential stop-and-go	The exponential stop-and-go profile generates smooth transitions between target speeds. It is useful to reproduce stop-and-go behaviour without introducing instantaneous speed jumps.

Table 3.4: Summary of the reference speed profiles implemented in the simulation framework.

3.6 Simulation scenarios

3.6.1 Base scenario

In the base scenario, only the EGO vehicle is present. The main objective is to isolate the tracking behaviour of the patient and assist controllers without external safety constraints. This scenario is appropriate for comparing patient profiles and checking the effect of delay, gain and saturation.

3.6.2 Lead-brake scenario

In the lead-brake scenario, a yellow lead vehicle is introduced ahead of the EGO vehicle. The leader initially accelerates towards its cruising speed and, when a set of gap and velocity conditions is satisfied, performs an emergency braking manoeuvre until standstill.

The main parameters used in the trapezoidal-reference lead-brake simulations are reported in Table 3.5.

Symbol	**Value**	**Description**
$v_{\text{lead},0}$	2.0 m/s	Initial leader velocity, inherited from the initial value of the trapezoidal profile.
$x_{\text{lead},0}$	120.0 m	Initial longitudinal position of the lead vehicle.
$x_{\text{ego},0}$	0.0 m	Initial longitudinal position of the EGO vehicle.
L_{lead}	4.8 m	Lead-vehicle length used to compute the net inter-vehicle gap.
g_{init}	115.2 m	Initial net gap, obtained as $x_{\text{lead},0} - x_{\text{ego},0} - L_{\text{lead}}$.
$v_{\text{lead,max}}$	10.0 m/s	Cruising-speed limit assigned to the lead vehicle.
$a_{\text{lead,max}}$	1.0 m/s^2	Maximum leader acceleration during the initial phase.
g_{trig}	100.0 m	Net-gap threshold enabling the emergency braking event.
$a_{\text{lead,br}}$	-6.0 m/s^2	Constant signed braking acceleration applied to the leader until standstill. Its magnitude is $b_{\text{lead}} = 6.0$ m/s^2.
g_0	12.0 m	Static minimum distance included in the dynamically required gap.
T_h	2.5 s	Following time headway used to convert the gap error into a safe-velocity correction.
b_{ego}	5.5 m/s^2	Available EGO braking-deceleration magnitude, corresponding to `ASSIST_TOTAL_A_MIN` $= -5.5$ m/s^2.
τ_r	0.0 s	Reaction delay used in the automatic safety-reference calculation. The patient perception delay is modelled separately.

Table 3.5: Parameters used in the lead-brake simulation scenario.

The net gap is measured between the front of the EGO vehicle and the rear of the leader:

$$g_k = x_{\text{lead},k} - x_{\text{ego},k} - L_{\text{lead}}. \tag{3.63}$$

The emergency braking event is activated only when all the following conditions are simultaneously satisfied:

$$g_k \leq g_{\text{trig}}, \tag{3.64}$$

$$v_{\text{lead},k} \geq 0.95\, v_{\text{lead,max}}, \tag{3.65}$$

$$v_{\text{ego},k} \geq 0.98\, v_{\max}. \tag{3.66}$$

These additional velocity conditions prevent the leader from braking during the initial acceleration phase of the simulation.

The safety calculation is based on a simplified constant-deceleration model. Road slope, aerodynamic effects and actuation dynamics are neglected. Under these assumptions, the stopping distance of a vehicle travelling at speed v with an available deceleration magnitude b is

$$d_{\text{stop}} = \frac{v^2}{2b}. \tag{3.67}$$

For the lead vehicle, considering a cruising speed of 10 m/s and a braking deceleration of 6 m/s^2,

$$d_{\text{lead}} = \frac{10^2}{2 \cdot 6} \approx 8.33 \text{ m}. \tag{3.68}$$

The corresponding braking time is

$$t_{\text{lead}} = \frac{10}{6} \approx 1.67 \text{ s}. \tag{3.69}$$

For the EGO vehicle, considering the maximum road speed $v_{\max} = 13.89$ m/s and an available deceleration magnitude of 5.5 m/s^2,

$$d_{\text{ego}} = \frac{13.89^2}{2 \cdot 5.5} \approx 17.54 \text{ m}. \tag{3.70}$$

For the online safety calculation, the stopping distances are evaluated at each simulation step using the current velocities of the two vehicles:

$$d_{\text{ego},k} = \frac{v_{\text{ego},k}^2}{2b_{\text{ego}}}, \tag{3.71}$$

$$d_{\text{lead},k} = \frac{v_{\text{lead},k}^2}{2b_{\text{lead}}}, \tag{3.72}$$

where the braking-deceleration magnitudes are reported in Table 3.5.

The actual gap g_k is then compared with a dynamically required gap, defined as

$$g_{\text{req},k} = g_0 + v_{\text{ego},k}\tau_r + \max\left(0, d_{\text{ego},k} - d_{\text{lead},k}\right), \tag{3.73}$$

where g_0 is the static minimum distance and τ_r is the reaction delay included in the automatic safety calculation. Their values are reported in Table 3.5.

Therefore, the implemented required gap can be written as

$$g_{\text{req},k} = 12 + \max\left(0, \frac{v_{\text{ego},k}^2}{2 \cdot 5.5} - \frac{v_{\text{lead},k}^2}{2 \cdot 6.0}\right). \tag{3.74}$$

For example, considering the nominal velocities at the braking instant, $v_{\text{ego}} = 13.89$ m/s and $v_{\text{lead}} = 10$ m/s, the required gap is

$$g_{\text{req}} = 12 + \max\left(0, \frac{13.89^2}{2 \cdot 5.5} - \frac{10^2}{2 \cdot 6}\right) \approx 21.21 \text{ m}. \tag{3.75}$$

The term $d_{\text{ego},k} - d_{\text{lead},k}$ represents the additional distance required when the EGO vehicle needs more space than the leader to come to a complete stop.

The safe velocity is computed by converting the difference between the actual and required gaps into a velocity correction:

$$v_{\text{safe},k} = \max\left[0, v_{\text{lead},k} + \frac{g_k - g_{\text{req},k}}{T_h}\right], \tag{3.76}$$

where T_h is the following time headway reported in Table 3.5. It converts the gap error $g_k - g_{\text{req},k}$ into a velocity correction: a larger value of T_h produces a smoother and less aggressive reference adaptation.

When $g_k < g_{\text{req},k}$, the safe velocity is reduced below the current leader velocity. Conversely, when $g_k \geq g_{\text{req},k}$, the safe velocity may be higher than the leader velocity, since the available distance is greater than the required one.

The safety layer is not allowed to increase the nominal reference velocity. Therefore, the effective reference is defined as

$$v_{\mathrm{ref},k} = \min\left(v_{\mathrm{ref},k}^{\mathrm{base}}, v_{\mathrm{safe},k}\right). \tag{3.77}$$

Consequently, $v_{\mathrm{safe},k}$ acts as an upper bound on the nominal reference and modifies the effective reference only when it is lower than $v_{\mathrm{ref},k}^{\mathrm{base}}$.

In the current implementation, this gap-aware reference adaptation is applied to the control architecture only in the lead-brake scenario when the assistance mode is enabled. When assistance is disabled, the safety quantities are still computed for logging and analysis, but the patient continues to follow the base reference and reacts to the leader braking event after the patient-specific perception delay.

It is important to distinguish the different distance parameters used in the scenario. The 100 m threshold is only an event trigger that initiates the sudden braking manoeuvre of the leader. The value $g_0 = 12$ m is the static minimum distance, whereas $g_{\mathrm{req},k}$ is the dynamic distance required according to the current vehicle velocities and braking capabilities. Under the nominal conditions considered above, this dynamic distance is approximately 21.21 m. The patient perception delay, which ranges from 0.6 to 1.6 s depending on the selected profile, is modelled separately and is not included in the automatic safety calculation, since $\tau_r = 0$ s.

3.7 Simulation loop and data logging

This section describes the simulation loop and the outputs it produces. The flowchart in Figure 3.4 shows the operations performed by the Python pipeline in detail:

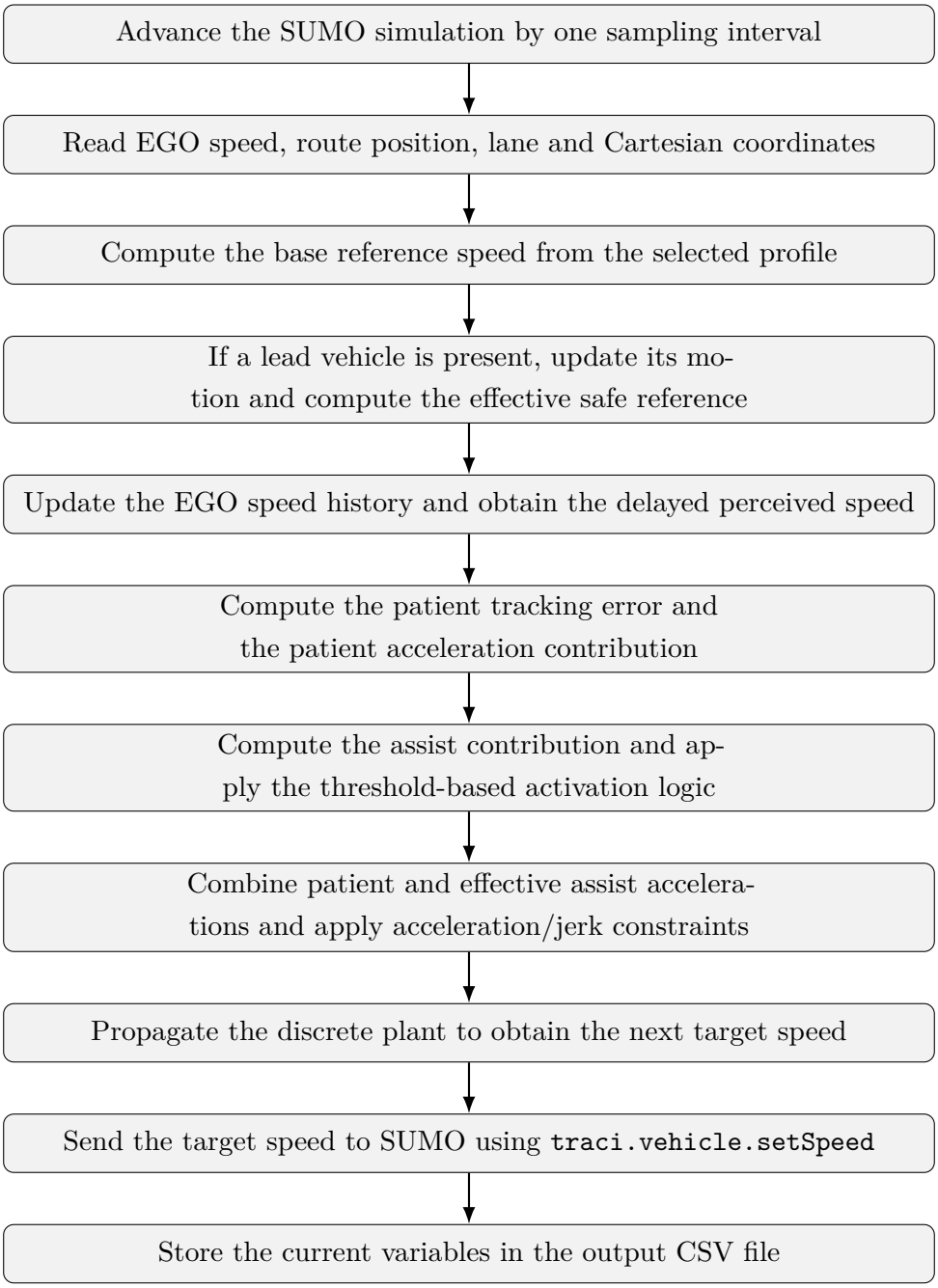


Figure 3.4: Main operations performed during each TraCI control cycle.

This cycle makes the connection between the mathematical model and the executable simulation explicit: the controller is evaluated once per sampling interval, and the acceleration command is converted into a new speed target through the discrete plant equation.

3.7.1 CSV output

Each simulation produces one CSV file with one row per time step. The most relevant columns are summarised in Table 3.6.

CSV column	Description
`time_s`	Simulation time.
`route_position_m`	EGO vehicle position along the route.
`speed_ms`	Actual EGO vehicle speed.
`base_reference_ms`	Ideal speed reference before safety adaptation.
`reference_ms`	Effective reference speed after possible lead-vehicle adaptation.
`perceived_speed_ms`	Delayed speed perceived by the patient.
`error_ms`	Patient tracking error.
`patient_accel_ms2`	Acceleration contribution generated by the patient controller.
`assist_accel_ms2`	Acceleration contribution generated by the assist controller.
`accel_ms2`	Total commanded acceleration applied to the vehicle.
`lead_gap_m`	Actual distance from the lead vehicle, when present.
`perceived_lead_gap_m`	Delayed perceived distance from the lead vehicle.

Table 3.6: Main columns stored in the simulation CSV output file.

3.7.2 Simulation summary

The simulation results are evaluated using a set of tracking, assistance and safety metrics, whose main columns are summarised in Table 3.7.

Column	Description
Simulation configuration	
`patient`	Selected patient profile.
`profile_source`	Origin of the patient profile, e.g. predefined or questionnaire-based.
`patient_class`	Patient class associated with the selected profile.
`discomfort_index`	Discomfort index used for questionnaire-generated profiles.
`scenario`	Selected simulation scenario.
`reference_profile`	Reference speed profile used in the simulation.
`assist_enabled`	Whether the assist controller is enabled.
Simulation outcome	
`arrived`	Indicates if the EGO vehicle reached the end of the route.
`collision_or_teleport`	Whether a collision or SUMO teleport occurred.
`duration_s`	Total simulation duration.
`final_position_m`	Final longitudinal position reached by the EGO vehicle.
Tracking performance	
`rmse_ms`	Root mean square speed-tracking error.
`mae_ms`	Mean absolute speed-tracking error.
`max_abs_error_ms`	Maximum absolute speed-tracking error.
`max_overshoot_ms`	Maximum amount by which the EGO speed exceeds the reference speed.
`overshoot_time_s`	Total time spent above the reference speed.
`settling_time_s`	Settling time, mainly used for step-reference simulations.
Assistance and safety indicators	
`assist_intervention_percentage`	Percentage of samples with threshold-based assistance active.
`min_lead_gap_m`	Minimum distance between the EGO vehicle and the lead vehicle, when present.
`time_below_safe_gap_s`	Time below the required safety gap in the lead-brake scenario.

Table 3.7: Main columns stored in the `summary.csv` output file.

Unless otherwise specified, all tracking metrics use the effective-reference error $e_k \equiv e_{a,k}$ defined in Equation 3.48.

For N recorded simulation samples, the root mean square error and mean absolute error are defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} e_k^2}, \tag{3.78}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{k=1}^{N} |e_k|. \tag{3.79}$$

The maximum absolute tracking error is

$$e_{\max} = \max_k |e_k|. \tag{3.80}$$

The maximum overshoot is evaluated as the maximum amount by which the actual vehicle speed exceeds the reference:

$$M_{\mathrm{os}} = \max_k \left[\max\left(v_k - v_{\mathrm{ref},k}, 0\right)\right]. \tag{3.81}$$

The total time spent above the reference velocity is computed as

$$t_{\mathrm{os}} = T_s \sum_{k=1}^{N} \mathbb{I}\left(v_k > v_{\mathrm{ref},k}\right), \tag{3.82}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

The percentage of time during which the assistance is active is defined as

$$P_{\mathrm{assist}} = \frac{100}{N}\sum_{k=1}^{N} \lambda_{\mathrm{assist},k}. \tag{3.83}$$

For the lead-brake scenario, the minimum measured gap is

$$g_{min,measured} = \min_k g_k, \tag{3.84}$$

while the time spent below the required safety gap is

$$t_{\mathrm{unsafe}} = T_s \sum_{k=1}^{N} \mathbb{I}\left(g_k < g_{\mathrm{req},k}\right). \tag{3.85}$$

4. Simulation scenarios and performance evaluation

This chapter presents a set of simulation cases involving different patient profiles, reference velocity trajectories and operating scenarios. The main objectives of the analysis are:

- to compare the tracking behaviour of the predefined patient profiles when following different reference velocity trajectories;
- to quantify the tracking performance through the selected error, overshoot, settling-time and assistance-intervention metrics;
- to evaluate the contribution of the assist controller and determine how frequently external support is required;
- to analyse the response of the patient and assist controllers under a critical condition, represented by the sudden braking of a lead vehicle;
- to assess whether the simulated longitudinal behaviour satisfies the tracking and safety requirements defined for each scenario.

The ON/OFF comparisons illustrate selected profile–reference pairs: delayed with the trapezoidal reference, cautious with the sinusoidal reference, normal with the step reference, and reactive with the exponential reference. The delayed profile is also examined with assistance in the lead-brake scenario because it collides in the unassisted case. These cases cover different baseline behaviours and task demands; they do not constitute a complete factorial comparison. Benefits must therefore be assessed within each pair, rather than ranked across different profiles and references.

All speed and speed-error plots and tables use SI units (m/s); assistance thresholds are expressed in the same units.

The predefined profiles should be interpreted as engineering parameterisations of different levels of responsiveness, delay and comfort constraints. They are not intended to represent clinically validated patient conditions.

4.1 Base scenario with trapezoidal reference

4.1.1 All patient profiles - assist OFF

Without assistance, RMSE ranges from 0.393 m/s for the reactive profile to 2.611 m/s for the delayed profile. The delayed profile also has the largest overshoot, 4.580 m/s, identifying a critical baseline for the subsequent ON/OFF comparison.

Objective: Evaluate the ability of different patient profiles to follow a simple reference composed of acceleration, cruise and final deceleration.

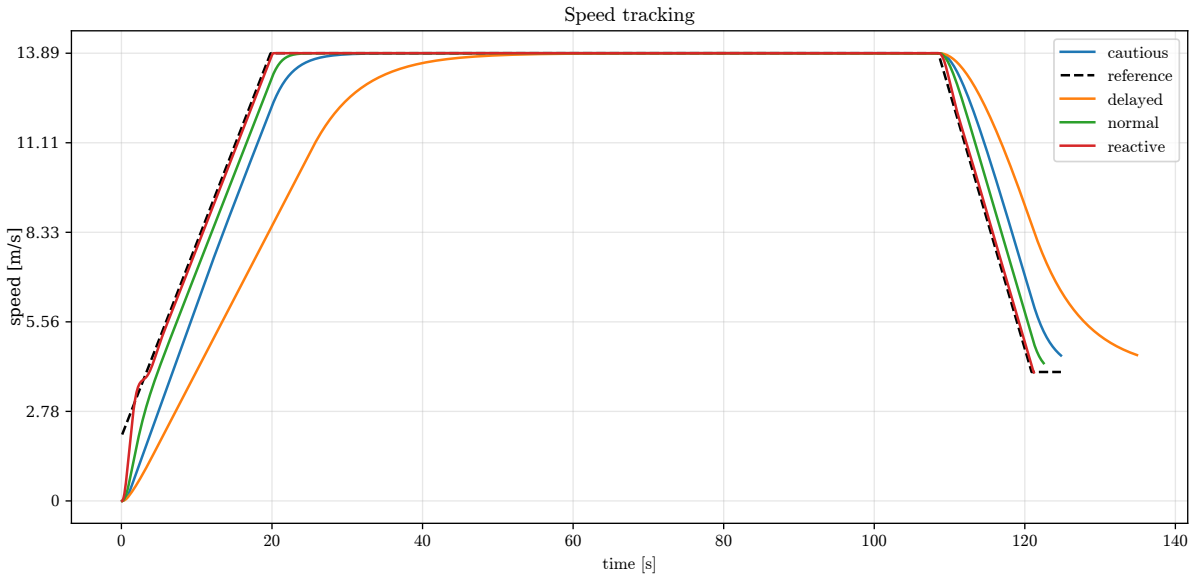


Figure 4.1: Speed tracking related to all patient profiles - trapezoidal reference, with assistance disabled.

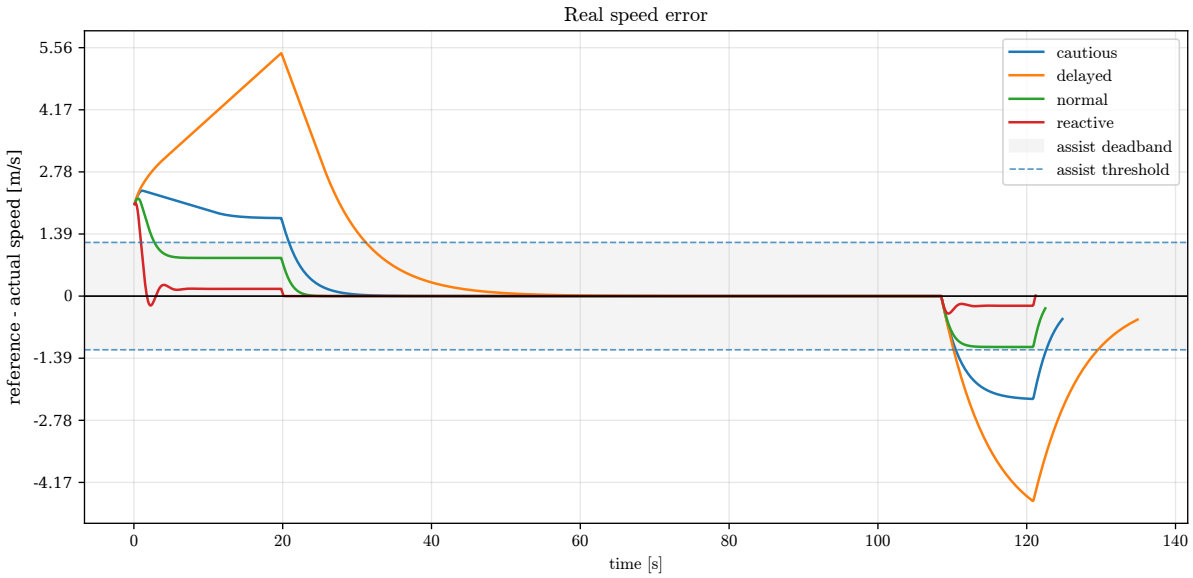


Figure 4.2: Tracking error related to all patient profiles - trapezoidal reference, with assistance disabled.

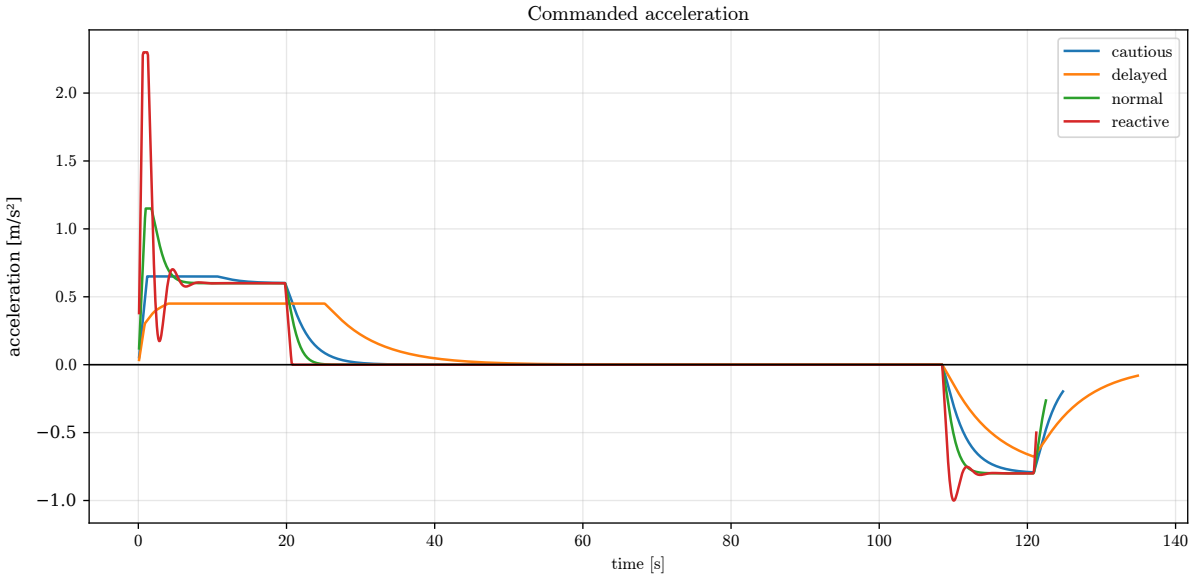


Figure 4.3: Acceleration command - trapezoidal reference, with assistance disabled.

Metric	Normal	Cautious	Reactive	Delayed
Duration [s]	122.5	124.8	121.2	134.9
RMSE [m/s]	0.832	1.402	0.393	2.611
MAE [m/s]	0.434	0.772	0.179	1.633
Max error [m/s]	2.660	3.170	2.442	6.153
Max overshoot [m/s]	1.138	2.301	0.390	4.580
Overshoot time [s]	14.0	16.3	13.8	26.4

Table 4.1: Summary metrics for the base scenario with trapezoidal reference and assistance disabled.

4.1.2 Focus on delayed profile - assist ON/OFF

The delayed profile RMSE falls from 2.611 m/s to 0.651 m/s (75.1%), while maximum overshoot decreases from 4.580 m/s to 1.246 m/s. Assistance is active for 12.9% of the run, demonstrating a substantial benefit with intermittent support.

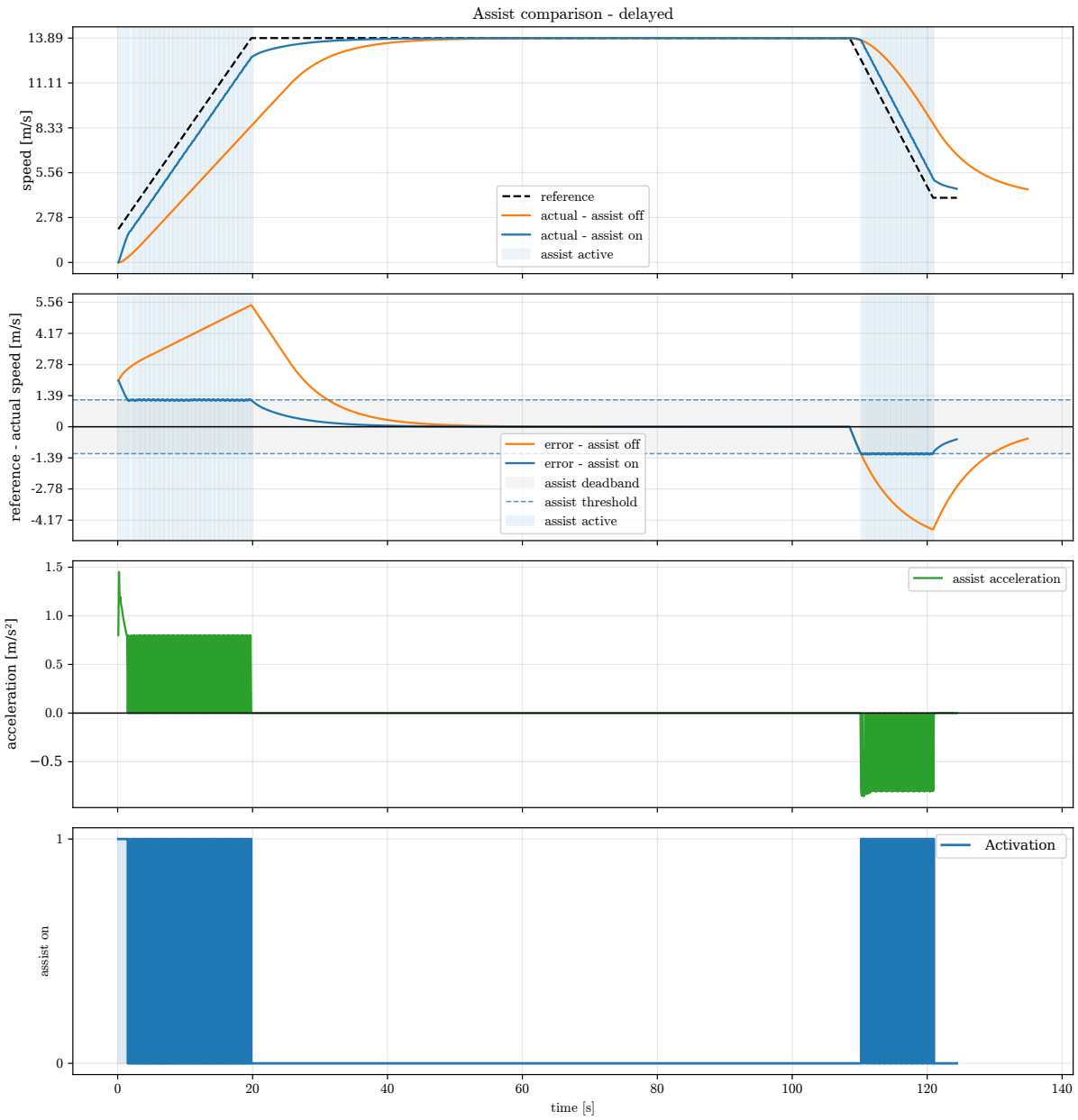


Figure 4.4: Assist ON/OFF comparison and assist intervention logic - trapezoidal reference, delayed profile.

Metric	Assist OFF	Assist ON	Improvement
Duration [s]	134.9	124.4	7.8%
RMSE [m/s]	2.611	0.651	75.1%
MAE [m/s]	1.633	0.386	76.3%
Max absolute error [m/s]	6.153	2.060	66.5%
Max overshoot [m/s]	4.580	1.246	72.8%
Overshoot time [s]	26.4	15.9	39.8%
Assist intervention [%]	0.0	12.9	–

Table 4.2: Assist OFF/ON comparison for the delayed patient profile in the base scenario with trapezoidal reference.

4.2 Base scenario with sinusoidal reference

Objective: Evaluate the ability of different patient profiles to follow a sinusoidal velocity reference, in order to test the patient's response to periodic accelerations and decelerations.

4.2.1 All patient profiles - assist OFF

The reactive profile has the lowest RMSE (0.720 m/s), while the cautious and delayed profiles reach 3.305 m/s and 4.387 m/s, respectively. Repeated speed changes reveal the effect of limited responsiveness even with a smooth reference.

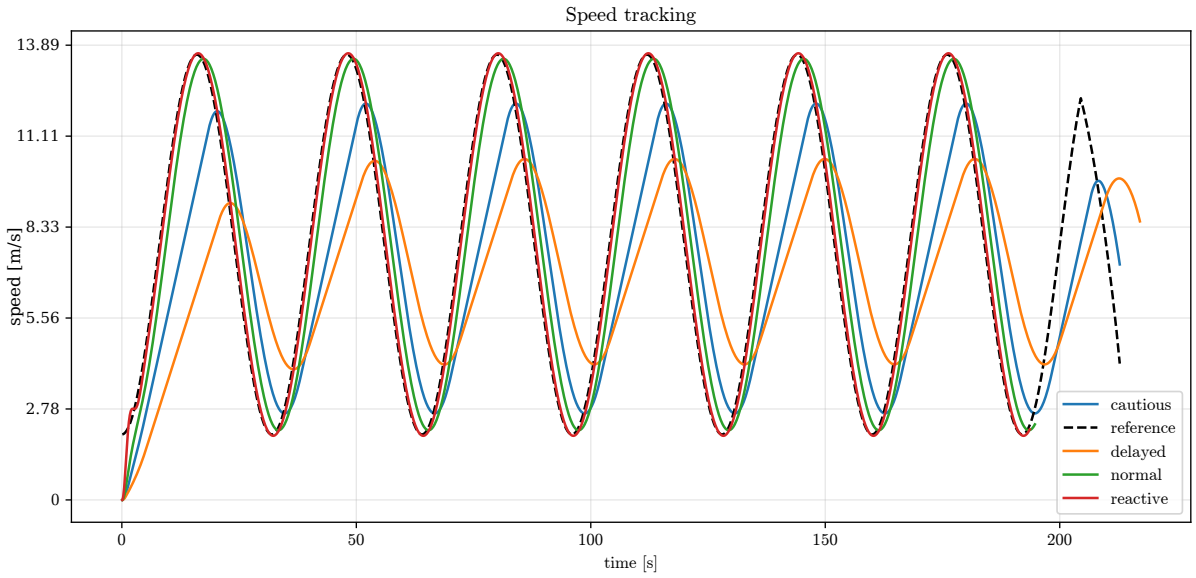


Figure 4.5: Speed tracking related to all patient profiles - sinusoidal reference, with assistance disabled.

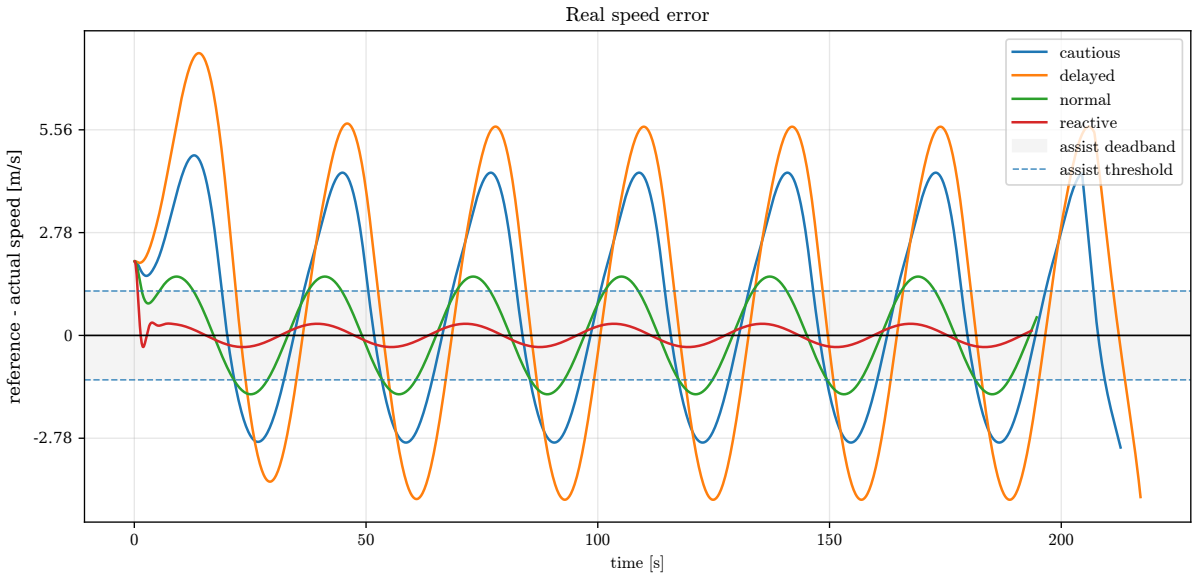


Figure 4.6: Tracking error related to all patient profiles - sinusoidal reference, with assistance disabled.

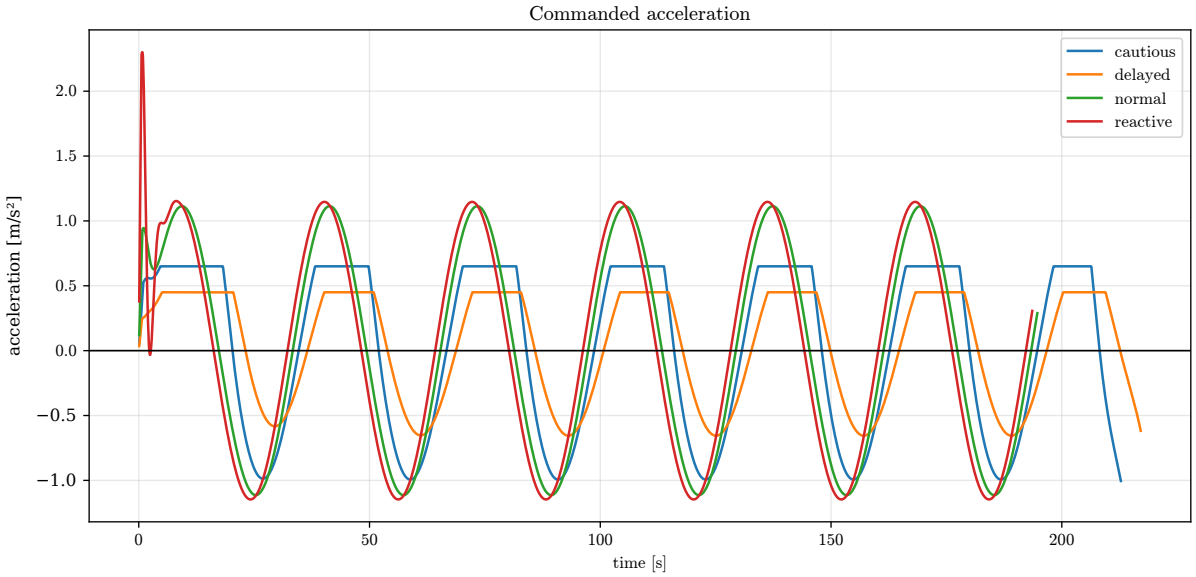


Figure 4.7: Acceleration command - sinusoidal reference, with assistance disabled.

Metric	Normal	Cautious	Reactive	Delayed
Duration [s]	194.7	212.8	193.6	217.1
RMSE [m/s]	1.755	3.305	0.720	4.387
MAE [m/s]	1.584	2.972	0.642	3.924
Max error [m/s]	2.475	5.581	2.055	8.348
Max overshoot [m/s]	1.590	3.029	0.315	4.441
Overshoot time [s]	96.0	92.8	97.3	92.7

Table 4.3: Summary metrics for the base scenario with sinusoidal reference and assistance disabled.

4.2.2 Focus on cautious profile - assist ON/OFF

For the cautious profile, RMSE decreases from 3.305 m/s to 1.062 m/s and maximum overshoot from 3.029 m/s to 1.252 m/s, with assistance active for 29.7% of the run. Time above the reference increases from 92.8 s to 96.0 s; therefore improved error amplitude does not imply improvement in every metric.

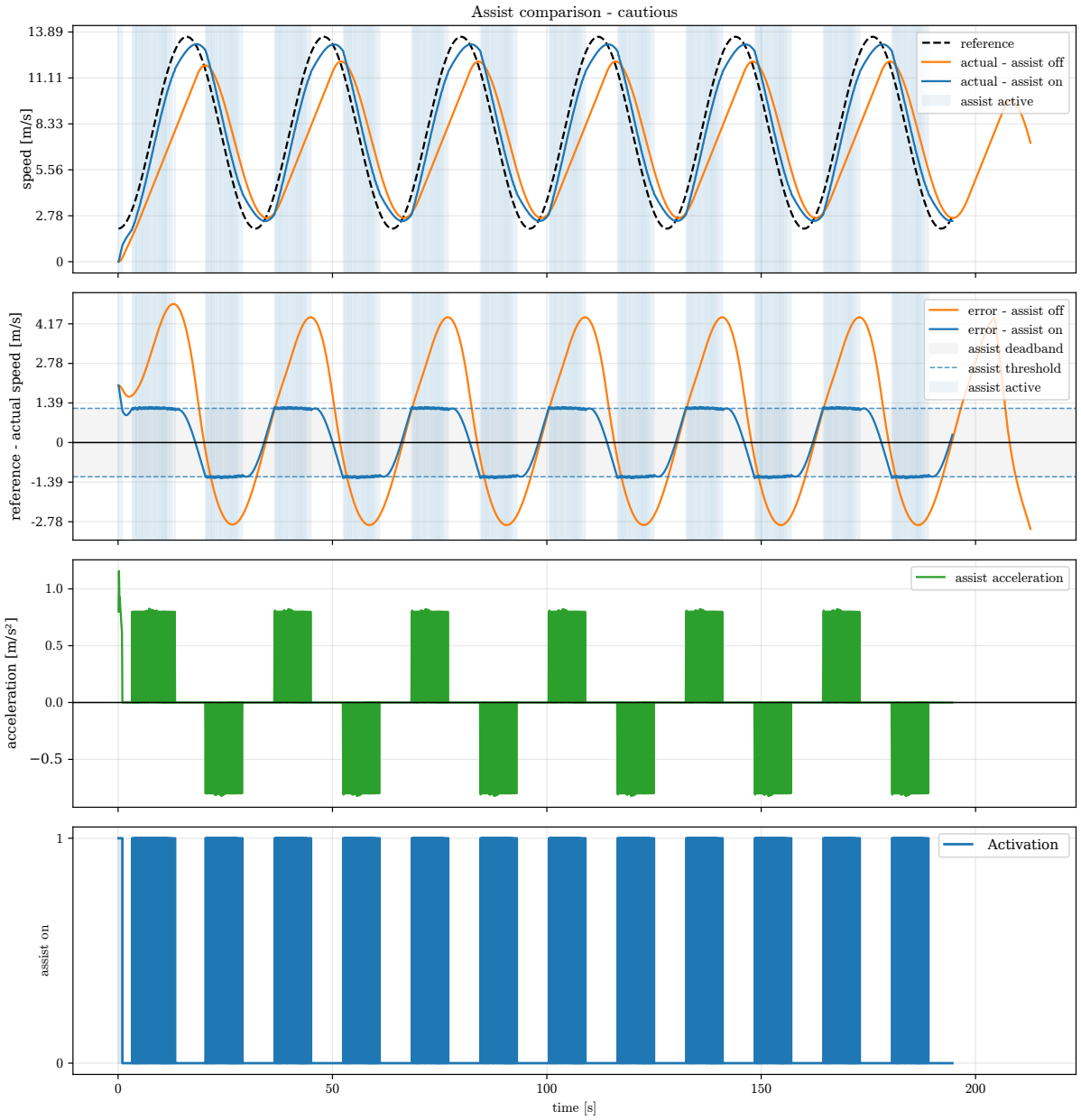


Figure 4.8: Assist ON/OFF comparison and assist intervention logic - sinusoidal reference, cautious profile.

Metric	**Assist OFF**	**Assist ON**	**Improvement**
Duration [s]	212.8	194.6	8.6%
RMSE [m/s]	3.305	1.062	67.9%
MAE [m/s]	2.972	1.008	66.1%
Max absolute error [m/s]	5.581	2.001	64.1%
Max overshoot [m/s]	3.029	1.252	58.7%
Overshoot time [s]	92.8	96.0	-3.4%
Assist intervention [%]	0.0	29.7	–

Table 4.4: Assist OFF/ON comparison for the cautious patient profile in the base scenario with sinusoidal reference.

4.3 Base scenario with step reference

Objective. Analyse maximum error, overshoot and settling time after abrupt reference changes.

4.3.1 All patient profiles - assist OFF

The step reference produces maximum errors above 10 m/s for all profiles, since speed cannot follow an instantaneous reference change. Settling time ranges from 46.9 s for the reactive profile to 62.6 s for the delayed profile.

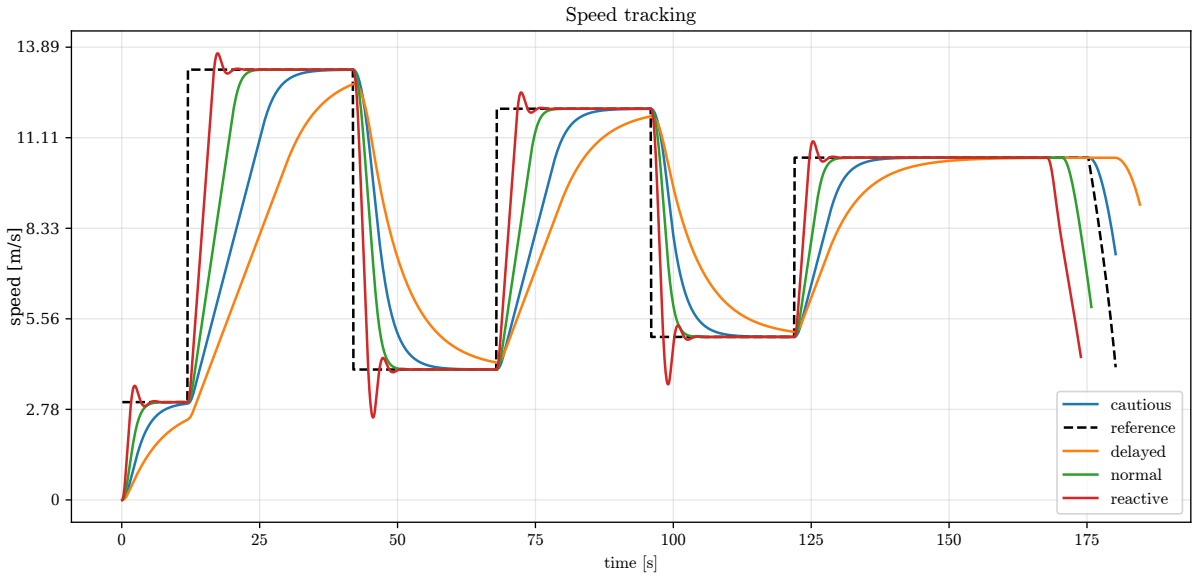


Figure 4.9: Speed tracking related to all patient profiles - step reference, with assistance disabled.

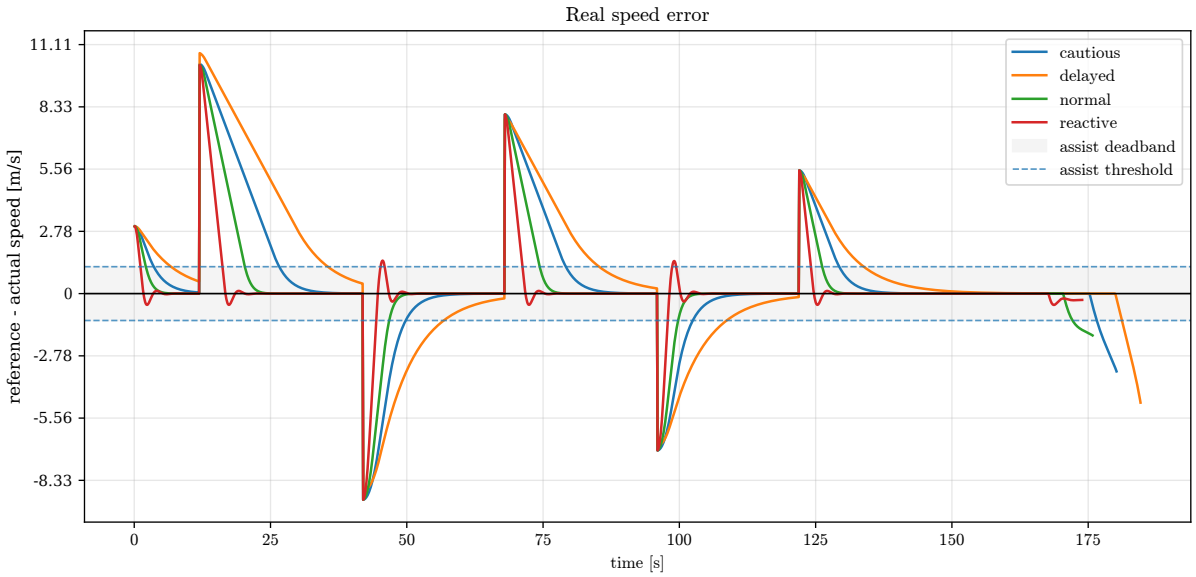


Figure 4.10: Tracking error related to all patient profiles - step reference, with assistance disabled.

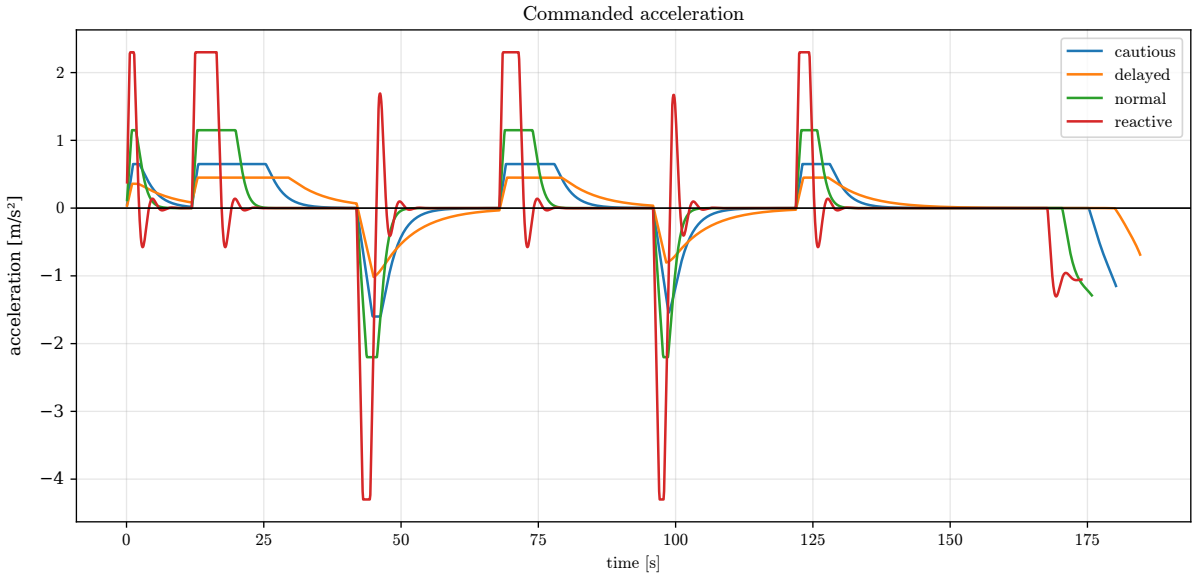


Figure 4.11: Acceleration command - step reference, with assistance disabled.

Metric	Normal	Cautious	Reactive	Delayed
Duration [s]	175.8	180.2	173.9	184.6
RMSE [m/s]	2.605	3.229	1.989	3.852
MAE [m/s]	1.123	1.770	0.667	2.631
Max error [m/s]	10.200	10.268	10.200	10.881
Max overshoot [m/s]	9.200	9.197	9.200	8.764
Overshoot time [s]	40.8	57.0	53.0	56.7
Settling time [s]	53.8	58.2	46.9	62.6

Table 4.5: Summary metrics for the base scenario with step reference and assistance disabled.

4.3.2 Focus on normal profile - assist ON/OFF

For the normal profile, RMSE decreases from 2.605 m/s to 1.469 m/s, and settling time decreases from 53.8 s to 31.0 s, with assistance active for 8.7% of the run. The maximum error and overshoot remain unchanged, consistent with abrupt reference changes and bounded acceleration.

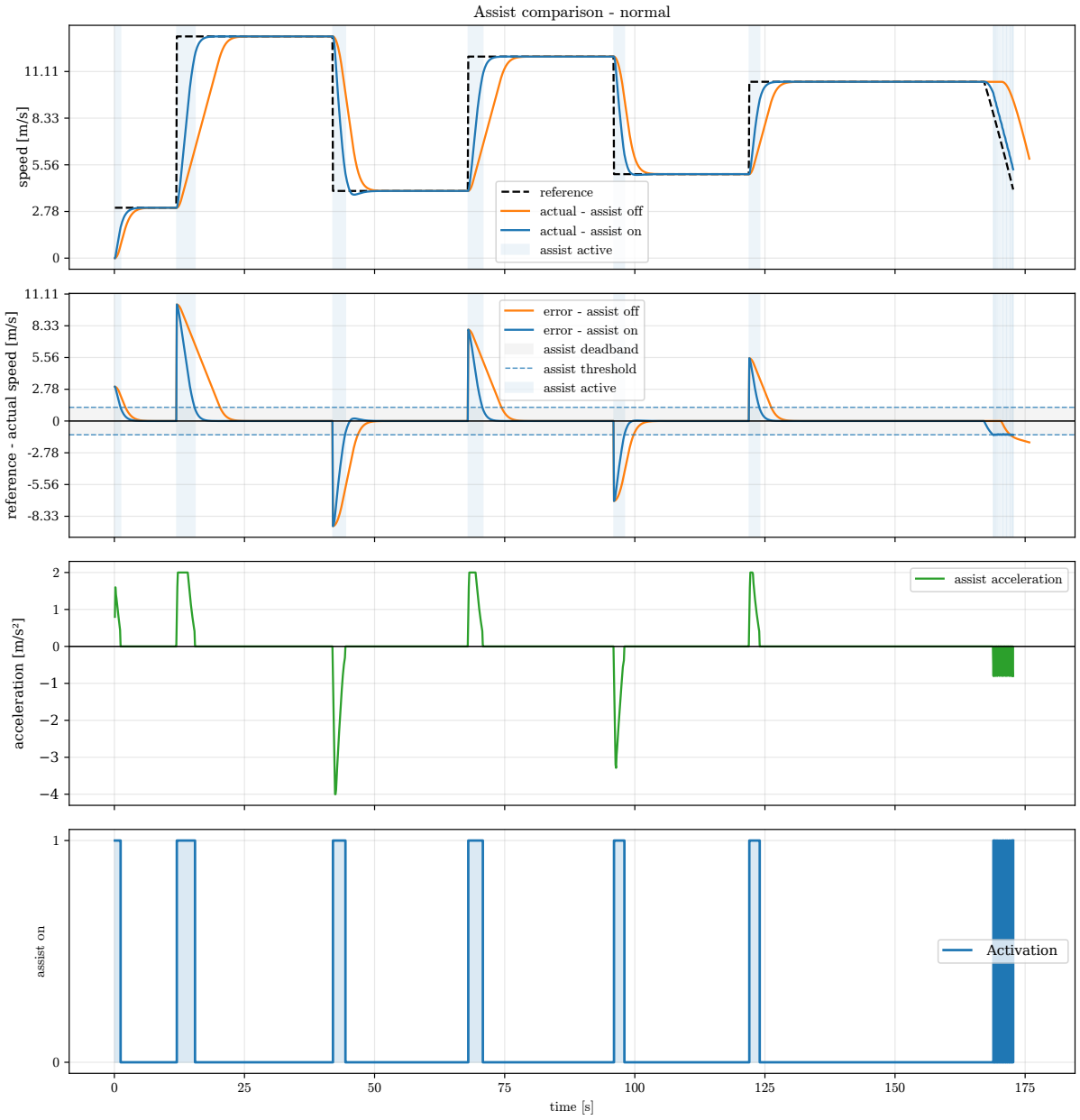


Figure 4.12: Assist ON/OFF comparison and assist intervention logic - step reference, normal profile.

Metric	Assist OFF	Assist ON	Improvement
Duration [s]	175.8	172.7	1.8%
RMSE [m/s]	2.605	1.469	43.6%
MAE [m/s]	1.123	0.424	62.2%
Max absolute error [m/s]	10.200	10.200	0.0%
Max overshoot [m/s]	9.200	9.200	0.0%
Overshoot time [s]	40.8	37.3	8.6%
Settling time [s]	53.8	31.0	42.4%
Assist intervention [%]	0.0	8.7	–

Table 4.6: Assist OFF/ON comparison for the normal patient profile in the base scenario with step reference.

4.4 Base scenario with exponential stop-and-go reference

Objective: Evaluate the response of the patient profiles to a stop-and-go speed demand characterised by smooth acceleration and deceleration transitions. This scenario is used to test the ability of the microscopic controller to follow repeated speed changes without the discontinuities introduced by the step reference.

4.4.1 All patient profiles - assist OFF

The smoother stop-and-go reference yields RMSE values from 0.489 m/s for the reactive profile to 2.694 m/s for the delayed profile. The delayed profile also exhibits the largest overshoot (5.484 m/s), showing that smooth transitions alone do not remove patient limitations.

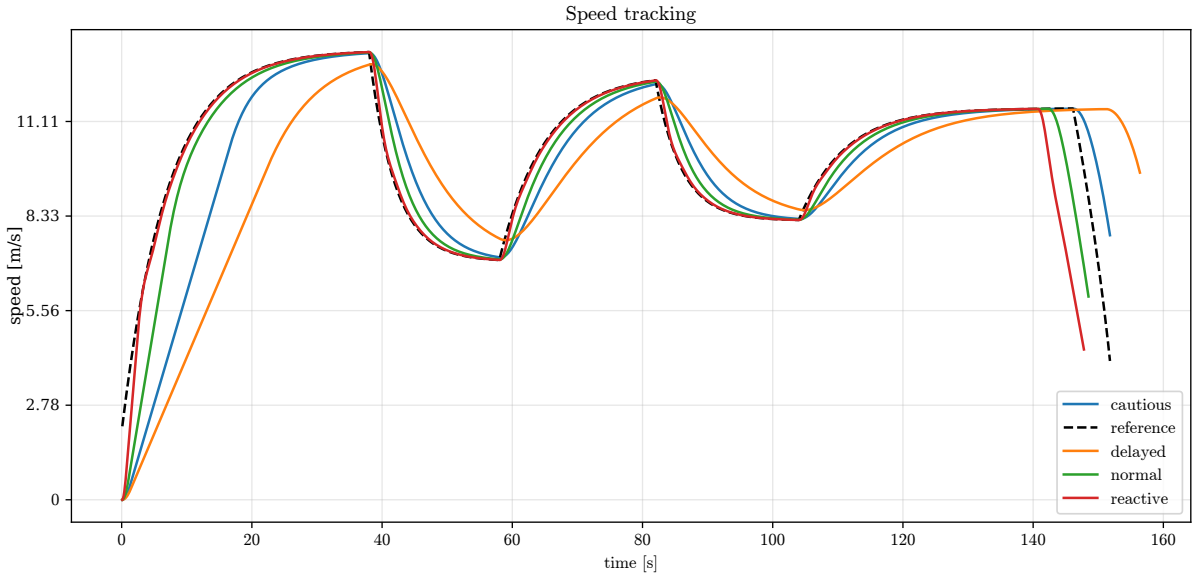


Figure 4.13: Speed tracking related to all patient profiles - exponential stop-and-go reference, with assistance disabled.

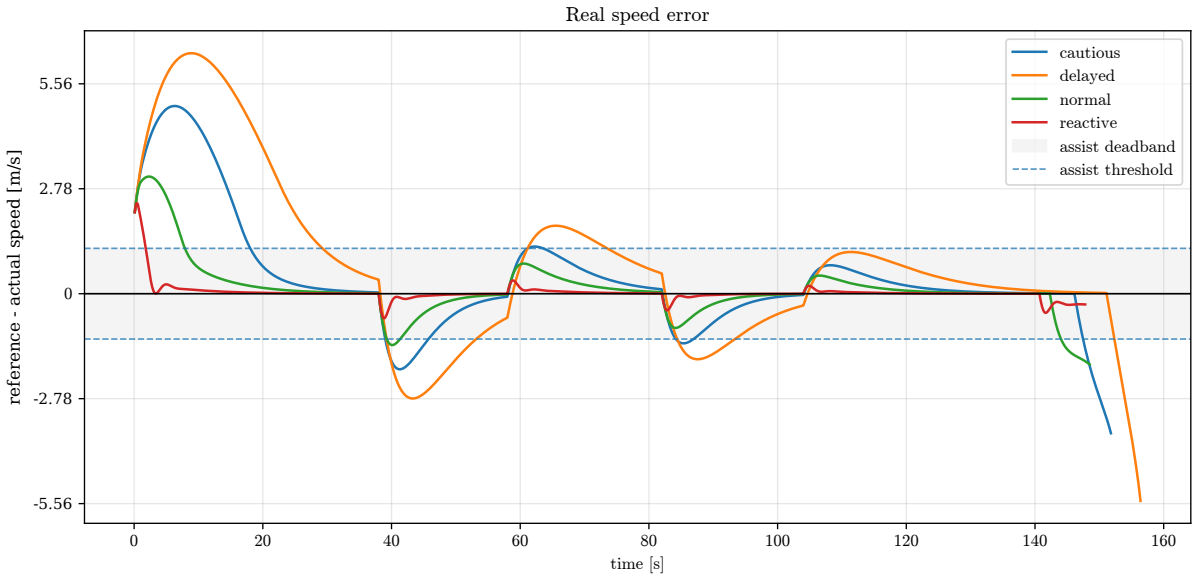


Figure 4.14: Tracking error related to all patient profiles - exponential stop-and-go reference, with assistance disabled.

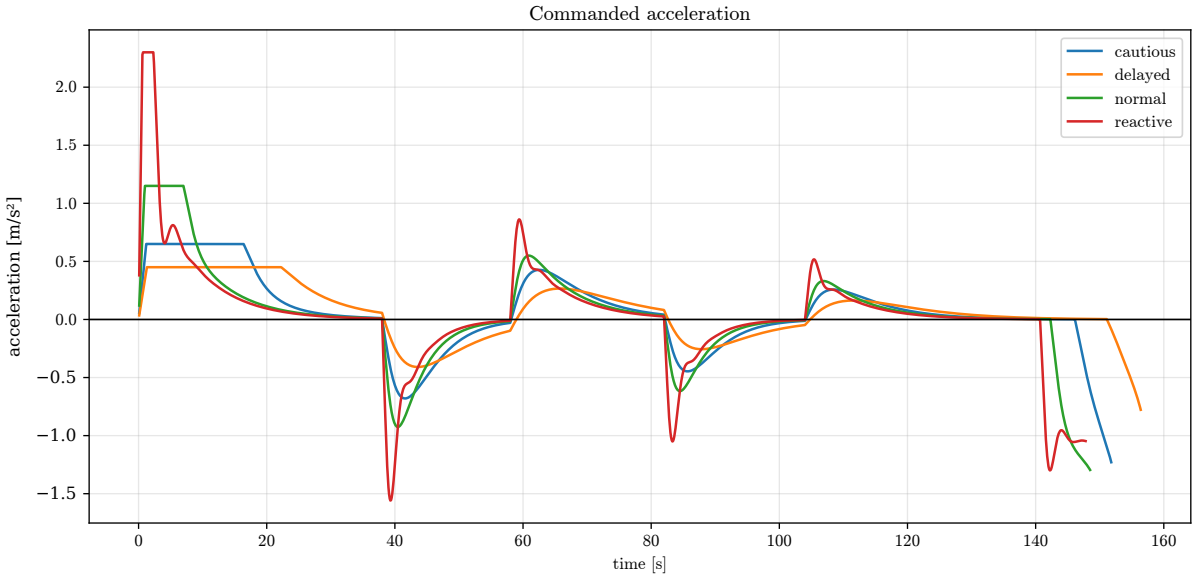


Figure 4.15: Acceleration command - exponential stop-and-go reference, with assistance disabled.

Metric	Normal	Cautious	Reactive	Delayed
Duration [s]	148.5	151.8	147.8	156.4
RMSE [m/s]	1.061	1.867	0.489	2.694
MAE [m/s]	0.611	1.157	0.245	1.924
Max error [m/s]	4.022	5.683	3.263	7.086
Max overshoot [m/s]	1.873	3.694	0.651	5.484
Overshoot time [s]	48.2	47.6	49.2	47.9

Table 4.7: Summary metrics for the base scenario with exponential reference and assistance disabled.

4.4.2 Focus on reactive profile - assist ON/OFF

The reactive profile already has an unassisted RMSE of 0.489 m/s; assistance reduces it to 0.199 m/s while intervening for only 0.6% of the run. Maximum overshoot and time above the reference remain unchanged, so the benefit concerns selected tracking-error measures rather than all response characteristics.

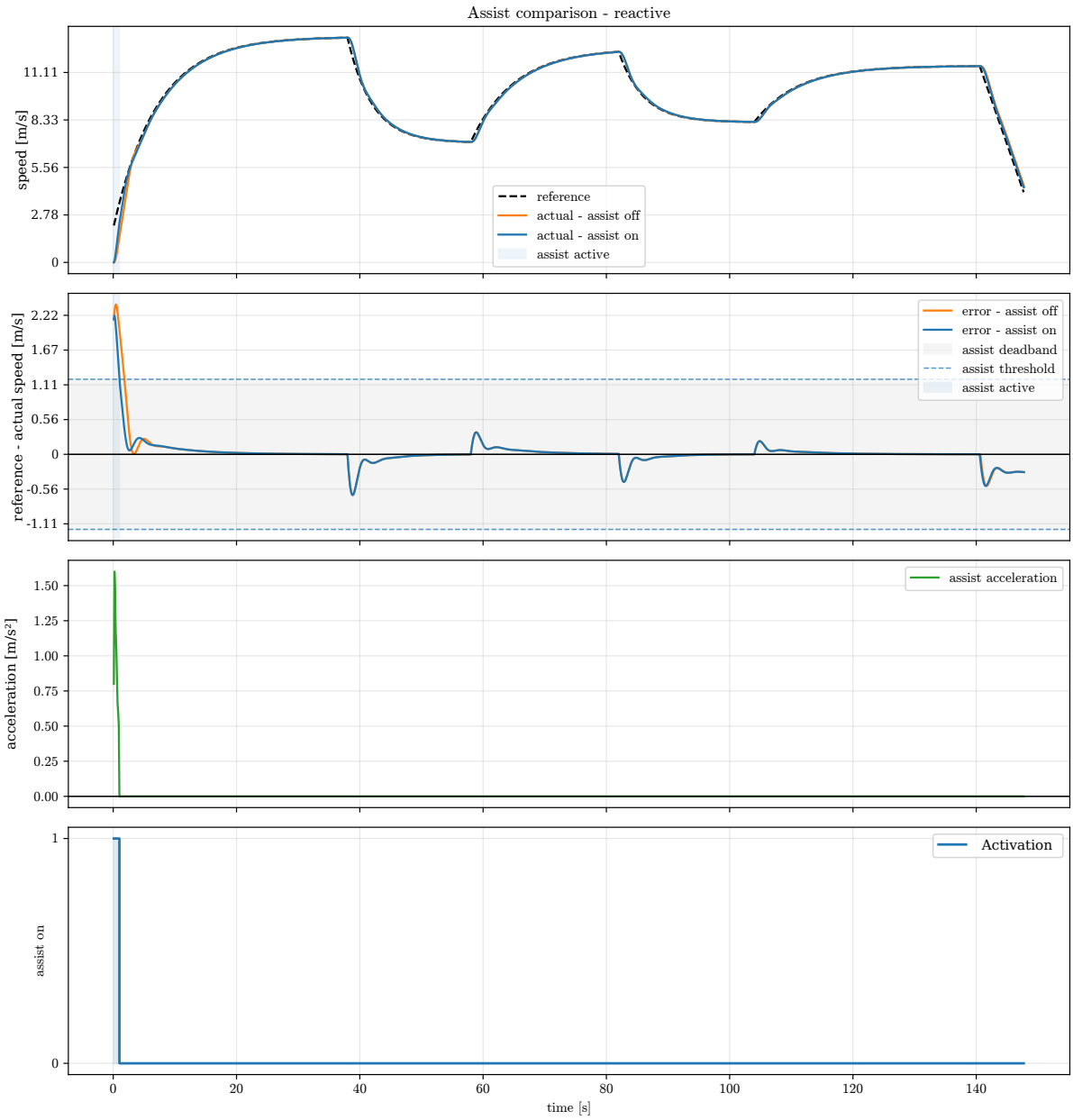


Figure 4.16: Assist ON/OFF comparison and assist intervention logic - exponential stop-and-go, reactive profile.

Metric	Assist OFF	Assist ON	Improvement
Duration [s]	147.8	147.7	0.1%
RMSE [m/s]	0.489	0.199	59.3%
MAE [m/s]	0.245	0.078	68.1%
Max absolute error [m/s]	3.263	2.215	32.1%
Max overshoot [m/s]	0.651	0.651	0.0%
Overshoot time [s]	49.2	49.2	0.0%
Assist intervention [%]	0.0	0.6	–

Table 4.8: Assist OFF/ON comparison for the reactive patient profile in the base scenario with exponential reference.

4.5 Lead-brake scenario with trapezoidal reference

Objective. Compare the unassisted behaviour of the patient profiles when a lead vehicle brakes ahead of the EGO vehicle. This test is used to identify the most critical profile in terms of both tracking performance and longitudinal safety, considering metrics such as the minimum lead gap and the time spent below the required safety distance.

4.5.1 All patient profiles - assist OFF

The reactive, normal and cautious profiles remain collision-free, with minimum gaps of 70.749 m, 41.800 m and 13.847 m. The delayed profile collides at 60.1 s, after 1.3 s below the required safety gap; its terminated run should not be compared directly with complete runs.

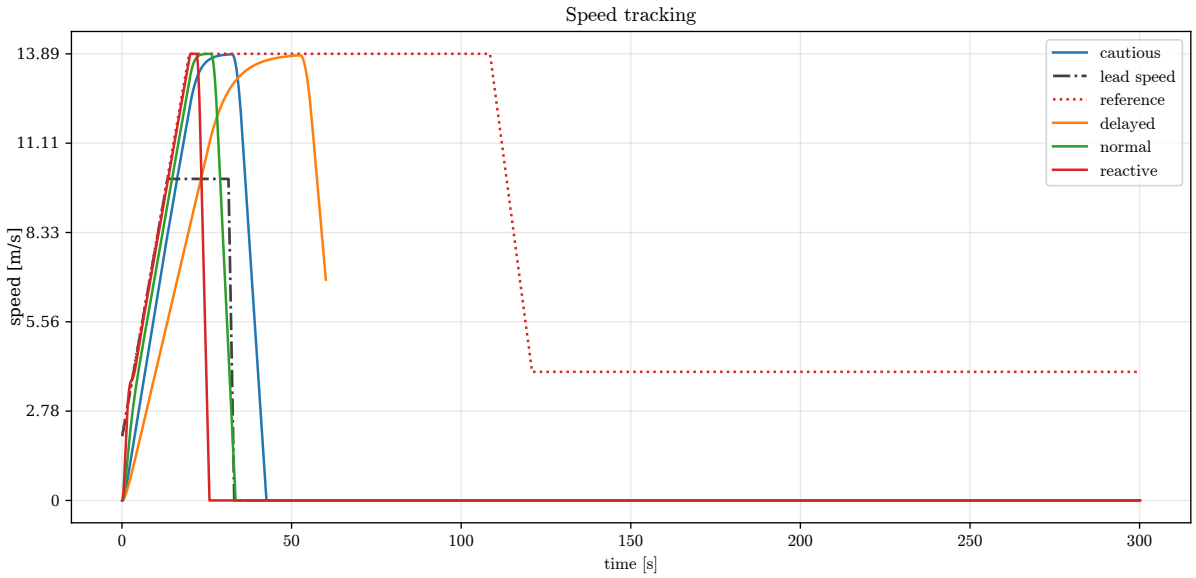


Figure 4.17: Speed tracking related to all patient profiles - trapezoidal reference with assist disabled - lead-brake scenario.

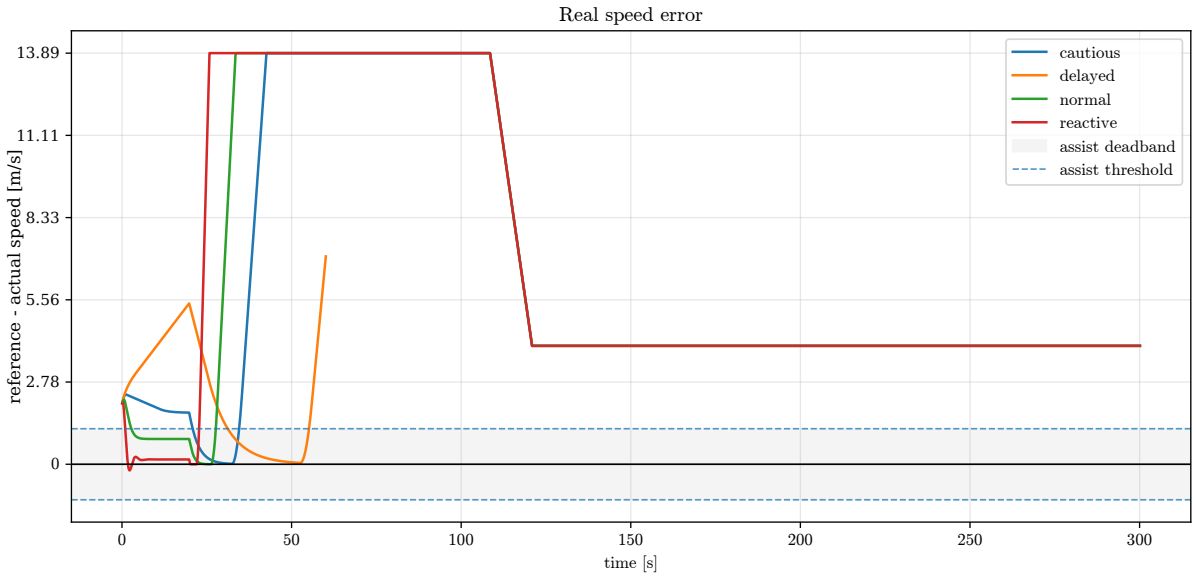


Figure 4.18: Tracking error related to all patient profiles - trapezoidal reference with assist disabled - lead-brake scenario.

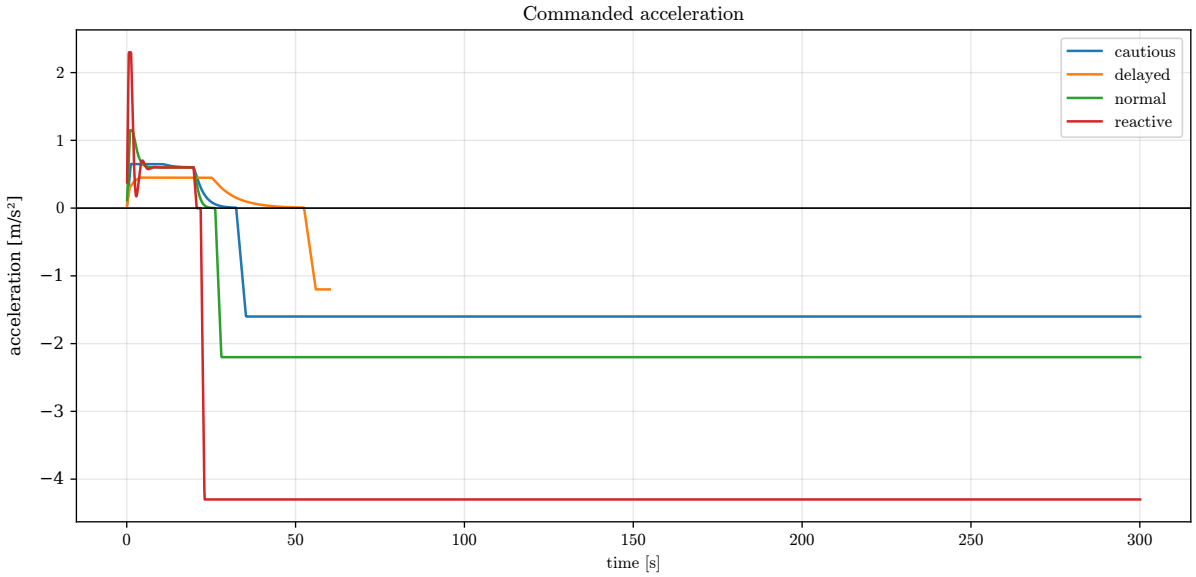


Figure 4.19: Acceleration command - trapezoidal reference with assist disabled - lead-brake scenario.

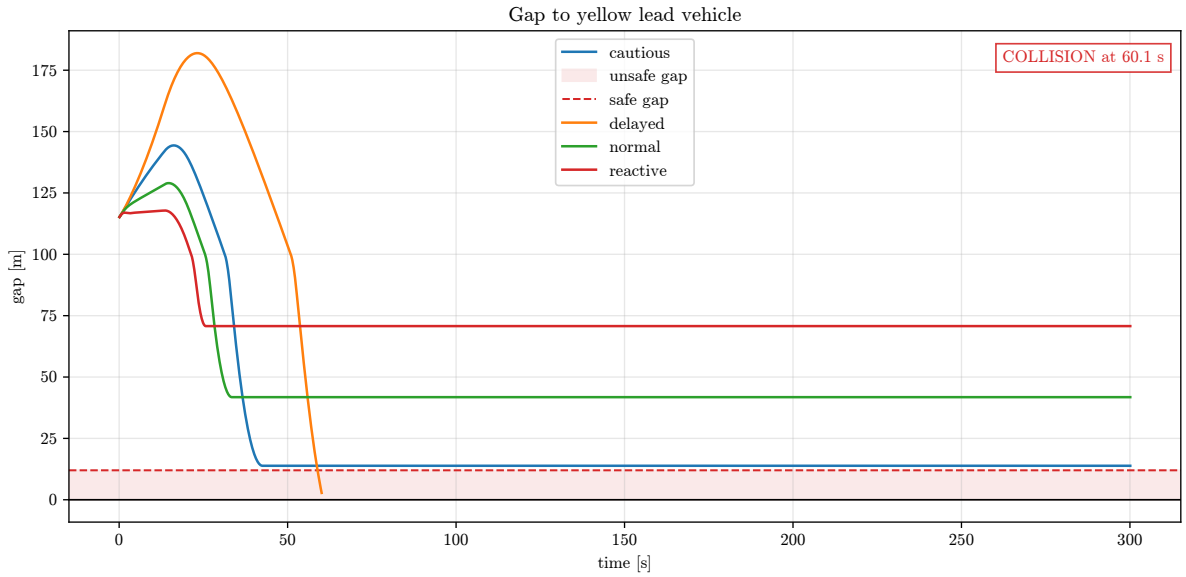
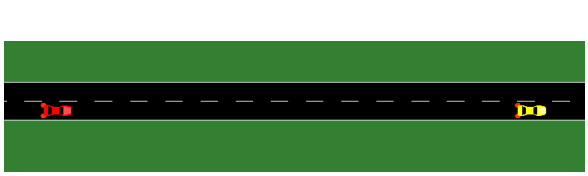
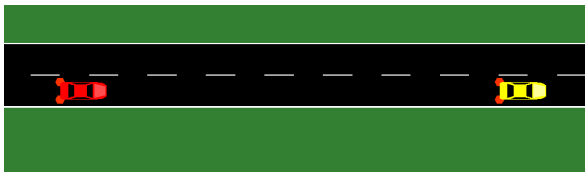


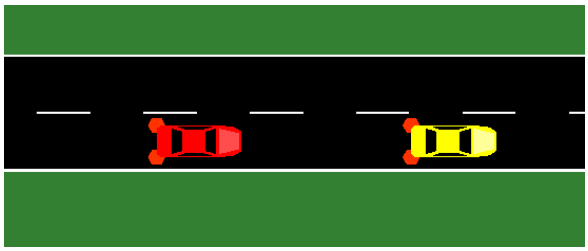
Figure 4.20: Gap to the lead vehicle for all four patient profiles with assistance disabled. The safe-gap boundary and unsafe region are indicated; the delayed-profile run terminates after a collision.



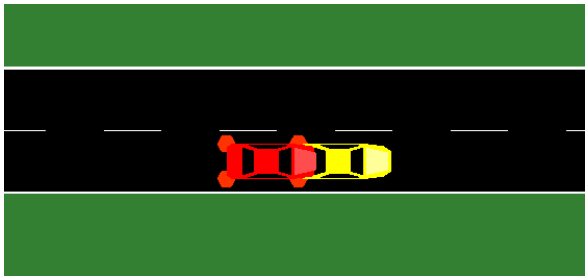
(a) Reactive profile



(b) Normal profile



(c) Cautious profile



(d) Delayed profile

Figure 4.21: Emergency braking lead-vehicle scenario for the four patient profiles - SUMO GUI view.

Metric	Reactive	Normal	Cautious	Delayed
Collision	No	No	No	Yes
Minimum lead gap [m]	70.749	41.800	13.847	–
Below safe gap	No	No	No	Yes
Time below safe gap [s]	0.0	0.0	0.0	1.3
Final position [m]	212.400	282.300	369.300	576.300
Simulation duration [s]	300.1	300.1	300.1	60.1

Table 4.9: Safety-oriented summary metrics for the emergency braking lead-vehicle scenario with assistance disabled.

4.5.2 Focus on delayed profile - assist ON

With assistance enabled, the delayed profile avoids the collision observed in the unassisted case. The minimum gap reported in Figure 4.24 is approximately 14.1 m; this positive separation is a result for this simulation and does not establish a general safety guarantee.

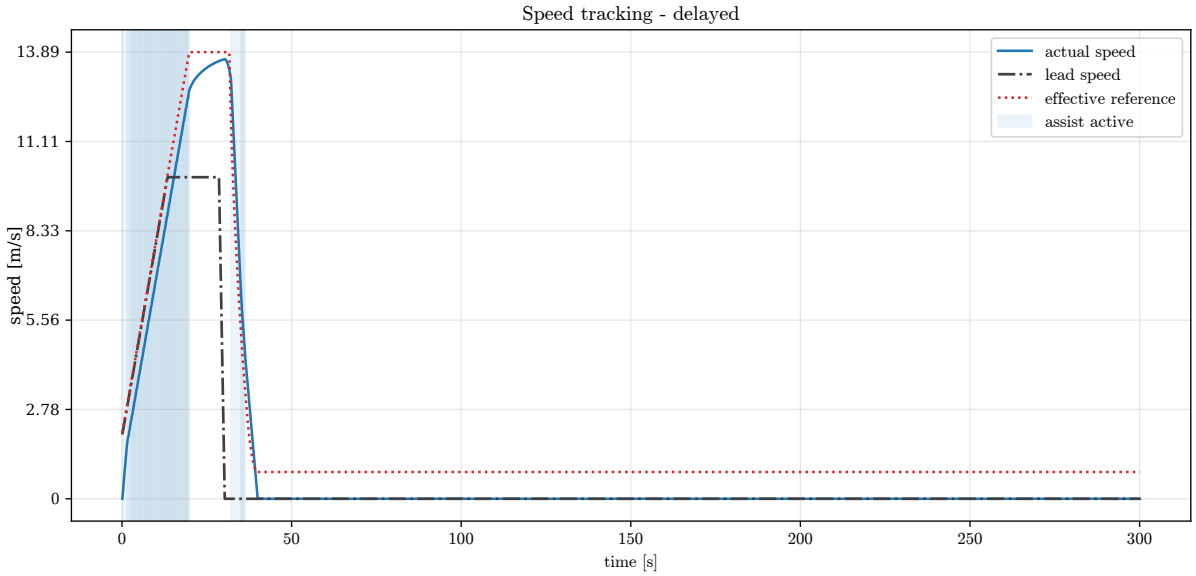


Figure 4.22: EGO, lead and effective-reference speeds for the delayed profile in the lead-brake scenario, with assistance enabled.

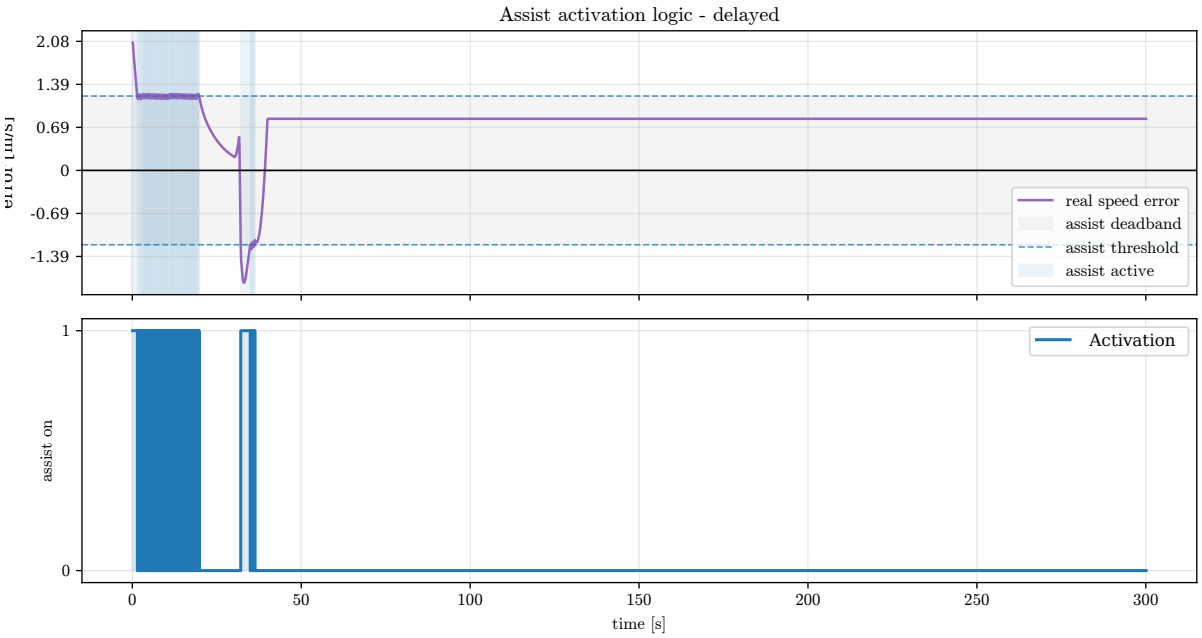


Figure 4.23: Tracking error and threshold-based assistance activation for the delayed profile in the lead-brake scenario, with assistance enabled.

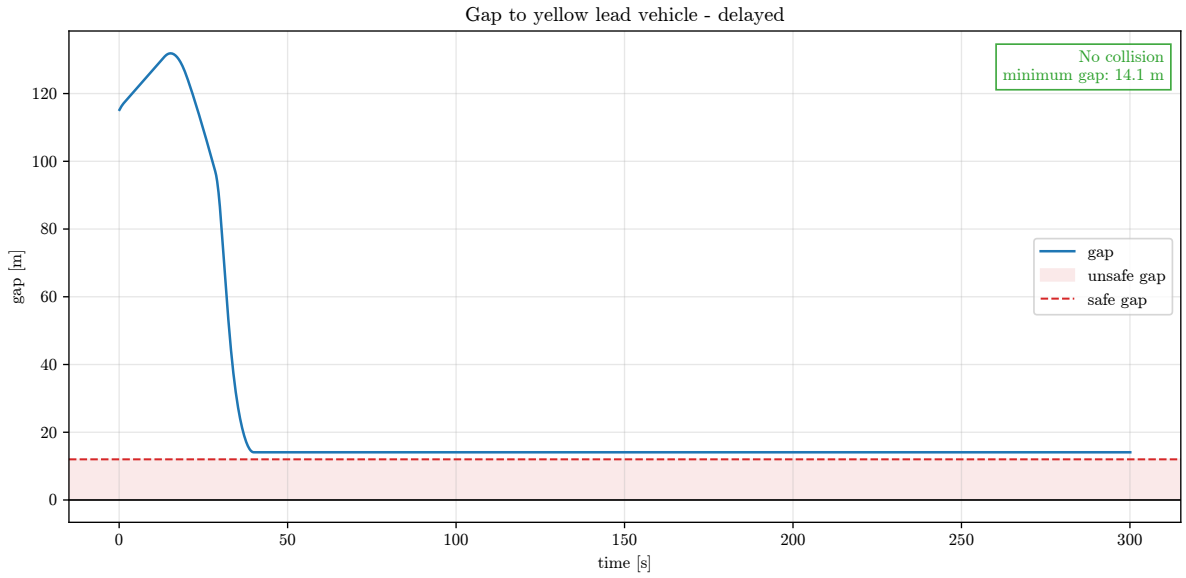


Figure 4.24: Lead-vehicle gap for the delayed profile with assistance enabled. The minimum recorded separation is approximately 14.1 m, and no collision occurs in this run.

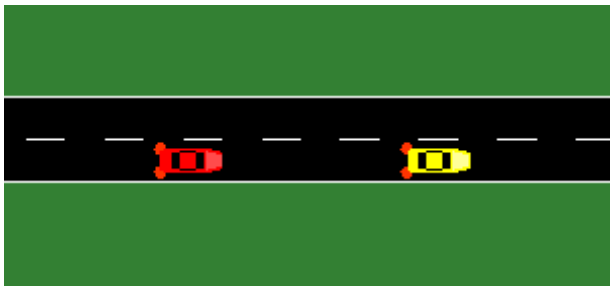


Figure 4.25: Emergency braking lead-vehicle scenario for delayed profile - assist ON.

Indicator	**Assist OFF**	**Assist ON**
Collision	Yes	No
Minimum lead gap [m]	–	≈ 14.1

Table 4.10: Safety comparison for the delayed profile in the lead-brake scenario. The unassisted run terminates following a collision.

4.6 Discussion of simulation results

The cases presented in the previous section are useful to evaluate and demonstrate the behaviour of the proposed framework.

In particular, starting from the results, it is possible to compare the response of different patient profiles, to analyse the influence of the reference speed profile, and to evaluate the effectiveness of the assist controller when the patient response is not sufficient.

The following subsections discuss patient parameters, reference demands, assistance benefits, the lead-brake case and the limitations of the framework. The overall findings are summarised in the Conclusions.

4.6.1 Different interpretations of the patient profiles

The comparison between the four patient profiles shows that the parameters used to describe the patient have a direct effect on the tracking performance.

The reactive profile generally provides the best tracking results, since it is characterised by a higher gain, a shorter reaction delay and less restrictive acceleration and jerk limits. As a consequence, this profile is able to react more promptly to changes in the reference speed.

The normal profile represents an intermediate condition. Its performance is worse than the reactive profile, but it remains acceptable in most of the tested scenarios. This behaviour is consistent with the idea of a patient who is able to follow the reference trajectory, but with a certain delay and with smoother acceleration commands.

The cautious and delayed profiles are more critical. The cautious profile reacts smoothly but less promptly, while the delayed profile combines a low gain with a large perception delay and restrictive acceleration limits. This produces larger tracking errors, longer manoeuvre durations and higher overshoot, especially when the reference speed changes rapidly.

Therefore, these profiles are useful to stress the control framework and to evaluate when the assist controller becomes necessary.

From another perspective, the profiles may also be interpreted as illustrative levels of longitudinal driving capability within the same modelling framework. In a future clinically validated implementation, variations in the identified parameters could potentially be used to describe changes in a patient's driving performance over the rehabilitation process. However, the predefined profiles adopted in this work are not intended to represent validated stages of clinical recovery.

As a hypothesis for future clinical investigation, changes in identified parameters could be associated with changes in an individual patient's driving ability during rehabilitation. The simulations do not demonstrate transitions between the predefined profiles or establish that they correspond to stages of recovery.

4.6.2 Interpretation of the reference speed profiles

The reference profiles impose different tracking demands: acceleration and cruising, repeated smooth variations, abrupt changes and stop-and-go motion. The selected ON/OFF cases illustrate assistance under these different demands; they do not establish a clinically validated correspondence between a task and a patient condition.

Indeed, the trapezoidal reference represents the simplest case, because it is composed of a smooth acceleration phase, a constant-speed phase and a final deceleration phase. In this scenario, the differences between patient profiles are clearly visible, but the tracking task remains relatively regular. The delayed profile is the most critical one, showing the largest tracking error and overshoot.

The sinusoidal reference is more demanding because the vehicle has to continuously accelerate and decelerate. In this case, the effect of patient delay becomes more evident. A delayed or cautious response introduces a phase lag with respect to the reference, increasing the tracking error even if the reference profile itself is smooth. This confirms that the performance of the patient model does not depend only on the amplitude of the reference speed, but also on how frequently the reference changes.

The step reference is the most severe tracking test. Since the reference speed changes abruptly, all patient profiles show high maximum errors and overshoot. This result is expected, because the vehicle speed cannot change instantaneously and the acceleration command is limited by physical and comfort-related constraints. Therefore, the step profile is useful as a stress test, but it is less representative of a comfortable real driving manoeuvre.

After the deliberately severe step-reference test, a smoother stop-and-go profile is considered to reproduce a more realistic urban driving pattern. It introduces repeated speed changes, but with smoother transitions than the step profile. For this reason, it is more realistic for urban stop-and-go behaviour. The results show that smoother references are easier to track and reduce the need for aggressive assistance.

4.6.3 Effectiveness of the assist controller

The assist controller improves the tracking performance especially when the patient profile is weak or delayed. This is evident in the trapezoidal case, where the delayed profile

benefits significantly from the activation of the assistance. The reduction of the root mean square error, mean absolute error and overshoot shows that the assist controller is able to compensate part of the patient delay and limited responsiveness.

An important aspect is that the assistance is not continuously active. Thanks to the threshold-based activation logic, the assist controller intervenes only when the real tracking error exceeds the predefined acceptable region. This makes the control strategy less invasive, because the patient remains the main controller when the tracking error is small.

The simulation results also show that the benefit of the assistance depends on the initial patient condition. For example, the reactive profile in the exponential-reference case requires assistance for only 0.6% of the run. Its RMSE decreases by 59.3%, although the absolute reduction is relatively small (0.290 m/s). This is a positive result, because it means that the assistance does not unnecessarily replace a patient who is already able to perform the task.

On the contrary, for cautious or delayed profiles, the assistance becomes more relevant. In these cases, the controller provides an additional corrective action that reduces the tracking error and improves the overall response. Therefore, the assist controller can be interpreted as a threshold-activated support mechanism: it is almost inactive for good driving performance, but becomes active when the patient response is not sufficient.

4.6.4 Analysis of the lead-brake scenario

The lead-brake scenario introduces an external disturbance and allows the safety behaviour of the microscopic framework to be evaluated. In this case, the EGO vehicle does not only have to follow the reference speed, but also has to react to the sudden braking of a vehicle ahead.

The results show that the reactive and normal profiles maintain a comfortable safety margin, while the cautious profile approaches the required safety boundary more closely without violating it. The delayed profile represents the critical case, leading to a collision when assistance is disabled. This result is consistent with the structure of the model: a large perception delay and restrictive braking capability reduce the ability of the patient to react promptly to the lead vehicle deceleration.

This scenario is particularly important because it shows that good speed tracking is not the only objective of the control framework. In a realistic driving situation, safety constraints must also be considered. A patient may be able to follow a reference speed in a simple base scenario, but may still become unsafe when an unexpected event occurs. For this reason, the lead-brake scenario provides useful information about the limits of each patient profile.

4.6.5 Strengths, limitations and possible extensions

The framework connects the discrete-time model to its Python implementation and uses SUMO to visualise and record vehicle motion. Different patient behaviours can be compared within the same architecture by varying gains, delay and acceleration or jerk limits.

The main limitation is the simplified vehicle model: a speed integrator omits mass, aerodynamic drag, rolling resistance, powertrain dynamics, actuator delay and tyre–road interaction. The results therefore concern high-level longitudinal control under the stated assumptions.

A second limitation is related to the patient model. The predefined patient profiles are useful for simulation purposes, but they are not derived from experimental clinical data. Therefore, they should be interpreted as representative behaviours rather than validated medical profiles.

Indeed, a first possible improvement can consist in an integration of clinical-medical data with the control solutions. The questionnaire-based procedure introduced in Section 3.4.3 provides a preliminary method for selecting patient parameters, but it should currently be regarded only as a mock-up.

The assist controller is implemented as a PID controller with threshold-based activation. A possible extension is a microscopic MPC that predicts vehicle motion while considering tracking, comfort and safety constraints. Its benefit would need to be measured against the current PID baseline using the same scenarios, together with solver execution time, the ability to complete optimisation within the sampling interval of 0.1 s, implementation complexity and computational hardware requirements. No such comparison is performed here.

Furthermore, the simulated environment is kept intentionally simple. The road is straight and the number of surrounding vehicles is limited. Future developments could introduce background traffic into the microscopic longitudinal-control experiments, which currently use only the EGO vehicle or an EGO–lead pair. Background traffic is already present in the traffic-level framework described in Chapter 2. Further vehicle-level tests could consider multiple lead vehicles, urban intersections, traffic lights, lane changes and variable speed limits.

Finally, a relevant future improvement would be the integration of the proposed microscopic control framework with the macroscopic MPC-based controller developed in parallel by A. Galassi [13]. The macroscopic controller is designed to generate optimal vehicle speed profiles according to traffic-level objectives. These profiles could replace the predefined reference trajectories currently used in the microscopic simulation, providing the EGO vehicle with dynamically optimised speed references.

Figure 4.26 illustrates the proposed integration through a conceptual scheme of the nested

traffic-level and vehicle-level control loops.

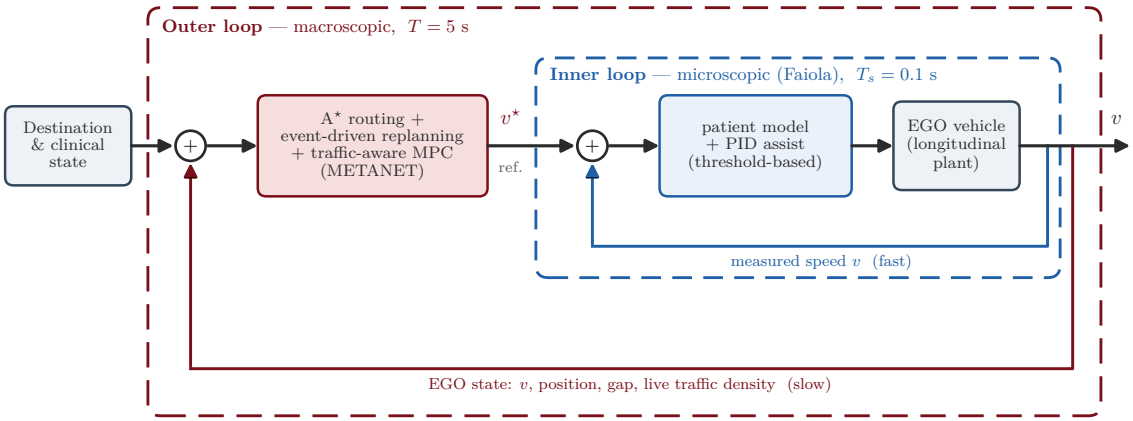


Figure 4.26: Conceptual nested-loop architecture for the future integration of traffic-level MPC and microscopic longitudinal control. Diagram produced by A. Galassi.

Conclusions

The work developed in this thesis was motivated by a gap that emerged from the analysis of the literature. Driving rehabilitation, human-driver modelling, shared-control strategies and traffic simulation are often investigated as separate topics, while only limited attention is devoted to frameworks in which patient-specific limitations, assistive control and traffic interaction are considered together.

The proposed work was therefore aimed at providing a first integrated engineering framework in which the patient remains an active component of the driving task, while automated assistance is introduced only when the patient's response becomes insufficient. From this point of view, the obtained results are coherent with the original objective of the thesis.

The simulation campaign showed that the selected patient parameters are able to generate clearly different longitudinal behaviours. More responsive profiles are generally capable of following the reference with limited error, whereas cautious and delayed profiles show larger tracking errors and a greater difficulty in reacting to variations in the driving task. This confirms that the framework is able to reproduce different levels of driving capability without changing the overall control architecture.

The assist-as-needed strategy also produced encouraging results. The benefit of the assistance was found to depend on the initial patient condition: when the patient was already able to perform the task satisfactorily, the assist controller remained almost inactive, while its contribution became more relevant for the more critical profiles. In the delayed case, for example, significant reductions in tracking error were obtained without requiring continuous intervention. This behaviour is consistent with the rehabilitation objective of supporting the patient rather than completely replacing his or her action.

The critical lead-brake scenario further showed that the evaluation of the system cannot be limited to reference tracking alone. A patient may exhibit an acceptable response under nominal conditions and still become unsafe when an unexpected event occurs. The collision observed for the delayed profile in the unassisted case highlights the importance of considering responsiveness, safety and assistance requirements together within the same simulation environment.

Overall, the results suggest that the proposed framework represents a useful proof of

concept for the study of personalised longitudinal assistance in a rehabilitation-oriented driving context. Its main value does not lie in the complexity of the individual control components, but in their integration within a common structure that makes it possible to compare patient conditions, driving tasks and assistance requirements in a repeatable way.

At the same time, the conclusions that can be drawn from the present work must be interpreted in light of its limitations. The vehicle dynamics are intentionally simplified, and the predefined patient profiles are engineering parameterisations rather than clinically validated models. Similarly, the questionnaire-based personalisation procedure should currently be regarded as a preliminary concept rather than a medical assessment tool. For these reasons, the numerical results should not be interpreted as direct predictions of the behaviour of real patients.

Nevertheless, within these assumptions, the developed framework achieved the main objective of the thesis: demonstrating that patient-specific behaviour, assist-as-needed longitudinal control and traffic simulation can be combined in a coherent and modular architecture. This provides a meaningful basis for more realistic and clinically grounded developments, while preserving the central idea of a rehabilitation system in which automation supports the patient only when necessary.

Bibliography

[1] H. M. Finestone, M. Guo, P. O'Hara, L. Greene-Finestone, S. C. Marshall, L. Hunt, J. Biggs, and A. Jessup, "Driving and reintegration into the community in patients after stroke," *PM&R*, vol. 2, no. 6, pp. 497–503, 2010.

[2] M. A. Hird, A. Vetivelu, G. Saposnik, and T. A. Schweizer, "Cognitive, on-road, and simulator-based driving assessment after stroke," *Journal of Stroke and Cerebrovascular Diseases*, vol. 23, no. 10, pp. 2654–2670, 2014.

[3] C. A. Unsworth and A. Baker, "Driver rehabilitation: A systematic review of the types and effectiveness of interventions used by occupational therapists to improve on-road fitness-to-drive," *Accident Analysis & Prevention*, vol. 71, pp. 106–114, 2014.

[4] K. Golisz, "Occupational therapy interventions to improve driving performance in older adults: A systematic review," *American Journal of Occupational Therapy*, vol. 68, no. 6, pp. 662–669, 2014.

[5] A. E. Akinwuntan, W. De Weerdt, H. Feys, J. Pauwels, G. Baten, P. Arno, and C. Kiekens, "Effect of simulator training on driving after stroke: A randomized controlled trial," *Neurology*, vol. 65, no. 6, pp. 843–850, 2005.

[6] H. Devos, A. E. Akinwuntan, A. Nieuwboer, M. Tant, S. Truijen, L. De Wit, C. Kiekens, and W. De Weerdt, "Comparison of the effect of two driving retraining programs on on-road performance after stroke," *Neurorehabilitation and Neural Repair*, vol. 23, no. 7, pp. 699–705, 2009.

[7] S. George, M. Crotty, I. Gélinas, and H. Devos, "Rehabilitation for improving automobile driving after stroke," *Cochrane Database of Systematic Reviews*, vol. 2014, no. 2, p. CD008357, 2014.

[8] M. Marcano, S. Díaz, J. Pérez, and E. Irigoyen, "A review of shared control for automated vehicles: Theory and applications," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 6, pp. 475–491, 2020.

[9] J. Sarabia, M. Marcano, J. Pérez, A. Zubizarreta, and S. Díaz, "A review of shared control in automated vehicles: System evaluation," *Frontiers in Control Engineering*, vol. 3, p. 1058923, 2023.

[10] C. J. Hasson, J. Manczurowsky, E. C. Collins, and M. Yarossi, "Neurorehabilitation robotics: How much control should therapists have?" *Frontiers in Human Neuroscience*, vol. 17, p. 1179418, 2023.

[11] A. A. Blank, J. A. French, A. U. Pehlivan, and M. K. O'Malley, "Current trends in robot-assisted upper-limb stroke rehabilitation: Promoting patient engagement in therapy," *Current Physical Medicine and Rehabilitation Reports*, vol. 2, no. 3, pp. 184–195, 2014.

[12] J. Xu, K. Kendrick, and A. R. Bowers, "Clinical report: Experiences of a driver with vision impairment when using a Tesla car," *Optometry and Vision Science*, vol. 99, no. 4, pp. 417–421, 2022.

[13] A. Galassi, "Traffic-simulated driving assistance in a rehabilitative framework," Master's thesis, University of Pavia, 2026.

[14] H. Devos, M. Tant, and A. E. Akinwuntan, "On-road driving impairments and associated cognitive deficits after stroke," *Cerebrovascular Diseases*, vol. 38, no. 3, pp. 226–232, 2014.

[15] N. Lodha, P. Patel, J. M. Shad, A. Casamento-Moran, and E. A. Christou, "Cognitive and motor deficits contribute to longer braking time in stroke," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, p. 7, 2021.

[16] H. Devos, A. E. Akinwuntan, A. Nieuwboer, S. Truijen, M. Tant, and W. De Weerdt, "Screening for fitness to drive after stroke: A systematic review and meta-analysis," *Neurology*, vol. 76, no. 8, pp. 747–756, 2011.

[17] A. Vander Veen, L. Johnston, J. Holmes, P. Tucker, and L. Alvarez, "Screening fitness to drive after stroke across demographic subgroups: A systematic review," *OTJR: Occupational Therapy Journal of Research*, vol. 46, no. 3, pp. 573–590, 2026.

[18] P. M. Aslaksen, M. Ørbo, R. Elvestad, C. Schäfer, and A. Anke, "Prediction of on-road driving ability after traumatic brain injury and stroke," *European Journal of Neurology*, vol. 20, no. 9, pp. 1227–1233, 2013.

[19] C. C. MacAdam, "Understanding and modeling the human driver," *Vehicle System Dynamics*, vol. 40, no. 1–3, pp. 101–134, 2003.

[20] C. J. Nash, D. J. Cole, and R. S. Bigler, "A review of human sensory dynamics for application to models of driver steering and speed control," *Biological Cybernetics*, vol. 110, no. 2–3, pp. 91–116, 2016.

[21] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000.

[22] P. G. Griffiths and R. B. Gillespie, "Sharing control between humans and automation using haptic interface: Primary and secondary task performance benefits," *Human Factors*, vol. 47, no. 3, pp. 574–590, 2005.

[23] F. Flemisch, J. Kelsch, C. Löper, A. Schieben, J. Schindler, and M. Heesen, "Cooperative control and active interfaces for vehicle assistance and automation," in *Proceedings of the FISITA World Automotive Congress*, Munich, Germany, 2008, paper F2008-02-045.

[24] D. A. Abbink, M. Mulder, and E. R. Boer, "Haptic shared control: Smoothly shifting control authority?" *Cognition, Technology & Work*, vol. 14, pp. 19–28, 2012.

[25] J. L. Emken, J. E. Bobrow, and D. J. Reinkensmeyer, "Robotic movement training as an optimization problem: Designing a controller that assists only as needed," in *Proceedings of the 9th IEEE International Conference on Rehabilitation Robotics*, 2005, pp. 307–312.

[26] E. T. Wolbrecht, V. Chan, D. J. Reinkensmeyer, and J. E. Bobrow, "Optimizing compliant, model-based robotic assistance to promote neurorehabilitation," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 3, pp. 286–297, 2008.

[27] R. Rajamani, S. B. Choi, B. K. Law, J. K. Hedrick, R. Prohaska, and P. Kretz, "Design and experimental implementation of longitudinal control for a platoon of automated vehicles," *Journal of Dynamic Systems, Measurement, and Control*, vol. 122, no. 3, pp. 470–476, 2000.

[28] L. Guo, P. Ge, D. Sun, and Y. Qiao, "Adaptive cruise control based on model predictive control with constraints softening," *Applied Sciences*, vol. 10, no. 5, p. 1635, 2020.

[29] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[30] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[31] Eclipse SUMO, "TraCI – traffic control interface," SUMO Documentation, 2026, accessed: September 2026. [Online]. Available: https://eclipse.dev/sumo/docs/TraCI/index.html

[32] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, Pasadena, CA, USA, 2008, pp. 11–15.

[33] A. Ferrara, S. Sacone, and S. Siri, *Freeway Traffic Modelling and Control*, ser. Advances in Industrial Control. Springer International Publishing, 2018.

[34] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 7th ed. Pearson, 2014.

[35] A. Ferrara, "Process control: Slides of the course - part 4," KIRO, University of Pavia, 2025.

[36] Y. Sugimoto and C. Sauer, "Effectiveness estimation method for advanced driver assistance system and its application to collision mitigation brake system," in *Proceedings of the 19th International Technical Conference on the Enhanced Safety of Vehicles (ESV)*, Washington, DC, USA, Jun. 2005, pp. 1–8.

[37] M. Poliak, L. Svabova, J. Benus, and E. Demirci, "Driver response time and age impact on the reaction time of drivers: A driving simulator study among professional-truck drivers," *Mathematics*, vol. 10, no. 9, p. 1489, 2022. [Online]. Available: https://doi.org/10.3390/math10091489

List of Figures

2.1 Execution sequence of the Python–SUMO framework. 19

2.2 TraCI execution flow, including the repeated exchange of measurements and commands. 21

2.3 Time-driven traffic-aware rerouting procedure. 27

2.4 Road network used for the example simulation, visualised in SUMO NetEdit. 29

2.5 NetworkX representation of the SUMO road network and EGO route in the example simulation. 30

2.6 Evolution of traffic density on the selected road segment. 32

2.7 Evolution of mean traffic speed on the selected road segment. 32

2.8 Evolution of traffic flow on the selected road segment. 33

2.9 Flow–density relationship obtained from the simulated background traffic. 34

2.10 Speed–density relationship obtained from the simulated background traffic. 34

2.11 Space–time trajectories of vehicles on lane 26599252_0. 35

2.12 Evolution of the EGO vehicle during the traffic simulation at different simulation times. 36

2.13 Speed profile of the EGO vehicle during the simulation. 37

2.14 Acceleration profile of the EGO vehicle during the simulation. 38

3.1 Compact architecture of the microscopic longitudinal control framework. . 42

3.2 Conceptual procedure for test-based patient characterisation. 55

3.3 Parallel patient–assist longitudinal control architecture with threshold-based assistance activation and final command saturation. 59

3.4 Main operations performed during each TraCI control cycle. 66

4.1 Speed tracking related to all patient profiles - trapezoidal reference, with assistance disabled. 71

4.2 Tracking error related to all patient profiles - trapezoidal reference, with assistance disabled. 72

4.3 Acceleration command - trapezoidal reference, with assistance disabled. . . 72

4.4 Assist ON/OFF comparison and assist intervention logic - trapezoidal reference, delayed profile. 74

4.5 Speed tracking related to all patient profiles - sinusoidal reference, with assistance disabled. 76

4.6 Tracking error related to all patient profiles - sinusoidal reference, with assistance disabled. 77

4.7 Acceleration command - sinusoidal reference, with assistance disabled. . . . 77

4.8 Assist ON/OFF comparison and assist intervention logic - sinusoidal reference, cautious profile. 79

4.9 Speed tracking related to all patient profiles - step reference, with assistance disabled. 81

4.10 Tracking error related to all patient profiles - step reference, with assistance disabled. 82

4.11 Acceleration command - step reference, with assistance disabled. 82

4.12 Assist ON/OFF comparison and assist intervention logic - step reference, normal profile. 84

4.13 Speed tracking related to all patient profiles - exponential stop-and-go reference, with assistance disabled. 86

4.14 Tracking error related to all patient profiles - exponential stop-and-go reference, with assistance disabled. 87

4.15 Acceleration command - exponential stop-and-go reference, with assistance disabled. 87

4.16 Assist ON/OFF comparison and assist intervention logic - exponential stop-and-go, reactive profile. 89

4.17 Speed tracking related to all patient profiles - trapezoidal reference with assist disabled - lead-brake scenario. 91

4.18 Tracking error related to all patient profiles - trapezoidal reference with assist disabled - lead-brake scenario. 92

4.19 Acceleration command - trapezoidal reference with assist disabled - lead-brake scenario. 92

4.20 Gap to the lead vehicle for all four patient profiles with assistance disabled. The safe-gap boundary and unsafe region are indicated; the delayed-profile run terminates after a collision. 93

4.21 Emergency braking lead-vehicle scenario for the four patient profiles - SUMO GUI view. 93

4.22 EGO, lead and effective-reference speeds for the delayed profile in the lead-brake scenario, with assistance enabled. 94

4.23 Tracking error and threshold-based assistance activation for the delayed profile in the lead-brake scenario, with assistance enabled. 95

4.24 Lead-vehicle gap for the delayed profile with assistance enabled. The minimum recorded separation is approximately 14.1 m, and no collision occurs in this run. 95

4.25 Emergency braking lead-vehicle scenario for delayed profile - assist ON. . . 96

4.26 Conceptual nested-loop architecture for the future integration of traffic-level MPC and microscopic longitudinal control. Diagram produced by A. Galassi. 101

List of Tables

2.1 Role of the main Python files in the SUMO–TraCI simulation framework. . 18

2.2 Information transferred from the SUMO network to the NetworkX graph. . 23

2.3 Main parameters of the example traffic simulation. 29

3.1 Role of the main Python files. 40

3.2 Main variables used in the discrete-time formulation. 43

3.3 Patient profiles available in the current configuration. 49

3.4 Summary of the reference speed profiles implemented in the simulation framework. 61

3.5 Parameters used in the lead-brake simulation scenario. 62

3.6 Main columns stored in the simulation CSV output file. 67

3.7 Main columns stored in the `summary.csv` output file. 68

4.1 Summary metrics for the base scenario with trapezoidal reference and assistance disabled. 73

4.2 Assist OFF/ON comparison for the delayed patient profile in the base scenario with trapezoidal reference. 75

4.3 Summary metrics for the base scenario with sinusoidal reference and assistance disabled. 78

4.4 Assist OFF/ON comparison for the cautious patient profile in the base scenario with sinusoidal reference. 80

4.5 Summary metrics for the base scenario with step reference and assistance disabled. 83

4.6 Assist OFF/ON comparison for the normal patient profile in the base scenario with step reference. 85

4.7 Summary metrics for the base scenario with exponential reference and assistance disabled. 88

4.8 Assist OFF/ON comparison for the reactive patient profile in the base scenario with exponential reference. 90

4.9 Safety-oriented summary metrics for the emergency braking lead-vehicle scenario with assistance disabled. 94

4.10 Safety comparison for the delayed profile in the lead-brake scenario. The unassisted run terminates following a collision. 96

Acknowledgements

I would like to sincerely thank Prof. Antonella Ferrara for her support, for the opportunity to work on this project, and above all for the trust she placed in us throughout its development. Her willingness to let us work independently gave us the freedom to explore our ideas and develop the project with creativity and responsibility.

I am also deeply grateful to Dr. Nikolas Sacchi for his constant availability, his valuable advice, and for always encouraging and supporting our ideas throughout the development of this work.

I would like to thank my family for their continuous support throughout my academic journey. I hope that this achievement makes them proud and reflects, at least in part, all the support they have given me over the years.

A special thanks also goes to all my course mates, with whom I shared these two years of study, challenges and experiences.

Finally, a special thanks goes to Alessandro and Gianluca, with whom I shared the early stages of this project. Their company, ideas and countless off-topic conversations made the work lighter and the whole experience much more enjoyable. I wish them all the best for their future, both personally and professionally.