Daniel Silvestre

FirePuma Executive Summary

Forest Surveillance using Autonomous Vehicles for Fire Prevention

January 15, 2025

Project FirePuma financially supported by Fundação para a Ciência e Tecnologia through 10.54499/PCIF/MPG/0156/2019

Acknowledgements

This work reproduces in a coherent form with respect to the objectives of project FirePuma work that was done in collaboration with Professor Rita Cunha, Professor Guilherme Ramos, Professor Francisco Rego and that is present in the dissertations of Francisco Santos, Duarte Silva and Hugo Matias.

Contents

Acronyms

UAV	Unmanned Aerial Vehicle
AES	Advanced Encryption Standard
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
DL	Download Link
ECC	Error Correction Code
GSM	Global System for Mobile Communication
IST	Instituto Superior Técnico
LTI	Linear Time-Invariant
LTV	Linear Time-Varying
LPV	Linear Parameter-Varying
LPWAN	Low Power Wide Area Network
LTE-A	Long Term Evolution Advanced
LTE	Long Term Evolution
LoRaWAN	Long Range Wide Area Network
CCG	Constrained Convex Generator
CZ	Constrained Zonotope
RPI	Robust Positively Invariant
LoRa	Long Range
MCU	Micro-Controller Unit
MISO	Master In Slave Out
MOP	Maximum Output Power
MOSI	Master Out Slave In
NB-IOT	Narrowband Internet of things
RMSE	Root-Mean-Square-Error
RSSI	Received Signal Strength Indicator
RTD	Round-Trip Delay
SCK	Serial Clock
SNR	Signal-To-Noise Ratio
SPI	Serial Peripheral Interface

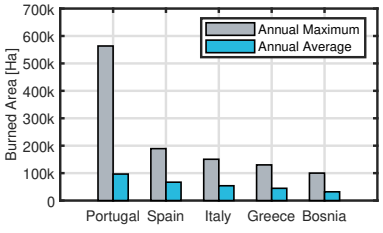
SPN	Substitution-Permutation Network
SS	Slave Select
TOA	Time On Air
UL	Upload Link
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
WiFi-6	Wireless Fidelity version 6
CF	Characteristic Function
CBF	Control Barrier Function
HF	Hybrid Filter
GMF	Gaussian Mixture Filter
GMM	Gaussian Mixture Model
KF	Kalman Filter
KL	Kullback–Leibler
PF	Particle Filter
PDF	Probability Density Function
MPC	Model Predictive Control
PDF	Probability Density Function
ODE	Ordinary Differential Equation
FOV	Field of View
EKF	Extended Kalman Filter
NLP	Nonlinear Program
PID	Proportional-Integral-Derivative
ROS	Robot Operating System
PF	Particle Filter
HITL	Hardware In The Loop
SITL	Software In The Loop
GPS	Global Positioning System
API	Application Programming Interface

Chapter 1

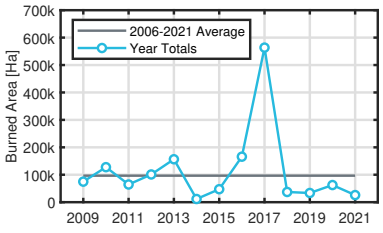
Forest Surveillance Challenges when using Autonomous Vehicles

1.1 Motivation

Over the past few decades, wildfires have emerged as a significant global threat. A combination of factors, including insufficient planning, more extreme weather conditions, and a lack of forest maintenance, has led to consecutive disasters. As a result, these events have imposed substantial financial burdens, claimed numerous human lives, and caused extensive damage to forests. In Europe, particularly, several countries witness an average yearly destruction of more than 50,000 hectares (ha) of land due to wildfires, with some years surpassing 100,000 ha [1]. Among these, Portugal has been the most severely impacted, with nearly 600,000 ha of land being burned in 2017 [2], as shown in Figure 1.1.



(a) Land burned by wildfires in Europe



(b) Land burned by wildfires fires in Portugal

Fig. 1.1: A look at wildfires in Europe, with an emphasis on Portugal (obtained from *Statista*).

According to a study conducted by the European Commission, Portugal experienced approximately 21,000 wildfires in 2017, which resulted in 117 deaths and caused nearly 1,500 million euros in damage costs [3, "Super Case Study 4"]. In

particular, the wildfire of Pedrógão Grande resulted in a burned area of roughly 45,000 acres and caused nearly 500 million euros in damage costs to the Portuguese government [4]. This outbreak also had a significant impact on the population, either directly through the destruction of agricultural resources and private property or indirectly through the impact on public infrastructures such as roads, energy networks, and telecommunications.

In light of such consecutive disasters, there has been a growing recognition of the urgent need to find technological solutions to prevent wildfires. As a result, governments and authorities have been interested in searching for solutions within the scientific community. The Portuguese government, in particular, has been promoting scientific research and innovation to improve the national forest defense system against wildfires, opening calls for research and development projects on the subject [5]. In particular, project Portuguese Fundação para a Ciência e a Tecnologia (FCT) funded project FirePuma (https://doi.org/10.54499/PCIF/MPG/0156/2019), which this document details the main outcomes.

1.2 Project Idea

Project FirePuma aimed at developing a surveillance system for the rural areas that would take data either from the population or sensor networks and encompass all stages until the design of optimized routes for autonomous vehicles to follow in their inspection of the forest. Namely, three questions were the driver for the project:

- how to detect and eliminate false data crowdsourced from the population?
- how to efficiently maintain a map of the likelihood of fire or the uncertainty regarding whether an area is safe or not?
- how to define at a mission level the optimal route for the autonomous mechanisms in order to reduce the uncertainty of existence of fire in problematic areas?

The data, mapping and mission levels cannot work independently if one is building an efficient system for surveillance. Though all can have an impact on a future strategy, they suffer from shortcomings only mitigated through integration. If estimating the likelihood of fires based on whether and terrain conditions, the system is oblivious to criminal malice or people with an economic agenda. On the other hand, this can be mitigated if the information is merged with crowdsourced data from citizens willing to cooperate in preventing fire destruction through early detection. If using just citizens' data, one is left with parts of the terrain that are not visited often and where fires can gain momentum. Such a shortcoming can be avoided using autonomous vehicles that can inspect remote areas. Nevertheless, these automatic methods are insufficient on their own due to a high needed investment if considering covering the whole area. In the context of Portugal alone, there are 9 221 200 ha (3.2

million ha according to the PEFC Portugal website - https://www.pefc.pt/) which renders these solutions expensive to be applied on their own to the whole territory. Therefore, substantial gains can be attained if these 3 areas of research are combined. The main objective would be to maintain a real-time map regarding the probability of fires based on the probability of occurrence given weather and other conditions and on the level of uncertainty measured from the last submitted data of citizens using mobile applications. An optimal trajectory for surveillance can be computed using autonomous systems by reducing the probability of fire through the minimization of the uncertainty in the map.

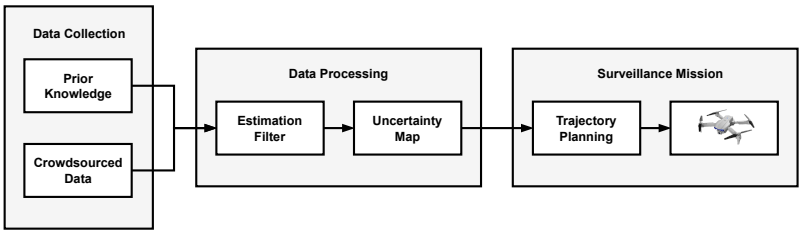


Fig. 1.2: FirePuma system blocks.

As depicted in Figure 1.2, the data collection and its processing to remove erroneous entries will be presented in Chapter 2. In particular, the team developed a LoRa protocol for communication that is tailored to the specifications needed for a sensor network to be deployed in the forest along with a prototype for the devices. The main contribution of Chapter 2 is then a reputation-based method that works both for synchronous and asynchronous communication that is able to detect intruders and incorrect data.

The second stage of FirePuma is to use an estimation filter to maintain either the support or the PDF that describe the evolution of a dynamical process. The developed algorithms are presented in Chapter 3, which can then be used to design an uncertainty map that serves to identify the areas with the most relevance for the autonomous surveillance. We remark to the reader that the developed methods can also be used for the estimation of the state of the drones and incorporate that information in a controller with collision avoidance mechanism.

The last step in the system is to take the uncertainty map and view it as an utility function that represents the importance of each location. In Chapter 4, we provide an optimization-based method that designs the trajectories and provides the control actions, which is efficient to be run in onboard embedded systems of each drone. A summary of the outputs of the project and some conclusions are also offered in Chapter 5.

Chapter 2

Resilient Data Acquisition using a LoRa-based Wireless Sensor Network

The first block of project FirePuma concerns with the acquisition of data and its processing to remove contributions that are either outliers or the input of malicious contributors trying to steer the overall surveillance. The task associated with this development has looked into the following problems:

- segmentation of images if cameras are to be used in [6], [7], [8] and [9];
- the development of a sensor network in [10] and [11];
- the desynchronization of transmitters to avoid packet collision [12];
- the efficient distribution of content [13] and packets [14], [15] in an unknown topology;
- the elimination of erroneous data through the definition of reputation-based mechanisms [16], [17] which were shown to be competitive and even outperform outlier removal techniques [18];
- evaluation of the degree of privacy in the topology and used algorithm that can be guaranteed for the proposed sensor network [19].

In the remainder of this chapter, we present the details of the final proposal that result from condensing the contributions and learning outcomes from the mentioned research.

2.1 Selection of Communication Protocol - LoRa

A sensor network to acquire forest data for a surveillance system must ensure economic viability if FirePuma is ever to be implemented in practice. On that note, the devices need to be sparsely deployed to reduce the required number for coverage as well as minimal cost to maintain the economic viability of the project. From the

several options like Wireless Local Area Network (WLAN) (i.e.: Wireless Fidelity version 6 (WiFi-6)), Bluetooth, Cellular (i.e.: Global System for Mobile Communication (GSM), Long Term Evolution Advanced (LTE-A), 5G) and Low Power Wide Area Network (LPWAN) (i.e.: Sigfox, Narrowband Internet of things (NB-IOT), Long Range (LoRa)) [20], [21], we evaluate briefly how they fair with respect to the economic viability metric. Table 2.1 shows a list of the main specifications for all the reviewed networks. Namely, the physical layer used by the technology, the spectrum frequency in which the communications are handled, the maximum bandwidth allowed per channel, the maximum theoretical throughput achievable, the Maximum Output Power (MOP) and the maximum range where packets can still be transmitted successfully. It should be noted that the MOP for WiFi-6 is strongly router-dependent which makes it a less relevant comparison factor and is therefore not mentioned in the table.

		PHY	Spectrum	Bandwidth	Throughput	MOP	Range
Bluetooth	Bluetooth 5.0	LE 1M uncoded, LE 1M coded, LE 2M uncoded	2.4 GHz	2 MHz	1.4 Mbps	20 dBm	< 100 m
WLAN	WiFi 6.0	IEEE 802.11ax	2.4 GHz 5 GHz	40 MHz 160 MHz	9.6 Gbps	—	< 100 m
Cellular	GSM	Based on GMSK	890-915 MHz, 935-960 MHz	200 KHz	270 Kbps	—	1 to several km
	LTE-A	DL:OFDM, UL:SC-FDMA	Same as GSM	100 MHz	DL:1 Gbps, UL:500 Mbps	30 dBm	1 to several km
	5G	5G-NR	mmWave: 30-300 GHz	500 MHz	1 Gbps	28 dBm	< 1 km
LPWAN	Sigfox	BPSK	868 MHz	100 Hz	100 bps	14 dBm	< 40 km
	NB-IOT	Same as LTE	Same as LTE	180/200 KHz	100 Kbps	35 dBm	< 35 km
	LoRaWAN	LoRa (CSS)	867-869 MHz	125 KHz	250 bps - 50 Kbps	14 dBm	< 15 km

Table 2.1: Main specifications of the networks approached in this section (all values refer to Europe specifications).

To complement the aforementioned Table 2.1, a radar graph is also presented in Figure 2.1. In this graph, a comparison is presented related to the main groups of technologies being analysed. As a result, and by using as metrics the range and coverage, throughput, latency performance, cost efficiency and battery life, this graph facilitates the reader to understand a technology choice for a certain application.

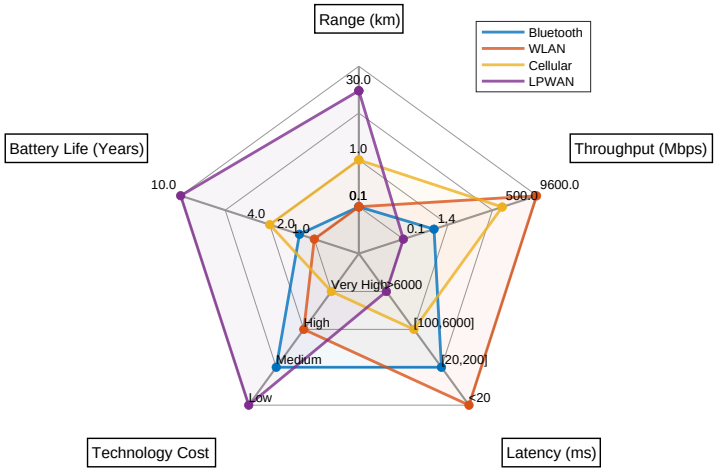


Fig. 2.1: Performance comparison across the categories of range, throughput, latency, technology cost and battery life between the main technologies approached: Bluetooth, WLAN, Cellular and LPWAN.

In order to maintain the economic viability, the network protocol must have long range capabilities (at least 1 km between the nodes and the gateway) and low power consumption (allowing for battery operated sensor devices). Additionally, a low cost technology is preferable. Since only sensor data with low sample size (mostly binary values) and low sample rate needs to be transmitted, a network with high bandwidth capabilities is not necessary. From Figure 2.1, it is clear that LPWAN is the most promising network available. Within the LPWANs, Sigfox was ruled out first due to its coverage requirement and limitation on the number of transmitted messages which can be a security risk for this project since a node will be blocked out if it does not comply with the regulations. In addition, the use of Sigfox comes with a subscription fee which increases the technology cost [22]. For the remaining technologies, given that all the requirements had been met, the main decision factor was the technology cost. The implementation of a LoRa enabled network is several times more affordable than its rival NB-IOT. Additionally, a LoRa gateway can easily be deployed in any area whereas NB-IOT requires Long Term Evolution (LTE) coverage to function which may not be available at the deployment site for this project. Hence, from the analyzed networks, the LoRa specification from the LPWANs is the choice which best fits the aforementioned requirements and it will be used to provide the communications network for FirePuma.

2.2 LoRa-based Wireless Sensor Network

The protocol Long Range Wide Area Network (LoRaWAN) was a possible option as a LoRa network protocol as it comes with features like data encryption, device classes for different energy consumption applications and device addressing. However, LoRaWAN also has some disadvantages: it requires the use of specific LoRaWAN compatible devices, which are more expensive than just plain LoRa modems; it imposes additional regulations on the transmission duty cycle (which further decreases the allowed number of transmitted messages) and it introduces additional delays due to the use of slotted receive windows in the end-devices. These are relevant drawbacks since they can lead to nodes not being able to communicate with the gateway which is a security flaw. For these reasons, a tailored protocol was developed for FirePuma, which we detail in the remainder of this section using n_i and g_i to represent a node and a gateway with id i, respectively.

2.2.1 LoRa Technology Review

The LoRa physical layer is based on Chirp Spread Spectrum (CSS) modulation technology, which is commonly used for radio applications. With CSS modulation, wide band frequency impulses which increase or decrease over time are used to transport the encoded data [23], with various parameters to be configured, namely:

a) **Carrier Frequency.** A specific band is reserved for LoRa communications with the range 867 MHz to 869 MHz for Europe.

b) **Transmission Power.** Can be configured depending on the LoRa module from 2 dBm up to 20 dBm, but in Europe it is limited to 14 dBm or 25 mW.

c) **Bandwidth.** Values can range from 7.8 KHz up to 500 KHz with an increase in signal bandwidth allowing the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity.

d) **Coding Rate.** As it is a fraction between the number of raw bits over the number of encoded bits after Error Correction Code (ECC), and it works with chunks of 4 bits and can encode them into 5, 6, 7 or 8 bits.

e) **Spreading Factor.** The LoRa modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the spread information is sent is referred to as the symbol rate (R_S). The spreading factor represents the number of raw bits that can be encoded per symbol as well as the number of chips contained in each symbol (2^{SF}); meaning that increasing the spreading factor will increase the number of chips per symbol which as a result allows for smaller and even negative Signal-To-Noise Ratio (SNR) values at the receiver, increasing the sensitivity, link budget and range.

f) **Implicit Header Mode.** Option to decide to include in the packet the header information: payload length, coding rate and if a Cyclic Redundancy Check (CRC) is used.

g) **Low Data Rate Optimization.** Increases robustness of a LoRa link at higher spreading factors and is mandatory for a symbol time (T_S) larger than 16 ms.

h) **Cyclic Redundancy Check.** Enables the inclusion of a CRC for the packet.

A LoRa transmission rate is better characterized by

$$R_b = \mathrm{SF} \times \frac{\mathrm{BW}}{2^{\mathrm{SF}}} \times \mathrm{CR}. \tag{2.1}$$

The performance of the network regarding delay times requires the definition of LoRa packet Time On Air (TOA). Using the *SX1276* modem datasheet [24], the TOA depends on a number of radio parameters: the Spreading Factor (SF), the Bandwidth (BW), the use of CRC (CRC), the Coding Rate (CR), the use of Implicit Header mode (IH) and the use of Low Data Rate Optimization (DE); as well as packet parameters like the preamble length (N_{PREAMBLE}) and payload length (PL). In this way, the symbol rate and symbol time are defined as:

$$R_S = \frac{\mathrm{BW}}{2^{\mathrm{SP}}}\,, \tag{2.2a}$$

$$T_S = \frac{1}{R_S}\,. \tag{2.2b}$$

The preamble time can now be calculated as a function of the preamble length in symbols:

$$T_{\mathrm{PREAMBLE}} = (N_{\mathrm{PREAMBLE}} + 4.25) \times T_S\,. \tag{2.3}$$

To calculate the payload time, the number of symbols in the payload must first be calculated:

$$N_{\mathrm{PAYLOAD}} = 8 + \max\left(\left\lceil \frac{8\mathrm{PL} - 4\mathrm{SF} + 28 + 16\mathrm{CRC} - 20\mathrm{IH}}{4(\mathrm{SF} - 2\mathrm{DE})} \right\rceil \times (\mathrm{CR} + 4), 0\right)\,, \tag{2.4}$$

Where $\lceil\ \rceil$ is used as a round up operation. The payload time is therefore given by:

$$T_{\mathrm{PAYLOAD}} = N_{\mathrm{PAYLOAD}} \times T_S\,. \tag{2.5}$$

Finally, the total TOA can be calculated by:

$$T_{\text{PACKET}} = T_{\text{PREAMBLE}} + T_{\text{PAYLOAD}} . \quad (2.6)$$

In this case of modem *SX1276*, the preamble size has the default value of 8 symbols and the payload length is of 18 bytes. Additionally, the CRC is enabled, the DE parameter has a value of 0 as there is no low data rate optimization and the IH parameter has a value of 1 as no explicit header is being used. The remaining parameters are dependent on the implementation and configuration of the network and must be decided according to the circumstances of each implementation.

2.2.2 Custom Network Protocol

The developed custom protocol was designed with the purpose of enabling bidirectional communication between the base station and the sensor devices, decrease payload size and its corresponding TOA given the data required for the FirePuma system, which in turn saves power by decreasing the time that the modem is transmitting.

The proposed protocol defines three device types: **end-nodes** (able to be battery operated, can be equipped with a number of sensors and/or actuators), **range-extenders** (able to be battery operated, extend the range of the gateway) and **gateways** (responsible for the communication with the end-nodes and bridging the network to a base-station). Additionally, it must meet the following specifications: **Device addressing** - so communication can be made between the base station and a single node; **Data encryption** - so that the sensitive data in the payload is protected; **Bidirectional communication** - so that both Upload Link (UL) and Download Link (DL) messages are supported; **Message delivery acknowledgement and retransmission** - given the use of a public frequency spectrum susceptible to interference; **Detection of Transmission Errors** - using a CRC of the data to detect corrupted packets.

2.2.2.1 Device Addressing

Each node device has a unique identifier in the form of a byte value which is defined at compile time (when the code is compiled and uploaded to the board) is what allows device addressing to work. When sending a message, this identifier is included as part of the message header (detailed in Section 2.2.4).

2.2.2.2 Data Encryption

Data encryption in the implemented in the custom network protocol resorted to the 256-bits *Advanced Encryption Standard (AES)* encryption algorithm. AES uses a fixed block size of 128-bits and a key size of 256-bits according to our selection. This algorithm is based on the Substitution-Permutation Network (SPN) principle where several rounds of substitution and permutation operations are applied to the *plaintext* (the text to be encrypted) in order to generate the *ciphertext* (the encrypted text). The number of rounds applied depends on the size of the used key and corresponds to 14 rounds for a 256-bit key. To this extent, every node device has a unique symmetric 256-bit encryption key, which is used for both UL and DL communications. The gateway device has information of all the remaining keys at compile time. This allows for end-to-end encryption between the nodes and the gateway.

2.2.2.3 Bidirectional Communication

Communication between the nodes and the gateway is defined in the following way: nodes only send data to the gateway (corresponding to an UL message) and the gateway only sends data to nodes (corresponding to a DL message). To achieve this, a feature from the LoRa modem called *IQ inversion* is used. In sum, by enabling *IQ inversion*, the preamble consisting on a defined number of chirps, used so the LoRa modem can identify a packet, becomes inverted (the up-chirps are swapped for down-chirps). Figure 2.2 illustrates the concepts of preamble, up-chirp and down-chirp. Finally, the gateway uses no *IQ inversion* for packet reception and uses *IQ inversion* for packet transmission. In contrast, nodes use *IQ inversion* for packet reception and use no *IQ inversion* for packet transmission allowing for correct bidirectional communication according to specifications.

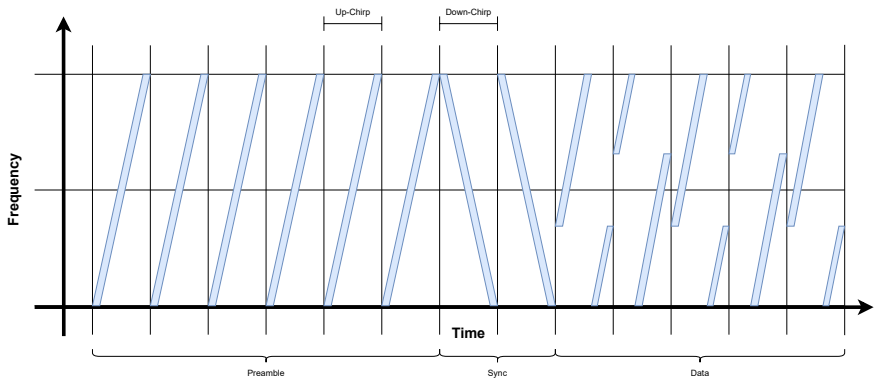


Fig. 2.2: LoRa modulation on an example packet. (No *IQ inversion*.)

2.2.2.4 Message delivery acknowledgement and retransmission

In this project, it is important for a device in a network to know if a message has been correctly received as the undetected loss of messages can be a security vulnerability. To achieve this, every time a message is sent, the sender awaits for an acknowledgement confirming the correct reception. In case the acknowledge message is not received after a transmission, the sender will wait for a designated amount of time (defined at compile time, meaning that after deployment, physical presence at the node location is required in order to reprogram the node to change this value) and try to re-transmit the message as depicted in Figure 2.3. This process will be repeated up to a maximum number of retries (defined at compile time) after which the message is discarded. It should be noted that the gateway cannot know when an UL message is dropped. However, it knows when a DL message is dropped and can inform the base-station of that occurrence. Additionally, we remark that re-transmitting a message can interfere with the process of handling the remaining messages of the queue for that node. For instance, if a message initially fails to be transmitted by a sensor node and, during its retransmission, a new trigger of a sensor occurs, then a new message will be generated and transmitted at the same time. To handle this, a message queue is implemented, which is illustrated for a practical case in Figure 2.4. In this example, a network with three nodes, where the nodes n_1 and n_2 are offline and the node n_3 is online, is used. Additionally, the network is configured with a retry timeout of 1200 ms, a maximum number of retries of 5 and queue size of 5. This means that the gateway will only send a DL message every 1200 ms and, in case of failure, a message will be re-transmitted up to 5 times before being dropped. The executed test consists of sending a status request message to each node sequentially three times every three seconds. As a result, Figure 2.4 shows the queue size as a function of time where every data point represents a sent message. Each data point is labeled with the corresponding message ID. Moreover, the DL message commands sent from the server are also shown along with the responses to the status request messages.

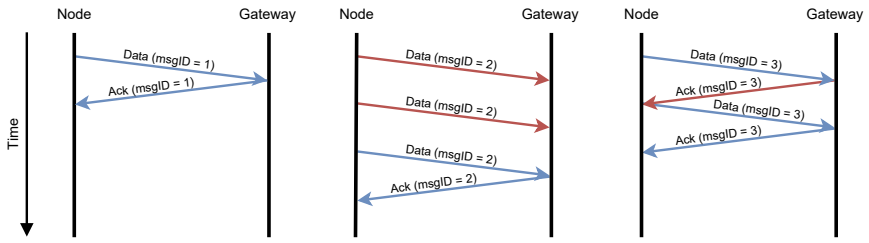


Fig. 2.3: Sequence diagram for three different examples of how a message can be successfully transmitted. (A packet that was received correctly is shown in blue and a failed packet in red.)

As depicted in Figure 2.4, the DL commands from the server are represented by the orange dots and, as expected, appear separated by a three-second interval. The blue dots represent a response to a sent message, which can either be a failure to send the message or a successful response from the node. Looking at the message IDs, it can be seen that the gateway sends messages corresponding to the nodes n_1 and n_2 5 times each. This happens due to the nodes being offline and the gateway retrying to send the message before triggering a failure response message. In contrast, messages sent to the node n_3 are only sent once as the node successfully responds to the received messages. It should be noted that around the 28 second mark, the final DL message command is sent from the server. However, as shown, the queue size at that time is already equal to the maximum queue size. As a result, this message cannot be added to the queue, being immediately dropped and a failure response triggered (represented by the blue dot labeled with the message ID 9).

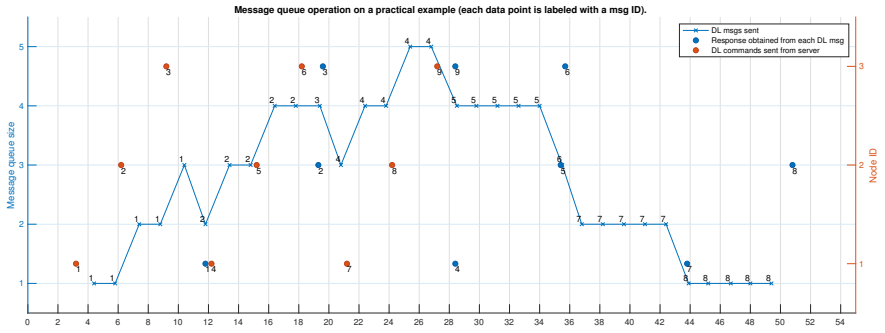


Fig. 2.4: Practical example of the message queue operation and message retransmission executed on a network with three nodes where the nodes n_1 and n_2 are offline and only the node n_3 is online.

2.2.2.5 Detection of Transmission Errors

The LoRa modem used (*SX1276* Section 2.3) includes an option to enable a CRC. By doing this, a 16-bit value calculated using the remainder of a polynomial division of the payload contents is added to the packet. On reception, the same calculation is repeated and if the result is different signals a corrupted packet. In addition, the LoRa modulation process already includes cyclic ECC, which allows for the data in corrupted packets to be recovered (up to a certain extent) giving LoRa some tolerance to interference.

This protocol is intended for a local deployment where only one gateway is utilized. The gateway is connected via serial to a processing unit which processes all

communications. For nodes outside the range of the gateway, it is required the use of range-extenders.

2.2.3 Network Topology

The sensor network to be deployed will only require nodes to send messages to the gateway that is in charge of forwarding the data to the processing unit. Therefore, we selected a star topology as it offers the best power savings, which is depicted in Figure 2.5.

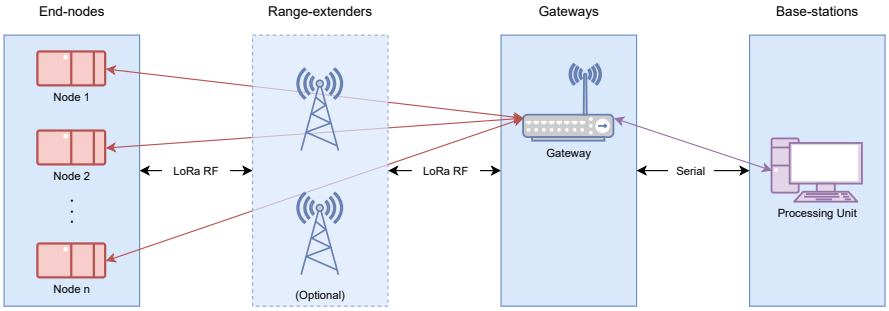


Fig. 2.5: WSN Architecture Diagram.

2.2.4 Message Formatting and Data Packets

A requisite for a successful communication is the correct formatting of messages to be exchanged, which in the current protocol are:

- Status update message (UL);
- Sensor data message (UL);
- Status request message (DL);
- Control message (DL);
- Acknowledge message (UL, DL).

The aforementioned message types allow for the gateway to receive sensor data and status data from the end-devices as well as to request a status update from any node or to send a message to control an actuator. To meet all the specifications

mentioned in Section 2.2.2, a custom packet format was defined with a plain header containing a network ID, necessary to distinguish different networks as well as due to the spectrum used to communicate. The network ID is therefore used to identify the messages that come from devices in this network. Additionally, the header also includes the node ID, which represents the destination ID in case of a DL message or the sender ID in case of an UL message. The payload is encrypted using the *AES* encryption algorithm [25] which only encrypts blocks of 16 bytes, explaining the fixed size of current messages. The entire packet format is depicted in Figure 2.6 in blue.

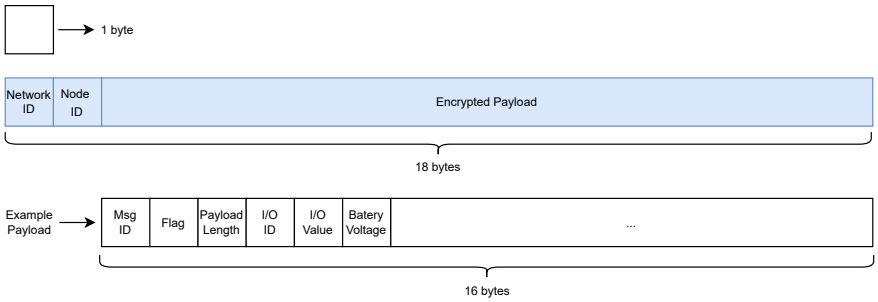


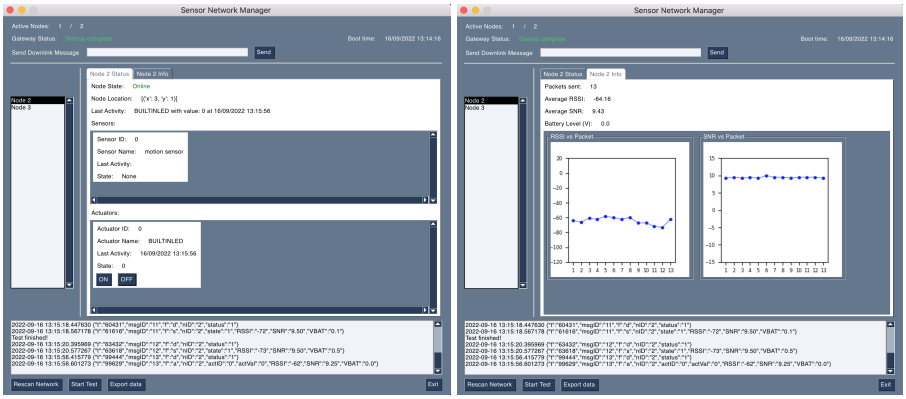
Fig. 2.6: Packet and payload format.

Also depicted in Figure 2.6 is a payload example containing a unique message ID used by the packet recipient to send back an acknowledgement, a flag byte to indicate the message type and a payload length byte indicating its size. Finally, there are bytes allocated for I/O (sensor ID and value in case of UL message or actuator ID and value in case of DL message) and for the battery voltage if the sender device is equipped with one. It should be noted that henceforward, since the encryption algorithm imposes a payload size of at least 16 bytes there is room for additional data to be included in the payload.

2.2.5 Network Manager Interface

In order to monitor the network, a tool was developed in the form of a python graphical user interface that runs on the processing unit connected to the gateway. The interface reads a *yaml* configuration file containing information regarding the network devices (including sensor and actuator information) and is updated in real-time whenever an UL message is received by the gateway.

The data monitored by the **Network Manager Interface** can be seen in Figure 2.7 and includes the following information:



(a) Focus on the Info tab.

(b) Focus on the Stats tab.

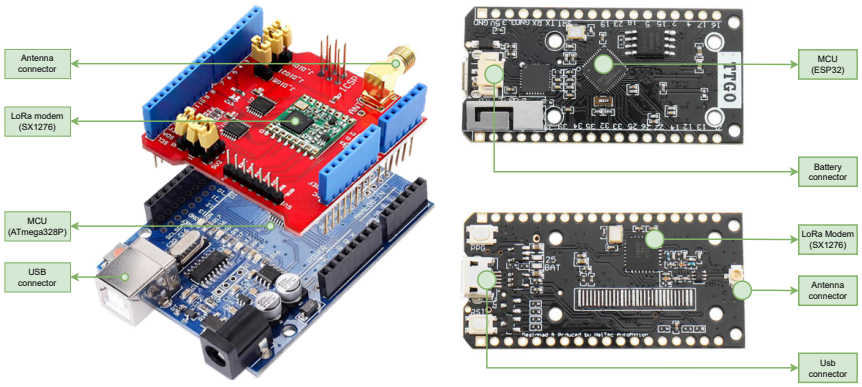
Fig. 2.7: Sensor Network Manager python interface.

- Gateway status;
- Number of active end nodes;
- Node status;
- Node location;
- Node last activity;
- Node sensors information;
- Node actuators information, including binary control options;
- Node statistics, including packets sent, average Received Signal Strength Indicator (RSSI), average SNR, and current battery level;
- Node RSSI and SNR plots over the packets sent;
- Terminal output for debug purposes.

Additionally, there is an input field for sending DL messages, a “Rescan Network” button that sends a status request to all nodes, an “Export Data” button that exports a comma-separated values file containing all the communications made since the interface boot time and a “Start Test” button which runs a predefined test on the network.

2.3 Hardware Equipment

The hardware equipment needed to deploy the Wireless Sensor Network (WSN) can be characterized by two main components: the processing unit and the radio transceiver. The processing unit is a Micro-Controller Unit (MCU) that can be the *ESP32* [26] or *Arduino Uno* [27] modules. As for the radio transceiver, the *SX1276* chip by *Semtech* [24] was used. The *Arduino Uno* serves as the gateway having the LoRa modem connected using the *Dragino LoRa Shield*. This assembly is depicted in Figure 2.8a. Alternatively, the *TTGO ESP32 SX1276* was utilized as the platform for the nodes and can be seen in Figure 2.8b.



(a) Arduino Uno (bottom) and Dragino Shield (top).

(b) TTGO ESP32 SX1276.

Fig. 2.8: Hardware equipment used in the deployment of the network.

The *SX1276* is equipped with the LoRa long range modem which has the main characteristics of providing ultra-long range spread spectrum communication and high interference tolerance whilst minimizing current consumption and is ideal for applications requiring long range or robustness. It should be noted that the LoRa modem communicates using the available Serial Peripheral Interface (SPI). SPI is a synchronous serial data protocol used by MCUs for communication with one or more peripheral devices. With an SPI connection there is a master device (usually the MCU) which controls the peripheral devices. In order to do so, there are three lines common to all devices: Master In Slave Out (MISO) (slave line for sending data to the master), Master Out Slave In (MOSI) (master line for sending data to the peripherals) and Serial Clock (SCK) (the clock pulse for synchronization purposes); and one line specific to each device: Slave Select (SS) (allows each device to be controlled individually while sharing the aforementioned lines) [28].

2.4 Network Testing

In order to evaluate the performance and ensure the correct functioning of the WSN, a network was established with a gateway based on the *Arduino Uno* with the *Dragino SX1276 Shield* (Figure 2.8a) and 3 nodes based on the *TTGO ESP32 SX1276* (Figure 2.8b). It was tested the following metrics: RSSI, SNR, packet loss and packet delay. The sequence of steps consist in sending status request messages to the nodes sequentially and repeatedly. The nodes will respond to the gateway request by sending a corresponding status message. This allows the gateway to measure the RSSI and SNR values on the received messages and it allows the delay and packet loss to be calculated based on the corresponding sent and received messages.

2.4.1 Basic Experiment

The purpose of the basic test is to ensure the network works properly under ideal conditions and to establish a base-line. The first type of measurement was related to packet delay given by the TOA (T_{packet}) under different circumstances. It should be noted that all communications in the network include an acknowledge message, meaning that each message corresponds to two packets. The overall theoretical Round-Trip Delay (RTD) is therefore equal to RTD $= 2 \times T_{\text{packet}}$. Given this, the network was initialized with a bandwidth of $BW = 125$ KHz, a coding rate of CR $= 4/5$ and a spreading factor of SF $= 7$ which correspond to the default values for the *SX1276* modem (Table 2.2).

Parameter	**Value**
Spreading Factor	7
Bandwidth	125 KHz
Coding Rate	4/5

Table 2.2: LoRa modem initialization parameters.

The spreading factor was then varied across different values (SF $\in \{7, 9, 11\}$), as were the bandwidth (BW $\in \{125$ KHz, 250 KHz$\}$) and the coding rate (CR $\in \{4/5, 4/8\}$), with each test run having only one parameter changed from the initial configuration. Finally, the test was ran for each configuration, where 50 status request messages are sent to node n_1 (with a distance of $d = 40$cm to the gateway) and the response status messages are received. Table 2.3 shows the results obtained.

CR	BW (KHz)	SF	Packet Loss(%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)	Theoretical RTD (ms)	RTD Diff. (ms)
4/5	125	7	0	-32.56	9.37	186.68	92.68	94.00
4/5	125	9	0	-32.84	12.81	457.54	329.72	127.82
4/5	125	11	0	-32.00	10.19	1426.18	1318.92	107.26
4/5	250	7	0	-33.26	10.13	133.46	46.34	87.12
4/8	125	7	0	-34.48	9.63	222.70	123.40	99.30

Table 2.3: Results obtained from sending the status request message to a node 50 times with different parameters. (Varied parameters shown in orange.)

As expected, the RTD follows the calculated theoretical value. Moreover, it can be concluded that if the range has to be extended by increasing the spreading factor, the RTD will also increase. Regarding the bandwidth, it also followed the theoretical RTD which makes sense given that an increase in bandwidth also increases the bit-rate, which in turn will decrease the delay. As for the coding rate, the results are also as expected. By including more redundancy bits in the ECC, the payload size increases and therefore the delay also increases. In this basic setup, there were no packets lost from the 50 sent messages. The RSSI and SNR are not dependent on the way the packet is transmitted as they are only affected by the conditions of the environment (like the distance to gateway, the existence of line-of-sight between the node and gateway and the amount of existing noise). Finally, there is an approximately constant difference of approximately 100 ms in the measured RTD and the theoretical one due to the processing times on the MCUs.

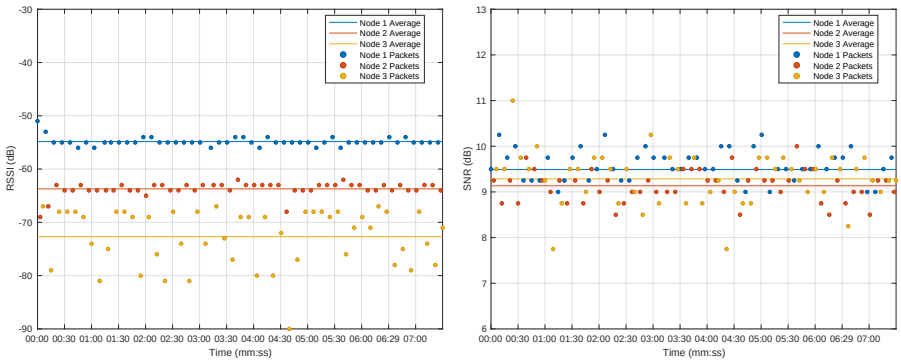
A second test was performed with the same nodes in the network and following the parameters in Table 2.2 with 50 status request messages being sent to the nodes n_1, n_2 and n_3, sequentially and repeatedly. The goal of this test is to evaluate the behaviour of the network with multiple nodes. The results obtained are shown in Table 2.4.

Aligned with the same conclusions from the previous test, in ideal conditions, all the packets are successfully delivered to all three nodes resulting in a zero packet loss. Additionally, the RTD delay stays consistent with the value on Table 2.3 (regarding the initialization parameters) showing once again that the delay does not depend on the distance or other parameters apart from the ones used in (2.6).

One observation is the way that the RSSI value changes with distance. As the distance between a node and the gateway increases, the power of the transmitted signal is the same. However, by having to cover a larger distance in the medium, the signal loses more strength resulting in a received signal with a lower RSSI value. This can be seen in Figure 2.9a where the RSSI values are plotted for each node.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	1	0	0	-54.84	9.49	204.04
2	4	0	0	-63.72	9.14	202.68
3	7	0	0	-72.68	9.29	203.34

Table 2.4: Results obtained from sending the status request message to each node 50 times sequentially and repeatedly.



(a) RSSI over time.

(b) SNR over time.

Fig. 2.9: Results obtained from sending the status request message to each node 50 times sequentially and repeatedly.

Furthermore, as the distance increases, the RSSI deviation from the average also increases, as the signal may be subjected to more interference. The values for the SNR are also plotted in Figure 2.9b. Given that the SNR is a ratio between the signal to noise power, it is reasonable that for the small distances in this test we obtained high SNR. Moreover, the values in Figure 2.9b, seem to follow a normal distribution around a constant value for each node, which can be caused by the noise interference in the medium since the signal power is constant for each node.

2.4.2 Rural Experiment

In this test, the aim is to evaluate the performance of the network in rural conditions. The nodes are located the farthest from the gateway (when compared to the basic

and urban experiments) in this test so as to find the maximum range where communication is still possible. A location was selected where electromagnetic interference is minimal and where the nodes can be in line of sight of the gateway or with only some bushes or trees in the way while keeping a large distance to the gateway. In this way, the performance of the network can be evaluated through larger communication distances in both line of sight and lightly obstructed conditions.

Fig. 2.10: Measurement location for the rural experiment. "A" and "B" represent gateway locations while the labels with the distances represent node locations. (Adapted from Google Maps.)

Depicted in Figure 2.10 are the device locations where the measurements took place. The letters "A" and "B" show the gateway locations and the distance values show the node locations. It should be noted that all node locations had line of sight to the gateway with exception of the closest one (438 m) in which a few trees were in the path.

The tests run in this experiment are similar to the ones in the basic experiment, where for each node, 50 status request messages are sent and responses are recorded. For each node location, a battery of 5 tests were run with different physical layer configurations where the relevant parameters were varied: SP $\in \{7, 9, 11\}$, BW $\in \{125\ \text{KHz}, 250\ \text{KHz}\}$, CR $\in \{4/5, 4/8\}$. Given this, and similarly to the previous experiment, the network was initialized with a bandwidth of BW = 125 KHz, a coding rate of CR = 4/5, and a spreading factor of SF = 7, which match default settings for the *SX1276* modem (Table 2.2) and only one parameter is changed from the initialization per run.

Tables 2.5 to 2.9 show in detail the results obtained from this experiment. For each node location and physical layer configuration, information is provided regarding the

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	1.96	0	-98.76	8.16	186.30
2	612	3.84	0	-105.34	6.57	185.24
2	956	1.96	0	-106.42	4.66	185.20

Table 2.5: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=7.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	9.26	2	-100.18	8.70	458.39
2	612	0	0	-105.22	9.62	457.04
2	956	0	0	-106.00	5.84	456.92

Table 2.6: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=9.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	30.88	6	-100.60	8.65	1426.70
2	612	10.90	2	-106.12	7.01	1425.60
2	956	24.59	8	-106.59	3.95	1425.40

Table 2.7: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=11.

main performance evaluation parameters. Packet loss shows the percentage of packets that were lost during a run including any packet retransmissions. On the other hand, the message loss shows the percentage of lost messages out of the total 50. These two metrics help to measure how packet retransmission affects the overall performance of the network. Additionally, it is reported averages for the RSSI, SNR and RTD values for each test run.

Looking at the average RTD value for each run and physical layer configuration, it is noticeable that the RTD stays consistent regardless of the distance to the gateway,

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	0	0	-98.98	7.04	134.36
2	612	1.96	0	-103.74	3.83	133.12
2	956	1.96	0	-103.85	0.55	133.03

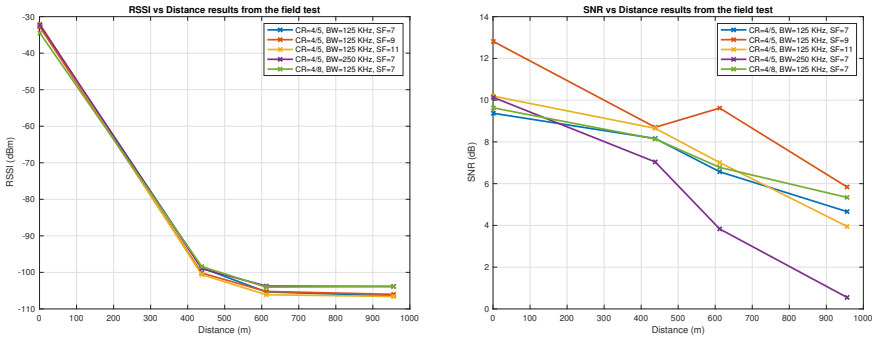
Table 2.8: Results obtained with the following physical layer configuration: CR=4/5, BW=250 KHz, SF=7.

Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
1	438	5.66	0	-98.40	8.14	223.58
2	612	0	0	-104.04	6.78	222.02
2	956	3.92	2	-103.86	5.34	222.04

Table 2.9: Results obtained with the following physical layer configuration: CR=4/8, BW=125 KHz, SF=7.

as expected from the basic tests. Moreover, the values obtained here follow the values obtained on the basic experiment closely, reinforcing that the RTD value does not depend on the distance to the gateway but only on the physical layer configuration. Regarding the averages for RSSI and SNR, plots in Figure 2.11 help better compare the results across different physical layer configurations.

In Figure 2.11a, it is shown that the tested physical layer configurations barely affect the strength of the received signal. This is expected as the RSSI value is determined only by the transmitted signal output power (the power at which the the LoRa modem transmits modulated signal), the antenna gain of the transmitter and receiver and by the gain loss due to the medium. As all these parameters are constant on each test, the RSSI value is expected to remain constant as result. Due to these same reasons, the plot clearly shows a decrease in the RSSI value as the distance to the gateway increases. This happens due to the medium gain loss. In this case, there is a drop in the RSSI value from the 438 m test to the 612 m test but from the 612 m test to the 956 m test the RSSI value remains more or less constant. This can be explained due to the few trees that block the line of sight at the 438 m test. On the other hand, the remaining tests have line of sight to the gateway and therefore the signal is only blocked by the air reducing the gain loss of the medium when compared to obstacles such as trees. When it comes to SNR values, the results shown in Figure 2.11b describe how the SNR value varies over distance and across



(a) RSSI over distance and physical layer configuration.

(b) SNR over distance and physical layer configuration.

Fig. 2.11: Results obtained from the field test.

physical layer configurations. As expected, as the distance increases the amount of noise that the signal is subjected to also increases and the SNR decreases as a result. However, in this case the SNR value never goes below 0, showing that there is little to no noise on this rural environment, as there are no obstacles or other signals being transmitted causing interference. Regarding physical layer configurations, all of them follow the same trend of worse SNR values over distance.

Finally, Figure 2.12 shows data regarding the network performance on packet loss. Overall, the packet loss is higher on the 438 m test. Intuitively, it would be expected that packet loss increased with distance. This is generally true but it must be taken into account that the 438 m test does not have an entirely clear line of sight to the gateway and this is most likely the reason why the packet loss is higher on this test.

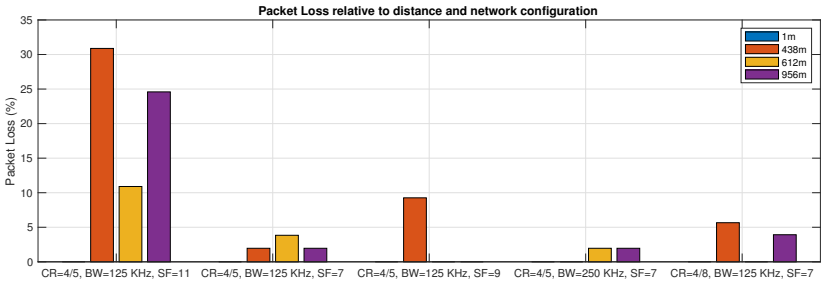


Fig. 2.12: Results obtained from the field test regarding packet loss over distance and physical layer configuration.

In summary, the rural experiment showed communication to be possible with a range close to 1km while still maintaining a relatively strong signal. This points to

the possibility of even larger distances being covered by the WSN. However, when compared to the advertised range of over 15 km, the results obtained fall short of that number. Additionally, an overall message loss of 1.33% was obtained given that out of the 50 messages per 3 nodes and per 5 physical layer configurations (750 in total) only 10 messages were dropped. Also in this subject, it should be noticed that the implemented packet retransmission capability plays an important role in reducing the message loss as often the message loss value is lower than the packet loss value proving how this feature improves the robustness of the network. When comparing the various tested physical layer configurations, interesting results were obtained: increasing the spreading factor actually lead to an increase in packet loss, which was not expected (the benefits of increasing the spreading factor may be more visible at larger communication distances); additionally, changing the coding rate and bandwidth had little effect on packet loss. In this way, the configuration which best suits a rural environment is: a bandwidth of $BW = 125$ KHz, a coding rate of CR = $4/5$ and a spreading factor of SF = 7.

2.4.3 Urban Experiment

The rural experiment is the most interesting with respect to the envisioned application of the surveillance system being designed. Nevertheless, it is also possible that some cases of areas to be inspected can have buildings or target surveillance of events in cities. Therefore, an urban case was also experimented which provide other researchers with a complete picture in terms of communication characteristics in a real-life test of both a rural and urban scenarios. Thus, this test is conducted at the Alameda campus of Instituto Superior Técnico (IST) in the center of Lisbon. The campus counts with two 12-story towers covered in glass and smaller buildings. The gateway was placed on the 8^{th} floor of the IST North tower close to a south-oriented window with the test node being moved over 5 different locations, some of which do not have line of sight to the gateway. This information is depicted in Figure 2.13.

Figure 2.13 shows the location used for this experiment along with the node and gateway locations where the measurements took place. It should be noted that the location point **E** is on the 5^{th} floor of the North tower while the remaining positions are at ground level. The remaining points were selected since they offer different conditions in terms of line of sight to the gateway. The location point **A** has direct line of sight to the gateway while **B** and **C** have a slightly obstructed view. Positions **D** and **E** are inside a building and, therefore, completed obstructed.

In comparison with the rural experiment, the urban test also compares the effect of two different antennas on the node device while sending 25 messages each (maintaining a 50-message test per configuration and per location). The battery of 5 tests encompass different physical layer configurations where the relevant varied parameters are: SP $\in \{7, 9, 11\}$, BW $\in$ {125 KHz, 250 KHz}, CR $\in \{4/5, 4/8\}$.

Fig. 2.13: Measurement location for the urban experiment. “G” represents the gateway location while the other letters represent the test node location. (Adapted from Google Maps.)

Given this, and similarly to the previous experiment, the network was initialized with a bandwidth of BW = 125 KHz, a coding rate of CR = 4/5, and a spreading factor of SF = 7, which match the default settings for the *SX1276* modem (Table 2.2) and only one parameter is changed from the initialization per run.

Tables 2.10 to 2.14 show the results obtained from this experiment in a similar fashion to the rural case reporting the main performance evaluation parameters: packet loss, message loss, average RSSI, average SNR and average RTD.

Regarding RTD, the results are as expected. For each physical layer configuration the delay stays approximately constant. Moreover, the values obtained for each physical layer configuration are close to the ones obtained when testing two devices in close proximity. When comparing the RTD values for both antennas, it can be concluded that the delay times are independent as the RTD is the same. With respect to RSSI and SNR, Figure 2.14 helps visualizing the results.

In Figure 2.14a the behaviour of the RSSI is shown against multiple communication distances. Furthermore, to widen the scope of this analysis, two scenarios were considered: an indoors scenario (where both the gateway and the node are inside a building) and an outdoors scenario (where only the gateway is inside a building). Immediately, a few observations can be made. The RSSI values for the node with the small antenna are consistently lower than the ones for the node with the big antenna. Since the two nodes stay at the exact same place, the medium is the same for both of them and the difference in the RSSI values is most likely caused by different output powers. Given that the same LoRa modem transmission power is selected for both

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-96.64	7.64	185.60
A	2	171	0	0	-102.04	5.98	184.36
B	1	73	0	0	-95.52	8.42	204.44
B	2	73	0	0	-101.80	6.95	203.16
C	1	193	0	0	-93.08	8.54	185.76
C	2	193	8	8	-103.78	6.81	184.52
D	1	110	8	8	-105.17	3.79	186.18
D	2	110	76.67	52	-110.57	-4.71	184.43
E	1	34	40	40	-103.27	0.32	185.67
E	2	34	81.25	76	-105.50	-3.50	184.50

Table 2.10: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=7.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-97.88	7.40	457.88
A	2	171	4	4	-103.08	5.04	456.54
B	1	73	0	0	-95.12	8.96	457.50
B	2	73	3.84	0	-101.56	8.09	456.15
C	1	193	0	0	-96.52	9.15	457.44
C	2	193	0	0	-106.76	5.93	456.24
D	1	110	0	0	-104.88	3.35	457.76
D	2	110	82.14	80	-109.80	-8.80	456.00
E	1	34	48	48	-102.23	0	457.93
E	2	34	76	76	-104.67	-6.50	456.33

Table 2.11: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=9.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	7.40	0	-98.28	7.13	1781.20
A	2	171	0	0	-104.12	5.85	1765.70
B	1	73	3.84	0	-96.40	8.17	1426.20
B	2	73	0	0	-101.88	7.40	1425.10
C	1	193	0	0	-97.76	7.23	1998.20
C	2	193	3.84	0	-106.68	6.84	1998.50
D	1	110	12	12	-108.04	2.92	1795.80
D	2	110	85.19	84	-113	-3.25	1795.80
E	1	34	32	32	-104.41	0.06	1426.10
E	2	34	68	68	-105.88	-4.37	1425.10

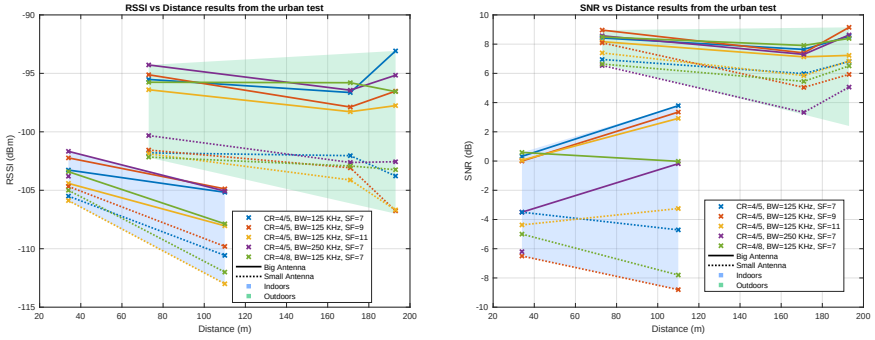
Table 2.12: Results obtained with the following physical layer configuration: CR=4/5, BW=125 KHz, SF=11.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-96.44	7.30	133.76
A	2	171	4	4	-102.62	3.33	132.37
B	1	73	0	0	-94.28	8.57	133.96
B	2	73	0	0	-100.32	6.54	132.44
C	1	193	0	0	-95.16	8.63	133.72
C	2	193	0	0	-102.56	5.06	132.28
D	1	110	46.15	44	-105.0714	-0.18	133.71
D	2	110	100	100	-	-	-
E	1	34	77.78	76	-101.67	-3.50	134.33
E	2	34	89.36	80	-103.80	-6.20	132.20

Table 2.13: Results obtained with the following physical layer configuration: CR=4/5, BW=250 KHz, SF=7.

Location	Node ID	Distance to Gateway (m)	Packet Loss (%)	Message Loss (%)	Average RSSI (dBm)	Average SNR (dB)	Average RTD (ms)
A	1	171	0	0	-95.80	7.91	222.80
A	2	171	0	0	-102.92	5.45	221.48
B	1	73	0	0	-95.76	8.49	222.84
B	2	73	3.84	0	-102.16	6.64	221.44
C	1	193	0	0	-96.56	8.39	222.88
C	2	193	0	0	-103.24	6.51	221.52
D	1	110	54.55	40	-107.87	-0.02	222.67
D	2	110	85.29	80	-112	-7.80	221.60
E	1	34	32	32	-103.41	0.59	222.76
E	2	34	96.15	96	-105	-5	222

Table 2.14: Results obtained with the following physical layer configuration: CR=4/8, BW=125 KHz, SF=7.



(a) RSSI over distance and physical layer configuration.

(b) SNR over distance and physical layer configuration.

Fig. 2.14: Results obtained from the urban test.

nodes, it means that the larger antenna has a higher gain. When it comes to node location, it is also clear that the indoor test runs yielded worse results in comparison with the outdoor setting. This is once again to be expected as there is no line of sight to the gateway and there are several walls and other obstacles in the signal path. Finally, the same conclusions taken from the previous experiment also apply here: RSSI has a small dependence on the physical layer configuration and overall it gets lower when the distance to the gateway increases.

As for the SNR, Figure 2.14b shows its behaviour for different communication distances. Similarly to the RSSI, the SNR tends to be lower for the small antenna. This should be explained by the fact that with the small antenna there is a lower signal output power resulting in a higher noise level by comparison. Again, in the indoor test runs, the results are worse. This happens due to the additional noise and interference caused by all the obstacles to the signal propagation. One interesting observation is that on the test run in location **D**, the SNR values obtained were generally better in comparison to using location **E** (even though location **E** was further away from the gateway). This could be explained by the difference in the medium through which the signal must pass. In location **E**, the node was three floors below the gateway, resulting in concrete obstacles while **D** is inside a different building (the buildings had line of sight to each other), leading to less signal interference.

Figure 2.15 shows the packet loss for all the test locations across multiple physical layer configurations. An expected trend is the higher packet loss for indoor test (34 m and 110 m) and the use of the small antenna since a lower antenna gain generates a lower RSSI. Regarding message loss, it is noted how this value is often lower than the packet loss, showing how packet retransmission is improving the network reliability.

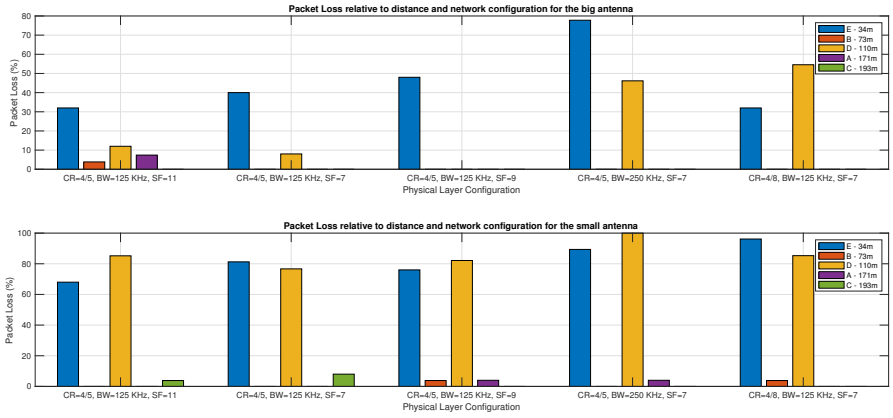


Fig. 2.15: Results obtained from the urban test regarding packet loss over distance and physical layer configuration.

To sum up, the urban experiment explored the capabilities of the developed LoRa based WSN in an urban environment, which can be quite challenging for a network using LoRa as its high efficiency property makes use of a very low signal transmission power. In addition, two different antennas were also tested. The results in this experiment showed how communication is possible from devices in different buildings and how well the network performs in better conditions, such as when the node is outside and with line of sight to the gateway. Similarly to the rural experiment, the default physical layer configuration appeared to show the best results overall. With this configuration a maximum packet loss of 40 % for the big antenna and 81.25 % for the small antenna was obtained for communication between devices in two different buildings with a communication distance of up to 110 m. For communication where only the gateway is inside a building and with a communication distance of up to 193 m, a maximum packet loss of 0 % and 8 % was obtained for the big and small antennas respectively.

2.5 Desynchronization Problem

The proposed WSN based on LoRa was tested under periodic communication where a single node would transmit. That raises the problem of having multiple collisions between transmitters. A possible solution is to desynchronize in a distributed manner the nodes of the sensor network as to avoid collisions. Moreover, if this is done so, nodes can turn on only periodically in order to conserve battery power. Each sensor can schedule its uptime to listen to only two neighbors, thus forming a ring. We can have a protocol on top of the communication one that periodically broadcasts a *fire message* or a *pulse* and listen to the medium for the messages of their neighbors. Each node $i \in \{1, \cdots, n\}$ has a *phase* variable $\theta_i(t)$ that depends on the period

$$\theta_i(t) = \frac{t}{T} + \phi_i(t) \mod 1, \tag{2.7}$$

where $\phi_i \in [0, 1]$ is the so-called phase offset of node i and $\mod$ notation stands for the modulo arithmetic. Every node i broadcasts a pulse when its phase reaches the unity (i.e., every T time units) and then resets it to zero. Every time a node receives a pulse, it will adjust its offset ϕ according to an update equation. The work in [29] showed that the problem of adjusting $\phi^{(k)}$ that stacks all phase offsets for individual nodes ϕ_i for each cycle k of pulse messages can be cast as the minimization of a quadratic function, where the steady-state solution would render each transmitter with equal phase offsets and able to use the medium without collisions. This contrasts with the Pulse Coupled Oscillators algorithm implemented in IEEE 802.15.4 [30] that corresponds to a consensus algorithm as in [31], [32].

Proposition 2.1 (Desynchronization as an optimization [29]) Let $\phi^{(k)}$ denote the phases of all nodes at updating cycle k. The state of desynchronization corresponds

to the solution of the following optimization problem:

$$\underset{\phi}{\text{minimize}} \quad g(\phi) := \frac{1}{2}\|D\phi - v\mathsf{1}_n + \mathsf{e}_n\|_2^2 \tag{2.8}$$

where $v = 1/n$, $\mathsf{1}_n$ is the vector of ones, $\mathsf{e}_n = (0, 0, \cdots, 0, 1)^\intercal$, and

$$D = \begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & 0 & -1 & 1 \\ 1 & \cdots & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{2.9}$$

During the project, we investigated optimal solutions for the update rule as to achieve fast convergence.

2.6 Optimal Fixed-parameter Nesterov-based Desynchronization Algorithm

The state-of-the-art approach to solve this problem was known as Fast-Desync [29], which corresponded to a time-varying parameter version of the Nesterov method as a fast algorithm to desynchronize the transmitters. Fast-Desync has the advantage of not requiring knowledge of the number of transmitters in the network at the expenses of a sub-optimal choice of parameters. In this section, we present other optimization algorithms and compute closed-form expressions for the parameters and worst-case convergence rates.

In [33], it is shown that the Gradient descent, Heavy-ball and Nesterov methods given by the iterations:

$$\begin{aligned} &\textsc{Gradient}: && x^{(k+1)} = x^{(k)} - \beta\nabla g(x^{(k)}) \\ &\textsc{Heavy-ball}: && x^{(k+1)} = x^{(k)} - \beta\nabla g(x^{(k)}) + \gamma(x^{(k)} - x^{(k-1)}) \\ &\textsc{Nesterov}: && \begin{aligned} x^{(k+1)} &= \xi^{(k)} - \beta\nabla g(\xi^{(k)}) \\ \xi^{(k)} &= (1+\gamma)x^{(k)} - \gamma x^{(k-1)} \end{aligned} \end{aligned} \tag{2.10}$$

and can be analyzed as dynamical systems to compute optimal parameters β and γ. This reformulation allows to use standard techniques from linear systems theory to compute the convergence rate in the worst-case trajectory, i.e., the positive constant $\lambda < 1$ such that $\|x^{(k)} - x^\star\| \leq c\lambda^k \|x^{(0)} - x^\star\|$, for some constant $c > 0$, where $x^\star$ corresponds to the steady state value for the system with initial conditions $x^{(0)}$.

However, the results presented in [33] apply to strongly convex functions, which, since g is quadratic, means that matrix Q must satisfy $mI_n \preceq Q \preceq LI_n$, which is equivalent to say that the eigenvalues of Q lie in the interval $[m, L]$ for $m > 0$. In the next lemma, the results are generalized to the case when function g is only convex but the eigenvectors associated with the zero eigenvalues are part of the minima. The result translates that the expressions in [33] for the parameters of the optimization algorithms achieving the best worst-case convergence rate for strongly convex functions are valid, provided that we consider the minimum non-zero eigenvalue as a replacement for the minimum eigenvalue of Q (which would be zero).

Lemma 2.1 *Consider a convex quadratic function $g(x) = x^\intercal Qx + c^\intercal x$ with Hessian matrix $0 \preceq Q \preceq LI_n$ and a subspace $\mathcal{S}$ containing any vector s resulting from a linear combination of eigenvectors of Q associated with zero eigenvalues. If any vector $s \in \mathcal{S}$ is a minimizer of the function, i.e.,*

$$\forall s \in \mathcal{S} : g(s) = g^\star, \tag{2.11}$$

where $g^\star$ is the global minimum of function g, then, optimal parameters achieving the best worst-case convergence rate for linear first-order optimization algorithms depend solely on m and L, where m is the minimum non-zero eigenvalue of Q.

Proof. Given that any linear first-order algorithm can be described by a transition matrix T

$$T = A + BQC, \tag{2.12}$$

where A, B and C represent the operation of the particular algorithm, its convergence rate depends on the spectra of T. From the statement of the lemma, all vectors in the null space of Q are global minima of function g. Thus, the error analysis needs only to consider initial conditions that do not start in $\mathcal{S}$. Moreover, from the results in [33], the spectra of T is given by the eigenvalues of $A_1 + B_1\lambda_i(Q)C_1$, where A_1, B_1 and C_1 correspond to the matrices A, B, C applied to a function g with domain in $\mathbb{R}$. Given any eigenvalue $\lambda_i(Q)$, we only need to consider $\lambda_i(Q) > 0$ since the initial conditions aligned with the eigenvectors of $\lambda_i(Q) = 0$ correspond to minima of function g, and the conclusion follows. □

Using Lemma 2.1, and assuming a known number of transmitters in the network, allows to compute optimal parameters for the three optimization algorithms, which is given in the next theorem. The expressions are given in terms of n which can be hardcoded in the transmitters software.

Theorem 2.1 *Consider the* Desync *problem in* (2.8) *with n transmitting sources. Then,*

Gradient descent with parameter:

$$\beta = \begin{cases} \frac{1}{3-\cos\left(\frac{2\pi}{n}\right)}, & \text{if } n \text{ is even} \\ \frac{1}{2-\cos\left(\frac{2\pi}{n}\right)-\cos\left(\frac{(n-1)\pi}{n}\right)}, & \text{if } n \text{ is odd} \end{cases} \tag{2.13}$$

achieves worst-case convergence rate ρ_G*:*

$$\rho_G = \begin{cases} \frac{1+\cos\left(\frac{2\pi}{n}\right)}{3-\cos\left(\frac{2\pi}{n}\right)}, & \text{if } n \text{ is even} \\ \frac{\cos\left(\frac{2\pi}{n}\right)-\cos\left(\frac{(n-1)\pi}{n}\right)}{2-\cos\left(\frac{2\pi}{n}\right)-\cos\left(\frac{(n-1)\pi}{n}\right)}, & \text{if } n \text{ is odd} \end{cases} \tag{2.14}$$

Heavy-ball with parameters:

$$\beta = \begin{cases} \frac{1}{\left(1+\sin\left(\frac{\pi}{n}\right)\right)^2}, & \text{if } n \text{ is even} \\ \frac{1}{\left(\sin\left(\frac{\pi}{n}\right)+\sin\left(\frac{(n-1)\pi}{2n}\right)\right)^2}, & \text{if } n \text{ is odd} \end{cases} \tag{2.15}$$

and

$$\gamma = \begin{cases} \left(\frac{1-\sin\left(\frac{\pi}{n}\right)}{1+\sin\left(\frac{\pi}{n}\right)}\right)^2, & \text{if } n \text{ is even} \\ \left(\frac{\sin\left(\frac{(n-1)\pi}{2n}\right)-\sin\left(\frac{\pi}{n}\right)}{\sin\left(\frac{(n-1)\pi}{2n}\right)+\sin\left(\frac{\pi}{n}\right)}\right)^2, & \text{if } n \text{ is odd} \end{cases} \tag{2.16}$$

achieves worst-case convergence rate ρ_H*:*

$$\rho_H = \begin{cases} \frac{1-\sin\left(\frac{\pi}{n}\right)}{1+\sin\left(\frac{\pi}{n}\right)}, & \text{if } n \text{ is even} \\ \frac{\sin\left(\frac{(n-1)\pi}{2n}\right)-\sin\left(\frac{\pi}{n}\right)}{\sin\left(\frac{(n-1)\pi}{2n}\right)+\sin\left(\frac{\pi}{n}\right)}, & \text{if } n \text{ is odd} \end{cases} \tag{2.17}$$

Nesterov with parameters:

$$\beta = \begin{cases} \frac{2}{7-\cos\left(\frac{2\pi}{n}\right)}, & \text{if } n \text{ is even} \\ \frac{2}{4-3\cos\left(\frac{(n-1)\pi}{n}\right)-\cos\left(\frac{2\pi}{n}\right)}, & \text{if } n \text{ is odd} \end{cases} \tag{2.18}$$

and

$$\gamma = \frac{1-2\sqrt{\beta}\sin\left(\frac{\pi}{n}\right)}{1+2\sqrt{\beta}\sin\left(\frac{\pi}{n}\right)} \tag{2.19}$$

achieves worst-case convergence rate ρ_N:

$$\rho_H = 1 - 2\sqrt{\beta}\sin\left(\frac{\pi}{n}\right) \tag{2.20}$$

Proof. We start by noting that function g has an infinite number of global minima corresponding to one solution rotated in the circle, meaning that g is not strongly convex. Therefore, taking any $x^\star$ and adding a scaled vector from the null space of Q results in a global minimum, i.e.,

$$\forall s \in \mathbb{R} : g(x^\star) = g(x^\star + s\mathbb{1}_n) \tag{2.21}$$

Using Lemma 2.1, the algorithms convergence rates depend on $m = \lambda_2(Q)$ and $L = \lambda_n(Q)$. Given the particular structure of Q, both m and L have closed-form expressions:

$$m = 2 - 2\cos\left(\frac{2\pi}{n}\right) \tag{2.22}$$

whereas

$$L = \begin{cases} 4, & \text{if } n \text{ is even} \\ 2 - 2\cos\left(\frac{(n-1)\pi}{n}\right), & \text{if } n \text{ is odd} \end{cases} \tag{2.23}$$

Using the optimal parameter value $\beta = \frac{4}{m+L}$ (see [33] or [34]) makes the convergence rate $\rho_G = \frac{\kappa-1}{\kappa+1}$ for $\kappa = L/m$. Replacing for the values of m and L from (2.22) and (2.23), we obtain the expressions in (2.13) and (2.14).

The Heavy-ball parameters can be found when the eigenvalues of T_2 and T_n have equal magnitude, resulting in $\beta = \frac{4}{\left(\sqrt{L}+\sqrt{m}\right)^2}$ and $\gamma = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$. Replacing the values of m and L and using the fact that $1 - \cos(x) = 2\sin^2(x/2)$ yields the values for the parameters in (2.15) and (2.16), and the worst convergence rate is given by $\rho_H = \sqrt{\gamma}$ which yields (2.17).

The last algorithm corresponding to Nesterov and solving such that three of the eigenvalues of T_2 and T_n have equal magnitude results in $\beta = \frac{4}{3L+m}$ and $\gamma = \frac{\sqrt{3\kappa+1}-2}{\sqrt{3\kappa+1}+2}$. Replacing the values of m and L and using the fact that $1 - \cos(x) = 2\sin^2(x/2)$, after some algebraic manipulations, yields the values for the parameters in (2.18) and (2.19), and using the expressions in [34], the worst convergence rate is given by $\rho_N = 1 - \frac{2}{\sqrt{3\kappa+1}}$ which yields (2.20). □

The main consequence of Theorem 2.1 is that if the number of transmitters to be desynchronized is known, it is possible to achieve optimal worst-case convergence rates. In a sense, the Fast-Desync algorithm in [29] using parameters $0 \leq \beta \leq \frac{1}{\max \lambda_i(Q)}$ and $\gamma^{(k)} = \frac{k-1}{k+2}$ is convergent because it uses the suboptimal maximum eigenvalue of Q of 4 for the even case (in the odd case this is a reasonable approximation only as $n \gg 1$) and disregards the minimum eigenvalue. In

the next theorem, we find a novel explicit formula for the convergence rate of this time-varying parameter version and show that the convergence rate is governed by $1 - 1/\kappa$ in comparison with $1 - 1/\sqrt{\kappa}$ when selecting the optimal fixed parameters as seen in Theorem 2.1. Thus, the current proposal of using the fixed-parameter Nesterov method for cases of known number of nodes n outperforms the current state-of-the-art.

Theorem 2.2 *The Nesterov method with* $\beta = \frac{1}{\max \lambda_i(Q)} = \frac{1}{4}$ *and* $\gamma^{(k)} = \frac{k-1}{k+2}$ *has a worst-case convergence rate, at time instant* k, $\lambda_{FD}^{(k)}$ *given by:*

$$\lambda_{FD}^{(k)} = \begin{cases} 1, & \text{if } k = 0 \\ \varphi, & \text{if } k = 1 \\ \left|\varphi\left((1+\gamma^{(k-1)})\lambda_{FD}^{(k-1)} - \gamma^{(k-1)}\lambda_{FD}^{(k-2)}\right)\right|, & \text{if } k \geq 2 \end{cases} \tag{2.24}$$

for $\varphi = 1 - \frac{1}{\kappa}$. *Moreover,* $\lambda_{FD}^{(k)}$ *is* $O(1/\kappa)$ *whereas* λ_N *is* $O(1/\sqrt{\kappa})$.

Proof. The Fast-Desync algorithm can be modeled through the Linear Time-Varying (LTV) model for dynamical systems using the following matrices:

$$A^{(k)} = \begin{bmatrix} (1+\gamma^{(k)})I_n & -\gamma^{(k)}I_n \\ I_n & 0_n \end{bmatrix}, B = \begin{bmatrix} -\beta I_n \\ 0_n \end{bmatrix}, \tag{2.25}$$

$$C^{(k)} = \begin{bmatrix} (1+\gamma^{(k)})I_n & -\gamma^{(k)}I_n \end{bmatrix}. \tag{2.26}$$

The transition matrix corresponding to the evolution of $x^{(k+1)} - x^\star$ is given by:

$$T^{(k)} = A^{(k)} + BQC^{(k)}. \tag{2.27}$$

Writing the error equation results in the relationship:

$$x^{(k+1)} - x^\star = \mathcal{T}^{(k+1)}(x^{(0)} - x^\star) \tag{2.28}$$

with matrix $\mathcal{T}^{(k+1)}$ being the transition matrix from the initial conditions to the current time instant, i.e.,

$$\mathcal{T}^{(k)} = T^{(k)} \cdots T^{(1)}. \tag{2.29}$$

All matrices $T^{(k)}$ admit the same eigenvalue decomposition using orthogonal matrix U, which allows writing $\mathcal{T}^{(k)}$ as

$$\begin{bmatrix} U & 0_n \\ 0_n & U \end{bmatrix} (A^{(k)} + B\Lambda C^{(k)}) \cdots (A^{(1)} + B\Lambda C^{(1)}) \begin{bmatrix} U & 0_n \\ 0_n & U \end{bmatrix}^\intercal, \tag{2.30}$$

since $U^\intercal U = I$. Since the spectrum of $\mathcal{T}^{(k)}$ is equivalent to that of $(A^{(k)} + B\Lambda C^{(k)}) \cdots (A^{(1)} + B\Lambda C^{(1)})$, we can study the spectrum of $(A_1^{(k)} + B_1\lambda_i C_1^{(k)}) \cdots (A_1^{(1)} + B_1\lambda_i C_1^{(1)})$ for all eigenvalues $\lambda_i(Q)$, and where matrices with subscript equal to one correspond to setting n to one.

As a consequence of the previous transformation, the convergence rate at each time instant k is given by the product of all matrices from 1 to k in the form:

$$T_1^{(k)} = \begin{bmatrix} (1+\gamma^{(k)})(1-\beta\lambda_i(Q)) & -\gamma^{(k)}(1-\beta\lambda_i(Q)) \\ 1 & 0 \end{bmatrix} \tag{2.31}$$

and, since $\beta = 1/4$, we get the worst-case value for $1 - \beta\lambda_i(Q) = 1 - \frac{1}{\kappa} = \varphi$. In addition, since $\gamma^{(1)} = 0$ (by definition of the algorithm as there is no momentum term at the first iteration), we get

$$T_1^{(1)} = \begin{bmatrix} \varphi & 0 \\ 1 & 0 \end{bmatrix} \tag{2.32}$$

which means that any matrix $\mathcal{T}_1^{(k)}$ will have the second column equal to zeros. As a consequence, the eigenvalues are always going to be a zero and the first entry of the matrix since it is a lower triangular. Therefore, the convergence rate for each time instant k evolves according to the sequence:

$$\lambda_{FD}^{(k)} = \varphi\left(\lambda_{FD}^{(k-1)} + \gamma^{(k-1)}(\lambda_{FD}^{(k-1)} - \lambda_{FD}^{(k-2)})\right). \tag{2.33}$$

Since the minimum of the eigenvalues is achieved after n iterations, i.e., when variable λ_{FD} goes from 1 to below zero, then we can propose the lower bound for the rate that corresponds to:

$$\lambda_{FD}^{(k)} = \varphi\left(\lambda_{FD}^{(k-1)} - \gamma^{(k-1)}\frac{1}{n}\right) \tag{2.34}$$

which has non-recursive definition given by

$$\lambda_{FD}^{(k)} = \varphi^k(1 - \frac{k}{n}) \tag{2.35}$$

thus, reaching the conclusion since the optimal fixed parameters achieves convergence rate of $1 - \frac{2}{\sqrt{3\kappa+1}}$. □

Remark 2.1 The result at Theorem 2.2 hints that the convergence rate is slower as the size of the network increases. However, using the expression for the exact convergence rate, we have depicted in Fig. 2.16 the comparison between the rates

for three sizes of networks. In each case, the rate achieved with the optimal fixed parameter is always faster than that using the time-varying version.

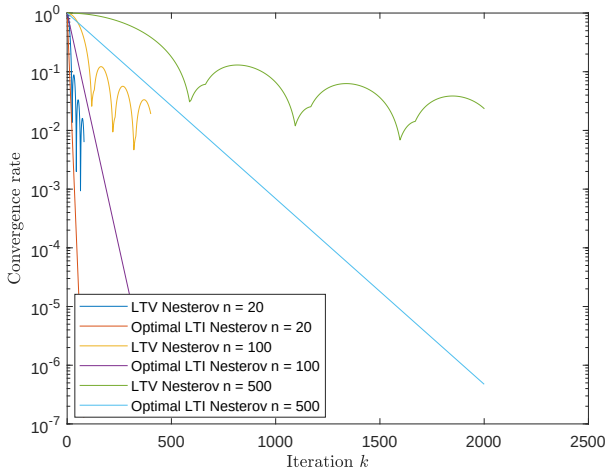


Fig. 2.16: Logarithmic evolution of the convergence rate for the time-varying Nesterov (LTVnesterov) and optimal fixed parameter Nesterov for networks sizes of 20, 100 and 500.

2.7 Desynchronization using Gauss-Seidel iterations

In the previous section, we have shown that the current state-of-the-art underperforms in comparison with setting an optimal fixed-parameter for the Nesterov method with the expressions being given in Theorem 2.1. However, in some scenarios, it will be infeasible to have all transmitters know the number of nodes in the entire network. In this section, we present results when viewing the problem in (2.8) as the solution of a linear equation, as was done in [35] for the PageRank problem. We first present the Gauss-Seidel algorithm for completeness and then apply it to $\nabla g(\phi) = 0$ in order to obtain a faster update rule without the need to set up parameters.

For a general system $Ax = b$, with $A = L + D + U$ decomposed in lower, diagonal and upper matrices, the Gauss-Seidel method has the following update rule:

$$x(k+1) = (L+D)^{-1}(b - Ux(k)) \tag{2.36}$$

which, by taking advantage of the triangular form of $L + D$, can be sequentially updated for each i using forward substitution, leading to the desynchronization

algorithm:

$$\begin{aligned}\phi_1^{(k+1)} &= \frac{1}{2}\left(1 - \phi_2^{(k)} - \phi_n^{(k)}\right)\\ \phi_i^{(k+1)} &= \frac{1}{2}\left(-\phi_{i-1}^{(k+1)} - \phi_{i+1}^{(k)}\right), 2 \le i \le n-1\\ \phi_n^{(k)} &= \frac{1}{2}\left(-1 - \phi_1^{(k+1)} - \phi_{n-1}^{(k+1)}\right)\end{aligned} \tag{2.37}$$

which requires communication with the immediate neighbors akin the original problem and exploits the inherent sequential behavior of the Desync algorithm. In this setup, nodes use the most updated values for the phases.

The next theorem provides the exponential rate of convergence of the iteration in (2.37).

Theorem 2.3 (Convergence Rate of Gauss-Seidel) The iterative method (2.37) asymptotically converges to a desynchronization state with exponential convergence rate λ_{GS}, i.e.,

$$\phi^{(k+1)} - \phi^\star \le \lambda_{GS}^{k+1}(\phi^{(0)} - \phi^\star) \tag{2.38}$$

where

$$\lambda_{GS} = |\lambda_2(T_{GS})|$$

and

$$T_{GS} = \sum_{j=0}^{n-1}\left(\frac{1}{2}\right)^{j+1} E^j E^\intercal, \tag{2.39}$$

$$E = \begin{bmatrix} \mathsf{0}_{n-1}^\intercal & 0 \\ I_{n-1} & \mathsf{0}_{n-1}\end{bmatrix} + \mathsf{e}_n \mathsf{e}_1^\intercal \tag{2.40}$$

with $|\lambda_n(T_{GS})| \le |\lambda_{n-1}(T_{GS})| \le \cdots \le |\lambda_1(T_{GS})|$.

Proof. The inequality in (2.38) comes directly from seeing the Gauss-Seidel algorithm as a linear time-invariant system where $T_{GS} := -(D+L)^{-1}U$ is the transition matrix for a general linear equality $Ax = b$ as in (2.36).

The first step in the proof consists of writing the matrix T_{GS} for the Desync problem. Given the partition $A = L + D + U$, the matrix T_{GS} has the following expression:

$$\begin{aligned}T_{GS} &= (2I_n - E)^{-1}E^\intercal\\ &= \frac{1}{2}(I_n - \frac{1}{2}E)^{-1}E^\intercal\end{aligned} \tag{2.41}$$

with the strictly lower triangular matrix E being defined as in (2.40).

We remark that

$$(I_n + N)^{-1} = I_n + \sum_{k=1}^{n-1} (-1)^k N^k$$

for a general strictly lower triangular matrix N. Using the above equality and after some algebraic manipulations, (2.41) simplifies to (2.39).

The second step is to show stability by proving that the spectral radius of T_{GS} is within the unit circle, i.e., $\rho(T_{GS}) \leq 1$. Matrix T_{GS} is row stochastic since its elements are trivially nonnegative and

$$
\begin{aligned}
T_{GS}\mathsf{1}_n &= \left(\sum_{j=0}^{n-1}\left(\frac{1}{2}\right)^{j+1} E^j\right)(\mathsf{1}_n + D^\intercal \mathsf{e}_n) \\
&= \frac{1}{2}(\mathsf{1}_n + D^\intercal \mathsf{e}_n) + \frac{1}{2^2}\begin{bmatrix} 0 \\ 2 \\ \mathsf{1}_{n-3} \\ 3 \end{bmatrix} + \frac{1}{2^3}\begin{bmatrix} \mathsf{0}_2 \\ 2 \\ \mathsf{1}_{n-3} \end{bmatrix} \\
&\quad + \frac{1}{2^4}\begin{bmatrix} \mathsf{0}_3 \\ 2 \\ \mathsf{1}_{n-4} \end{bmatrix} + \cdots + \frac{1}{2^n}\begin{bmatrix} \mathsf{0}_{n-1} \\ 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ \frac{1}{2} + \frac{2}{2^2} \\ \sum_{j=1}^{2} \frac{1}{2^j} + \frac{2}{2^3} \\ \vdots \\ \sum_{j=1}^{n-2} \frac{1}{2^j} + \frac{2}{2^{n-1}} \\ \frac{3}{2^2} + \sum_{j=3}^{n-1} \frac{1}{2^j} + \frac{2}{2^n} \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ \frac{1}{2} + \frac{1}{2} \\ 1 - \frac{1}{2^2} + \frac{1}{2^2} \\ \vdots \\ 1 - \frac{1}{2^{n-2}} + \frac{1}{2^{n-2}} \\ \frac{3}{2^2} - \frac{1}{2^{n-1}} + \frac{1}{2^2} + \frac{1}{2^{n-1}} \end{bmatrix} \\
&= \mathsf{1}_n.
\end{aligned}
\tag{2.42}
$$

By noticing that the first row is equal to two times the nth minus the $n-1$th rows, the following equality is true

$$UT_{GS} = \begin{bmatrix} 0 & 0_{n-1}^{\intercal} \\ 0_{n-1} & T_{GS}^{sub} \end{bmatrix}$$

where

$$U = \left[\begin{array}{c|c} 1 & 0_{n-3}^{\intercal}\ 1\ -2 \\ \hline 0_{n-1} & I_{n-1} \end{array}\right]$$

and the matrix T_{GS}^{sub} is a submatrix of T_{GS} obtained by removing the first row and column. Since the matrix U implements elementary row operations, the multiplication has no effect on the spectra of T_{GS}, meaning that $\lambda_i(T_{GS}) = \lambda_i(UT_{GS}), \forall 1 \leq i \leq n$. In particular, from the format of UT_{GS}, it follows that

$$\{\lambda_i(T_{GS}), 1 \leq i \leq n\} = \{\lambda_i(T_{GS}^{sub}), 1 \leq i \leq n-1\} \cup \{0\}$$

Similarly, T_{GS}^{sub} remains row stochastic and its support graph is strongly connected since the last row and columns are full (meaning that the correspondent node would have edges to and from all the remaining nodes in the graph). As a consequence, $\lambda_1(T_{GS}^{sub}) = 1$ and $|\lambda_2(T_{GS}^{sub})| < 1$ by the Perron-Frobenius Theorem and the conclusion about the convergence rate also follows. □

2.8 Simulation Results for Desynchronization Algorithm

In this section, simulations are presented using the toolbox in [36] in order to illustrate whether the theoretical rates represent an advantage in practical sense. All simulations have considered an initial starting phase state $\phi^{(0)} = 1_n/n$ corresponding to all nodes sharing the same phase and being completely synchronized.

Figure 2.17 compares the evolution of the convergence rates for Gradient with $\beta = \frac{1}{4}$ (which is equivalent to the PCO-based Desync when $\beta = \frac{\alpha}{2}$ as demonstrated in [29]), Gauss-Seidel, Nesterov and Heavy-Ball for the fixed parameters $\beta = \frac{1}{4}, \gamma = \frac{1}{2}$. This hints at the fact that indeed considering both optimization methods and iterative algorithms for solving linear equations yields improvements in performance in comparison with the PCO model. As expected, as the number of nodes increases, so does the convergence rate, which is approaching the unity as n grows to infinity.

An important remark is that the version of the Nesterov method proposed in [29] has time-varying parameters (in particular $\gamma = \frac{k-1}{k+2}$) that might contribute to increase the speed of convergence. In order to compare the method proposed in [29], a simulation of a $n = 5$ node network was conducted and the logarithm of the error norm is presented is Fig. 2.18.

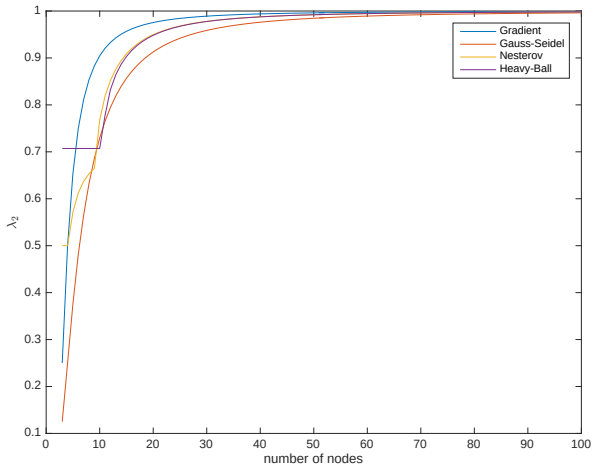


Fig. 2.17: The convergence rate λ_2 for the different algorithms depending on the number of nodes n.

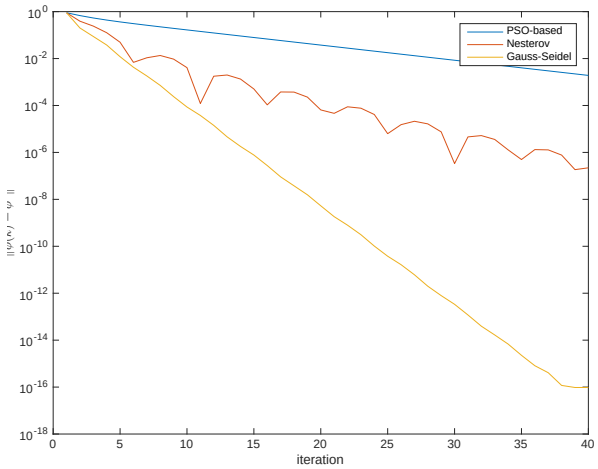


Fig. 2.18: Logarithmic evolution of the error norm for the PCO-based, Nesterov and Gauss-Seidel algorithms.

Figure 2.18 shows that the Gauss-Seidel iteration achieves a faster convergence at a fixed rate in comparison with the algorithm in [29]. Both methods present a clear advantage when compared to the PCO-based method with parameter $\alpha = 0.2$. Additional simulations were conducted to assess the potential advantage of the Nesterov method with a time-varying parameter. A network of $n = 20$ nodes was also simulated and the results are depicted in Fig. 2.19.

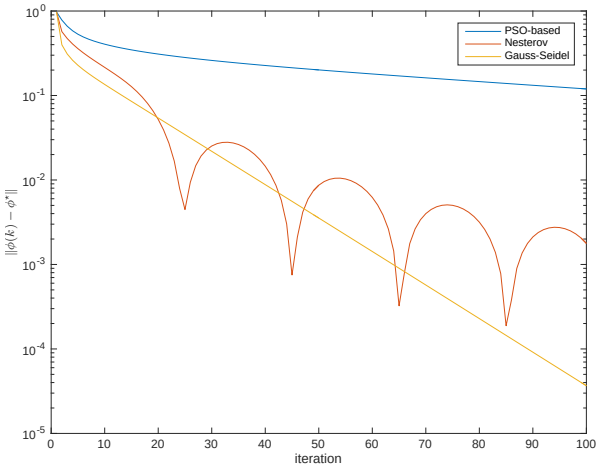


Fig. 2.19: Logarithmic evolution of the error norm for the PCO-based, Nesterov and Gauss-Seidel algorithms for the 20 node network.

The main observation from the evolution of the error in Figs. 2.18 and 2.19 is that as n increases, the behavior of the error norm changes. For small networks, the Gauss-Seidel method outperforms the Nesterov algorithm. When increasing n, the error decreases faster using the Gauss-Seidel up to a small tolerance and then the Nesterov method becomes faster. The observed oscillations tend to fade for larger n.

The simulations presented so far only considered algorithms for which there is no knowledge of the number of transmitters n. In the remainder of the simulations, both the Gauss-Seidel and the LTV version of Nesterov from [29] are compared against the Nesterov and Heavy-Ball algorithms selecting optimal parameters from Theorem 2.1. The initial state is set to $1_n/n$ and we report the same error function as previously.

In Fig. 2.20 it is depicted the error evolution for a small size network of 6 nodes. In small networks, simulations show that the LTV Nesterov has the worst performance while, as expected, the Heavy-Ball algorithm has the best performance given that it is the fastest for quadratic functions of all methods. As proven in Theorem 2.2, the convergence of LTV Nesterov is not monotonous. The Gauss-Seidel outperforms the optimal Nesterov method but with a similar behavior.

In order to test for medium-sized networks, a similar simulation was conducted for a 20 node network. The Gradient with optimal parameter is the slowest and the oscillatory behavior of the LTV Nesterov is heightened and still underperfoms in comparison with the Gauss-Seidel. For this case, both the Nesterov and the Heavy-Ball methods achieve a better convergence since it scales with $1/\sqrt{\kappa}$. Another curious fact that starts to emerge is the bad performance of the Heavy-Ball in the beginning of the simulation.

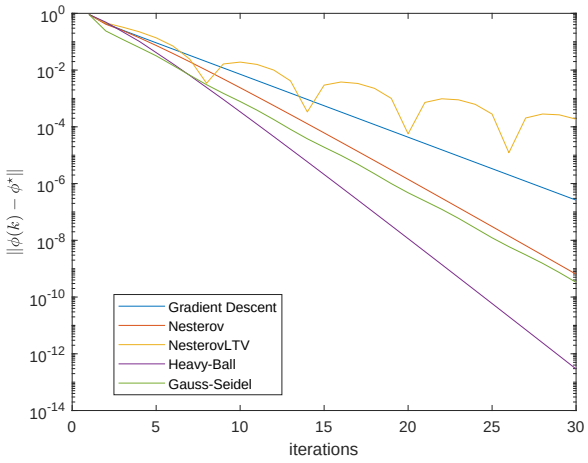


Fig. 2.20: Logarithmic evolution of the error norm for the PCO-based (Gradient Descent), Nesterov, LTV Nesterov, Heavy-Ball and Gauss-Seidel algorithms for a 6 node network.

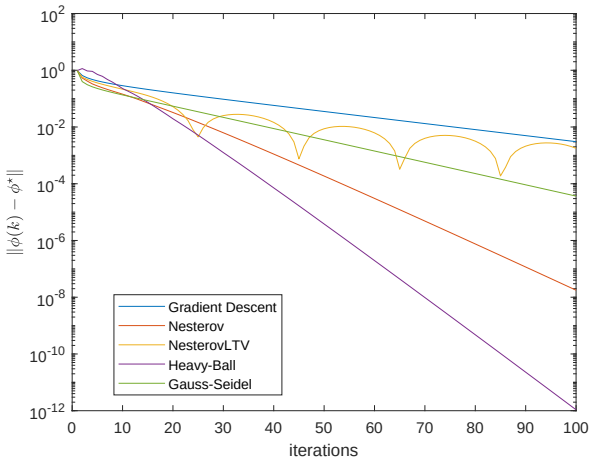


Fig. 2.21: Logarithmic evolution of the error norm for the PCO-based (Gradient Descent), Nesterov, LTV Nesterov, Heavy-Ball and Gauss-Seidel algorithms for a 20 node network.

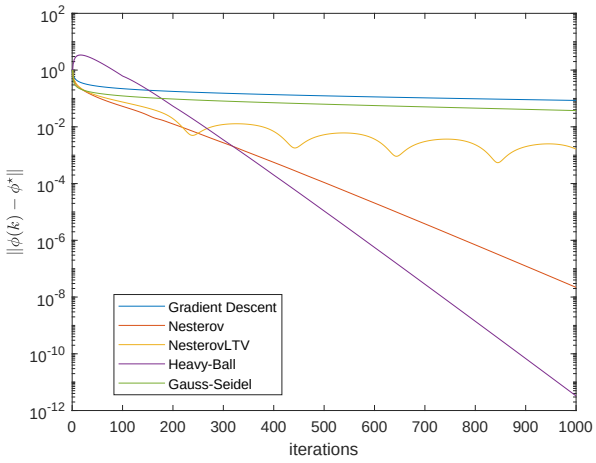


Fig. 2.22: Logarithmic evolution of the error norm for the PCO-based (Gradient Descent), Nesterov, LTV Nesterov, Heavy-Ball and Gauss-Seidel algorithms for a 200 node network.

In order to illustrate the behavior of the algorithms for large networks, a 200 transmitter network is also simulated with the error evolution being presented in Fig. 2.22. For such cases, the behavior of the Gauss-Seidel approaches the optimal Gradient Descent albeit faster but both underperform in comparison with the Nesterov and Heavy-Ball. The LTV Nesterov has a performance in between these two classes and still maintains its oscillatory behavior. The initial increase in the error for the Heavy-Ball is noticeable which might discourages its application for large networks and error tolerances around 10^{-3} or 10^{-4} for which the Nesterov method produces equivalent results without the initial increase in the error.

2.9 Resilient Data Acquisition with Reputation-Based Consensus

One important aspect of a sensor network is to be able to cope with incorrect data generated by some of its nodes or by an attacker that might spoof messages. In practice, if we have multiple sensors measuring common values, they might perform a consensus protocol to achieve a better estimate. We note that simply removing outliers or attackers can also be accomplished using a similar strategy as was presented in [37] or by assessing the variance of transmitted data [38]. In this section, we address the case where the WSN has nodes measuring temperature, humidity and other parameters to provide data for estimating the risk distribution.

Let us consider a network of agents forming a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{(k)})$, where $\mathcal{V}$ is a nonempty set of *nodes*, and $\mathcal{E}^{(k)} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of *edges* and time k and with initial states $x_v^{(0)} \in \mathbb{R}$, for $v \in \mathcal{V}$. In the non-attacked scenario, agents can reach consensus through the use of a distributed linear iterative algorithm with dynamics given by:

$$x^{(k+1)} = W^{(k)} x^{(k)}, \tag{2.43}$$

where $x^{(k)}$ is the vector collecting the n agents states at time step $k > 0$, and the matrix $W^{(k)} \in \mathbb{R}^{n \times n}$ is such that: (i) $W_{u,v} = 0$ if the agents u and v do not communicate, and (ii) the agents converge to the same quantity, i.e., $\lim_{k \to \infty} x^{(k)} = x^{\infty}$.

We consider a set of attacked agents $\mathcal{A} \subset \mathcal{V}$. If the agents $a \in \mathcal{A}$ do not follow the consensus update rule, then each regular agent, $v \in \mathcal{V} \setminus \mathcal{A}$, should identify and discard the attacked agents' values.

The assumption we made is typical in the state-of-the-art methods to ensure resilient consensus. We remark that the assumption we do make is equivalent to the r-robustness ($(r, 1)$-robust) defined in [39].

The previous assumption is reasonable as each regular agent has to divide his neighbors into the set of normal nodes and the set of attacked ones, comparing the information that it receives. Thus, if the majority of the information is not legitimate there is no redundancy to allow identifying the attacked neighbors.

2.9.1 Attacker model

Subsequently, we consider an attacker that may corrupt the state of the nodes in the subset $\mathcal{A}$ by adding an unbounded signal. The attacked dynamics are a corrupted version of (2.43) as follows:

$$x^{(k+1)} = W^{(k)} x^{(k)} + \Delta^{(k)}, \tag{2.44}$$

where $\Delta^{(k)} \in \mathbb{R}^n$, which entails the assumption that the attacker cannot corrupt the communication between nodes to send different messages to distinct neighbors. Observe that this assumption allows the attacker to change the state of a subset of agents to (possibly) different values. Also, the attacker cannot create artificial nodes nor change the network topology, i.e., the structure of $W^{(k)}$, and the dimension n are fixed in (2.44).

2.9.2 Reputation-based consensus *(RepC)*

Next, we propose a reputation-based consensus algorithm (RepC). The idea is that each time an agent obtains information from its neighbors, it measures how discrepant is, on average, the state from one neighbor regarding the states of the remaining ones and its own state. This metric also translates how much can a single node influence the entire network [40]. The RepC is composed by two phases: (i) **identification of the attacked nodes**; (ii) computation of the **consensus**.

We propose a fully distributed discrete-time consensus that works for synchronous and asynchronous networks. Agents only need a low computational power to do computations with the neighbors' values. This method can also be complemented with the one in [41], [42] to allow recovery.

2.9.2.1 Synchronous communication *RepC*

Given the maximum number of allowed attacked nodes f, the identification of the attacked nodes is performed by the iterative scheme in Algorithm 1.

Note that, in Algorithm 1, $c_{ij}^{(k+1)} = 0$ if $j \notin \mathcal{N}_i$, and $c_{ii}^{(k+1)} = 1$, and where $c_{ij}^{(0)} = 1$ for $j \in \bar{\mathcal{N}}_i$ and $c_{ij}^{(0)} = 0$ otherwise, and $x_i^{(0)}$ is the initial value of each agent i. Further, $\varepsilon \in]0, 1[$ is a *confidence factor* which guarantees that each agent does not discard immediately values that are discrepant from its neighbors' average.

Notice that the selected value for ε must be small to have a negligible impact on the agents' consensus states. Also, a large ε may cause an agent to do not detect an attacked neighbor. This, in turn, makes the asymptotic consensus deviate from the consensus without attacked agents toward a combination of the attacked agents' asymptotic states. We illustrate this property in Section 2.10.

The proposed method computes a weighted average of the agents' values. So, the final state is a convex combination of the initial ones.

2.9.2.2 Asynchronous communication *RepC*

The asynchronous version of algorithm RepC consists of, at each instance of time, the agents that communicate, $\mathcal{A}' \subset \mathcal{A}$, follow Equation (2.45), where $\bar{\mathcal{N}}_i$ is replaced by $\bar{\mathcal{N}}_i \cap \mathcal{A}'$.

The iterative scheme (2.45) may also be used in the scenario where the network of agents evolves with time. The results in Section 2.9.2 can be restated for this scenario by considering that the set of neighbors of a node is dynamic, and by verifying, at each time, that each agent has more than two neighbors and more than half of them

Algorithm 1 Synchronous communication RepC

input: Network of agents $\mathcal{G} = (\mathcal{V}, \mathcal{E}(k))$, agents initial states $x^{(0)}$, number of time steps $\mathbf{T}$, and confidence factor $\varepsilon \in]0, 1[$
output: agents final states $x^{(\mathbf{T})}$
for $k = 1, \ldots, \mathbf{T}$ **do**
    for $i = 1, \ldots, |\mathcal{V}|$ **do**

Reputation update:

$$\tilde{c}_{ij}^{(k+1)} = \begin{cases} 1 - \sum\limits_{v \in \tilde{\mathcal{N}}_i} \dfrac{|x_j^{(k)} - x_v^{(k)}|}{|\mathcal{N}_i|}, & j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases}$$

Normalized Reputation update:

$$\tilde{\tilde{c}}_{ij}^{(k+1)} = \begin{cases} \dfrac{\tilde{c}_{ij}^{(k+1)} - \min\limits_{v \in \tilde{\mathcal{N}}_i}{}^{\mathrm{f}}\, \tilde{c}_{iv}^{(k+1)}}{\max\limits_{v \in \tilde{\mathcal{N}}_i} \tilde{c}_{iv}^{(k+1)} - \min\limits_{v \in \tilde{\mathcal{N}}_i}{}^{\mathrm{f}}\, \tilde{c}_{iv}^{(k+1)}}, & i \neq j \\ 1, & \text{otherwise} \end{cases} \tag{2.45}$$

Normalized Reputation update with confidence ε:

$$c_{ij}^{(k+1)} = \begin{cases} \tilde{\tilde{c}}_{ij}^{(k+1)}, & \text{if } \tilde{\tilde{c}}_{ij}^{(k+1)} > 0, \\ \varepsilon^{k+1}, & \text{otherwise} \end{cases}$$

Consensus state update:

$$x_i^{(k+1)} = \sum_{j \in \tilde{\mathcal{N}}_i} c_{ij}^{(k)} x_j^{(k)} \Big/ \sum_{j \in \tilde{\mathcal{N}}_i} c_{ij}^{(k)}$$

    end for
end for

are regular agents. In Section 2.10.4 and 2.10.5, we illustrate the dynamic network of agents and dynamic network with noisy agents scenarios.

First, we show that RepC converges. To simplify the proof, we assume that we are in the scenario of synchronous communication. The general proof follows the same steps, but it is more complex, and it needs more complex notation to denote the set of neighbors with which a node communicates at each time. Let us define $\tilde{x}_i^{(0)} = (x_i^{(0)} - x_{\min}^{(0)}) \big/ (x_{\max}^{(0)} - x_{\min}^{(0)})$, which yields a re-scaling of the agents' states that simplifies the methods' proof of convergence. This re-scaling is to make the proof less extensive.

Further, in the following proofs and for technical reasons, we assume that each attacked agent shares a state converging to some value. Although, in practice, the algorithm is still effective under other circumstances, as we illustrate in Section 2.10.

Moreover, to have guarantees of resilient consensus and derive theoretical result, we make the following additional assumptions:

- For each regular agent, $v \in \mathcal{V} \setminus \mathcal{A}$, more than half of the neighbors are regular agents, i.e., $|\mathcal{N}_v \cap \mathcal{A}| < |\mathcal{N}_v|/2$ and the network of normal nodes is connected.
- The attack cannot target the initial state, i.e., $\Delta^{(0)} = 0$ in (2.44). This scenario would be undetectable. The sequences of state values for the attacked version would be the same as a normal execution of the algorithm with the attack value as the initial state.

Lemma 2.2 *If for any $i \in \mathcal{V}$ we have that $|\mathcal{N}_i| > 2$, then each agent that follows the iterative scheme in* (2.45) *converges.*

Proof. For the proof, suppose that the initial states are re-scaled to be in $[0, 1]$. We have that $\left\|x^{(k+1)} - x^{(k)}\right\|_\infty = \max_i \left|x_i^{(k+1)} - x_i^{(k)}\right|$ and, hence, assuming w.l.o.g. that:
(†) $\|c_i^{(k)}\|_1 \leq \|c_i^{(k+1)}\|_1$

$$
\begin{aligned}
\left|x_i^{(k+1)} - x_i^{(k)}\right| &\overset{\substack{\text{consensus state}\\ \text{update definition}}}{=} \left|\frac{c_i^{(k+1)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} - \frac{c_i^{(k)} \cdot x^{(k-1)}}{\|c_i^{(k)}\|_1}\right| \\
&= \left|\frac{c_i^{(k+1)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} - \frac{c_i^{(k)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} + \frac{c_i^{(k)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} - \frac{c_i^{(k)} \cdot x^{(k-1)}}{\|c_i^{(k)}\|_1}\right| \\
&\overset{(\dagger)}{\leq} \left|\frac{c_i^{(k+1)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} - \frac{c_i^{(k)} \cdot x^{(k)}}{\|c_i^{(k+1)}\|_1} + \frac{c_i^{(k)} \cdot x^{(k)}}{\|c_i^{(k)}\|_1} - \frac{c_i^{(k)} \cdot x^{(k-1)}}{\|c_i^{(k)}\|_1}\right| \qquad (2.46) \\
&\leq \frac{\max_{j\in\mathcal{N}_i} |c_{ij}^{(k+1)} - c_{ij}^{(k)}|}{\|c_i^{(k+1)}\|_1} + \frac{\max_{j\in\mathcal{N}_i} |x_j^{(k)} - x_j^{(k-1)}|}{\|c_i^{(k)}\|_1} \\
&\overset{(\dagger)}{\leq} \frac{\max_{j\in\mathcal{N}_i} |c_{ij}^{(k+1)} - c_{ij}^{(k)}|}{\|c_i^{(k)}\|_1} + \frac{\max_{j\in\mathcal{N}_i} |x_j^{(k)} - x_j^{(k-1)}|}{\|c_i^{(k)}\|_1},
\end{aligned}
$$

because we are assuming that $x_v^{(m)}, c_{ij}^{(m)} \in]0, 1[$. Now we need to compute $\max_{j\in\mathcal{N}_i} |c_{ij}^{(k+1)} - c_{ij}^{(k)}|$. First, we notice that we cannot have that $\max_{j\in\mathcal{N}_i} |c_{ij}^{(k+1)} - c_{ij}^{(k)}| = |\varepsilon^{k+1} - \varepsilon^k|$, because there is always a $j \in \mathcal{N}_i$ such that $c_{ij}^{(k+1)} > \varepsilon^{k+1}$ and all the other $k \neq j \in \mathcal{N}_i$ are such that $c_{ij}^{(k+1)} \geq \varepsilon^{k+1}$. Therefore, we need to consider only three cases:

1. $c_{ij}^{(k+1)} = \varepsilon^{k+1}$ and $c_{ij}^{(k)} = \tilde{c}_{ij}^{(k)}$;
2. $c_{ij}^{(k+1)} = \tilde{c}_{ij}^{(k+1)}$ and $c_{ij}^{(k)} = \varepsilon^k$;
3. $c_{ij}^{(k+1)} = \tilde{c}_{ij}^{(k+1)}$ and $c_{ij}^{(k)} = \tilde{c}_{ij}^{(k)}$.

For case 1) we have that $\left|c_{ij}^{(k+1)} - c_{ij}^{(k)}\right| = \left|\varepsilon^{k+1} - \tilde{c}_{ij}^{(k)}\right| < \left|\tilde{c}_{ij}^{(k+1)} - \tilde{c}_{ij}^{(k)}\right|$, since $c_{ij}^{(k+1)} = \varepsilon^{k+1}$ implies that $\tilde{c}_{ij}^{(k+1)} \leq 0$. Using the same reasoning, for case 2), we have that

$\left|c_{ij}^{(k+1)} - c_{ij}^{(k)}\right| = \left|\tilde{c}_{ij}^{(k+1)} - \varepsilon^k\right| < \left|\tilde{c}_{ij}^{(k+1)} - \tilde{c}_{ij}^{(k)}\right|$. We only need to compute 3)

$$\begin{aligned}
\left|\tilde{c}_{ij}^{(k+1)} - \tilde{c}_{ij}^{(k)}\right| &= \frac{\left|\tilde{c}_{ij}^{(k+1)} - \tilde{c}_{ij}^{(k)}\right|}{\max_{v\in\bar{\mathcal{N}}_i} \tilde{c}_{iv}^{(k+1)} - \min^{\mathrm{f}}{}_{v\in\bar{\mathcal{N}}_i} \tilde{c}_{iv}^{(k+1)}} \\
&\leq |\tilde{c}_{ij}^{(k+1)} - \tilde{c}_{ij}^{(k)}| \\
&= \frac{1}{|\bar{\mathcal{N}}_i|} \sum_{v\in\bar{\mathcal{N}}_i} \left(|x_j^{(k)} - x_v^{(k)}| - |x_j^{(k-1)} - x_v^{(k-1)}|\right) \\
&\overset{\text{by def. of } \alpha}{=} \frac{1}{|\bar{\mathcal{N}}_i|} |\bar{\mathcal{N}}_i| \left(|x_j^{(k)} - x_\alpha^{(k)}| - |x_j^{(k-1)} - x_\alpha^{(k-1)}|\right) \qquad (2.47) \\
&= \left(|x_j^{(k)} - x_\alpha^{(k)}| - |x_\alpha^{(k-1)} - x_j^{(k-1)}| + |x_\alpha^{(k-1)} - x_j^{(k-1)}| - |x_j^{(k-1)} - x_\alpha^{(k-1)}|\right) \\
&\leq \left(|x_j^{(k)} - x_j^{(k-1)}| + |x_\alpha^{(k)} - x_\alpha^{(k-1)}|\right) \\
&\leq 2 \max_{j\in\bar{\mathcal{N}}_i} |x_j^{(k)} - x_j^{(k-1)}|,
\end{aligned}$$

where $\alpha = \arg\max_{v\in\bar{\mathcal{N}}_i} \left(|x_j^{(k)} - x_v^{(k)}| - |x_j^{(k-1)} - x_v^{(k-1)}|\right)$. Now, plugging (2.47) in (2.46), we have that

$$\begin{aligned}
\|x^{(k+1)} - x^{(k)}\|_\infty &\leq \frac{3}{\|c_i^{(k))}\|_1} \max_{j\in\bar{\mathcal{N}}_i} |x_j^{(k)} - x_j^{(k-1)}| \\
&\leq \frac{3}{\|c_i^{(k)}\|_1} \|x^{(k)} - x^{(k-1)}\|_\infty \leq \frac{3}{|\bar{\mathcal{N}}_i|} \|x^{(k)} - x^{(k-1)}\|_\infty
\end{aligned}$$

Therefore, the iterative scheme converges whenever $\frac{3}{|\mathcal{N}_i|+1} < 1$, which is equivalent to $|\mathcal{N}_i| > 2$. □

A regular agent should assess at least 3 states to distinguish whether nodes are following the consensus update rule or not. With 2 neighbors, their reputation can alternate between iterations.

The previous result states that RepC converges. It is still missing to show that each regular agent converges to the same value, i.e., all regular agents *agree*. The next lemma assesses that: (i) either an agent v converges to a unique value; (ii) or for any other agent, the reputation of agent v is zero, i.e. $c_{uv}^\infty = 0$.

Lemma 2.3 *Consider the iterative scheme 2.45. For any agent $j \in \mathcal{V}$ one of the following holds:*
(i) $\lim_{k\to\infty} x_j^{(k)} = x^\infty$*; (ii)* $\forall_{i\in\mathcal{N}_j} \lim_{k\to\infty} c_{ij}^{(k)} = 0$ *(neighbors of j assign it reputation zero).*

Proof. By Lemma 2.2, we have that (2.45) converges for each agent $i \in \mathcal{V} \setminus \mathcal{A}$. We just need to show that for a given node $u \in \mathcal{V} \setminus \mathcal{A}$ and for each of its neighbors $v \in \mathcal{N}_u$ either (i) or (ii) happens. Let $u \in \mathcal{V} \setminus \mathcal{A}$ and $v \in \mathcal{N}_u$, we show by induction on the number of neighbors of u, $|\mathcal{N}_u|$, that for each neighbor either its reputation is zero or it converges to the same value as u. The basis is when $|\mathcal{N}_u| = 1$, and we have that $x_u^\infty = \frac{x_u^\infty + c_{uu}^\infty x_v^\infty}{1+c_{uv}^\infty}$. Thus, either $c_{uv}^\infty = 0$, or $c_{uv}^\infty > 0$ and $x_v^\infty = x_u^\infty$. When $|\mathcal{N}_u| = N + 1$, we have that $x_u^\infty = \sum_{j\in\bar{\mathcal{N}}_u} c_{uj}^\infty x_j^\infty \Big/ \sum_{j\in\bar{\mathcal{N}}_u} c_{uj}^\infty$. Since the reputation that u assigns to itselft is $c_{uu}^\infty = 1$, we can rewrite the previous as

$$x_u^\infty = \frac{x_u^\infty + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty x_j^\infty + c_{uv}^\infty x_v^\infty}{1 + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty + c_{uv}^\infty} \tag{2.48}$$

Further, using the induction hypothesis, either (i) or (ii) is true for any set of N neighbors of u. Hence, for $j \in \mathcal{N}_u \setminus \{v\}$ either $x_j^\infty = x^\infty$ or $c_{uj}^\infty = 0$. In any of the cases, we have that

$$x_u^\infty + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty x_j^\infty = x^\infty \left(1 + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty\right) \tag{2.49}$$

By replacing (2.49) in (2.48), it follows that

$$x_u^\infty = x^\infty = \frac{x^\infty \left(1 + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty\right) + c_{uv}^\infty x_v^\infty}{\left(1 + \sum_{j\in\mathcal{N}_u\setminus\{v\}} c_{uj}^\infty\right) + c_{uv}^\infty},$$

implying that either $c_{uv}^\infty = 0$ or $c_{uv}^\infty > 0$ and $x_v^\infty = x_u^\infty = x^\infty$. By transitivity, we can apply the same to each neighbor of all neighbors of u, and so forth. Thus, the result yields for all $i \in \mathcal{V} \setminus \mathcal{A}$. □

As a corollary, the detection yields no false positives.

Corollary 2.1 *Let $v \in \mathcal{V} \setminus \mathcal{A}$ and $u \in \mathcal{V}$. By using the iterative scheme* (2.45)*, if $c_{uv}^\infty = 0$ then $u \in \mathcal{A}$.*

The proof of Lemma 2.2 also hints that half of each agent's neighbors should not be under attack so that each normal node identifies the attacked agents correctly. This property emerges from the following.

Lemma 2.4 *Suppose that the iterative scheme* (2.45) *converges to a value different from that broadcasted by the attacked agents. If for each agent $i \in \mathcal{V} \setminus \mathcal{A}$, less than half of its neighbors are not attacked agents, i.e. $|\bar{\mathcal{N}}_i \cap \mathcal{A}| < |\bar{\mathcal{N}}_i \setminus \mathcal{A}|$, then $\lim_{k\to\infty} c_{ia}^{(k)} = 0$, for $a \in \mathcal{A}$ and $\lim_{k\to\infty} c_{iv}^{(k)} = 1$ for $v \in \bar{\mathcal{N}}_i \setminus \mathcal{A}$.*

Proof. By Lemma 2.2, we have that the each regular agent using the iterative scheme in (2.45) converges to x^∞. Let y denote the value that all the attacked agents in $\mathcal{A}$ share with the neighbors. For a regular agent ($i \notin \mathcal{A}$), an attacked agent's reputation ($a \in \mathcal{A}$) satisfies

$$\begin{aligned}\tilde{c}_{ia}^{\infty} = \lim_{k\to\infty} \tilde{c}_{ia}^{(k)} &= 1 - \frac{1}{|\mathcal{N}_i|} \sum_{v\in\mathcal{N}_i} \left| y - \lim_{k\to\infty} x_v^{(k)} \right| \\ &= 1 - \frac{|\mathcal{N}_i \setminus \mathcal{A}|}{|\mathcal{N}_i|} |y - x^\infty| ,\end{aligned}$$

and the limit of the reputation of a regular user, $j \notin \mathcal{A}$, is given as

$$\tilde{c}_{ij}^{\infty} = \lim_{k\to\infty} \tilde{c}_{ij}^{(k)} = 1 - \frac{|\mathcal{N}_i \cap \mathcal{A}|}{|\mathcal{N}_i|} |x^\infty - y| .$$

Since $|\mathcal{N}_i \cap \mathcal{A}| < |\mathcal{N}_i \setminus \mathcal{A}|$ and $y \neq x^\infty$, then $\tilde{c}_{ij}^{\infty} > \tilde{c}_{ia}^{\infty}$, and because reputations values are normalized to be between 0 and 1, we have that, for all i, $1 = c_{ij}^{\infty} > c_{ia}^{\infty} = 0$. □

Now, we need to show that a regular agent using RepC identifies the attacked neighbors and study the method's convergence rate.

Lemma 2.5 *Let $v \in \mathcal{V} \setminus \mathcal{A}$ and $u \in \mathcal{V}$. By using the iterative scheme* (2.45)*, if $u \in \mathcal{A}$ and $|\mathcal{A}| \leq f$ then $c_{uv}^{\infty} = 0$.*

Proof. Let $a \in \mathcal{A}$ be an attacked node. We want to show that for a regular agent, $v \in \mathcal{V} \setminus \mathcal{A}$, the reputation of agent a strictly decreases with time. Let $v \in \mathcal{V} \setminus \mathcal{A}$, we have that $|x_a - x_v^{(k+1)}| - |x_a - x_v^{(k)}| \leq |x_v^{(k+1)} - x_v^{(k)}|$. □

Proposition 2.2 *Consider the iterative scheme in* (2.45) *and let $N = \min_{i\in\mathcal{V}} |\mathcal{N}_i|$ and $\lambda = \frac{3}{N+1}$. If $N > 3$, then* (2.45) *converges with exponential rate and we have that $\|x^{(k+1)} - x^{(k)}\|_\infty \leq \lambda^k$. Further, to achieve an error of at most $\delta > 0$ between the last two iterations, we need to run the iterative scheme at most $k = \lceil \log_\lambda(\delta) \rceil$ times.*

Proof. Let $N = \min_{i\in\mathcal{V}} |\mathcal{N}_i|$ and $\lambda = \frac{3}{N+1}$. Using the proof of Lemma 2.2, we have that $\|x^{(k+1)} - x^{(k)}\|_\infty \leq \frac{3}{|\mathcal{N}_i|} \|x^{(k)} - x^{(k-1)}\|_\infty \leq \lambda^k \|x^{(1)} - x^{(0)}\|_\infty \leq \lambda^k$. Hence, the iterative scheme converges with an exponential rate of λ^k. To achieve an error between iterations of at most ε, we need to have that $\lambda^k \leq \delta$, which is equivalent to having that $k \leq \log_\lambda(\delta) \leq \lceil \log_\lambda(\delta) \rceil$. Therefore, if we run the iterative scheme at most $\lceil \log_\lambda(\delta) \rceil$ times, we obtain an error between the last two iterations of at most ε. □

2.9.3 Complexity Analysis

Next, we investigate the complexity analysis of the proposed algorithm RepC when the network communication is synchronous.

Proposition 2.3 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a network of agents, $l = \max_{v \in \mathcal{V}} |\mathcal{N}_v|$, then, for i iterations and for each agent, the iterative scheme* (2.45) *has time complexity of $O(l^2 i)$.*

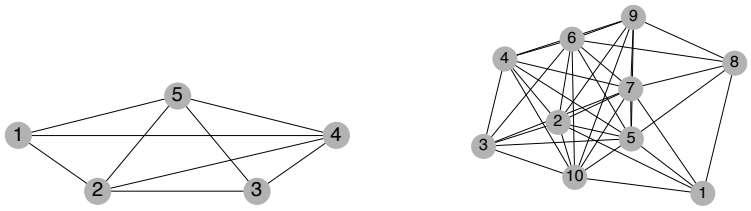
Proof. Given a network of agents $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for time step k and agent v, the time complexity of (2.45) is the sum of the time complexities of computing $\tilde{c}_{vu}^{(k)}$, $\tilde{\tilde{c}}_{vu}^{(k)}$, $c_{vu}^{(k)}$, for each $u \in \mathcal{N}_i$, and $x_v^{(k)}$. Computing $\tilde{c}_{vu}^{(k)}$ costs $O(|\mathcal{N}_v|^2)$, because there are $O(|\mathcal{N}_v|^2)$ pairs of neighbors values to compute the absolute difference. Each of the remaining steps has time complexity of $O(|\mathcal{N}_v|)$. Hence, the sum of each step time complexity is $O(|\mathcal{N}_v|^2) + 4 \times O(|\mathcal{N}_v|) = O(|\mathcal{N}_v|^2)$. If $l = \max_{v \in \mathcal{V}} |\mathcal{N}_v|$, then $O(l^2)$ is a bound for the complexity that each incurs. For i iterations an agent incurs in $O(l^2 i)$ time complexity. □

2.10 Illustrative Examples

Subsequently, we illustrate the use of RepC for different kinds of attacks. Further, in the examples, we use $\varepsilon = 0.1$.

2.10.1 Same value for attacked nodes

In the next examples, consider the network of Fig. 2.23 (a).



(a) Network of agents $\mathcal{G}_A$. (b) Network of agents $\mathcal{G}_B$.

Fig. 2.23: Topologies used for the illustrative examples.

First, we illustrate algorithm RepC in the scenario of a network of agents **without attacked nodes**. The set of agents is $\mathcal{V}_1 = \{1, \ldots, 5\}$ and, thus, the set of attacked agents is $\mathcal{A} = \emptyset$. We set the parameter $f = 1$. Fig. 2.24 depicts the state evolution of each agent.

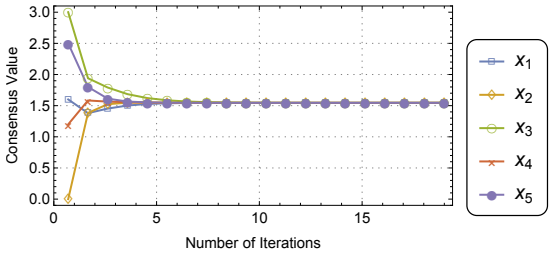


Fig. 2.24: Consensus of network $\mathcal{G}_A$ with agents $\mathcal{V}_1$ and set of attacked agents $\emptyset$.

Here, we explore the scenario where an **attacker** targets one agent to **share a value close to the consensus**, depicted in Fig. 2.23.

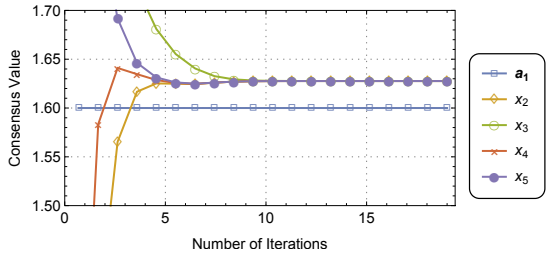


Fig. 2.25: Consensus of network $\mathcal{G}_A$ with agents $\mathcal{V}_1$ and set of attacked agents $\mathcal{A}_1$ (zoomed around the attacker's value.)

The set of agents is $\mathcal{V} = \{1, \ldots, 5\}$ and the set of attacked agents is $\mathcal{A} = \{1\}$. Fig. 2.25 depicts each agent consensus value. We can see that although the attacker value is very close to the consensus value, the neighbors of the attacked node assign zero to its reputation by using (2.45). Hence the attacked node shared values are discarded.

Next, in Fig. 2.26 and 2.27, we depict the evolution of the reputations that agents 2 and 3 assign to their neighbors.

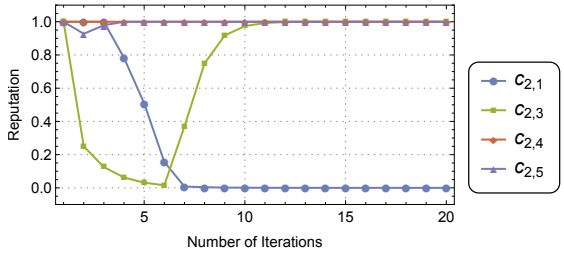


Fig. 2.26: Evolution of the reputations that agent 2 assigns to each of its neighbors.

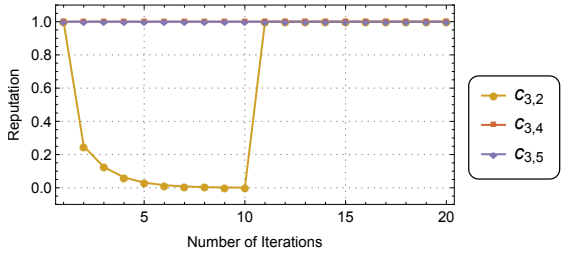


Fig. 2.27: Evolution of the reputations that agent 3 assigns to each of its neighbors.

2.10.2 Different values for attacked nodes

Next, we illustrate the scenario where attacked nodes share different values. For that end, we consider the set of agents $\mathcal{V} = \{1, \ldots, 10\}$, with $\mathcal{A} = \{1, 8\}$, and the network of agents depicted in Fig. 2.23 (b). We explore two scenarios with two attacked agents: (i) both attacked nodes share values (distinct) smaller than the consensus, see Fig. 2.28; (ii) one attacked node shares a value larger than the consensus while the other uses a smaller value than the consensus, see Fig. 2.29.

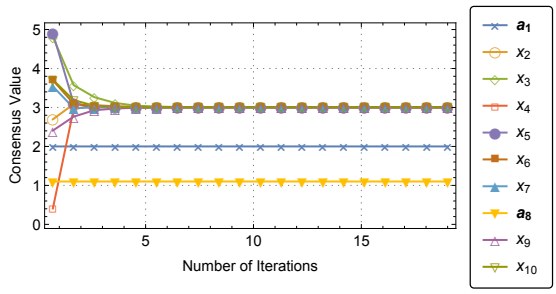


Fig. 2.28: Consensus of network $\mathcal{G}_B$ with agents $\mathcal{V}_3$ and set of attacked agents $\mathcal{A}_3$

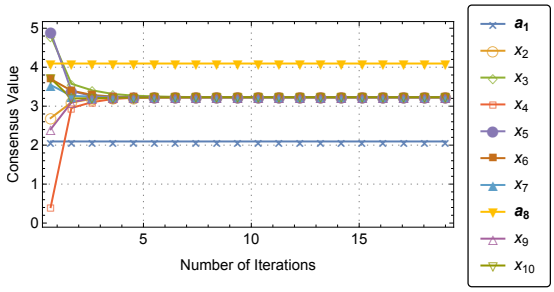


Fig. 2.29: Consensus of network $\mathcal{G}_B$ with agents $\mathcal{V}_2$ and set of attacked agents $\mathcal{A}_2$

2.10.3 Asynchronous Communication

We now illustrate the use of algorithm RepC in the case where the communication between nodes occurs asynchronously. To simulate this scenario, at each time instance, a random subset of agents communicates. The set of agents is $\mathcal{V} = \{1, \ldots, 5\}$, the network of agents is $\mathcal{G}_A$, and the set of attacked agents is $\mathcal{A} = \{1\}$. Fig. 2.30 depicts the state evolution of each agent when using the asynchronous version of algorithm RepC. Each normal node identifies and discards the information of the attacked agent.

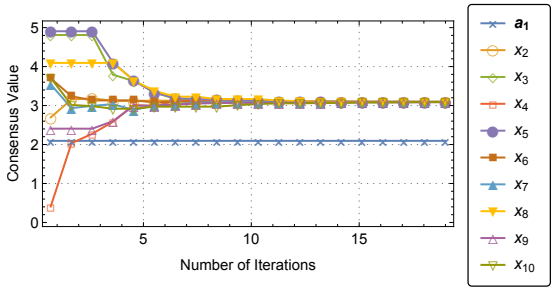
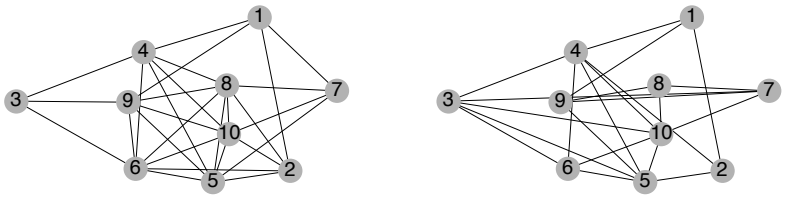


Fig. 2.30: Consensus of network $\mathcal{G}_A$ with agents $\mathcal{V}_1$, asynchronous communication, and set of attacked agents $\mathcal{A} = \{1\}$.

2.10.4 Dynamic network

Next, we test the scenario where the network of agents evolves with time and the attacked agents share the same value. We consider two networks composed of 10

agents, as depicted in Fig. 2.31, with a set of agents $\mathcal{V} = \{1, \ldots, 10\}$ and of attacked agents $\mathcal{A} = \{1\}$.



(a) Network of agents $\mathcal{G}_D$. (b) Network of agents $\mathcal{G}_C$.

Fig. 2.31

In the example, we consider that the dynamic network of agents for time instance $k > 0$ is given by $\mathcal{G}_1^{(k)} = \begin{cases} \mathcal{G}_C & \text{if } k \leq 10 \\ \mathcal{G}_D & \text{otherwise} \end{cases}$. The consensus value of each agent, using (2.45), is depicted in Fig. 2.32.

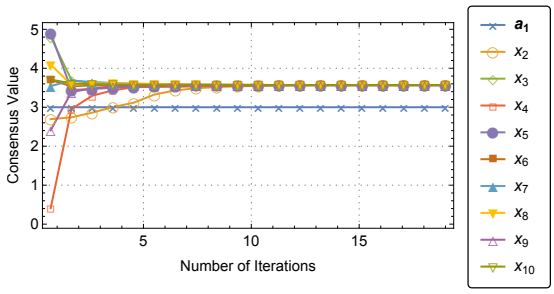


Fig. 2.32: Consensus of dynamic network $\mathcal{G}_1^{(k)}$ with agents $\mathcal{V}_1$, and set of attacked agents $\mathcal{A} = \{1\}$.

2.10.5 Dynamic network with noisy agents

Last, we consider the case where the network of agents evolves, and the attacked agents share different values (drawn from uniform random variables with a fixed mean). See examples in Fig. 2.33 and 2.34.

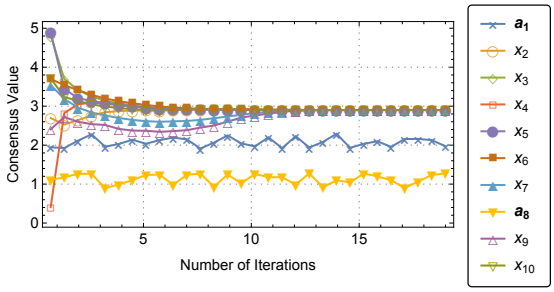


Fig. 2.33: Consensus of dynamic network $\mathcal{G}_1^{(k)}$ with agents $\mathcal{V}_1$, and set of attacked (noisy) agents $\mathcal{A} = \{1, 8\}$.

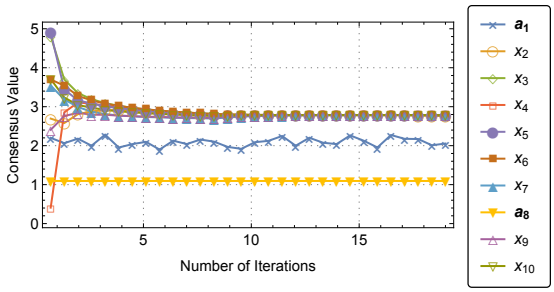


Fig. 2.34: Consensus of dynamic network $\mathcal{G}_1^{(k)}$ with agents $\mathcal{V}_1$, and set of attacked agents $\mathcal{A} = \{1, 8\}$, where agent 1 behaves as a noisy node and agent 8 as an attacked node.

2.10.6 Stochastic communication

When the communication between agents is stochastic, we may still successfully apply RepC. We consider the network $\mathcal{G}_E$ in Fig. 2.35, with $\mathcal{V}_1$, and the set of attacked agents $\mathcal{A} = \{1\}$. Further, at each time step, only a random subset of agents communicate between them. This case is depicted in Fig. 2.36, where the regular agents effectively detect the attacked node, achieving the true consensus.

2.10.7 RepC vs. state-of-the-art

Here, we illustrate how the proposed algorithm competes with the state-of-the-art approaches, based on the idea that each agent discards a set of maximum and minimum neighbor values.

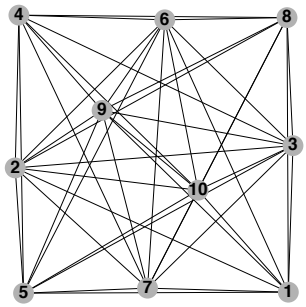


Fig. 2.35: Network of agents $\mathcal{G}_E$.

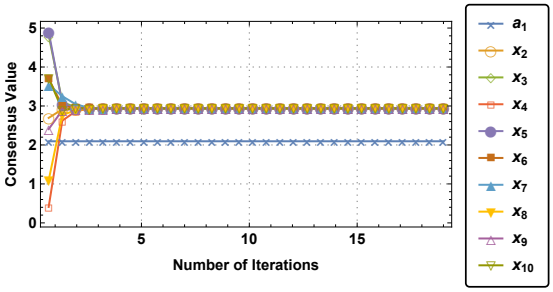
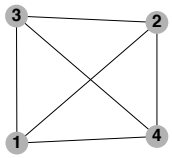
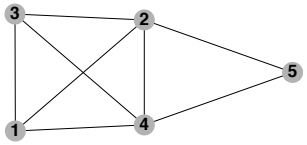


Fig. 2.36: Consensus using a RepC method, with network $\mathcal{G}_E$, set of agents $\mathcal{V}_1$, set of attacked agents $\mathcal{A} = \{1\}$ and stochastic communication.

In the next examples, we use the two networks in Fig. 2.37.



(a) Network of agents $\mathcal{G}_F$.



(b) Network of agents $\mathcal{G}_G$.

Fig. 2.37: Additional topologies for the examples.

In the first example, consider the set of agents $\mathcal{V}_2 = \{1, 2, 3, 4\}$, with the complete network (Fig. 2.37 (a)) and attacked agents $\mathcal{A} = \{1\}$.

Using the state-of-the-art, i.e., when each agent discards the maximum and minimum neighbors' values, we obtain the result depicted in Fig. 2.38. The method cannot deter the attack, and the regular agents converge to the attacker value. Using

RepC, as illustrated in Fig. 2.39, the regular agents converge to a value close to the true value, with a small deviation caused by the influence of the ε parameter.

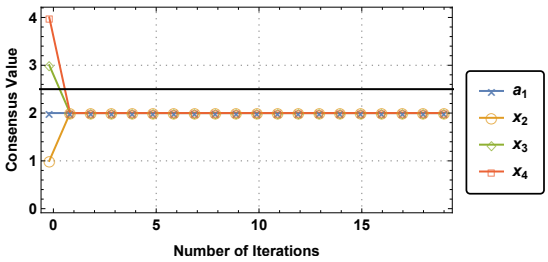


Fig. 2.38: Consensus using a **state-of-the-art** method, with network $\mathcal{G}_F$, set of agents $\mathcal{V}_2$, and set of attacked agents $\mathcal{A} = \{1\}$. The black line is the true consensus value.

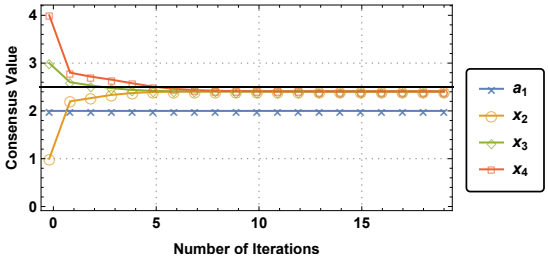


Fig. 2.39: Consensus using RepC, with network $\mathcal{G}_F$, set of agents $\mathcal{V}_2$, and set of attacked agents $\mathcal{A} = \{1\}$. The black line is the true consensus.

In the second example, we consider the network of agents depicted in Fig. 2.37 (b), the set of agents $\mathcal{V}_3 = \{1, 2, 3, 4, 5\}$ and attacked agents set $\mathcal{A} = \{1\}$. The example portrays the scenario where an attacker stubbornly sends to the neighbors the true consensus value.In Fig. 2.40, the agents cannot converge to the true consensus value when using the state-of-the-art approach.

We present the consensus state of the agents when using RepC in Fig. 2.41, and the agents converge to the true network consensus.

2.10.8 Consensus final error

To explore how different is the final consensus value produced by RepC and the consensus value without attacked nodes, we use the complete network of 5 agents

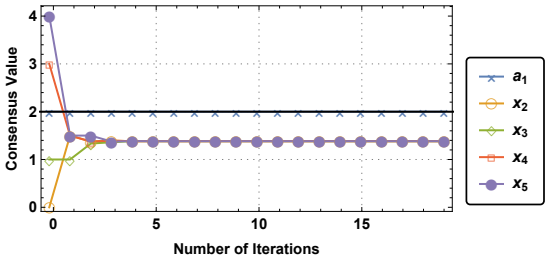


Fig. 2.40: Consensus using a **state-of-the-art** method, with network $\mathcal{G}_G$, set of agents $\mathcal{V}_3$, and set of attacked agents $\mathcal{A} = \{1\}$. The black line is the true consensus value.

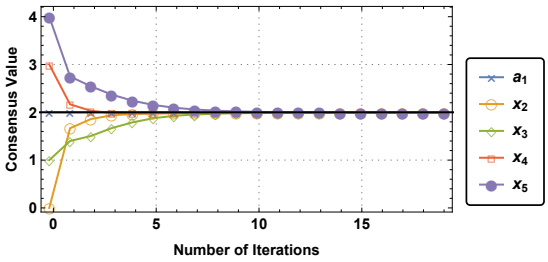


Fig. 2.41: Consensus using RepC, with network $\mathcal{G}_F$, set of agents $\mathcal{V}_3$, and set of attacked agents $\mathcal{A} = \{1\}$. The black line is the true consensus.

depicted in Fig. 2.23 (a), with agents' initial states $x^{(0)} = [\,1\ 0\ 3\ 1.2\ 2.5\,]^\intercal$, where agent 1 is under attack and shares values from a Gaussian noise with mean μ and standard deviation σ. The consensus, without attacked nodes, is 1.489. We compute the absolute difference between the consensus value found with RepC in the non-attacked case and the consensus value obtained with RepC when the attacker follows the mentioned strategy. We ranged μ from 0 to 1 in steps of 0.005 and ranged σ from 0.1 to 1 in steps of 0.005, repeating each scenario 20 times, see Fig. 2.42.

We can see from Fig. 2.42 that, on average, we get a small final consensus error. When μ is close to 1, and σ is close to 0.1, the attacked node state value is close to what it would be in the non-attacked scenario ($x_1^{(0)} = 1$), taking more time to be classified as an attacker and yielding a slightly larger (final) consensus error.

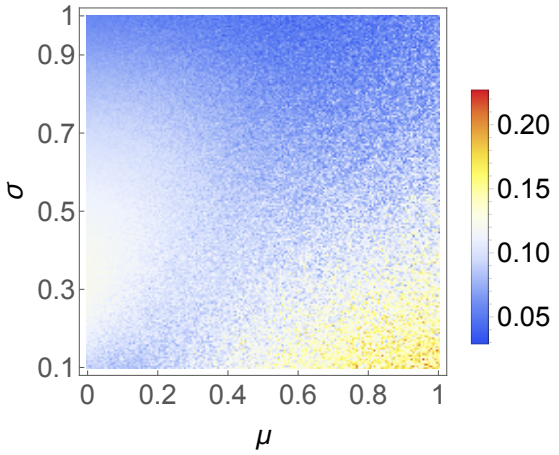


Fig. 2.42: Absolute difference between the consensus resulting from Algorithm RepC when node one 1 is under attack to share the a Gaussian noise with mean μ and standard deviation σ.

Chapter 3

Deterministic and Stochastic Estimation for Linear Dynamical Systems

The second stage in FirePuma consists in using the acquired data and build an utility map to represent the areas of interest. In essence, we would like to design the mathematical tools to apply linear dynamics to the estimates (for instance to account for how a variable like velocity changes the position) but also to incorporate recently acquired measurements to remove uncertainty from those areas. In a broader picture, this is the same problem that arises in the control of autonomous vehicles as controllers often use the information about the state to decide on the actuation. Therefore, a prior task is to obtain estimates for the state from measurements obtained using the onboard sensors. In this chapter, we detail the technique mostly for linear dynamics, albeit this can be extended for the nonlinear case by using the Taylor expansion and resorting to the linear terms and bound in a suitable manner the sum of the higher order elements, i.e., the Lagrange remainder.

The task associated with this development has looked into the following problems:

- deterministic estimation for linear systems by introducing the definition of Constrained Convex Generators (CCGs) [43], [44];
- reduce the size of the data structures for CCGs [45];
- stochastic estimation for linear systems accounting for general noise and disturbance Probability Density Functions (PDFs) using Characteristic Functions (CFs) [46];
- comparison of different data structures to represent sets [47];
- extending the deterministic estimation to the case of uncertain linear systems [48], [49] and [50];
- extending the estimation to nonlinear systems using an overbound for the Lagrange remainder [51] or the Koopman operator [52];
- computing the reachable set of neural networks [53];

- calculating the Robust Positively Invariant (RPI) set to be used in MPC controllers [54].

3.1 Deterministic Estimation for Linear Dynamical Systems

In this chapter, we start by considering a linear dynamical model with no uncertainties:

$$x(k+1) = F_k x(k) + B_k u(k) + w(k) \tag{3.1}$$

where $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^{n_u}$ and $w(k) \in \mathbb{R}^n$ are respectively the state, input and disturbance signals. Following the derivation in [55], one can assume the model for a vehicle like a quadrotor to be a double integrator subject to constraints on the velocity and acceleration, which fits the formulation for a linear dynamical system. The objective is to estimate the state $x(k)$ using a set reachability approach from bearing and range measurements. Let us assume that there are j nodes providing measurements, which will be referred by towers. Under such conditions, we are interested in 3 different problems, namely:

Problem 3.1 (State estimation using range/bearing data) Let us consider a Linear Time-Varying (LTV) model for a vehicle as in (3.1) with position evolving in $\mathbb{R}^p$ and j towers with known locations $\text{tower}_1, \cdots, \text{tower}_j$.

- If range measurements are available, they will be denoted by $y^r(k)$ defined as follows:

$$y^r(k) = \begin{bmatrix} \|x_{[1,\cdots,p]}(k) - \text{tower}_1\|_2 \\ \vdots \\ \|x_{[1,\cdots,p]}(k) - \text{tower}_j\|_2 \end{bmatrix} + v(k), \tag{3.2}$$

- If bearing measurements are available, they will be denoted by $y^b(k)$ defined as follows:

$$y^b(k) = \begin{bmatrix} \text{ang}(x_{[1,\cdots,p]}(k) - \text{tower}_1) \\ \vdots \\ \text{ang}(x_{[1,\cdots,p]}(k) - \text{tower}_j) \end{bmatrix} + v(k), \tag{3.3}$$

where we used the notation $x_{[1,\cdots,p]}(k)$ to select the entries 1 through p that correspond to the position of the vehicle and $v(k) \in \mathbb{R}^{n_v}$ is the noise signal. The transpose of a vector v is denoted by $v^\intercal$ and is used to define the Euclidean norm for vector x is represented as $\|x\|_2 := \sqrt{x^\intercal x}$, while the operator $\text{ang}(v)$ returns either the angle of vector v in polar coordinates when $p = 2$ or a vector of 2 angles in spherical

coordinates for v. Let us define the vector of all available measurements at time k by $y(k)$.

The problem is defined as computing a set of possible state values $X(k)$ for $k > 0$ such that $x(k) \in X(k), \forall w(k) \in W(k), \forall v(k) \in V(k)$, for some convex sets $W(k)$ and $V(k)$, from range measurements (i.e., $y(k) = y^r(k)$), bearing measurements (i.e., $y(k) = y^b(k)$, or range and bearing measurements (i.e., $y(k) = \begin{bmatrix} y^r(k) \\ y^b(k) \end{bmatrix}$).

3.2 Generalization of the Constrained Zonotope Representation

In this section, we first review the Constrained Zonotope (CZ) representation introduced in [56] to model polytopes, highlighting how the basic set operations can be viewed in terms of convexity-preserving operations of a basic generator set.

Let us first review the definition of CZ from [56] and the basic set operations: affine map (3.5), Minkowski sum (3.6) and intersection after a linear map (3.7). In the following, we will use the notation $\|x\|_\infty := \max_i |x_i|$, 0_n to denote the n-dimensional vector of zeros, the intersection after applying a matrix R to the first set by $\cap_R$ and the Minkowski sum of two sets by $\oplus$.

Definition 3.1 (Constrained Zonotope) A set Z is a CZ defined by the tuple $(G, c, A, b) \in \mathbb{R}^{n \times n_g} \times \mathbb{R}^n \times \mathbb{R}^{n_c \times n_g} \times \mathbb{R}^{n_c}$ such that:

$$Z = \{G\xi + c : \|\xi\|_\infty \leq 1, A\xi = b\}. \tag{3.4}$$

Definition 3.2 (Set operations) Consider three CZs as in Definition 3.1:

- $Z = (G_z, c_z, A_z, b_z) \subset \mathbb{R}^n$;
- $W = (G_w, c_w, A_w, b_w) \subset \mathbb{R}^n$;
- $Y = (G_y, c_y, A_y, b_y) \subset \mathbb{R}^m$;

and a matrix $R \in \mathbb{R}^{m \times n}$ and a vector $t \in \mathbb{R}^m$. The three set operations are defined as:

$$RZ + t = (RG_z, Rc_z + t, A_z, b_z) \tag{3.5}$$

$$Z \oplus W = \left(\begin{bmatrix} G_z & G_w \end{bmatrix}, c_z + c_w, \begin{bmatrix} A_z & 0 \\ 0 & A_w \end{bmatrix}, \begin{bmatrix} b_z \\ b_w \end{bmatrix} \right) \tag{3.6}$$

$$Z \cap_R Y = \left(\begin{bmatrix} G_z & 0 \end{bmatrix}, c_z, \begin{bmatrix} A_z & 0 \\ 0 & A_y \\ RG_z & -G_y \end{bmatrix}, \begin{bmatrix} b_z \\ b_y \\ c_y - Rc_z \end{bmatrix} \right). \tag{3.7}$$

Given the Definition 3.2 for the three major set operations that will be required, let us write the complete definition for these operations, which will make it easier to introduce the novel definition for the proposed sets. The affine map for a CZ Z can also be defined as:

$$\begin{aligned} RZ + t &= \{RG_z\xi + Rc_z + t : \|\xi\|_\infty \leq 1, A_z\xi = b_z\} \\ &= \{RG_z\xi + Rc_z + t : \xi \in C_z, A_z\xi = b_z\} \end{aligned} \tag{3.8}$$

where the second equation is implicitly assuming that C_z is the unit ℓ_∞-norm ball. In a similar fashion, we can present the same extended definition for the Minkowski sum and the intersection after linear map, where we use ξ_z, ξ_w and ξ_y as the auxiliary variables for the CZs Z, W and Y:

$$\begin{aligned} Z \oplus W = \{G_z\xi_z + G_w\xi_w + c_z + c_w : &A_z\xi_z = b_z, A_w\xi_w = b_w, \\ &\xi_z \in C_z, \xi_w \in C_w\} \end{aligned} \tag{3.9}$$

$$\begin{aligned} Z \cap_R Y = \{G_z\xi_z + c_z : &A_z\xi_z = b_z, A_y\xi_y = b_y, \xi_y \in C_y, \\ &RG_z\xi_z + Rc_z = G_y\xi_y + c_y, \xi_z \in C_z\} \end{aligned} \tag{3.10}$$

From the above definition, it becomes clear that there is nothing forcing the use of the unit ℓ_∞-norm ball as the generator and one could resort to any unit ball following a p-norm but also extend the definition to convex cones (and other convex sets).

Remark 3.1 The definition of CZ follow a more systematic representation of polytopes, which was already approached in a hyperplane representation within the literature in [57]–[64].

We are now in condition of presenting the definition for CCGs and a proposition establishing the equivalence between the set operations and its definition in the CCG format that will explore the relationship identified in (3.8), (3.9) and (3.10).

Definition 3.3 (Constrained Convex Generators) A Constrained Convex Generator (CCG) $Z \subset \mathbb{R}^n$ is defined by the tuple (G, c, A, b, C) with $G \in \mathbb{R}^{n \times n_g}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{n_c \times n_g}$, $b \in \mathbb{R}^{n_c}$, and $C := \{C_1, C_2, \cdots, C_{n_p}\}$ such that:

$$Z = \{G\xi + c : A\xi = b, \xi \in C_1 \times \cdots \times C_{n_p}\}. \tag{3.11}$$

In the above, we have used $\times$ to represent the cartesian product of sets.

Given Definition 3.3, we could present a proposition asserting all set operations for CCGs.

Proposition 3.1 *Consider three Constrained Convex Generators (CCGs) as in Definition 3.3:*

- $Z = (G_z, c_z, A_z, b_z, C_z) \subset \mathbb{R}^n$;
- $W = (G_w, c_w, A_w, b_w, C_w) \subset \mathbb{R}^n$;
- $Y = (G_y, c_y, A_y, b_y, C_y) \subset \mathbb{R}^m$;

and a matrix $R \in \mathbb{R}^{m \times n}$ and a vector $t \in \mathbb{R}^m$. The three set operations are defined as:

$$RZ + t = (RG_z, Rc_z + t, A_z, b_z, C_z) \tag{3.12}$$

$$Z \oplus W = \left(\begin{bmatrix} G_z & G_w \end{bmatrix}, c_z + c_w, \begin{bmatrix} A_z & 0 \\ 0 & A_w \end{bmatrix}, \begin{bmatrix} b_z \\ b_w \end{bmatrix}, \{C_z, C_w\} \right) \tag{3.13}$$

$$Z \cap_R Y = \left(\begin{bmatrix} G_z & 0 \end{bmatrix}, c_z, \begin{bmatrix} A_z & 0 \\ 0 & A_y \\ RG_z & -G_y \end{bmatrix}, \begin{bmatrix} b_z \\ b_y \\ c_y - Rc_z \end{bmatrix}, \{C_z, C_y\} \right). \tag{3.14}$$

Proof. In order to prove (3.12), let us define the set resulting from the affine map as $\{z' : z' = Rz + t, \forall z \in Z\}$ which can be expanded as:

$$\{z' : z' = Rz + t, z = G_z\xi_z + c_z, A_z\xi_z = b_z, \xi_z \in C_z\} \tag{3.15}$$

where we used the notation $\xi_z \in C_z$ to mean $\xi_z \in C_1 \times \cdots \times C_{n_p}$, $|C_z| = n_p$. Replacing the value of z in the expression we get:

$$\{z' : z' = RG_z\xi_z + Rc_z + t, A_z\xi_z = b_z, \xi_z \in C_z\} \tag{3.16}$$

which is precisely the definition for the CCG on the right-hand side of (3.12).

For the Minkowski sum, we can do a similar analysis by first defining the set corresponding to the application of this operation by $\{z' : z' = z + w, \forall z \in Z, \forall w \in W\}$ and expand it as:

$$\begin{aligned} \{z' : z' = z + w, &z = G_z\xi_z + c_z, A_z\xi_z = b_z, \xi_z \in C_z, \\ &w = G_w\xi_w + c_w, A_w\xi_w = b_w, \xi_w \in C_w\}. \end{aligned} \tag{3.17}$$

By replacing the values of z and w we can further simplify the definition as:

$$\begin{aligned} \{z' : z' = G_z\xi_z + c_z + G_w\xi_w + c_w, &A_z\xi_z = b_z, \xi_z \in C_z, \\ &A_w\xi_w = b_w, \xi_w \in C_w\}. \end{aligned} \tag{3.18}$$

Stacking ξ_z and ξ_w in a single vector, we obtain the following expression:

$$\left\{z' : z' = \begin{bmatrix} G_z & G_w \end{bmatrix} \begin{bmatrix} \xi_z \\ \xi_w \end{bmatrix} + (c_z + c_w), \begin{bmatrix} A_z & 0 \\ 0 & A_w \end{bmatrix} \begin{bmatrix} \xi_z \\ \xi_w \end{bmatrix} = \begin{bmatrix} b_z \\ b_w \end{bmatrix}, \begin{bmatrix} \xi_z \\ \xi_w \end{bmatrix} \in \{C_z, C_w\}\right\}, \tag{3.19}$$

which is the right-hand side of (3.13).

Lastly, the intersection after a linear map can be defined as $\{z' : z' = z, \forall z \in Z, Rz \in Y\}$, which can also be expanded to:

$$\begin{aligned} \{z' : z' = z, & z = G_z\xi_z + c_z, A_z\xi_z = b_z, \xi_z \in C_z, \\ & Rz = G_y\xi_y + c_y, A_y\xi_y = b_y, \xi_y \in C_y\}. \end{aligned} \tag{3.20}$$

Replacing the value of z in the expression yields:

$$\begin{aligned} \{z' : & z' = G_z\xi_z + c_z, A_z\xi_z = b_z, \xi_z \in C_z, \\ & RG_z\xi_z + Rc_z = G_y\xi_y + c_y, A_y\xi_y = b_y, \xi_y \in C_y\}, \end{aligned} \tag{3.21}$$

which by stacking the values of ξ_z and ξ_y into a single vector becomes:

$$\left\{z' : z' = \begin{bmatrix} G_z & 0 \end{bmatrix} \begin{bmatrix} \xi_z \\ \xi_y \end{bmatrix} + c_z, \begin{bmatrix} A_z & 0 \\ 0 & A_y \\ RG_z & -G_y \end{bmatrix} \begin{bmatrix} \xi_z \\ \xi_y \end{bmatrix} = \begin{bmatrix} b_z \\ b_y \\ c_y - Rc_z \end{bmatrix}, \begin{bmatrix} \xi_z \\ \xi_y \end{bmatrix} \in \{C_z, C_y\}\right\}, \tag{3.22}$$

which is the right-hand side of (3.14), thus concluding the proof. □

From the operations in Proposition 3.1 and the fact that by construction, CCG as defined in Definition 3.3 are convex sets, it means that they are well-suited to be applied to state estimation and fault detection of LTV models. Computationally speaking, it is required to store which type of generator we are using for which entries of the vector of auxiliary variables ξ. In the next section, we illustrate the use of CCGs and will only use unit ℓ_∞-norm balls, unit ℓ_2-norm balls and cones as the convex generators for designing an observer to estimate the state of a vehicle. Also notice that it is always possible to over-approximate CCG sets by the hyper-cubes resulting from interval analysis so the results in [65] regarding the boundedness of the hyper-volume of these sets can be directly applied.

We would like to point out that all the aforementioned set representations are subsets of CCGs, namely:

- an interval corresponds to $(G, c, [\,], [\,], \|\xi\|_\infty \leq 1)$, for a diagonal matrix G;
- a zonotope is given by $(G, c, [\,], [\,], \|\xi\|_\infty \leq 1)$;
- an ellipsoid is defined by $(G, c, [\,], [\,], \|\xi\|_2 \leq 1)$, for a square matrix G;
- a CZ or polytope is $(G, c, A, b, \|\xi\|_\infty \leq 1)$;
- a convex cone in $\mathbb{R}^n$ is $(G, c, [\,], [\,], \xi \geq 0)$;
- ellipsotopes are given by $(G, c, A, b, \|\xi\|_{p_1} \leq 1, \cdots, \|\xi\|_{p_m} \leq 1)$, for some $p_i > 0$, $1 \leq i \leq m$;
- AH-polytopes are given by $(G, c, [\,], [\,], A\xi \leq b)$.

3.3 State Estimation using Constrained Convex Generators (CCGs) with Range/Bearing Data

In this section, the state estimation strategy is presented by exploiting the properties of CCGs introduced in Section 3.2 and over-approximations to the exact non-convex measurement sets. The propagation equation of the estimates using the dynamical model in (3.1) can be accomplished using the set operations in Proposition 3.1, namely that with previous set-valued estimates $X(k)$ can be propagated to obtain set $X_{\text{prop}}(k+1)$ that contains all points that are consistent with the previous estimate and the dynamics in the following fashion:

$$X_{\text{prop}}(k+1) = F_k X(k) + B_k u(k) \oplus W(k), \tag{3.23}$$

meaning that $X_{\text{prop}}(k+1)$ is the result of an affine map on $X(k)$ using matrix F_k and vector $B_k u(k)$ and then the Minkowski sum with the disturbance set.

The update set of the observer requires performing an intersection following the linear map $C = \begin{bmatrix} \mathsf{e}_1 & \cdots & \mathsf{e}_p \end{bmatrix}^\mathsf{T}$, which is defined to obtain the first p entries of vector x that store the vehicle position, i.e., $x_{[1,\cdots,p]}(k) = Cx(k)$. This means that the set-valued estimates for state at time $k+1$, $X(k+1)$, can be obtained as an intersection between propagated set $X_{\text{prop}}(k+1)$ with the measurement set $Y(k+1)$ (which will be defined for the three cases of range, bearing and range/bearing data) as follows:

$$X(k+1) = X_{\text{prop}}(k+1) \cap_C Y(k+1). \tag{3.24}$$

Note that the set $Y(k+1)$ is going to be the intersection of the individual measurement set for each tower. For simplicity of exposition, we will present the various

$Y(k+1)$ assuming a single tower (and drop the subscript for that matter) but we can perform the intersection using the identity linear map over all the sets.

3.3.1 Bearing-only measurements

Let us define the minimum and maximum error on a single bearing measurement as b_l and b_u, meaning that we have in two dimensions that $\text{ang}(x_{[1,2]}(k) - \text{tower}) \in \begin{bmatrix} y^b(k) - b_l & y^b(k) + b_u \end{bmatrix}$.

Thus, the bearing-only measurement set $Y^b(k)$ is a cone, which can be expressed by the CCG:

$$Y^b(k) = \left(\begin{bmatrix} \cos(y^b(k) + b_u) & \cos(y^b(k) - b_l) \\ \sin(y^b(k) + b_u) & \sin(y^b(k) - b_l) \end{bmatrix}, \text{tower}, \mathbf{0}_2^\intercal, 0, \{\mathbb{R}_+^2\} \right), \tag{3.25}$$

where $\mathbb{R}_+^2$ is the nonnegative orthant in $\mathbb{R}^2$. The definition in (3.25) did an affine transformation of $\mathbb{R}_+^2$ where G was selected as to change the canonical vectors to the desired ones corresponding to the minimum and maximum angles allowed by the measurement $y^b(k)$. We remark that setting $A = \begin{bmatrix} 0 & 0 \end{bmatrix}$ in $Y^b(k)$ as we have done can be omitted and treated as the empty matrix provided that dimensions are kept consistent when using block diagonal operations. In 3 dimensions, one can address each angle separately and then use the intersection to construct the shape.

3.3.2 Bearing and Range measurements

Prior to presenting the range-only measurements, it is easier to first look at a segment of the annulus and then to resort to the equivalent of Zonotope Bundles [66] with CCGs instead. Let us define the minimum and maximum error on a single range measurement as r_l and r_u, meaning that we have in two dimensions that $\|x_{[1,2]}(k) - \text{tower}\|_2 \in \begin{bmatrix} y^r(k) - r_l & y^r(k) + r_u \end{bmatrix}$ in addition to the constraint from the bearing. Figure 3.1 hints at the need to first compute the four points that result from the intersection of each of the circles and the minimum and maximum angles (getting two outer points in the outer circle and conversely two inner points). The coordinates

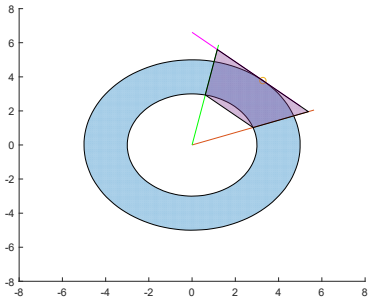


Fig. 3.1: Conservative case caused by a larger error in the angle.

of each point is simply $\rho \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix}$ for $\rho \in \{y^r(k) - r_l, y^r(k) + r_u\}$ and $\alpha \in \{y^b(k) - b_l, y^b(k) + b_u\}$. We also need a fifth point with the maximum range possible and $\alpha = (2y^r(k) - r_l + r_u)/2$. It is then straightforward to find the line equation that is parallel to the outer points and passes through the fifth point as well as the remaining line equations and write the trapezoidal shape as $Mx \leq m$. By applying the formula from Theorem 1 in [56], we obtain the CZ (G, c, A, b) that is equivalent to the CCG representation $Z_{\text{trap}} = (G, c, A, b, \{\mathcal{B}_\infty\})$ where $\mathcal{B}_\infty$ is the unit ℓ_∞-ball. The outer circle is given by the CCG $\Big((y^r(k) + r_u)I_2, \text{tower}, 0_2^\intercal, 0, \{\mathcal{B}_2\}\Big)$, where $\mathcal{B}_2$ is the unit ℓ_2-ball. Thus, the measurement set $Y^{br}(k)$ is given by:

$$Y^{br}(k) = Z_{\text{trap}} \cap_{I_2} \Big((y^r(k) + r_u)I_2, \text{tower}, 0_2^\intercal, 0, \{\mathcal{B}_2\}\Big). \tag{3.26}$$

3.3.3 Range-only measurements

The measurement set $Y^r(k)$ when there are only range measurements available cannot be approximated as done in the previous section. However, one can partition the full circle into segments with $2\pi/\Delta$ angles and use as minimum and maximum angles the values $2\pi q/\Delta$ and $2\pi(q+1)/\Delta$ for $q = 0, 1, \cdots \Delta - 1$ and repeat the computations done in Subsection 3.3.2 replacing the interval $\Big[y^r(k) - r_l \; y^r(k) + r_u\Big]$ by $\Big[2\pi q/\Delta \; 2\pi(q+1)/\Delta\Big]$. These Δ sets $Y_1^r(k), \cdots, Y_\Delta^r(k)$ need to be intersected independently with the propagated set $X_{\text{prop}}(k)$ to obtain $X(k) = \bigcup_{q=0}^{\Delta-1} Y_q^r(k) \cap_{I_2} X_{\text{prop}}(k)$. Naturally, this increases the computational complexity as in the subsequent time step we may end up with Δ times more sets to

propagate. Each set can be checked to see if it is empty and discarded if so to reduce the amount of future computation with unnecessary sets. In the envisioned scenario of a tower providing range measurements or a beacon emitting a sound or a signal to help the vehicle in its localization, most of these sets will be empty as the beacon will be far away.

3.4 Simulations for Estimation with Range/Bearing Measurements

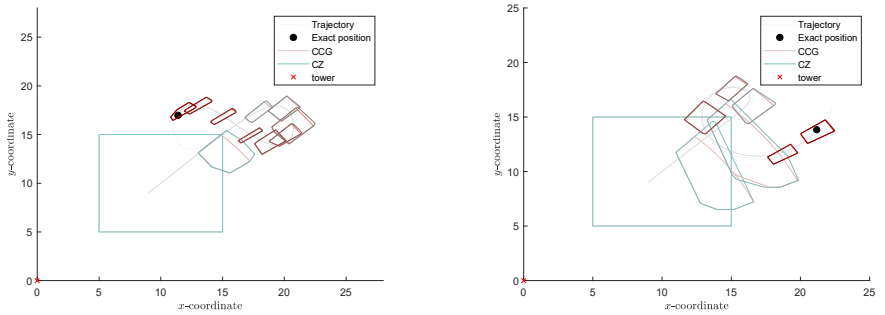
In this section, simulation results are presented with the objective of characterizing the behavior of the proposed CCG definition with respect to 4 important factors: i) size of the matrices and vectors used in the definitions; ii) computational time to achieve the representation; iii) solver time to check that a point belong to the set and also if it is empty; and, iv) comparison against the size of the estimate set produced by the CZs. Notice that i) is not of particular interest unless the set representation needs to be sent to other agents (as for example to implement a distributed state estimation algorithm such as in [67]) whereas ii) is crucial when running the observer online as the computation has to be faster than that of the sampling time. Lastly, checking for a collision with another convex obstacle can be achieved by modeling it as a CCG and checking whether the intersection is the empty set, which emphasizes the importance of iii). Moreover, computing the centroid of the set, what is the point maximizing some direction among other questions can all be formulated as an optimization problem which means that it should be efficient to solve a program constrained to a CCG.

In the following simulations, we adopted a double integrator dynamics (which is a fair model provided hard constraints are enforced on the velocity and trajectory [55]) for a vehicle moving in $\mathbb{R}^2$ motivated by applications in maritime and underwater vessels where range and bearing measurements are a relevant type of sensor information. The continuous dynamics are discretized with a sample and hold strategy and a sampling time of Ts = 0.1s and it is assumed to exist a state-feedback controller on board corresponding to the solution K of the discrete Linear Quadratic Regulator with parameters $Q = 10I_4$ and $R = I_2$:

$$K = \begin{bmatrix} 2.5857 & 0 & 3.4434 & 0 \\ 0 & 2.5857 & 0 & 3.4434 \end{bmatrix}. \tag{3.27}$$

Initial state is $x(0) = \begin{bmatrix} 9 & 9 & 0 & 0 \end{bmatrix}^\intercal$ (meaning that the vehicle is stopped at point with coordinates $\begin{bmatrix} 9 & 9 \end{bmatrix}^\intercal$) and the initial estimate $X(0) = \left(5I_4, \begin{bmatrix} 10 & 10 & 0 & 0 \end{bmatrix}^\intercal, 0_4^\intercal, 0, \{\mathcal{B}_\infty\}\right)$,

i.e., both position and velocity have an uncertainty of ±5 with the center of the estimate representing a stopped vehicle at coordinates $\begin{bmatrix} 10 & 10 \end{bmatrix}^\top$. Lastly, the source of the readings is placed at the origin and the simulations were run in Matlab R2018a running on a HP machine with a Intel Core i7-8550U CPU @ 1.80GHz and 12 GB of memory resorting to the overloaded plot function by Yalmip to depict the sets and using Mosek as the underlying solver. Videos of the simulations and figures can be found in `https://github.com/danielmsilvestre/CCGpaper`.



(a) Vehicle performing a 200 time instant trajectory of a figure 8.

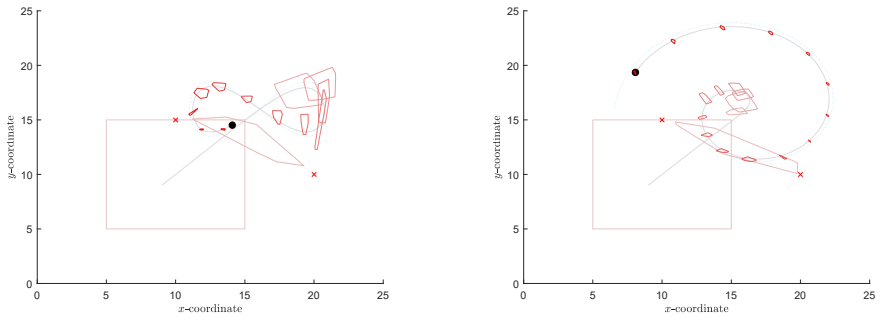
(b) Vehicle performing a 250 time instant trajectory of a of a spiral.

Fig. 3.2: Trajectory intended (light blue dots) and realized path done by the vehicle (grey) and the set-valued estimates obtained from Range and Bearing measurements drawn at each 15 iterations going from time instant 1 (lighter) to the end of the simulation (darker).

Figure 3.2a depicts the vessel doing a figure-8 with both range and bearing measurements during a simulation of 200 time instants corresponding to 20s. The sets are shown every 15 iterations to avoid cluttering the image. Over the 200 iterations, the construction of the sets took on average 8.1×10^{-3} seconds with a variance of 3.87×10^{-5}, which is much faster than the 10^{-1} for the sampling time. In terms of the sizes of the data structures, the CCG at time k required storing $4 + 8k$ generator variables, $1 + 9k$ linear equalities and an additional vector with $4 + 8k$ entries (one for each generator variable) storing a numeric value starting by an identifier number for the type of generator followed by a unique identifier so that we can group generator variables that are defined within the same generator. This means a linear growth of $2n_x$ and $2n_x + 1$ for this type of measurement. At each iteration, we solved a linear objective function constrained on the point belonging to $X(k)$. Over the 200 iterations, it took on average 0.0151 seconds with the maximum being 0.0516 seconds. For instance, a problem at iteration 200 took 0.024 seconds and a collision check took around 0.0197 seconds. In addition, we compared the hyper-volume of the sets produced by CZs against CCGs and on average throughout the whole simulation they are larger 6.6% with a maximum 34.35%. This percentage increases with the

difference $b_u - b_l$, i.e., with the error on the bearing measurement as seen in Figure 3.1.

In Figure 3.2b, we present a simulation with a different trajectory that promotes a sharper turn that offers the possibility for a better estimate since the control law is exciting the system from an earlier point in the simulation. This is confirmed by the sizes of the sets that decrease after the turn. With respect to the performance of constructing the sets and conservatism, the results are similar to those of Figure 3.2a.



(a) Vehicle performing a 200 time instant trajectory of a figure 8.

(b) Vehicle performing a 300 time instant trajectory of a spiral.

Fig. 3.3: Trajectory intended (light blue dots) with the actual trajectory done by the vehicle (grey) and the set-valued estimates drawn at each 15 iterations going from time instant 1 (lighter) to time instant 200 (darker) obtained using bearing-only measurements from the two towers (red crosses).

We have also simulated a case where the observer has access to bearing-only measurements corrupted by noise from two beacons assuming that the vessel is within 10 distance units. This is a case that cannot be represented by a CZ unless we assume some arbitrarily large constant to eventually bound the set as it is not possible to represent unbounded sets. The beacons are placed at positions $\begin{bmatrix} 10 & 15 \end{bmatrix}^\top$ and $\begin{bmatrix} 20 & 10 \end{bmatrix}^\top$ as to make segments of the trajectory be served by a different beacon. The estimation task has a better performance given that the measurement sets can be described using fewer generator variables. The mean time to compute the sets was 0.0016s with the worst-case taking 0.007s, which reinforces the usability of CCGs in real time. The number of generator variables increases as $4 + 2k$ given that the set $Y(k)$ can be described with 2 variables. The reduced description of the sets also decreases the time it took to check whether the set is empty or if a point belongs to the set, achieving a mean solver time of 0.0035s with the worst-case being 0.0073s.

The last simulation used a spiral trajectory in order get a richer set of measurements. The results are depicted in Figure 3.3b and show that the set shrinks rapidly and allows for a good estimation performance. No order reductions were performed and the computational times are very similar to the case of the figure 8 with bearing measurements.

3.5 Deterministic Estimation for Uncertain Linear Dynamical Systems

The problem of state estimation in uncertain Linear Parameter-Varyings (LPVs) can be cast as finding a set of possible values given the measurements, disturbance, noise and initial state bounds when the model is given by:

$$\begin{aligned} x(k+1) &= \left(F(\rho(k)) + \sum_{\ell=1}^{n_\Delta} \Delta_\ell(k) U_\ell\right) x(k) + B(\rho(k))u(k) + w(k) \\ y(k) &= C(\rho(k))x(k) + v(k) \end{aligned} \tag{3.28}$$

where $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^{n_u}$, $w(k) \in \mathbb{R}^n$, $y(k) \in \mathbb{R}^{n_y}$ and $v(k) \in \mathbb{R}^{n^y}$ are the system state, input, disturbance signal, output and noise, respectively. The parameter $\rho(k)$ is the part of the parameters that can be measured at time k, which can be treated as in the case of LTVs. The main challenge appears from considering the n_Δ uncertainties denoted by Δ_ℓ and the constant matrices U_ℓ that account for how the uncertainties affect the nominal dynamics matrix given by $F(\rho(k))$. To lighten the notation, we will consider $F_k := F(\rho(k))$ and similarly for all the remaining matrices in (3.28). Notice that we have to explicitly consider ρ to account for nonlinearities that enter the model in a linear fashion as will happen with unicycle model used in Section 3.7. Moreover, in order to have a well-posed problem, we assume that all unknown signals are bounded within a compact convex set denoted by the correspondent capital letter, i.e., $x(0) \in X(0)$, $w(k) \in W(k)$ and $v(k) \in V(k)$. Without loss of generality, we will assume that $\forall k, |\Delta_\ell(k)| \leq 1$.

The problem addressed in this paper is summarized as:

Problem 3.2 Given compact convex sets $X(0)$, $W(k)$ and $V(k)$ for all $k \geq 0$ and measurements $y(k)$, how to compute a set $X(k)$ such that it is guaranteed that $x(k) \in X(k)$, $\forall k \geq 0$.

Notice that Problem 3.2 is called *state estimation* although a converse definition could be presented for the output of the system (this is of particular interest in sensitivity analysis ([68]) and system distinguishability ([69])).

3.6 State Estimation for Uncertain LPVs using Constrained Convex Generators (CCGs)

The state estimation for the uncertain cases has to account for the polytopic uncertainty in the dynamics in the propagation phase using the model, namely, set $X_{\text{prop}}(k+1)$ after applying the dynamics to $X(k)$ can be written as:

$$\begin{aligned} X_{\text{prop}}(k+1) =& \text{cvxHull}\left(\bigcup_{\Delta \in \text{vertex}([-1,1]^{n_\Delta})} \left(F_k + \sum_{\ell=1}^{n_\Delta} \Delta_\ell(k) U_\ell\right) X(k)\right) \\ &+ B_k u(k) \oplus W(k), \end{aligned} \tag{3.29}$$

where cvxHull is the convex hull function.

The update phase corresponding to intersection with the measurement set $Y(k+1)$, i.e., all state values that could result in the measurement $y(k+1)$, that corresponds to:

$$X(k+1) = X_{\text{prop}}(k+1) \cap_C Y(k+1). \tag{3.30}$$

3.6.1 Convex Hull for CCGs

Let us start by defining the convex hull of two sets:

$$\begin{aligned} \text{cvxHull}\,(Z_1, Z_2) := \big\{ z : z =& \lambda z_1 + (1-\lambda) z_2, \\ & \lambda \in [0,1], z_1 \in Z_1, z_2 \in Z_2 \big\}. \end{aligned} \tag{3.31}$$

Let us introduce a specific instance of norm cones that are going to be used in the following result. For a norm unity ball C defined as $\|\xi\|_p \leq 1$, let us associate with it the correspondent norm cone of order zero $C^{(0)}(\xi, \lambda, a, b) := \|\xi\|_p + w_0 \lambda \leq v_0$ with the initialization of the row vector w_0 and column vector λ as empty and scalar $v_0 = 1$. In the base case, we can omit the arguments with a slight abuse of notation. We can now define norm cones of higher order of this operation in a recursive manner $C^{(\tau)}(\xi, \lambda, a, b) := \|\xi\|_p + \begin{bmatrix} a & b w_{\tau-1} \end{bmatrix} \lambda \leq b v_{\tau-1}$, such that the generator variables are $\lambda \in \mathbb{R}^\tau$ and ξ with the same dimension as the zero order cone and constant arguments a and b.

We can now state the main theorem introducing the closed-form expression for the convex hull of two CCGs and the complexity of this representation.

Theorem 3.1 *Consider two CCGs as in Definition 3.3:*

- $X = (G_x, c_x, A_x, b_x, C_x^{(\tau_x)}) \subset \mathbb{R}^n$;

- $Y = (G_y, c_y, A_y, b_y, C_y^{(\tau_y)}) \subset \mathbb{R}^n$;

such that $A_x \in \mathbb{R}^{n_c^x \times n_g^x}$, $A_y \in \mathbb{R}^{n_c^y \times n_g^y}$, $\xi_x \in C_x^{(\tau_x)} \implies \alpha\xi_x \in C_x^{(\tau_x)}$, *for* $\alpha \in [0, 1]$ *and similarly for* $C_y^{(\tau_y)}$. *The CCG corresponding to the exact convex hull* $Z_h = (G_h, c_h, A_h, b_h, C_h) \subset \mathbb{R}^n$ *is given by:*

$$\begin{aligned} G_h &= \begin{bmatrix} G_x & G_y & c_x - c_y \end{bmatrix}, c_h = \frac{c_x + c_y}{2}, \\ A_h &= \begin{bmatrix} A_x & 0 & -b_x \\ 0 & A_y & b_y \end{bmatrix}, b_h = \begin{bmatrix} \frac{1}{2}b_x \\ \frac{1}{2}b_y \end{bmatrix} \\ C_h &= \{C_x^{(\tau_x+1)}(\xi_x, \xi_\lambda, -1, 0.5), C_y^{(\tau_y+1)}(\xi_y, \xi_\lambda, 1, 0.5), \mathbb{R}\}, \end{aligned} \tag{3.32}$$

which has $n_g^x + n_g^y + 1$ *generators and* $n_c^x + n_c^y$ *constraints.*

Proof. Following Theorem 1 from ([70]), we write Z_h as:

$$\begin{aligned} Z_h = \{p_h = G_x\xi_x + \lambda c_x + G_y\xi_y + (1-\lambda)c_y : \\ 0 \le \lambda \le 1, A_x\xi_x = \lambda b_x, A_y\xi_y = (1-\lambda)b_y, \\ \|\xi_x\|_{\ell_x} \le \lambda, \|\xi_y\|_{\ell_y} \le (1-\lambda)\} \end{aligned} \tag{3.33}$$

when in the presence of unit balls.

By performing the substitution $\xi_\lambda = \lambda - 0.5$, we obtain a generator variable that belongs to the interval $[-0.5, 0.5]$ and after reorganizing to write everything in terms of $\xi_h = \begin{bmatrix} \xi_x^\intercal & \xi_y^\intercal & \xi_\lambda^\intercal \end{bmatrix}^\intercal$, we obtain:

$$\begin{aligned} Z_h = \{p_h = G_h\xi_h + c_h : \\ A_h\xi_h = b_h, \|\xi_x\|_{\ell_x} \le 0.5 + \xi_\lambda, \|\xi_y\|_{\ell_y} \le 0.5 - \xi_\lambda\}. \end{aligned} \tag{3.34}$$

where the norm cones correspond to $C_x^{(1)}(\xi_x, \xi_\lambda, -1, 0.5)$ and $C_y^{(1)}(\xi_y, \xi_\lambda, 1, 0.5)$. If on the other hand, we have a norm cones of order τ_x and τ_y, respectively, we obtain the expression $C_x^{(\tau_x+1)}(\xi_x, \xi_\lambda, -1, 0.5)$ and $C_y^{(\tau_y+1)}(\xi_y, \xi_\lambda, 1, 0.5)$. The number of generators and constraints results from the size of matrix A_h, which concludes the proof. □

Theorem 3.1 produces the exact convex hull as no approximation was required and is available in the toolbox ReachTool that can be found in `https://github.com/danielmsilvestre/ReachTool`. Figure 3.4 depicts an example of sets Z_1 and Z_2 with the respective set Z_h as given by Theorem 3.1 and what one would get if first converted the sets to CZs and then applied the exact convex hull given in ([71]). As observed, the proposed method is tight for CCGs and offers better accuracy in

comparison to the result from ([71]). Moreover, since CZs are an instance of CCGs where the generator sets are ℓ_∞ unit balls, a corollary from Theorem 3.1 is that the optimal representation of the convex hull of two CZs is possible only in the more general CCG format.

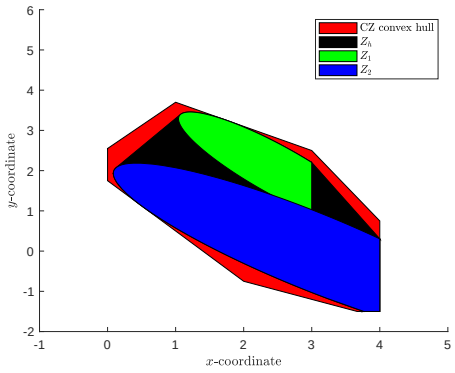


Fig. 3.4: Comparison between the set Z_h and the convex hull that one would obtain if first converted both Z_1 and Z_2 to CZs by overbounding all convex generators by the ℓ_∞ unit ball.

The convex hull operator increases linearly the number of auxiliary variables to $n_g^x + n_g^y + 1$, however, this step has to be performed for all vertices which are exponential in the number of uncertainties. Such an issue was already present in ([65]) for polytopic set descriptions using the optimal convex hull formulation.

In order to keep the computation time for each iteration bounded, we introduce the order reduction in Algorithm 2, which computes a CCG with a specified number of constraints γ using $n+\gamma$ generators which is of the form of a polytope. The procedure starts by constructing a collections of hyperplanes tangent to the surface and then converting to CCG representation. The min and max operations are element-wise.

3.7 Simulations for Estimation of an Uncertain LPV

In this section, simulations results are presented for a unicycle model of an autonomous vehicle in discrete-time for which there is a digital compass as an onboard sensor providing measurements of the orientation angle with a $\pm 5^\circ$ error. Simulations were run in Matlab R2018a running on a HP machine with a Intel Core i7-8550U CPU @ 1.80GHz and 12 GB of memory resorting to Yalmip as the language to model optimization problems and Mosek as the underlying solver. Videos, figures and code can be found in `https://github.com/danielmsilvestre/CCGExactConvexHull`

Algorithm 2 Order Reduction using points on the surface.

Require: Set $X(K) \subseteq \mathbb{R}^n$ and desired order γ.
Ensure: Calculation of $X(k) \subseteq X_{\text{red}}(k) \subseteq \mathbb{R}^n$ with $n_g = \gamma + n$ generators and $n_c = \gamma$ constraints.

1: /* *Get points p_i on the surface such that $p_i = \arg\max v_i^\intercal p_i$, $1 \leq i \leq \gamma$ for random v_i* */
2: $[v, p] = \text{sampleSurface}(X(k), \gamma)$
3: /* *Compute box $\tilde{Z}$ for the points p* */
4: $\tilde{Z} = \left(\frac{1}{2}\text{diag}(\max p - \min p), \frac{1}{2}(\max p + \min p), [\ ], [\ ], \|\tilde{\xi}\|_\infty \leq 1\right)$
5: /* *Calculate b and σ such that all entries $v_i^\intercal p_i \in [\sigma, b]$*/
6: $\sigma = \min v^\intercal p$
7: $b = \text{diag}(v^\intercal p)$
8: $X_{\text{red}}(k) = \left(\left[\tilde{Z}.G\ \ 0_{n\times\gamma}\right], \tilde{Z}.c, \left[v^\intercal \tilde{Z}.G\ \ \frac{1}{2}\text{diag}(\sigma - b)\right], \frac{b+\sigma}{2} - v^\intercal \tilde{Z}.c, \|\tilde{\xi}\|_\infty \leq 1\right)$

We recover the example considering unicycle dynamics described in ([72]). The vehicle schematic representation is given in Figure 3.5 and has the following dynamics in discrete-time:

$$\begin{bmatrix} p_i \\ q_i \end{bmatrix}(k+1) = \begin{bmatrix} p_i \\ q_i \end{bmatrix}(k) + \text{Ts}\ A_i(\theta_i) \begin{bmatrix} v_i \\ w_i \end{bmatrix}(k) \tag{3.35}$$

where the state (p_i, q_i) identify the position of the front of the ith vehicle and the inputs (v_i, w_i) account for the linear velocity and rotation. Moreover, Ts = 0.1 stands for the sampling time, θ_i (we omit the time dependence in k for a more compact presentation) for the orientation and matrix $A_i(\theta_i)$ is given as:

$$A_i(\theta_i) = \begin{bmatrix} \cos\theta_i & -l\sin\theta_i \\ \sin\theta_i & l\cos\theta_i \end{bmatrix}. \tag{3.36}$$

In this simulation, we consider a single vehicle running for a total of 15 seconds and, assuming that the compass takes measurements $\hat{\theta_1}$ of the true variable θ_1 that have a maximum of $\pm 5°$ following a uniform distribution. Therefore, at each iteration time k, matrix A_1 in the dynamics is not available to the observer and we have to consider $\hat{\theta_1}$ to generate the nominal dynamics and an uncertainty Δ_1 with maximum magnitude of 5°, which fits (3.28).

The trajectory-tracking control law used is:

$$\begin{bmatrix} v_i(k) \\ w_i(k) \end{bmatrix} = \frac{A_i^{-1}(\theta_i)}{\text{Ts}} \left(\tau(k+1) - \frac{\tau(k)}{2} - 0.5 \begin{bmatrix} p_i(k) \\ q_i(k) \end{bmatrix} + d(k) \right) \tag{3.37}$$

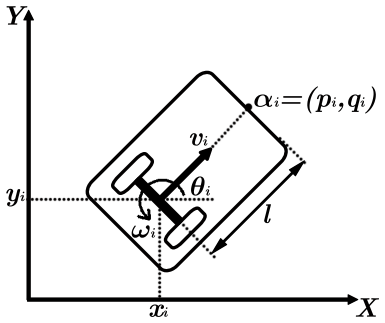


Fig. 3.5: Schematic of the unicycle model for the vehicles.

where $\tau(k)$ accounts for the discrete sequence of waypoints in the trajectory. Once again, we assume that there is a telemetry sensor that produces estimates corrupted by noise of the value of $p_1(k)$ and $q_1(k)$ and add the corresponding disturbance term $d(k)$ to account for those differences. Moreover, there are two beacons at positions $\begin{bmatrix} 5 & 25 \end{bmatrix}^\intercal$ and $\begin{bmatrix} 23 & 10 \end{bmatrix}^\intercal$ that can be detected within a 5 and 2 units of distance which allows to better localize the vehicle.

The vehicle performs a figure 8 trajectory such that it can only get measurements from each beacon in one time interval. Figure 3.6 illustrates the volume evolution for the set-valued estimates $X(k)$ when using CZs ([56]) and CCGs when both used the same order reduction method in Section 3.3. Since the vehicle is moving and most of the time performing dead reckoning with the uncertain LPV model, the volume keeps increasing and is lowered when the vehicle reaches the beacon areas. The main trend to observe is that the added accuracy of the ℓ_2 ball representing the range measurement from the beacon contributes to a better performance of the CCG filter.

In Figure 3.7, it is illustrated the trajectory executed by the vehicle and the corresponding set-valued estimates using both the CZ and CCG approaches. We have selected a small number of time instants to display the sets as to avoid cluttering the image, but the full video can be found in the GitHub repository associated with the paper.

A last relevant issue is the elapsed time in each iteration taken by both filters with different set representations. Figure 3.8 shows the computation times across iterations during the whole simulation. At the beginning, both filters have very similar behavior pointing out to the fact that the CCG is yet to have round facets and the order reduction produces equivalent representations. However, as the simulation progresses the set is intersected with the range measurements. The curved boundaries of the CCGs result in a more complex representation. When the vehicle finds the second beacon and the set is considerably reduced in size, the CZ approach has a better performance given that $X(k)$ has a shape close to an interval, where its accuracy is the worst. This result points out to the need to further develop order

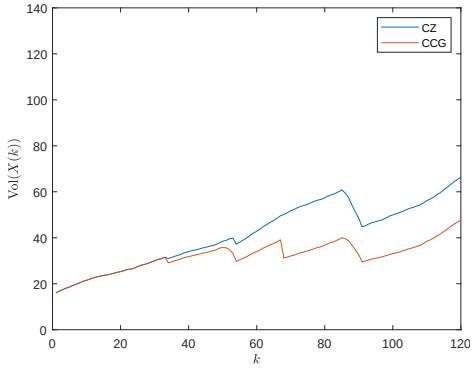


Fig. 3.6: Comparison of the volume for both set-valued estimates when using CZs and CCGs for the figure 8 trajectory.

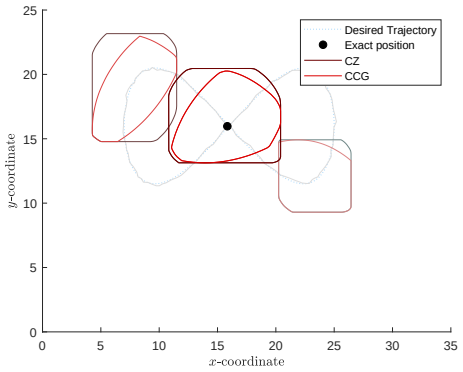


Fig. 3.7: Trajectory executed by the vehicle and the correspondent set-valued estimates at multiples of 40 iterations when using CZs and CCGs for the figure 8 trajectory.

reduction methods for CCGs that can exploit the nature of the sets. This is not a trivial task given the requirement of computing an outer-approximation to maintain the guaranteed feature in the estimation using set-membership approaches.

In order to illustrate an example where both filters should be similar, we simulated a spiral trajectory and increased the range of the beacons in 5 meters each. In this case, the trajectory is not taking advantage of the two beacons. However, the fact that the vehicle will receive the beacon more often should compensate. Figure 3.9 showcases that the volume is indeed much smaller for this trajectory since the vehicle performs dead reckoning less often. In this setup, the main difference between the two filters is precisely the representation of the circular shapes that benefits the CCGs.

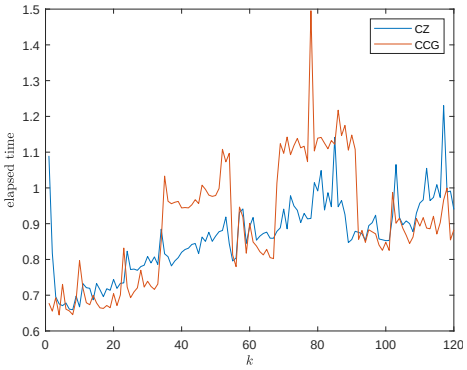


Fig. 3.8: Elapsed time for each iteration of both methods taking into account the constructiong of the set, approximation algorithm and volume computation.

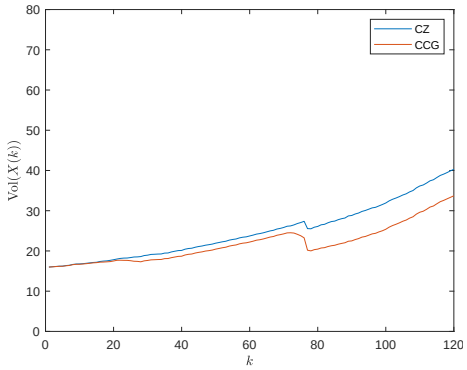


Fig. 3.9: Comparison of the volume for both set-valued estimates when using CZs and CCGs for the spiral trajectory.

In Figure 3.10, it is depicted the same snapshots for the trajectory where it is noticeable the rounded shapes corresponding to the range measurements. However, as seen in Figure 3.11, the more complicated set representation also reduces the performance of both filters. Similarly to the figure 8 trajectory scenario, both simulations illustrate a clear reduction in the conservatism without a very expressive increase in elapsed time for the overall computations. We remark that in terms of orders of magnitude, both filters in normal operation will take between 0.6 and 1.5 seconds, which is not viable for real-time applications and showcases the need to further purse efficient order reductions methods. We did not use the methods from CORA toolbox since we were obtaining even larger computing times.

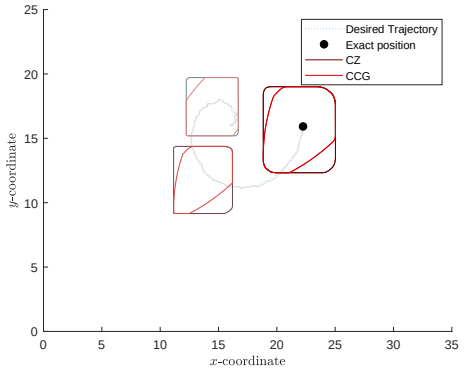


Fig. 3.10: Trajectory executed by the vehicle and the correspondent set-valued estimates at multiples of 40 iterations when using CZs and CCGs for the spiral trajectory.

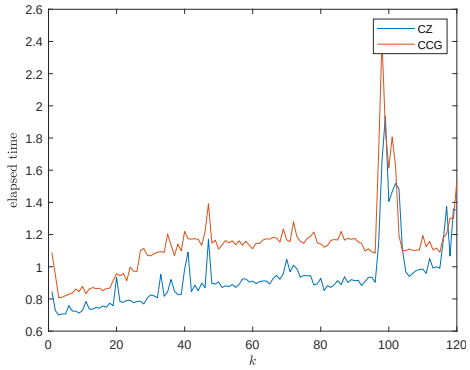


Fig. 3.11: Elapsed time for each iteration of both methods taking into account the constructiong of the set, approximation algorithm and volume computation in the spiral trajectory scenario.

3.8 Order Reduction Method of CCGs with Ellipsoids

One of the problems of using any type of observer for deterministic estimation is an inherent trade-off between accuracy and complexity of the data structure. In particular, CCGs are exact but their representation datastructures increase in size as more operations are executed. For systems with uncertainties, this is quite problematic as we had to consider all vertices of the polytopic uncertainty. For the previous simulations, we have used a straightforward method that samples the surface, however it becomes impractical for large state-spaces. For that reason, we have developed techniques to reduce the order of a CCG, which is defined as $\frac{n_g}{n}$. It represents the computational burden required to describe the set. The approach in

this section is a method to eliminate generators and another to reduce the number of constraints.

3.8.1 Order Reduction

Before proceeding to the order reduction algorithm we need the following results. First, an ellipsoid can overapproximate a CCG with the following procedure, which amounts to enclose the overall generator set $C_1 \times \ldots \times C_{n_p}$ within an ellipsoid.

Lemma 3.1 *Consider* $Z = \left(G, c, A, b, \{C_1, \cdots C_{n_p}\}\right)$*, where* G *and* A *have the following structure*

$$A \coloneqq \begin{bmatrix} A_1 \; A_2 \; \ldots \; A_{n_p} \end{bmatrix}, \tag{3.38a}$$

$$G \coloneqq \begin{bmatrix} G_1 \; G_2 \; \ldots \; G_{n_p} \end{bmatrix}, \tag{3.38b}$$

with $A_i \in \mathbb{R}^{n_c \times m_i}$ *and* $G_i \in \mathbb{R}^{n \times m_i}$*, and the ellipsoid* $\bar{Z}$ *given by*

$$\bar{Z} \coloneqq \{\bar{G}\xi + c : \bar{A}\xi = b, \|\xi\|_2 \leq 1\}, \tag{3.39}$$

with

$$\bar{A} \coloneqq \begin{bmatrix} \frac{a_1}{\sqrt{d_1}} A_1 \; \frac{a_2}{\sqrt{d_2}} A_2 \; \ldots \; \frac{a_{n_p}}{\sqrt{d_{n_p}}} A_{n_p} \end{bmatrix}, \tag{3.40a}$$

$$\bar{G} \coloneqq \begin{bmatrix} \frac{a_1}{\sqrt{d_1}} G_1 \; \frac{a_2}{\sqrt{d_2}} G_2 \; \ldots \; \frac{a_{n_p}}{\sqrt{d_{n_p}}} G_{n_p} \end{bmatrix}, \tag{3.40b}$$

$$a_i \coloneqq \max\left(1, \frac{\sqrt{m_i}}{m_i^{\frac{1}{p_i}}}\right). \tag{3.40c}$$

If constants d_i*,* $i \in \{1, \ldots, n_p\}$ *satisfy* $d_i > 0$ *and* $\sum_{i=1}^{n_p} d_i = 1$*, then,* $Z \subset \bar{Z}$*.*

Proof. Note first that, as shown in [73], a_i is the lowest constant such that for all $\xi_i \in \mathbb{R}^{m_i}$,

$$\|\xi_i\|_2 \leq a_i \|\xi_i\|_{p_i}. \tag{3.41}$$

Choose any $z \in Z$, consider the following generator transformation

$$\tilde{\xi} := \begin{bmatrix} \frac{\sqrt{d_1}}{a_1}\xi_1 \\ \frac{\sqrt{d_2}}{a_2}\xi_2 \\ \vdots \\ \frac{\sqrt{d_{n_p}}}{a_{n_p}}\xi_{n_p} \end{bmatrix}. \tag{3.42}$$

From the definition of the 2-norm and (3.41) we obtain

$$\|\tilde{\xi}\|_2^2 = \sum_{i=1}^{n_p} \frac{d_i}{a_i^2}\|\xi_i\|_2^2 \leq \sum_{i=1}^{n_p} d_i \|\xi_i\|_{p_i}^2. \tag{3.43}$$

Since the generator elements $\xi_i \in \mathbb{R}^{m_i}$ satisfy $\|\xi_i\|_{p_i} \leq 1$ and given that $\sum_{i=1}^{n_p} d_i = 1$ we have that

$$\sum_{i=1}^{n_p} d_i \|\xi_i\|_{p_i}^2 \leq \sum_{i=1}^{n_p} d_i = 1, \tag{3.44}$$

and therefore $\|\tilde{\xi}\|_2 \leq 1$. If we consider $\tilde{\xi}$ as the new vector of generators, we have to rewrite the matrices as:

$$G\xi = \sum_{i=1}^{n_p} G_i\xi_i = \sum_{i=1}^{n_p} \bar{G}_i \frac{\sqrt{d_i}}{a_i}\xi_i = \bar{G}\tilde{\xi}, \tag{3.45a}$$

$$A\xi = \sum_{i=1}^{n_p} A_i\xi_i = \sum_{i=1}^{n_p} \bar{A}_i \frac{\sqrt{d_i}}{a_i}\xi_i = \bar{A}\tilde{\xi}, \tag{3.45b}$$

and we can conclude that $z \in \bar{Z}$, given by (3.39) as we wanted to show. □

To determine the weights d_i one can set $d_i = \frac{1}{n_p}$ or, in the unconstrained case if more precision is required, solve the following convex optimization problem.

Lemma 3.2 *Given the ellipsoidal overapproximation* (3.39)*, for an unconstrained CCG choosing the weights d_i for $i \in \{1, \ldots, n_p\}$ that minimize the volume of $\bar{Z}$ amounts to solving the following convex optimization problem.*

$$\begin{aligned} \min_{d_i} \quad & -\log\left(\det\left(G^{\dagger\intercal}\,\mathrm{diag}\left(\frac{d_i}{a_i^2}I_{m_i}\right)G^{\dagger}\right)\right) \\ s.t. \quad & \sum_{i=1}^{n_p} d_i = 1. \end{aligned} \tag{3.46}$$

Proof. Given $x \in \mathbb{R}^n$ the vector ξ with minimum 2-norm such that

$$x = \bar{G}\xi \tag{3.47}$$

is given by

$$\xi = \bar{G}^{\dagger} x. \tag{3.48}$$

Note that since

$$\bar{G} = G \operatorname{diag}\left(\frac{a_i}{\sqrt{d_i}} I_{m_i}\right), \tag{3.49}$$

where in $\operatorname{diag}\left(\frac{a_i}{\sqrt{d_i}} I_{m_i}\right)$, $i \in \{1, \ldots, n_p\}$, we have that

$$\bar{G}^{\dagger} = \operatorname{diag}\left(\frac{\sqrt{d_i}}{a_i} I_{m_i}\right) G^{\dagger}. \tag{3.50}$$

Therefore $\|\xi\|_2 \leq 1$ is equivalent to

$$x^{\intercal} G^{\dagger\intercal} \operatorname{diag}\left(\frac{d_i}{a_i^2} I_{m_i}\right) G^{\dagger} x \leq 1. \tag{3.51}$$

The result ensues by noting that minimizing the volume of an ellipsoid given by (3.51) is equivalent to maximizing

$$\det\left(G^{\dagger\intercal} \operatorname{diag}\left(\frac{d_i}{a_i^2} I_{m_i}\right) G^{\dagger}\right), \tag{3.52}$$

that $\log(\cdot)$ is a monotonically increasing function and that $-\log(\det(\cdot))$ is a convex function. □

Based on Lemmas 11 and 12 in [74], a constrained ellipsoid can always be expressed without constraints as in the following Lemma, which corresponds to Lemma 13 in [74], making explicit the expressions for matrices T and t.

Lemma 3.3 *Given a constrained ellipsoid of the form*

$$X = (G, c, A, b, \|\xi\|_2 \leq 1), \tag{3.53}$$

it can be expressed with $n_g - n_c$ generators as

$$X \equiv \tilde{X} \coloneqq (GT, c + Gt, [\,], [\,], \|\xi\|_2 \leq 1), \tag{3.54}$$

where

$$t \coloneqq A^{\dagger} b, \tag{3.55a}$$

$$T \coloneqq \sqrt{1 - \|t\|^2}\, \operatorname{null}(A), \tag{3.55b}$$

$A^{\dagger}$ is the pseudo-inverse of A and $\operatorname{null}(A)$ *is an orthonormal basis of the null-space of A.*

Proof. To prove the lemma we must show that these sets are equivalent

$$\{\xi : A\xi = b, \|\xi\|_2 \leq 1\} \equiv \left\{T\tilde{\xi} + t : \|\tilde{\xi}\|_2 \leq 1\right\}. \tag{3.56}$$

This can be shown as follows. In the first set, we can perform the transformation $\xi = \bar{\xi} + t$, and noting that, from (3.55a), $At = b$ we obtain that

$$\{\xi : A\xi = b, \|\xi\|_2 \leq 1\} \tag{3.57}$$

is equivalent to

$$\left\{\bar{\xi} + t : A\bar{\xi} = \mathbf{0}, \|\bar{\xi} + t\|_2 \leq 1\right\}. \tag{3.58}$$

Given that $A\bar{\xi} = \mathbf{0}$, $\bar{\xi}$ is in the nullspace of A, and therefore without loss of generality we can make the transformation $\bar{\xi} = T\tilde{\xi}$. Given that, from (3.55), $A^{\dagger\intercal}\,\text{null}(A) = \mathbf{0}$ and $t^\intercal T = \mathbf{0}$, we have,

$$\|T\tilde{\xi} + t\|_2 = \sqrt{\|T\tilde{\xi}\|_2^2 + \|t\|_2^2}. \tag{3.59}$$

Then, from the fact that $\|T\tilde{\xi}\|_2 = \sqrt{1 - \|t\|_2^2}\|\tilde{\xi}\|_2$ we have that (3.58) is equivalent to

$$\left\{T\tilde{\xi} + t : \|\tilde{\xi}\|_2 \leq 1\right\}, \tag{3.60}$$

as we wanted to show. □

Finally, an ellipsoid can always be expressed with a vector of generators of size $\text{rank}(G)$ with the following method

Lemma 3.4 *An ellipsoid of the form*

$$X = (G, c, [\,], [\,], \|\xi\|_2 \leq 1), \tag{3.61}$$

can always be expressed with $\text{rank}(G)$ *generators by taking the singular value decomposition of* G

$$USV^\intercal = G \tag{3.62}$$

and computing

$$X \equiv \tilde{X} \coloneqq (US, c, [\,], [\,], \|\xi\|_2 \leq 1). \tag{3.63}$$

Proof. Performing the generator transformation $\tilde{\xi} = V^\intercal\xi$, since V is orthonormal we have that

$$\|\tilde{\xi}\|_2 \leq \|\xi\|_2 \leq 1, \tag{3.64}$$

thus concluding the proof. □

A similar exact order reduction method for ellipsoids was presented in [74]. However, since the order reduction method of Lemma 3.4 consists of performing a

singular value decomposition on $G \in \mathbb{R}^{n \times n_g}$ its complexity is $O(nn_g^2)$, whereas the complexity of the method in [74] is $O(n^3 + n_g^3)$.

Since we are interested in reducing the number of generators without changing the effect of some or most of the generators, we adopt a method based on the lift-then-reduce strategy for CZs [56] to partially reduce the order of a CCG.

Lemma 3.5 *Consider a CCG and a partition of G and A as*

$$A \coloneqq \begin{bmatrix} \bar{A}_1 & \bar{A}_2 \end{bmatrix}, \tag{3.65a}$$

$$G \coloneqq \begin{bmatrix} \bar{G}_1 & \bar{G}_2 \end{bmatrix}, \tag{3.65b}$$

where

$$\bar{A}_1 \coloneqq \begin{bmatrix} A_1 & \dots & A_{\bar{n}_p} \end{bmatrix}, \tag{3.66a}$$

$$\bar{A}_2 \coloneqq \begin{bmatrix} A_{\bar{n}_p+1} & \dots & A_{n_p} \end{bmatrix}, \tag{3.66b}$$

$$\bar{G}_1 \coloneqq \begin{bmatrix} G_1 & \dots & G_{\bar{n}_p} \end{bmatrix}, \tag{3.66c}$$

$$\bar{G}_2 \coloneqq \begin{bmatrix} G_{\bar{n}_p+1} & \dots & G_{n_p} \end{bmatrix}, \tag{3.66d}$$

for some integer $\bar{n}_p$ such that $1 \le \bar{n}_p \le n_p$ Given matrices $\tilde{G}_2 \in \mathbb{R}^{n \times (n+n_c)}$ and $\tilde{A}_2 \in \mathbb{R}^{n_c \times (n+n_c)}$ that yield the following ellipsoidal overbound

$$\left(\begin{bmatrix} \bar{G}_2 \\ \bar{A}_2 \end{bmatrix}, \mathbf{0}, [\,], [\,], C_{\bar{n}_p+1} \times \ldots \times C_{n_p} \right) \subset \left(\begin{bmatrix} \tilde{G}_2 \\ \tilde{A}_2 \end{bmatrix}, \mathbf{0}, [\,], [\,], \|\xi\|_2 \le 1 \right), \tag{3.67}$$

the CCG can be overapproximated by

$$\left(\begin{bmatrix} \bar{G}_1 & \tilde{G}_2 \end{bmatrix}, c, \begin{bmatrix} \bar{A}_1 & \tilde{A}_2 \end{bmatrix}, b, C_1 \times \ldots \times C_{\bar{n}_p} \times \tilde{C} \right), \tag{3.68}$$

where

$$\tilde{C} \coloneqq \{\xi : \|\xi\|_2 \le 1\} \subset \mathbb{R}^{n+n_c}. \tag{3.69}$$

Proof. Choose any $z \in Z$, which is equivalent to

$$\begin{bmatrix} z \\ \mathbf{0} \end{bmatrix} \in \bar{Z} \coloneqq \left(\begin{bmatrix} G \\ A \end{bmatrix}, \begin{bmatrix} c \\ -b \end{bmatrix}, [\,], [\,], C_1 \times \ldots \times C_{n_p} \right). \tag{3.70}$$

Notice that $\bar{Z}$ can be expressed as $\bar{Z} = Z_1 \oplus Z_2$ where

$$Z_1 \coloneqq \left(\begin{bmatrix} \bar{G}_1 \\ \bar{A}_1 \end{bmatrix}, \begin{bmatrix} c \\ -b \end{bmatrix}, [\,], [\,], C_1 \times \ldots \times C_{\bar{n}_p} \right), \tag{3.71a}$$

$$Z_2 \coloneqq \left(\begin{bmatrix} \bar{G}_2 \\ \bar{A}_2 \end{bmatrix}, \mathbf{0}, [\,], [\,], C_{\bar{n}_p+1} \times \ldots \times C_{n_p} \right). \tag{3.71b}$$

From (3.67) we have that $Z_2 \subset \bar{Z}_2$ where

$$\bar{Z}_2 \coloneqq \left(\begin{bmatrix} \tilde{G}_2 \\ \tilde{A}_2 \end{bmatrix}, \mathbf{0}, [\,], [\,], \|\xi\|_2 \leq 1 \right). \tag{3.72}$$

Using the expression for the Minkowski sum in Proposition 3.1 we have that $\begin{bmatrix} z^\intercal & \mathbf{0}^\intercal \end{bmatrix}^\intercal$ is in the set

$$\left(\begin{bmatrix} \bar{G}_1 & \tilde{G}_2 \\ \bar{A}_1 & \tilde{A}_2 \end{bmatrix}, \begin{bmatrix} c \\ -b \end{bmatrix}, [\,], [\,], C_1 \times \ldots \times C_{\bar{n}_p} \times \tilde{C} \right), \tag{3.73}$$

and we can conclude that z is in the set given by (3.68) as we wanted to show. □

Since we aim to eliminate only some of the generators we require a heuristic to select which generators to eliminate. Given Lemmas 3.1 and 3.3, one possible heuristic to estimate the size of the contribution of each generator to the final set is the following

Definition 3.4 Partition the matrix T from (3.55) as

$$T \coloneqq \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n_p} \end{bmatrix}, \tag{3.74}$$

where $T_i \in \mathbb{R}^{m_i \times (n_g - n_c)}, \forall i \in \{1, \cdots, n_p\}$. Then, the weight of each generator is defined as

$$w_i \coloneqq a_i \|G_i T_i\|_2, \forall i \in \{1, \cdots, n_p\}, \tag{3.75}$$

where a_i is defined in (3.40c).

The transformation T_i maps the new generators, constrained only by the two-norm, back to the original generator space, while G_i transforms from the space of the original generators to the space of the set. Consequently, w_i is an upper bound on the radius of a spheroid enclosing the generator set C_i, which serves as an approximation of the volume contribution attributed to individual generator sets. This approximation can be applied to any type of p-norm.

Given these methods, the order reduction algorithm to achieve a desired order r is given by Algorithm 3. We note that in practice, as will be seen in Section 3.9, it is often preferable to set $d_i = \frac{1}{n_p}$ than to solve (3.46) due to the increase in computational time that it involves. In this situation, the computational complexity of the algorithm is $O(n_c n_g^2 + n(n_g - rn)^2)$.

Algorithm 3 Order reduction algorithm $\tilde{Z} = \text{ord\_red}(Z, r)$.

Require: Z, r

1: Reorder the generators such that $w_{i+1} \leq w_i$.
2: Set $\bar{n}_p$ as the lowest number with $rn \leq \sum_{i=1}^{\bar{n}_p} m_i$.
3: Partition A and G as in (3.65a) and (3.65b)
4: Compute the overapproximation (3.67) using Lemma 3.1 either by setting $d_i = \frac{1}{n_p}$ or by solving (3.46), and Lemma 3.4.
5: Compute (3.68)

Remark 3.2 The adoption of the 2-norm is motivated by its compatibility with Lemma 3 and Definition 3, enabling the derivation of a heuristic for estimating the size of the reduced generator set. It is worth noting that our approach is flexible, as demonstrated by the potential adaptation of Lemma 1 to accommodate a variety of p-norms, thereby extending the applicability of the method to different norm settings.

3.8.2 Constraint Reduction

As in [56] to remove the constraint i and the generator j from the CCG we consider the following overapproximation, which adapts Proposition 5 in [56] to CCGs:

Lemma 3.6 *A CCG satisfies*

$$(G, c, A, b, C) \subset (G - \Lambda_G A, c + \Lambda_G b, A - \Lambda_A A, b - \Lambda_A b, C) \tag{3.76}$$

for every $\Lambda_G \in \mathbb{R}^{n \times n_c}$ *and* $\Lambda_A \in \mathbb{R}^{n_c \times n_c}$.

Proof. $z \in (G, c, A, b, C)$ if there exists $\xi \in C$ such that

$$\begin{bmatrix} z \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} G \\ A \end{bmatrix} \xi + \begin{bmatrix} c \\ -b \end{bmatrix}. \tag{3.77}$$

For any such ξ

$$\begin{bmatrix} z \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} G \\ A \end{bmatrix} \xi + \begin{bmatrix} c \\ -b \end{bmatrix} + \begin{bmatrix} \Lambda_G \\ \Lambda_A \end{bmatrix} (b - A\xi). \tag{3.78}$$

Therefore, $z \in (G - \Lambda_G A, c + \Lambda_G b, A - \Lambda_A A, b - \Lambda_A b, C)$. □

In this paper, instead of considering the Hausdorff distance to select which generator to remove as in [56], for each constraint i we consider removing the generator $j = \arg\max_k |a_{ik}|$. To remove one constraint we select the one which yields the smaller 2-norm of the overapproximation when removed. To remove the constraint i and the generator j we select, from [56],

$$\Lambda_G \coloneqq G E_{ji} a_{ij}^{-1}, \tag{3.79}$$

$$\Lambda_A \coloneqq A E_{ji} a_{ij}^{-1}, \tag{3.80}$$

where a_{ij} is the element of A in the ith row and jth column and $E_{ji} \in \mathbb{R}^{n_g \times n_c}$ is zero except for a one in the (j, i) position. Note that the ith row and jth column of $A - \Lambda_A A$, the jth row of $G - \Lambda_G A$ and the ith element of $b - \Lambda_A b$ are zero. Therefore, constraint i and generator element j may be removed from the set by removing these rows and columns and considering a generator set $\tilde{C}$ which is the projection of C in the hyperplane given by fixing the dimension j. That is, suppose that the generator j corresponds to the generator set l, that is,

$$\sum_{i=1}^{l-1} m_i < j \leq \sum_{i=1}^{l} m_i, \tag{3.81}$$

then, if $m_l = 1$, computing $\tilde{C}$ amounts to remove C_l from C, that is

$$\tilde{C} \coloneqq C_1 \times \ldots \times C_{l-1} \times C_{l+1} \times \ldots \times C_{n_p}. \tag{3.82}$$

If $m_l > 1$ then we must replace C_l by $\tilde{C}_l$ where

$$\tilde{C}_l \coloneqq \{\xi : \|\xi\|_{p_l} \leq 1\} \subset \mathbb{R}^{m_l - 1} \tag{3.83}$$

that is,

$$\tilde{C} \coloneqq C_1 \times \ldots \times \tilde{C}_l \times \ldots \times C_{n_p}. \tag{3.84}$$

In summary, the constraint reduction algorithm is given by Algorithm 4, which has a complexity of $O((n_c - \tilde{n}_c) n_c^2 n_g^2)$.

Algorithm 4 Constraint reduction algorithm $\tilde{Z} = \text{con\_red}(Z, \bar{n}_c)$.

Require: $Z, \bar{n}_c$
for $k \leftarrow 1$ to $n_c - \bar{n}_c$ **do**
  Set $Z^{prev} = Z \equiv (G, c, A, b, C)$
  Set $weight = \infty$
  for $i \leftarrow 1$ to $n_c - k + 1$ **do**
    Set $j = \arg\max_l |a_{il}|$
    Compute $Z^{prev} \subset (\tilde{G}, \tilde{c}, \tilde{A}, \tilde{b}, \tilde{C})$ as follows:
    Compute $\Lambda_G \coloneqq G E_{ji} a_{ij}^{-1}$.
    Compute $\Lambda_A \coloneqq A E_{ji} a_{ij}^{-1}$.
    $\tilde{G}$ is $G - \Lambda_G A$ removing column j.
    $\tilde{c} = c - \Lambda_G b$
    $\tilde{A}$ is $A - \Lambda_A A$ removing column j and row i.
    $\tilde{b}$ is $b - \Lambda_A b$ removing row i.
    $\tilde{C}$ is (3.82) or (3.84).
    Using Lemmas 3.1 and 3.3 compute:
    $(\tilde{G}, \tilde{c}, \tilde{A}, \tilde{b}, \tilde{C}) \subset (\bar{G}, \bar{c}, [\,], [\,], \|\xi\|_2 \leq 1)$
    Compute *new_weight* = $\|\bar{G}\|_2$
    if *new_weight* < *weight* **then**
      Set $Z = (\tilde{G}, \tilde{c}, \tilde{A}, \tilde{b}, \tilde{C})$
      Set *weight* = *new_weight*
    end if
  end for
end for

3.8.3 Guaranteed State Estimation with Order Reduction

In this section, we aim to leverage the order reduction procedure to have a bounded complexity in the guaranteed state estimation method. For simplicity, we will consider a Linear Time-Invariant (LTI) model given by :

$$x_{k+1} = Fx_k + Bu_k + w_k, \tag{3.85a}$$
$$y_k = Cx_k + v_k, \tag{3.85b}$$

where $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^{n_u}$, $w_k \in W \subset \mathbb{R}^n$, $y_k \in \mathbb{R}^{n_y}$, and $v_k \in V \subset \mathbb{R}^{n_y}$ represent the system state, input, disturbance signal, output, and noise, respectively. The recursive estimate for the state given $X_k \subset \mathbb{R}^n$ such that $x_k \in X_k$ and a measurement y_{k+1}, is the set $X_{k+1} \subset \mathbb{R}^n$ such that $x_{k+1} \in X_{k+1}$ that can be computed as

$$X_{k+1} = (FX_k \oplus W + Bu_k) \cap_C (y_{k+1} - V). \tag{3.86}$$

In this implementation, we assume that the sets W and V are represented as CCGs. In particular, we consider that

$$W \coloneqq (G_w, c_w, [\,], [\,], C_w)\,, \tag{3.87a}$$
$$V \coloneqq (G_v, c_v, [\,], [\,], C_v)\,, \tag{3.87b}$$

with $G_w \in \mathbb{R}^{n \times n_w}$ and $G_v \in \mathbb{R}^{n_y \times n_v}$.

In order to maintain the complexity of the set description manageable and to keep the computational burden low we adopt the following event-triggering order reduction mechanism, where $0 < \beta < 1$ must be small enough.

Algorithm 5 Set-valued CCG observer with event triggered order reduction

Require: X_0, V, W, T_c, β
Set $X_0^r = X_0$
for $k \geq 0$ **do**
  $X_{k+1} = (FX_k \oplus W + Bu_k) \cap_C (y_{k+1} - V)$
  $X_{k+1}^r = \left(FX_k^r \oplus W + Bu_k\right) \cap_C (y_{k+1} - V)$
  Try to compute under a time of T_c:
  $X_{k+1}^{or} = \text{con\_red}\,(X_{k+1}, \beta n_c)$.
  $X_{k+1}^{or} = \text{ord\_red}\left(X_{k+1}^{or}, \beta \frac{n_g}{n}\right)$.
  if Computation was successful **then**
    $X_{k+1}^r = X_{k+1}^{or}$
  else
    $X_{k+1} = X_{k+1}^r$
  end if
end for

The approach in Algorithm 5 guarantees that the computation time at each step does not exceed T_c but may be too conservative. A less conservative solution that does not guarantee a fixed upper bound on the computation time but ensures that the computation time is approximately T_d is given by Algorithm 6.

Algorithm 6 Set-valued CCG observer with event triggered order reduction

Require: X_0, V, W, T_d
Set $X_0^r = X_0$
for $k \geq 0$ **do**
  start the clock.
  $X_{k+1} = (FX_k \oplus W + Bu_k) \cap_C (y_{k+1} - V)$
  $X_{k+1}^{or} = \text{con\_red}\left(X_{k+1}, n_c - 2n_y\right)$.
  $X_{k+1}^{or} = \text{ord\_red}\left(X_{k+1}^{or}, \frac{n_g - 2(n_w + n_v) - (n_c - 2n_y)}{n}\right)$.
  if Elapsed time is greater than T_d **then**
    $X_{k+1} = X_{k+1}^{or}$
  end if
end for

3.9 Numerical Results for the Order Reduction Method

In order to assess the performance of the proposed methods, we start by considering the order reduction applied to random CZs in $\mathbb{R}^{10}$ on an Intel Core i7-12700H processor at 2.70 GHz. We consider that each element of G is drawn from a normal distribution centred at zero with a standard deviation of $\frac{1}{n_g}$, where n_g is the number of generators, each element of A is drawn from a normal distribution centred at zero with a standard deviation of one, and each element of b is drawn from a uniform distribution from -0.5 to 0.5. The order of a set representation is given by $\frac{n_g}{n}$. We consider an order reduction of the form

$$\tilde{Z} = \text{ord\_red}(Z, \frac{\tilde{n}_g}{n}). \tag{3.88}$$

Finally, we overbound the result with a CZ by changing the generator vector noting that

$$\{\xi : \|\xi\|_2 \leq 1\} \subset \{\xi : \|\xi\|_\infty \leq 1\} \tag{3.89}$$

and compare the results obtained with the order reduction method proposed in this paper with that of [56] for CZs and that of [74]. We represent a CZ as a CCG with $n_p = n_g$. We note that, in practice, the required time to solve (3.46) is much larger than the remaining of the algorithm and yields only marginal improvements. Therefore, we will consider $d_i = \frac{1}{n_p}$.

To have a more systematic assessment of the performance of the proposed method in Figures 3.12 and 3.13 we present the results of the difference of the obtained volume overapproximation of the enclosing hyperrectangles with both methods, $\frac{V}{V_{original}}$ where V is the volume of the enclosing hyperrectangle of the reduced set and $V_{original}$ is the volume of the enclosing hyperrectangle of the original set, and the computational time, respectively for different randomly generated CZs with different number of generators n_g while setting the number of constraints to $n_c = 0.4n_g$.

We can observe from Figure 3.12 that in this case, for low order proportions, [56] and [74] fare worse than the method of Algorithm 3 in terms of volume of the resulting set, while for higher order proportions, the volume of the resulting set with Algorithm 3 is higher, which may be partly due to the near-ellipsoidal shape of the described set. This trend shows that Algorithm 3 is competitive in situations that require sets expressed with a small order. From Figure 3.13 we observe that in terms of computational time, there is no clear advantage to either method.

To assess the performance of the constraint reduction method, we consider randomly generated CZs as before. We consider a constraint reduction of the form

$$\tilde{Z} = \text{con\_red}(Z, \tilde{n}_c). \tag{3.90}$$

Again, for a systematic assessment of the performance of the proposed constraint reduction method in Figures 3.14 and 3.15 we present the results of the difference of

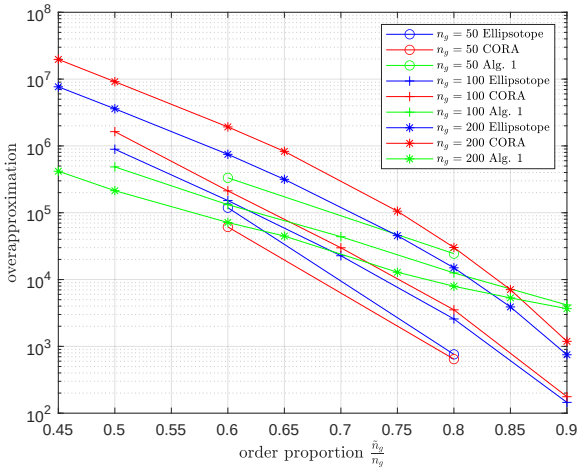


Fig. 3.12: Average of the obtained volume overapproximation $\frac{V}{V_{original}}$ with the proposed order reduction method (Alg. 1), the order reduction method in [56] (CORA), and the order reduction method in [74] for ellipsotopes (Ellipsotope), for different randomly generated CZs with different numbers of generators n_g. This would correspond to a uniform expansion on each side of $\left(V/V_{original}\right)^{\frac{1}{10}}$ times.

the obtained volume overapproximation and the computational times, respectively for different randomly generated CZs with different number of generators n_g, while setting the number of constraints to $n_c = 0.4n_g$.

We can observe from Figures 3.14 and 3.15 that in this case there is no clear advantage of either method regarding the volume of the overapproximation. Still, there are significant savings regarding computational time for Algorithm 4.

Regarding guaranteed state estimation, we test Algorithm 5 on system (3.85) with no input and $A \in \mathbb{R}^{3\times3}$ and $C \in \mathbb{R}^{3\times3}$ are random orthonormal matrices. The initial state, the process disturbance and the measurement noise satisfy $\|x_0\|_\infty \leq 100$, $\|w_k\|_\infty \leq 1$ and $\|v_k\|_\infty \leq 1$ for all k.

Figure 3.16 shows the mean over time of the volume of the state estimate for the method for CCGs proposed here with event-triggered order reduction (Algorithm 5), with $T_c = 0.5$ and $\beta = 0.6$ and the same algorithm using a CZ approximation with the order reduction method in [56] for CZs, with $\beta = 0.1$. Figure 3.17 shows the time to compute the description of the set at runtime. We can observe that with the methods proposed here, we obtain better overapproximations because the number of generators used to describe the set is larger.

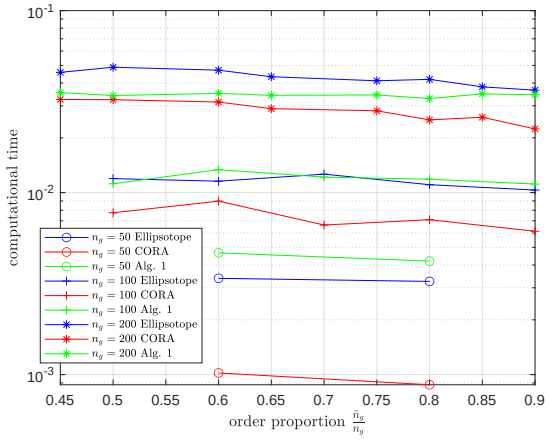


Fig. 3.13: Average of the computational times with the proposed order reduction method (Alg. 1), the order reduction method in [56] (CORA), and the order reduction method in [74] for ellipsotopes (Ellipsotope), for different randomly generated CZs with different numbers of generators n_g.

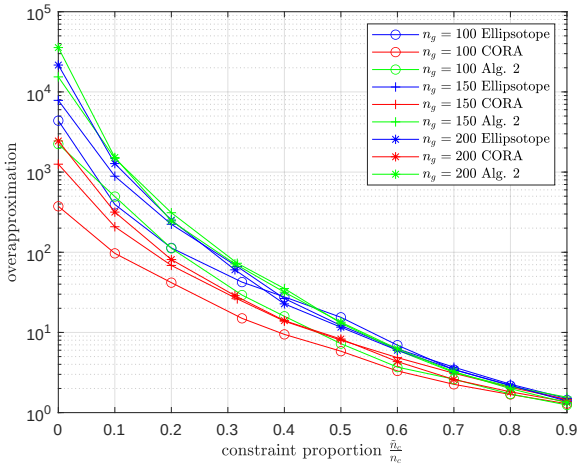


Fig. 3.14: Average of the obtained volume overapproximation $\frac{V}{V_{original}}$ with the proposed constraint reduction method (Alg. 2), the constraint reduction method in [56] (CORA), and the constraint reduction method in [74] for ellipsotopes (Ellipsotope), for different randomly generated CZs with different numbers of generators n_g.

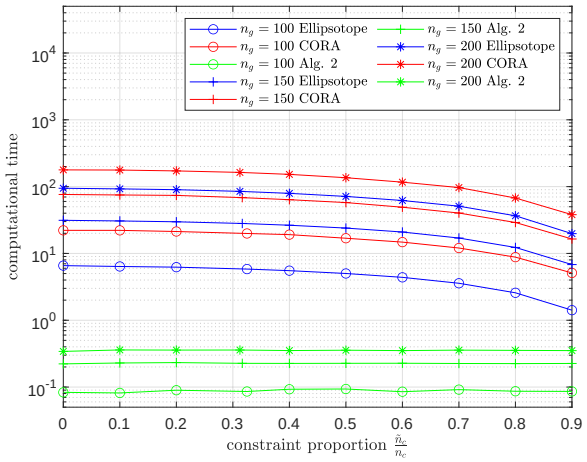


Fig. 3.15: Average of the computational times with the proposed constraint reduction method (Alg.2) and the constraint reduction method in [56] (CORA), and the constraint reduction method in [74] for ellipsotopes (Ellipsotope), for different randomly generated CZs with different numbers of generators n_g.

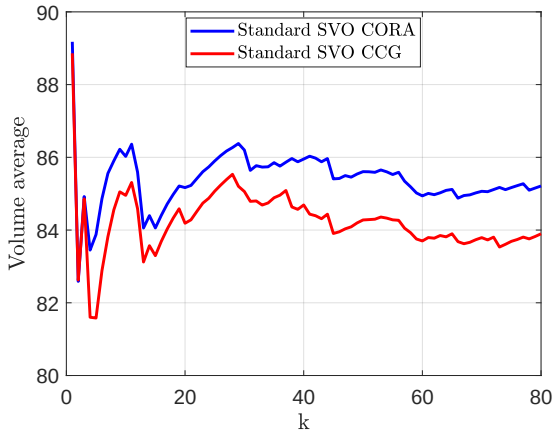


Fig. 3.16: Volume mean $\sum_{j=1}^{k} \frac{Vol_j}{k}$, where Vol_k is the volume at time k, of the state estimate at various iterations for the two set-valued observers (SVO).

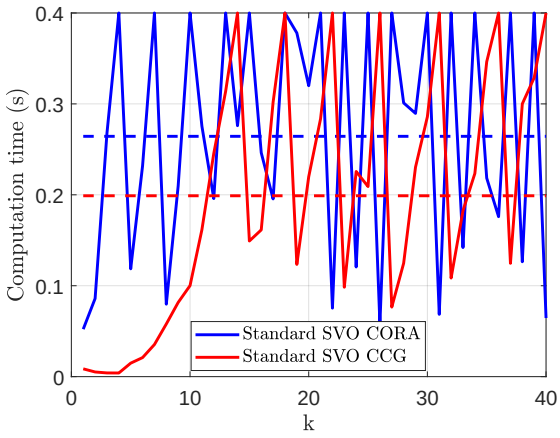


Fig. 3.17: Computation times for the two set-valued observers (SVO). The dashed horizontal lines correspond to the mean value for each method.

3.10 Explicit computation of guaranteed state estimates

In the previous section, we have introduced the idea of performing order reduction at every time instant to maintain a bounded complexity on the CCG description. However, in cases where the dynamics are stable, such a procedure is disregarding the inherent properties of the system and blindly removing generators and constraints. In this section, we propose an alternative for such cases by considering that the entire representation of a set in the past can be replaced by a conservative ellipsoidal estimate provided by some other filter as long as it is given by a closed-form expression (i.e., no iterative procedure). Other options could be found in the literature like in [75]–[77].

3.10.1 Ellipsoidal observer

Before obtaining a state estimate with low conservatism, we start with a coarse ellipsoidal state estimate based on a Luenberger observer, which for systems like in (3.85), is given by

$$\hat{x}_{k+1} = F\hat{x}_k + Bu_k + L\left(y_k - C\hat{x}_k\right), \tag{3.91}$$

where L is defined such that $\rho(F - LC) < 1$, where $\rho(\cdot)$ is the spectral radius. If the pair (F, C) is detectable, such matrix L always exists. Defining the estimation error as $e_k := x_k - \hat{x}_k$ from (3.85) we obtain

$$e_{k+1} = (F - LC)e_k + w_k - Gv_k. \tag{3.92}$$

Given that $\rho(F - LC) < 1$ there exists a symmetric matrix $P \in \mathbb{R}^{n\times n}$ such that

$$(F - LC)^\intercal P(F - LC) - P = -I_n, \tag{3.93}$$

and we may define a decrease rate as follows

$$a := \|P^{\frac{1}{2}}(F - LC)P^{-\frac{1}{2}}\|_2 = \sqrt{1 - \frac{1}{\sigma_{max}(P)}}, \tag{3.94}$$

where $\sigma_{max}(\cdot)$ is the maximum singular value. From (3.93), we have that $a < 1$. Therefore, defining

$$e_{init} := \max_{\xi \in X_0} \|P^{\frac{1}{2}}\xi\|_2 \tag{3.95}$$

$$e_{noise} := \max_{\xi \in W \oplus -LV} \|P^{\frac{1}{2}}\xi\|_2 \tag{3.96}$$

and applying Theorem 6 in [78], we have that $x_k \in \hat{X}_k$ for all $k \geq 0$, where

$$\hat{X}_k = \left\{\hat{x}_k + \xi : \|P^{\frac{1}{2}}\xi\|_2 \leq a^k e_{init} + \frac{e_{noise}}{1 - a}\right\}. \tag{3.97}$$

Given that the state estimate is an ellipsoid, it can be written in CCG format as follows

$$\hat{X}_k = \left(\left(a^k e_{init} + \frac{e_{noise}}{1 - a}\right) P^{-\frac{1}{2}}, \hat{x}_k, [\,], [\,], \|\xi\|_2 \leq 1\right). \tag{3.98}$$

We remark to the reader that the fact that CCGs allow for set operations between polytopes and ellipsoids, it is possible to use the conservative ellipsoidal estimate $\hat{X}_k$ and improve it by explicitly considering the exact iterations for some fixed number of time instants. Therefore, at some time k, the set $\hat{X}_{k-N}$ can be viewed as an implicit order reduction to the more accurate set X_{k-N} that would be obtained by the direct recursion in (3.86). These two fact will be useful in the next section to provide a set-valued observer that does not require order reduction methods and where most of the computations can be performed offline before the estimation procedure is run.

3.10.2 Explicit finite-horizon observer

The state estimate given by (3.98) can serve as a conservative set that can be improved by N iterations of the recursion (3.86). Specifically, defining for an integer l,

$$Y_k^l := \begin{bmatrix} y_{k-1} \\ \vdots \\ y_{k-l} \end{bmatrix}, \tag{3.99}$$

$$U_k^l := \begin{bmatrix} u_{k-1} \\ \vdots \\ u_{k-l} \end{bmatrix}, \tag{3.100}$$

we consider that the state estimate at time k is expressed by

$$X_k^l = \left(G_{X,k}^l, c_{X,k}^l, A_{X,k}^l, b_{X,k}^l, C_X^l \right) \subset \mathbb{R}^n, \tag{3.101}$$

where

$$G_{X,k}^l = \left[G^l \ \left(a^{k-l} e_{init} + \tfrac{e_{noise}}{1-a} \right) G_0^l \right], \tag{3.102a}$$

$$c_{X,k}^l = c^l + c_U^l U_k^l + c_0^l \hat{x}_{k-l}, \tag{3.102b}$$

$$A_{X,k}^l = \left[A^l \ \left(a^{k-l} e_{init} + \tfrac{e_{noise}}{1-a} \right) A_0^l \right], \tag{3.102c}$$

$$b_{X,k}^l = Y_k^l + b^l + b_U^l U_k^l + b_0^l \hat{x}_{k-l}. \tag{3.102d}$$

For $l = 0$ one recovers the ellipsoidal observer considering that G^0, c_U^0, A^0,A_0^0, b^0, b_U^0 and b_0^0 are empty matrices,

$$C_X^0 = \{ \xi : \|\xi\|_2 \leq 1 \}, \tag{3.103}$$

and

$$G_0^0 := P^{-\frac{1}{2}}, \tag{3.104a}$$

$$c^0 := \mathbf{0}_n, \tag{3.104b}$$

$$c_0^0 := I_n. \tag{3.104c}$$

By applying (3.86) we obtain after simple computations the main result of this paper.

Theorem 3.2 *Given a state estimate X_k^l such that $x_k \in X_k^l$, then $x_{k+1} \in X_{k+1}^{l+1}$ with*

$$C_X^{l+1} = C_w \times C_v \times C_X^l, \tag{3.105a}$$

$$G^{l+1} = \begin{bmatrix} G_w & \mathbf{0} & FG^l \end{bmatrix}, \tag{3.105b}$$

$$G_0^{l+1} = FG_0^l, \tag{3.105c}$$

$$c^{l+1} = Fc^l + c_w, \tag{3.105d}$$

$$c_U^{l+1} = \begin{bmatrix} B & Fc_U^l \end{bmatrix}, \tag{3.105e}$$

$$c_0^{l+1} = Fc_0^l, \tag{3.105f}$$

$$A^{l+1} = \begin{bmatrix} \mathbf{0} & G_v & CG^l \\ \mathbf{0} & \mathbf{0} & A^l \end{bmatrix}, \tag{3.105g}$$

$$A_0^{l+1} = \begin{bmatrix} CG_0^l \\ A_0^l \end{bmatrix}, \tag{3.105h}$$

$$b^{l+1} = \begin{bmatrix} -Cc^l - c_v \\ b^l \end{bmatrix}, \tag{3.105i}$$

$$b_U^{l+1} = \begin{bmatrix} \mathbf{0} & Cc_U^l \\ \mathbf{0} & b_U^l \end{bmatrix}, \tag{3.105j}$$

$$b_0^{l+1} = \begin{bmatrix} Cc_0^l \\ b_0^l \end{bmatrix}. \tag{3.105k}$$

Proof. We first consider that at time k a state estimate is given by (3.101) and (3.102). The Theorem follows by applying (3.86) with the CCG operations, where W and V are given by (3.87). □

Based on Theorem 3.2, the observer proposed in this paper consists of selecting a fixed horizon N and pre-computing the set C_X^N and matrices G^N, G_0^N, c^N, c_U^N, c_0^N, A^N, A_0^N, b^N, b_U^N, and b_0^N, offline with Algorithm 7.

After obtaining matrices G^N, G_0^N, c^N, c_U^N, c_0^N, A^N, A_0^N, b^N, b_U^N, and b_0^N with Algorithm 7, at runtime, for $k \geq N$, the observer consists of the Algorithm 8.

With Algorithm 5, to obtain the description of a guaranteed state estimate set we only have to perform a small number of computations proportional to the horizon length N, which may be significantly more efficient than performing (3.86) recursively. Given the finite-horizon nature of the algorithm, this approach is more

Algorithm 7 Pre-computation of CCG parameters

Require: $G^0, c_U^0, A^0, A_0^0, b^0, b_U^0, b_0^0 = [\,]$; C_X^0 is given by (3.103); G_0^0, c^0, c_0^0 are given by (3.104)

1: **for** $l \leftarrow 0$ to $N-1$ **do**
2: compute $C_X^{l+1}, G^{l+1}, G_0^{l+1}, c^{l+1}, c_U^{l+1}, c_0^{l+1}, A^{l+1}, A_0^{l+1}, b^{l+1}, b_U^{l+1}, b_0^{l+1}$ with (3.105)
3: **end for**
4: **return** $C_X^N, G^N, G_0^N, c^N, c_U^N, c_0^N, A^N, A_0^N, b^N, b_U^N, b_0^N$

Algorithm 8 Explicit finite-horizon observer

Require: $C_X^N, G^N, G_0^N, c^N, c_U^N, c_0^N, A^N, A_0^N, b^N, b_U^N, b_0^N, L, a, e_{init}\ e_{noise}, \hat{x}_0$

1: **for** $0 \le k < N$ **do**
2: $X_k = \left(\left(a^k e_{init} + \frac{e_{noise}}{1-a}\right) P^{-\frac{1}{2}}, \hat{x}_k, [\,], [\,], \|\xi\|_2 \le 1 \right)$
3: **end for**
4: **for** $k \ge N$ **do**
5: $\hat{x}_{k-N+1} = F\hat{x}_{k-N} + Bu_{k-N} + L\,(y_{k-N} - C\hat{x}_{k-N})$,
6: compute Y_{k+1}^N by storing y_k and discarding y_{k-N}
7: compute U_{k+1}^N by storing u_k and discarding u_{k-N}
8: $G_{X,k+1}^N = \left[G^N \ \left(a^{k+1-N} e_{init} + \frac{e_{noise}}{1-a}\right) G_0^N \right]$
9: $c_{X,k+1}^N = c^N + c_U^N U_{k+1}^N + c_0^N \hat{x}_{k+1-N}$
10: $A_{X,k+1}^N = \left[A^N \ \left(a^{k+1-N} e_{init} + \frac{e_{noise}}{1-a}\right) A_0^N \right]$
11: $b_{X,k+1}^N = Y_{k+1}^N + b^N + b_U^N U_{k+1}^N + b_0^N \hat{x}_{k+1-N}$
12: $X_{k+1} = \left(G_{X,k+1}^N, c_{X,k+1}^N, A_{X,k+1}^N, b_{X,k+1}^N, C_X^N \right)$
13: **end for**

conservative than applying (3.86) recursively. However, by increasing the horizon N the introduced conservatism tends to disappear.

We have to remark that to apply this method for $k < N$ would imply storing in memory all the coefficients from $l = 1$ to $l = N-1$. However, it greatly increases the memory requirements for large N and it would only have an effect in a small transient period. For that reason, we consider that for $k < N$ the state estimate is obtained with (3.98).

With the description of set X_k^N, an important operation is obtaining an estimate of the centre of the set. This can be done with an optimization algorithm by estimating the centre x_k^{center} as

$$x_k^{center} = c_{X,k}^N + G_{X,k}^N \operatorname{argmin}_{A_{X,k}^N \xi = b_{X,k}^N} \|\xi\|_2 \tag{3.106}$$

Alternatively, this can be computed algebraically as follows

$$x_k^{center} = c_{X,k}^N + G_{X,k}^N A_{X,k}^{N,\intercal} \eta_k, \tag{3.107}$$

where η_k is computed by solving the linear equation

$$A_{X,k}^{N} A_{X,k}^{N,\intercal} \eta_k = b_{X,k}^{N}. \tag{3.108}$$

Given that a^k tends to zero, one may neglect the term $a^{k-N} e_{init}$ after some time. Therefore, we may consider that

$$G_{X,k}^{N} \approx G_X^N := \left[G^N \ \tfrac{e_{noise}}{1-a} G_0^N \right], \tag{3.109a}$$

$$A_{X,k}^{N} \approx A_X^N := \left[A^N \ \tfrac{e_{noise}}{1-a} A_0^N \right], \tag{3.109b}$$

and we can pre-compute the matrix

$$Z_X^N := G_X^N A_X^{N,\intercal} \left(A_X^N A_X^{N,\intercal} \right)^{-1}, \tag{3.110}$$

obtaining significant computational time savings in the computation of the CCG center as

$$x_k^{center} = c_{X,k}^{N} + Z_X^N b_{X,k}^{N}. \tag{3.111}$$

3.11 Numerical Results for the Explicit Guaranteed State Observer

To assess the performance of the proposed algorithm we consider a random system generated with the MATLAB function *drss* with dimension 15, an output of size 3 and input of size 5, that is, $x_k \in \mathbb{R}^{15}$, $u_k \in \mathbb{R}^5$ and $y_k \in \mathbb{R}^3$, for all $k \geq 0$. We consider that the initial state is drawn from an initial state which is a CCG given by

$$X_0 = \left(G_{X,0}, c_{X,0}, [\ ], [\ ], C_{X,0} \right) \subset \mathbb{R}^n, \tag{3.112}$$

where

$$C_{X,0} = \{ \xi : \|\xi\|_\infty \leq 1 \} \times \{ \xi : \|\xi\|_2 \leq 1 \}, \tag{3.113a}$$

$$G_{X,0} = \left[2I_{15} \ I_{15} \right], \tag{3.113b}$$

$$c_{X,0} = \mathbf{0}_{15}. \tag{3.113c}$$

The disturbance and noise sets are expressed as (3.87) with parameters

$$C_W = \{\xi : \|\xi\|_\infty \leq 1\} \times \{\xi : \|\xi\|_2 \leq 1\}, \tag{3.114a}$$

$$G_W = \begin{bmatrix} 2I_{15} & I_{15} \end{bmatrix}, \tag{3.114b}$$

$$c_W = \mathbf{0}_{15}, \tag{3.114c}$$

$$C_V = \{\xi : \|\xi\|_\infty \leq 1\} \times \{\xi : \|\xi\|_2 \leq 1\}, \tag{3.114d}$$

$$G_V = \begin{bmatrix} I_3 & 2I_3 \end{bmatrix}, \tag{3.114e}$$

$$c_V = \mathbf{0}_3. \tag{3.114f}$$

The control input is constant and given by $u_k = 20\mathbf{1}_5$ for all $k \geq 0$.

Figure 3.18 shows the evolution in time of the projection of the first coordinate of the state estimate obtained with the ellipsoidal method of (3.98) (Ellipsoidal), the standard description obtained by applying recursively (3.86) (Standard), and the method proposed in this paper for various horizons N. From Figure 3.18, we observe that the performance of the algorithm approaches that of the standard case for large N.

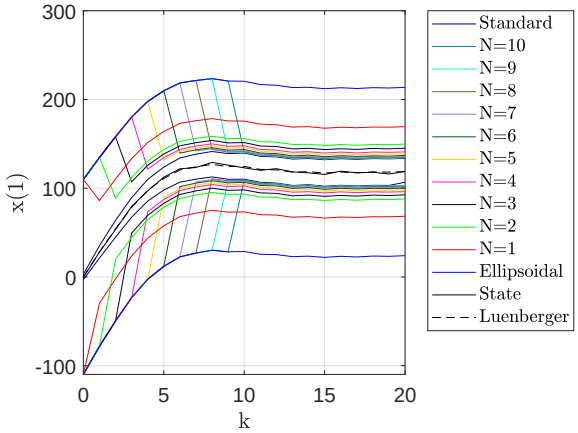


Fig. 3.18: Time plot of state enclosures for diverse state estimation methods for the first coordinate of the state. The black line in the middle represents the system's actual state, while the dashed line indicates the Luenberger state estimates $\hat{x}_k$.

In Figure 3.19, we plot the projection in the first two dimensions of the state estimate obtained with various methods and for different horizons N. As in Figure 3.18, we observe that the performance of the algorithm approaches that of the standard observer for large N. This fact can also be observed in Figure 3.20 which shows the size of the projection in the first dimension of the state estimate.

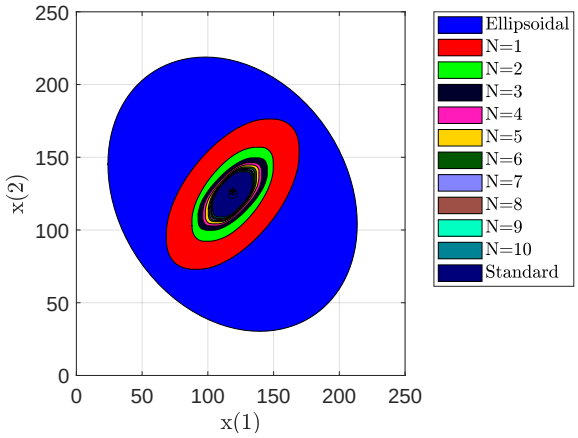


Fig. 3.19: State enclosures for various state estimation techniques when k is set to 20. The asterisk ($*$) represents the Luenberger state estimate $\hat{x}_k$ and the circle ($\circ$) represents the true state of the system.

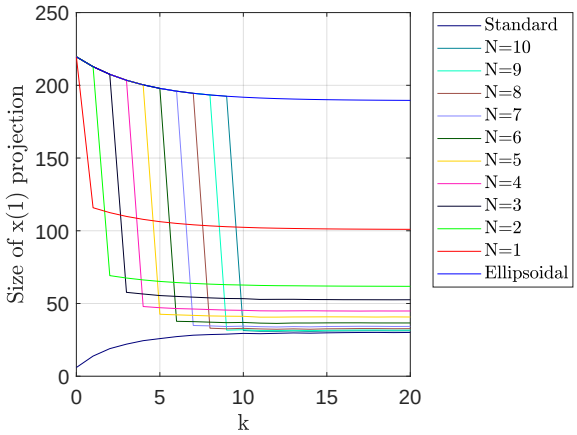


Fig. 3.20: Size of the projection of the first dimension of the state at various iterations for different state estimation methods and different horizons N.

Figure 3.21 shows the time to compute the description of the set at runtime with an Intel Core i7-12700H processor at 2.70 GHz. From Figure 3.21, we can observe that the computation times are significantly more competitive with the method proposed in this paper since most of the matrix computations are done offline.

The most significant advantage of the method proposed in this paper is the fact that the set description size remains constant. Therefore, as shown in Figure 3.22 while with the standard method, the computation time increases at every iteration, with the method of Algorithm 5 the computation time remains constant.

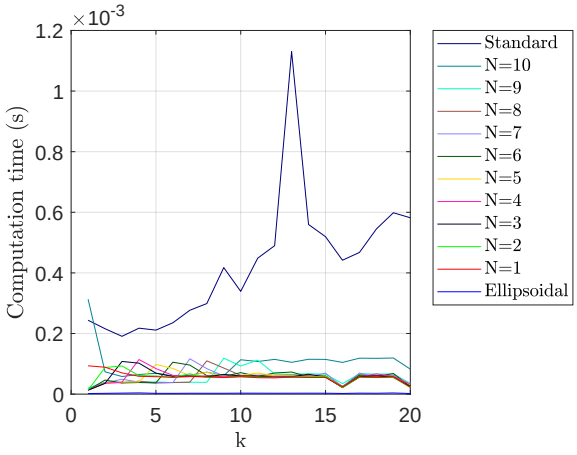


Fig. 3.21: Computation times for various state estimation methods.

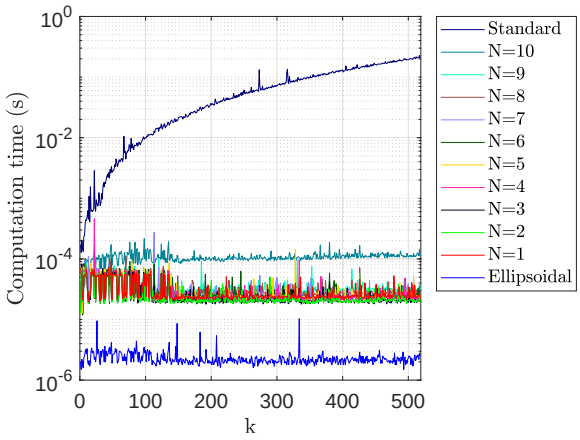


Fig. 3.22: Computation times until $k = 500$.

Figure 3.23 shows the computation times for various centre computation methods for the standard observer and horizons $N = 1$ and $N = 10$. We tested the method of centre computation of solving (3.106) with YALMIP and the MOSEK solver [79] (Opt), the algebraic method of (3.107) (Alg), and the method with pre-computed matrices of (3.111) (Pre). For improved efficiency, for the optimization approach, we adopted the simplification (3.109) and used the function *optimizer* to pre-compile the optimization algorithm. From Figure 3.23 we observe that it is significantly more advantageous to compute the relevant matrices beforehand, instead of solving a linear equation at every time.

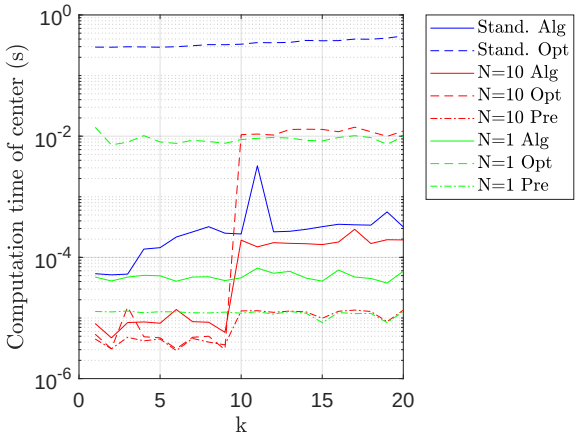


Fig. 3.23: Center computation times for various state estimation methods and various centre computation methods.

To highlight the main advantage of the proposed method, we used the same simulation for a larger number of iterations with the results being depicted in Figure 3.24. Since the description of the state estimate increases in size at each iteration, the computation of the center becomes more time-consuming, whereas, the proposed method benefits from the constant description and pre-computation of parts of the data structures being done offline. We remark that the presented method voids the need for an order reduction procedure, which is going to add conservatism and represent a time overhead.

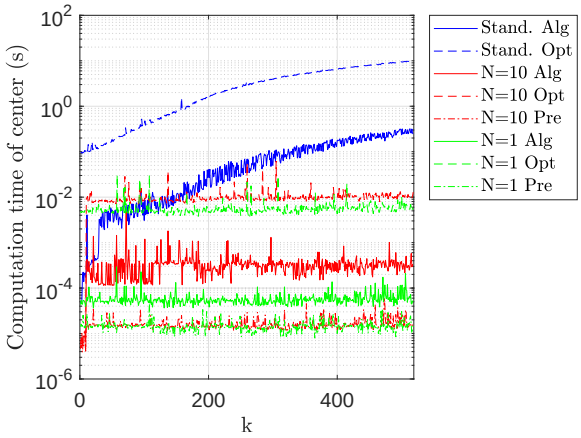


Fig. 3.24: Center computation times until $k = 500$.

3.12 Stochastic Estimation for Linear Dynamical Systems

In the previous sections, the problem was always viewed from a deterministic point-of-view, which is also commonly referred to as guaranteed state estimation or worst-case analysis. Those names translate the option/assumption that the PDF for each of the exogenous signals is not known and the methods work solely with the support of those functions. In the remainder of this chapter, we also explore stochastic state estimation in linear systems. The primary objective is to estimate the system state based on noisy observations, considering both Gaussian and non-Gaussian assumptions. We start by presenting some of the typical techniques like the Kalman Filter (KF), Particle Filter (PF), and the CF Filter in order to better place the presented method.

3.12.1 Bayes Filter

In the stochastic domain, the equivalent to the recursive estimation in the deterministic setting is given by the Bayes Filter. In a similar fashion to the guaranteed state estimation, it must first use the information of the model to *predict* the next state and then *update* those estimates using the measurements.

- **Prediction Step:** This step propagates the prior belief about the system's state forward in time, integrating the system's dynamics and the associated uncertainties. Mathematically, it is represented as:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}) \cdot p(x_{k-1}|z_{1:k-1}) dx_{k-1} \tag{3.115}$$

  Here, x_k denotes the state at time k, $z_{1:k-1}$ represents the sequence of observations up to time $k-1$, $p(x_k|x_{k-1})$ is the transition model that describes the system's dynamics, and $p(x_{k-1}|z_{1:k-1})$ is the prior state estimate at time $k-1$.

- **Update Step:** This step refines the prediction by incorporating the latest observation at time k. It adjusts the state probability based on the likelihood of the new observation given the predicted state:

$$p(x_k|z_{1:t}) = \frac{p(z_k|x_k) \cdot p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{3.116}$$

  In this expression, z_k is the new observation at time k, $p(z_k|x_k)$ represents the likelihood of the observation given the current state, and $p(z_k|z_{1:k-1})$ serves as the normalization factor, ensuring that the posterior distribution remains a valid probability distribution.

Under the Markov assumption

$$p(x_k|x_{k-1}, x_{t-2}, \ldots, x_0) = p(x_k|x_{k-1}) \tag{3.117}$$

the filter can simply use the previous measurement and estimate and recursively iterate the estimates over time. The next filters follow the same idea but posing additional assumptions that allows some of the expressions to be computed in a different form.

3.12.2 Kalman Filter

The KF [80] provides an optimal solution under the assumptions of system linearity and Gaussian noise and disturbances. As random variables with Gaussian distributions are closed for both additions and the update step, we can propagate the mean and covariance that completely define those variables:as it describes the evolution of the mean and covariance matrix associated with the state estimate

- **Prediction Step:** Let the estimate at time $k-1$ be described by the mean $\hat{x}_{k-1|k-1} \in \mathbb{R}^n$ and covariance matrix $P_{k|k-1} \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$ be the disturbance covariance matrix, the prediction equations are

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{3.118}$$
$$P_{k|k-1} = AP_{k-1|k-1}A^\top + Q. \tag{3.119}$$

- **Update Step:** With $K_k \in \mathbb{R}^{n\times p}$ as the Kalman gain and $z_k \in \mathbb{R}^p$ as the measurement at time k, $H \in \mathbb{R}^{p\times n}$ as the observation model matrix and $R \in \mathbb{R}^{p\times p}$ denoting the noise covariance, the update equation of the Bayes filter becomes

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}) \tag{3.120}$$
$$K_k = P_{k|k-1}H^\top(HP_{k|k-1}H^\top + R)^{-1} \tag{3.121}$$
$$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{3.122}$$

3.12.3 Particle Filter

The PF [81] poses the option to compute the PDF of the state as an approximation using the histogram of a collection of samples often called *particles*. As a consequence, the filter requires an extra step of initialization to sample all involved distributions

but the operations are rather trivial as we are using vector-valued samples and not random variables.

- **Initialization:** Initialize N particles $\{x_0^{(i)}\}$ from the prior distribution $p(x_0)$. Each particle $x_0^{(i)}$ is assigned an initial weight $w_0^{(i)}$, typically set to $\frac{1}{N}$:

$$x_0^{(i)} \sim p(x_0), \quad i = 1, \ldots, N \tag{3.123}$$

- **Prediction:** Predict the next state for each particle using the process model:

$$x_k^{(i)} \sim p(x_k | x_{k-1}^{(i)}, u_k) \tag{3.124}$$

  This step advances each particle according to the system's dynamics, incorporating control inputs and process noise.

- **Weight Update and Normalization:** Upon receiving a new measurement z_k, the weight $w_k^{(i)}$ of each particle is updated based on the likelihood $p(z_k | x_k^{(i)})$. Weights are subsequently normalized such that their sum equals one:

$$w_{k,\text{unnorm}}^{(i)} = w_{k-1}^{(i)} p(z_k | x_k^{(i)}) \tag{3.125}$$

$$w_k^{(i)} = \frac{w_{k,\text{unnorm}}^{(i)}}{\sum_{j=1}^{N} w_{k,\text{unnorm}}^{(j)}} \tag{3.126}$$

  This step evaluates how well each particle explains the observed data and adjusts the weights accordingly.

- **Resampling:** Resample N particles $\{x_k^{(i)'}\}$ based on the normalized weights $\{w_k^{(i)}\}$:

$$x_k^{(i)'} \sim \sum_{j=1}^{N} w_k^{(j)} \delta(x_k^{(i)'} - x_k^{(j)}) \tag{3.127}$$

  where $\delta(\cdot)$ is the Dirac delta function.

  Resampling addresses particle degeneracy by favoring particles with higher weights, thereby ensuring an adequate representation of the posterior distribution.

3.12.4 Gaussian Mixture Filter

The Gaussian Mixture Filter (GMF) [82] extends Bayesian estimation principles to state-space systems operating under non-Gaussian noise conditions by utilizing

Gaussian Mixture Models (GMMs). This class of PDFs is attractive because it can approximate arbitrarily well other distributions. A fundamental aspect of the GMF is the need to reduce the number of terms (much like an order reduction) using a technique based on the Kullback-Leibler divergence metric [83].

Considering the Bayesian filtering framework and state-space systems represented by a GMM, the prior distribution can be described as follows:

$$p\left(x_0\right) = \sum_{i=1}^{N_p} \alpha_i \mathcal{N}\left(x_0; \mu_0^i, P_0^i\right), \quad \sum_{i=1}^{N_p} \alpha_i = 1. \tag{3.128}$$

The process model is expressed in the form:

$$p\left(x_k \mid x_{k-1}\right) = \sum_{j=1}^{N_x} \beta_k^j \mathcal{N}\left(x_k; Ax_{k-1} + u_k^j, Q_k^j\right), \quad \sum_{j=1}^{N_x} \beta_k^j = 1. \tag{3.129}$$

The measurement model is similarly represented as:

$$p\left(z_k \mid x_k\right) = \sum_{\tau=1}^{N_z} \gamma_k^\tau \mathcal{N}\left(z_k; Cx_k + v_k^\tau, R_k^\tau\right), \quad \sum_{\tau=1}^{N_z} \gamma_k^\tau = 1. \tag{3.130}$$

In the above equations, $\mathcal{N}(x; \mu, P)$ denotes a standard multivariate Gaussian distribution. The mean offset terms u_k^j and v_k^γ facilitate the inclusion of input signals, such as control inputs or noise.

Assuming we have a predicted mixture available, it can be expressed as:

$$p\left(x_k \mid z_{1:k-1}\right) = \sum_{\ell=1}^{N_{k|k-1}} w_{k|k-1}^\ell \mathcal{N}\left(x_k; \widehat{x}_{k|k-1}^\ell, P_{k|k-1}^\ell\right), \sum_{\ell=1}^{N_{k|k-1}} w_{k|k-1}^\ell = 1. \tag{3.131}$$

Considering the measurement update step given in (3.116), this can be extended into the current GMM context as follows:

$$p\left(x_k \mid z_{1:k}\right) = \sum_{\ell=1}^{N_{k|k-1}} \sum_{\tau=1}^{N_z} w_{k|k-1}^\ell \gamma_k^\tau \frac{\mathcal{N}\left(z_k; Cx_k + v_k^\tau, R_k^\tau\right)}{p\left(z_k \mid z_{1:k-1}\right)} \cdot \mathcal{N}\left(x_k; \widehat{x}_{k|k-1}^\ell, P_{k|k-1}^\ell\right). \tag{3.132}$$

Due to the linear Gaussian structure, this expression can be reformulated as:

$$p\left(x_k \mid z_{1:k}\right) = \sum_{s=1}^{N_{k|k}} w_{k|k}^s \mathcal{N}\left(x_k; \widehat{x}_{k|k}^s, P_{k|k}^s\right), \quad \sum_{s=1}^{N_{k|k}} w_{k|k}^s = 1, \tag{3.133}$$

where for each $\tau = 1, \ldots, N_z$ and $\ell = 1, \ldots, N_{k|k-1}$, it holds that:

$$N_{k|k} = N_{k|k-1} N_z \tag{3.134}$$
$$s \triangleq N_z(\ell - 1) + \tau \tag{3.135}$$
$$\widehat{x}^s_{k|k} = \widehat{x}^\ell_{k|k-1} + K^s_k e^s_k \tag{3.136}$$
$$e^s_k = z_k - C\widehat{x}^\ell_{k|k-1} - v^\tau_k \tag{3.137}$$
$$\Sigma^s_k = CP^\ell_{k|k-1} (C)^\top + R^\tau_k \tag{3.138}$$
$$K^s_k = P^\ell_{k|k-1} C \left(\Sigma^s_k\right)^{-1} \tag{3.139}$$
$$P^s_{k|k} = P^\ell_{k|k-1} - K^s_k \Sigma^s_k \left(K^s_k\right)^\top \tag{3.140}$$
$$w^s_{k|k} = \frac{\bar{w}^s_{k|k}}{\sum_{s=1}^{N_{k|k}} \bar{w}^s_{k|k}}, \tag{3.141}$$
$$\bar{w}^s_{k|k} = w^\ell_{k|k-1} \gamma^\tau_k \tag{3.142}$$

The prediction step described in (3.115) becomes:

$$p\left(x_k \mid z_{1:k}\right) = \sum_{s=1}^{N_{k|k}} \sum_{j=1}^{N_x} w^s_{k|k} \beta^j_t \int \mathcal{N}\left(x_k; Ax_k + u^j_k, Q^j_k\right) \mathcal{N}\left(x_k; \widehat{x}^s_{k|k}, P^s_{k|k}\right) dx_k \tag{3.143}$$

again, due to the linear Gaussian densities involved, we can express this predicted mixture via

$$p\left(x_k \mid z_{1:k}\right) = \sum_{\ell=1}^{N_{k+1|k}} w^\ell_{k+1|k} \mathcal{N}\left(x_k; \widehat{x}^\ell_{k+1|k}, P^\ell_{k+1|k}\right), \quad \sum_{\ell=1}^{N_{k+1|k}} w^\ell_{k+1|k} = 1, \tag{3.144}$$

where for each $s = 1, \ldots, N_{k|k}$ and $j = 1, \ldots, N_x$, we have:

$$N_{k+1|k} = N_{k|k} N_x, \tag{3.145}$$
$$\ell \triangleq N_x(s - 1) + j, \tag{3.146}$$
$$\widehat{x}^\ell_{k+1|k} = A\widehat{x}^s_{k|k} + u^j_k, \tag{3.147}$$
$$P^\ell_{k+1|k} = AP^s_{k|k} A^\top + Q^j_k, \tag{3.148}$$
$$w^\ell_{k+1|k} = w^s_{k|k} \beta^j_k. \tag{3.149}$$

Given the growth in the number of gaussian components $N_{k|k}$, it is necessary to merge elements of the GMM to return to the form

$$\pi(x) = \sum_{i=1}^{N} w_i \pi_i(x), \quad \pi_i = \mathcal{N}(x; \mu_i, P_i), \quad \sum_{i=1}^{N} w_i = 1, \tag{3.150}$$

i.e., a mixture model with fewer components. This is achieved by defining a merging function that combines different components:

$$f(\pi_i(x), \pi_j(x)) = w_{ij} \mathcal{N}(x; \mu_{ij}, P_{ij}) \tag{3.151}$$

where f represents the merging function for two components $\pi_i(x)$ and $\pi_j(x)$ with w_{ij}, μ_{ij} and P_{ij} being the combined weight, mean and covariance, respectively. These parameters can be computed as

$$w_{ij} = w_i + w_j \tag{3.152}$$

$$\mu_{ij} = \frac{w_i}{w_{ij}} \mu_i + \frac{w_j}{w_{ij}} \mu_j \tag{3.153}$$

$$P_{ij} = \frac{w_i}{w_{ij}} P_i + \frac{w_j}{w_{ij}} P_j + \frac{w_i w_j}{w_{ij}^2} (\mu_i - \mu_j)(\mu_i - \mu_j)^\top. \tag{3.154}$$

The GMM reduction process proceeds by iteratively merging pairs of components. To guide the merging, a bound on the Kullback–Leibler (KL) divergence is used to quantify the information loss that occurs when two components are merged. This bound is denoted as $B(i, j)$ and is defined for the merging of the i-th and j-th components as:

$$B(i, j) \triangleq \frac{1}{2} \left[w_{ij} \log |P_{ij}| - w_i \log |P_i| - w_j \log |P_j| \right]. \tag{3.155}$$

Here, $|\cdot|$ represents the determinant of a matrix. The primary purpose of $B(i, j)$ is to identify the pair of components whose merging results in the smallest increase in the KL divergence value, thereby minimizing the overall information loss in the mixture. For a full derivation of this bound, refer to Section VI of [83].

It is worth noting that $B(i, j) = B(j, i)$ and $B(i, i) = 0$, which simplifies the search for the optimal pair of components to merge by reducing the evaluations to $\frac{1}{2}N(N-1)$ combinations, where N represents the number of components in the mixture prior to reduction.

3.12.5 Characteristic Function Filter

The main challenge in computing the Bayes equations lies in the convolution integral required for the predict stage. The CF filter resorts to the same concept that is used

when solving differential equations, by using the Fourier transform. Foundational work on this method is documented in [84], [85], with a comprehensive formulation provided in [84]. In the prediction step, the convolution is converted to a multiplications whereas the update step becomes a convolution. In cases where the number of measurements is small, it might be benefitial to resort to the CF filter.

The CF of a random variable $\mathbf{X}$, denoted as $\phi_{\mathbf{X}}(\boldsymbol{\nu})$, provides a complete characterization of its probability distribution and is defined as the expected value of $e^{i\boldsymbol{\nu}^\top \mathbf{x}}$, where i represents the imaginary unit, $\boldsymbol{\nu}$ is a vector in $\mathbb{R}^n$, and $f_{\mathbf{X}}(\mathbf{x})$ is the probability density function (PDF) of the random variable $\mathbf{X}$:

$$\mathbb{E}[e^{i\boldsymbol{\nu}^\top \mathbf{X}}] = \int_{\mathbb{R}^n} e^{i\boldsymbol{\nu}^\top \mathbf{x}} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}. \tag{3.156}$$

The steps of the CF filter are given as:

- **Prediction Step:** In the prediction step, given the characteristic function of the un-normalized posterior PDF after the measurement update, denoted as $\bar{\phi}_{X_{k-1}|Z_{k-1}}(\boldsymbol{\nu})$, the aim is to compute its propagated version, $\bar{\phi}_{X_k|Z_{k-1}}(\boldsymbol{\nu})$, after applying the time propagation model:

$$\tag{3.157}$$

$$\bar{\phi}_{X_k|Z_{k-1}}(\boldsymbol{\nu}) = \bar{\phi}_{X_{k-1}|Z_{k-1}}\left(A^\top \boldsymbol{\nu}\right) \phi_D\left(\boldsymbol{\nu}\right). \tag{3.158}$$

- **Update Step:** The updated characteristic function can be expressed as:

$$\bar{\phi}_{X_k|Z_k}(\boldsymbol{\nu}) = \frac{1}{(2\pi)^p} \int_{\xi} \bar{\phi}_{X_k|Z_{k-1}}\left(\boldsymbol{\nu} - H^\top \xi\right) \phi_V(-\xi) e^{j\xi^\top z_k} d\xi. \tag{3.159}$$

In the above equations, $x_k \in \mathbb{R}^n$ represents the system state vector at time k, and $z_k \in \mathbb{R}^p$ is the corresponding measurement vector. The matrices $A \in \mathbb{R}^{n\times n}$ and $H \in \mathbb{R}^{p\times n}$ are known system parameters that describe the state transition and observation models, respectively. The vector-valued process noise $d_t \in \mathbb{R}^n$ is characterized by its characteristic function $\phi_D\left(\boldsymbol{\nu}_d\right)$, where $\boldsymbol{\nu}_d \in \mathbb{R}^n$. Similarly, the measurement noise $v_t \in \mathbb{R}^p$ is described by its characteristic function $\phi_V\left(\boldsymbol{\nu}_v\right)$, with $\boldsymbol{\nu}_v \in \mathbb{R}^p$. The variable $\xi \in \mathbb{R}^p$ serves as an integration variable in the update step.

3.13 Stochastic State Estimation with Hybrid Filter

In order to avoid having to numerically evaluate convolution integrals, the proposed Hybrid Filter (HF) combines a predict step in the characteristic domain and an update in the original domain. However, converting between the two domains requires evaluating Fourier or inverse Fourier integrals. In order to avoid those steps, the

proposed filter aims to find an approximating CF for a known distribution such that its inversion can be direct. Therefore, between the predict and update steps, we need to solve the following optimization problem:

$$\begin{aligned} \min_{\theta} \quad & \int \left|\phi(\boldsymbol{\nu}) - \phi_{\text{approx.}}(\boldsymbol{\nu}, \theta)\right|^2 d\boldsymbol{\nu} \\ \text{s.t.} \quad & \text{constraints dependent on known CF} \end{aligned} \tag{3.160}$$

where ϕ and $\phi_{\text{approx.}}$ are the actual and approximated (known) characteristic functions, respectively, $\boldsymbol{\nu}$ is a vector in $\mathbb{R}^n$, and θ is the parameter vector whose dimension depends on the specific form of the approximating CF.

3.13.1 Hybrid Filter via Dirac Approximation

A possible implementation of the HF is using a dirac approximation of the PDFs, derived from their corresponding CFs. Recalling (3.156), the CF of a probability distribution is expressed as an expected value over kernel functions of the form $e^{i\boldsymbol{\nu}^\top \mathbf{x}}$. This representation implies that the CF can be viewed as a weighted sum of these exponential kernel functions. Consequently, the integral defining the CF can be discretized as:

$$\phi(\boldsymbol{\nu}) \approx \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}^\top \mathbf{x_k}} \tag{3.161}$$

where there are N points $\mathbf{x_k}$, representing discretized values of the variable $\mathbf{X}$, and p_k are weights corresponding to the probabilities at these points. This approximation transforms the integral into a finite sum, thereby facilitating numerical computation. Here, $e^{i\boldsymbol{\nu}^\top \mathbf{x_k}}$ corresponds to the CF of a Dirac distribution centered at $\mathbf{x_k}$, effectively approximating the original PDF by a weighted sum of Dirac distributions, resulting in a discretized PDF.

To derive the discretized PDF, we establish an optimization problem that aims to align the discrete approximation of the CF with the actual CF. The objective is to minimize the squared L^2-norm of the difference between the actual CF and its approximation, formulated as follows:

$$\begin{aligned} \min_{p_k} \quad & \int \left| \phi(\boldsymbol{\nu}) - \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}^\top \mathbf{x_k}} \right|^2 d\boldsymbol{\nu} \\ \text{s.t.} \quad & \sum_{k=1}^{N} p_k = 1 \\ & p_k \geq 0 \quad \text{for all } k = 1, \ldots, N. \end{aligned} \tag{3.162}$$

This minimization problem adjusts the weights p_k to ensure that the approximation closely matches the actual CF. Constraints are imposed to guarantee that the weights p_k form a valid probability distribution.

From a computational perspective, this optimization problem can be effectively solved by discretizing the integral over a grid of points $\boldsymbol{\nu}$ as in (3.163). This transformation results in a constrained linear least squares problem, where the objective is to determine the weights p_k that best fit the discrete version of the CF approximation to the actual CF at these grid points.

$$\begin{aligned} \min_{p_k} \quad & \sum_{l=1}^{M} \left| \phi(\boldsymbol{\nu}_l) - \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}_l^\top \mathbf{x_k}} \right|^2 \\ \text{s.t.} \quad & \sum_{k=1}^{N} p_k = 1 \\ & p_k \geq 0 \quad \text{for all } k = 1, \ldots, N. \end{aligned} \tag{3.163}$$

The selection of the grid points requires first determining the intervals for each of the variables. The concept of moments is critical to determine the set where the mass of the PDF is concentrated.

The first two moments, mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of a random vector $\mathbf{X}$ can be computed from the CF $\phi_{\mathbf{X}}(\boldsymbol{\nu})$ as

$$\boldsymbol{\mu} = -i \left. \frac{d\phi_{\mathbf{X}}(\boldsymbol{\nu})}{d\boldsymbol{\nu}} \right|_{\boldsymbol{\nu}=0} \tag{3.164}$$

$$\boldsymbol{\Sigma} = - \left. \frac{d^2\phi_{\mathbf{X}}(\boldsymbol{\nu})}{d\boldsymbol{\nu}^2} \right|_{\boldsymbol{\nu}=0} - \boldsymbol{\mu}\boldsymbol{\mu}^\top. \tag{3.165}$$

To determine the interval of discretization over $\mathbf{X}$, we utilize the moment information contained in the original CF. Specifically, we extract the first and second moments using (3.164) and (3.165), respectively. Then, we calculate the interval I_j for each dimension j of the grid as follows:

$$I_j = \left[\mu_j - 3.5\Sigma_{j,j}^{\frac{1}{2}}, \mu_j + 3.5\Sigma_{j,j}^{\frac{1}{2}} \right]. \tag{3.166}$$

The discretization intervals over $\boldsymbol{\nu}$ are determined using an approximation based on the CF of a zero-mean Gaussian distribution. Given that the significant portion of the CF is within the range where its magnitude is non-negligible, we set a threshold ϵ to define this effective range. Assuming a zero-mean Gaussian CF, the magnitude decays as $e^{-\frac{\boldsymbol{\nu}^\top \Sigma \boldsymbol{\nu}}{2}}$, where Σ is the covariance matrix. The threshold is set where the magnitude of the CF approximates zero:

$$e^{-\frac{\boldsymbol{\nu}^\top \Sigma \boldsymbol{\nu}}{2}} \leq \epsilon \tag{3.167}$$

Taking the logarithm of both sides and solving for $\boldsymbol{\nu}$, we obtain:

$$\boldsymbol{\nu}^\top \Sigma \boldsymbol{\nu} \geq 2 \log\left(\frac{1}{\epsilon}\right) \tag{3.168}$$

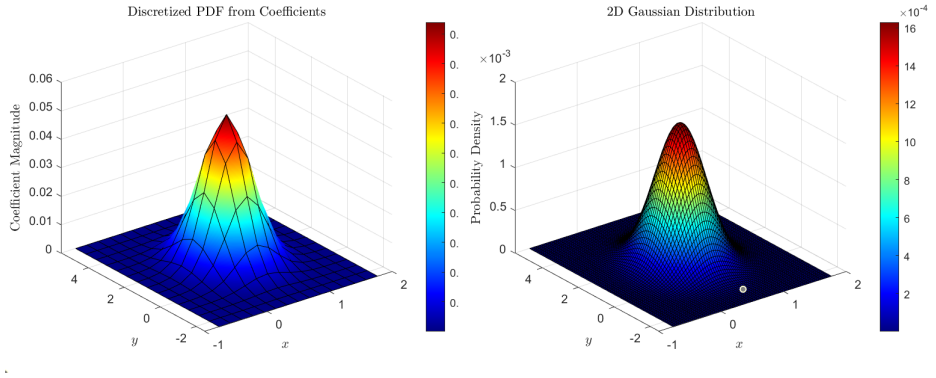
For each dimension j, we approximate the range of ν_j by considering only the diagonal elements $\Sigma_{j,j}$, simplifying the expression to:

$$|\nu_j| \geq \sqrt{\frac{2 \log\left(\frac{1}{\epsilon}\right)}{\Sigma_{j,j}}} \tag{3.169}$$

This approximation provides a practical means of selecting discretization intervals over $\boldsymbol{\nu}$ that are expected to capture the essential behavior of the CF for the purpose of estimating the PDF. Once these intervals are determined, uniform random sampling is employed to generate the N points $\mathbf{x_k}$ and the M points $\boldsymbol{\nu}_l$.

Following this approximation, we obtain a discretized representation of the PDF, which corresponds to particles within the PF framework. This is where the PF update step becomes particularly useful, as it can now operate on these particles. After completing this step, it is possible to transition back to the frequency domain by incorporating the updated particles and their associated weights into the CF expression as in (3.161), substituting the $\mathbf{x_k}$ and p_k values accordingly.

To better illustrate the proposed approximation, Figure 3.25 shows how a Gaussian distribution is approximated by the sum of dirac distributions in the original domain with Figure 3.26 showing the equivalent representation in the Fourier domain.



(a) (b)

Fig. 3.25: Gaussian *PDF*, approximation on the left and real *PDF* on the right.

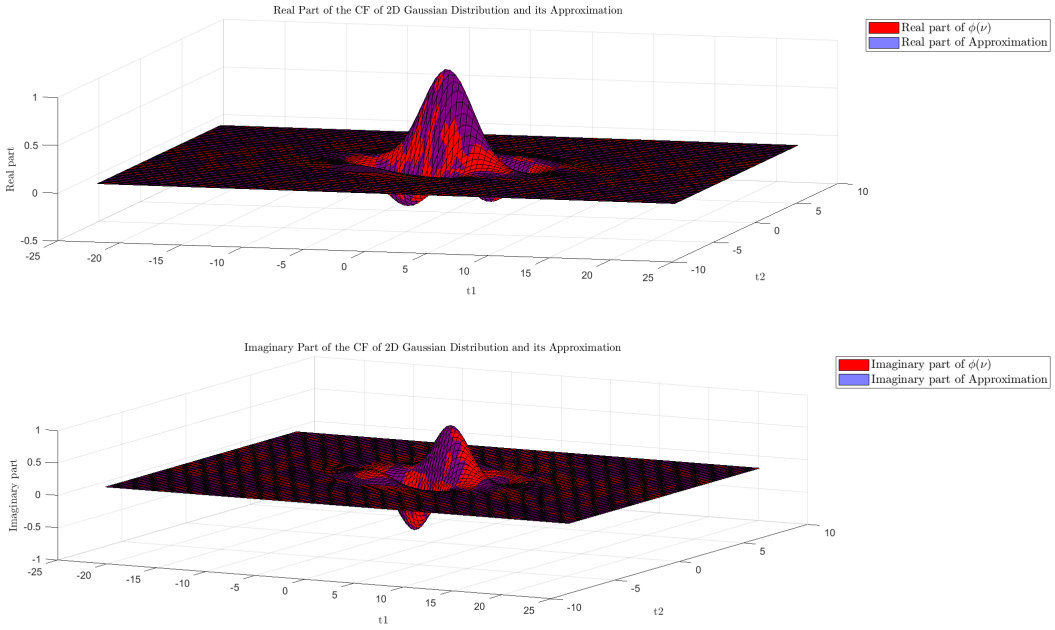
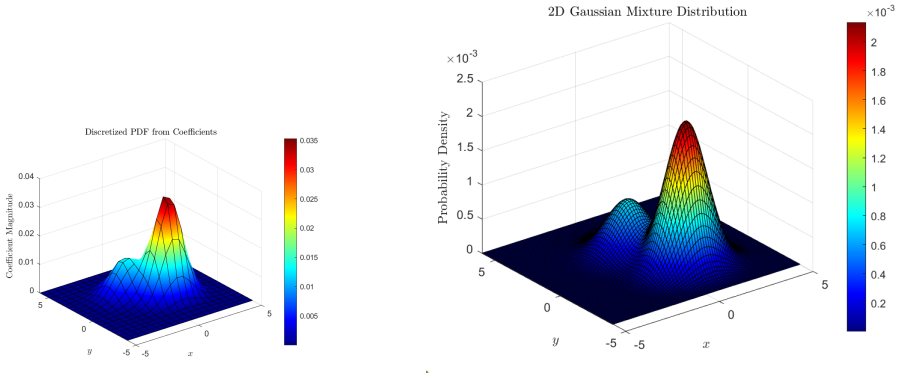


Fig. 3.26: Real and approximated CFs of gaussian random variable.

Similarly, Figure 3.27 provides an example for the real and approximated GMM with its representation in the Fourier domain given in Figure 3.28.



(a) (b)

Fig. 3.27: Gaussian Mixture *PDF*, approximation on the left and real *PDF* on the right.

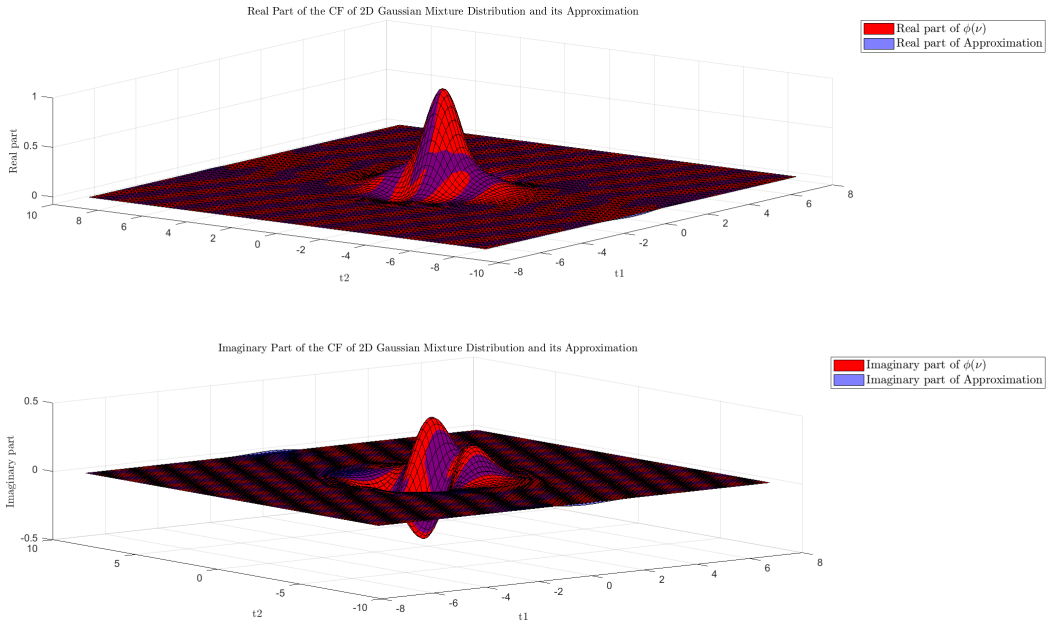
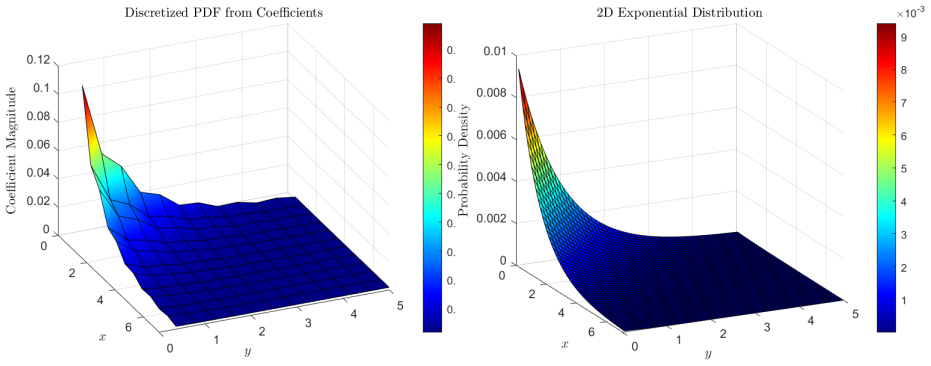


Fig. 3.28: Real and approximated *CF's* of gaussian mixture random variable.

Lastly, Figure 3.29 and Figure 3.30 show the same example for an exponential distribution to assess the quality of the approximation in a distribution that is not centered and presents a tail.



(a) (b)

Fig. 3.29: Exponential *PDF*, approximation on the left and real *PDF* on the right.

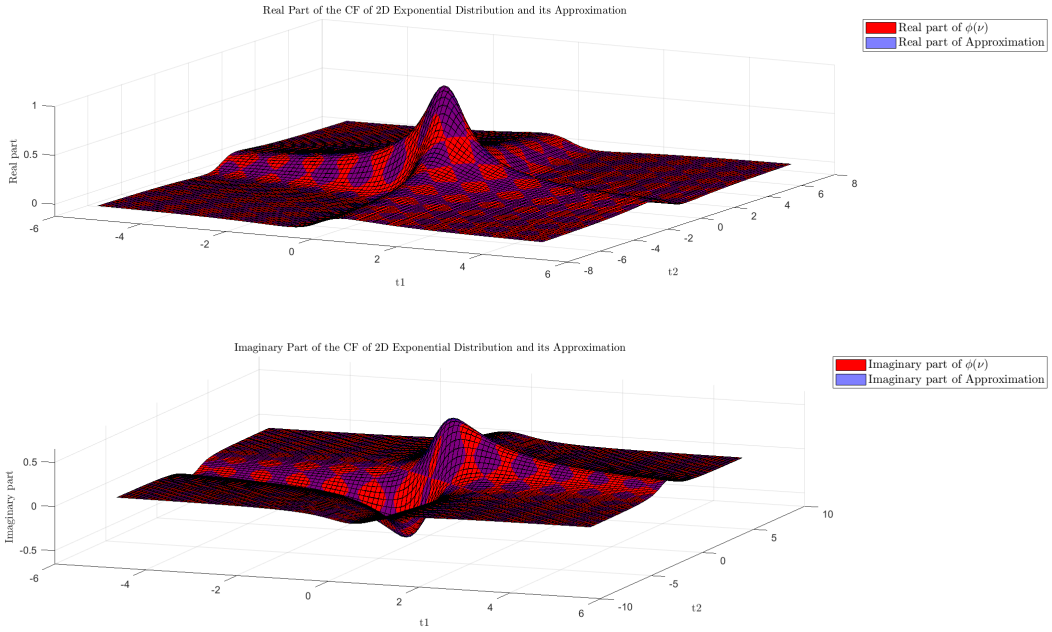


Fig. 3.30: Real and approximated *CF's* of exponential random variable.

3.13.2 Hybrid Filter via Gaussian Mixture Approximation

Instead of approximating the CF with a sum of dirac distributions, we can also consider a GMM as the kernel function. Since the CF of a Gaussian distribution centered at $\boldsymbol{\mu}$ with covariance $\boldsymbol{\Sigma}$ is given by

$$\phi_{\text{Gaussian}}(\boldsymbol{\nu}) = e^{i\boldsymbol{\nu}^\top \boldsymbol{\mu}} e^{-\frac{1}{2}\boldsymbol{\nu}^\top \boldsymbol{\Sigma}\boldsymbol{\nu}}, \tag{3.170}$$

we can directly express the CF of a GMM as a weighted sum of the individual CFs of its Gaussian components and obtain the following expression for the approximation

$$\phi(\boldsymbol{\nu}) \approx \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}^\top \boldsymbol{\mu}_k} e^{-\frac{1}{2}\boldsymbol{\nu}^\top \boldsymbol{\Sigma}_k \boldsymbol{\nu}}, \tag{3.171}$$

where N is the number of Gaussian components, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the k-th Gaussian, and p_k are the corresponding weights.

To approximate the PDF, we formulate a minimization problem akin to the Dirac approximation approach. Here, the objective is to minimize the squared L^2-norm of the difference between the actual CF and its Gaussian mixture approximation:

$$\begin{aligned} \min_{p_k,\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k} \quad & \int \left| \phi(\boldsymbol{\nu}) - \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}^\top \boldsymbol{\mu}_k} e^{-\frac{1}{2}\boldsymbol{\nu}^\top \boldsymbol{\Sigma}_k \boldsymbol{\nu}} \right|^2 d\boldsymbol{\nu} \\ \text{s.t.} \quad & \sum_{k=1}^{N} p_k = 1 \\ & p_k \geq 0 \quad \text{for all } k = 1, \ldots, N, \\ & \boldsymbol{\Sigma}_k \succ 0 \quad \text{for all } k = 1, \ldots, N. \end{aligned} \tag{3.172}$$

The constraints ensure that the weights p_k form a valid GMM and that the covariance matrices $\boldsymbol{\Sigma}_k$ are positive definite. This optimization problem is inherently more challenging than the Dirac-based approach, as it requires solving for not only the weights p_k, but also the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ for each Gaussian component.

In practice, solving this optimization problem involves discretizing the domain $\boldsymbol{\nu}$ into grid points and transforming the continuous integral into a finite summation over these points. This transformation reduces the problem to a constrained non-linear optimization, which can be addressed using standard numerical methods, such as gradient-based approaches.

$$\begin{aligned} \min_{p_k,\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k} \quad & \sum_{l=1}^{M} \left| \phi(\boldsymbol{\nu}_l) - \sum_{k=1}^{N} p_k e^{i\boldsymbol{\nu}_l^\top \boldsymbol{\mu}_k} e^{-\frac{1}{2}\boldsymbol{\nu}_l^\top \boldsymbol{\Sigma}_k \boldsymbol{\nu}_l} \right|^2 \\ \text{s.t.} \quad & \sum_{k=1}^{N} p_k = 1 \\ & p_k \geq 0 \quad \text{for all } k = 1, \ldots, N, \\ & \boldsymbol{\Sigma}_k \succ 0 \quad \text{for all } k = 1, \ldots, N. \end{aligned} \tag{3.173}$$

Furthermore, we can leverage the moments of the CF to guide the initialization of the Gaussian parameters. The mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the underlying distribution can be directly derived from the CF using (3.164) and (3.165).

Similar to the Dirac-based approach, these moments can be used to define the discretization intervals (Equations 3.166 and 3.169), providing the ranges needed to generate the N points $\mathbf{x_k}$ and the M points $\boldsymbol{\nu}_l$ via uniform random sampling.

3.14 Simulation Results for the Stochastic State Estimation

The simulations consider estimating the position of a vehicle modeled using discrete double integrator dynamics with a sampling time of $T_s = 0.1$ s performing a random walk. This means that the control input u_k is sampled from either a Gaussian distribution or a GMM with two or three components. We use the observation matrix H defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with noise v_t sampled from either a Gaussian, Gaussian Mixture, or an exponential distributions. These variations in both process and observation noise provide a comprehensive testbed for evaluating the HF under diverse noise scenarios. The initial state of the vehicle is assumed to be a random variable following a zero-mean Gaussian distribution. Therefore, using all combinations results in nine setups:

- Gaussian walk with Gaussian noise.
- Gaussian walk with exponential noise.
- Gaussian walk with GMM noise.
- GMM (2 peaks) walk with Gaussian noise.
- GMM (2 peaks) walk with exponential noise.
- GMM (2 peaks) walk with GMM noise.
- GMM (3 peaks) walk with Gaussian noise.
- GMM (3 peaks) walk with exponential noise.
- GMM (3 peaks) walk with GMM noise.

For each of these scenarios, the Dirac approach was tested with different numbers of particles, while the Gaussian Mixture approach varied the number of Gaussian components in the mixture, keeping the particle count fixed at 256.

To provide a benchmark, both the KF and PF were used as comparative methods. The KF represents an optimal solution under the Gaussian assumption, while the PF was configured with the same number of particles as the Hybrid Filter to ensure a fair comparison.

The filters were evaluated over 50 independent runs, each consisting of 150 time steps. In each run, the vehicle's initial velocity was set to $\begin{bmatrix} 0 & 0 \end{bmatrix}^\top$, while its initial position was uniformly sampled within the square defined by vertices at $(-8, -8)$,

$(-8,-10)$, $(-10,-10)$, and $(-10,-8)$. During each run, the type of random walk and observation noise remained fixed, allowing for a controlled assessment of filter performance under specific conditions.

The performance of each filter was evaluated using two key metrics: cumulative Root-Mean-Square-Error (RMSE) over the 150 time steps to assess estimation accuracy, and the trace of the covariance matrix to evaluate the uncertainty associated with the estimates at each time step.

3.14.1 Hybrid Filter via Dirac Approximation

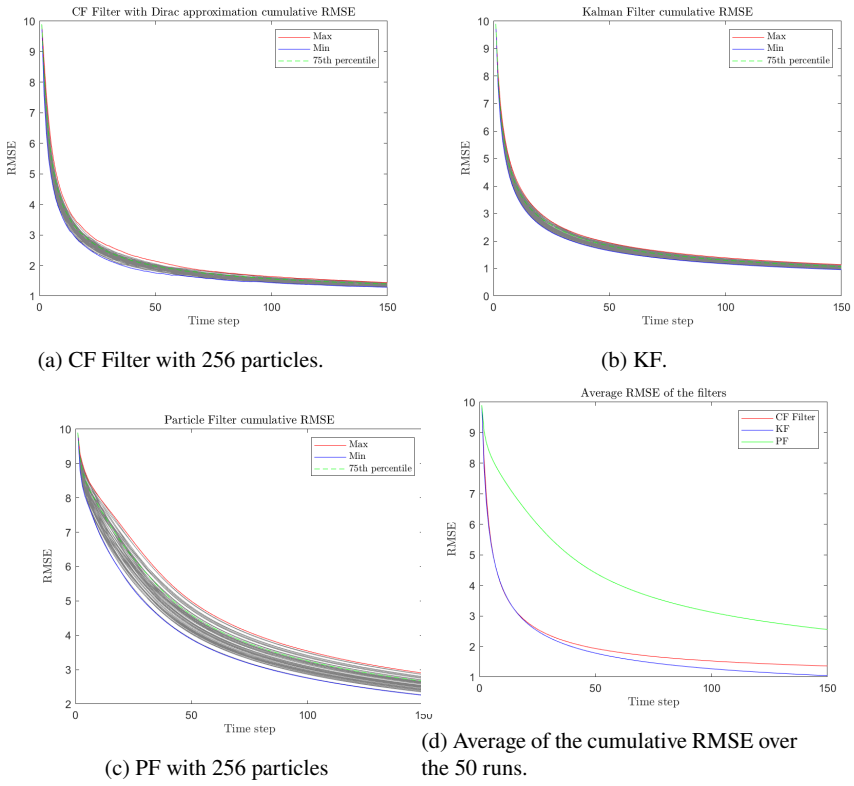
For the Dirac approximation approach, the nine distinct scenarios were tested with different numbers of particles — specifically, 256, 625, and 1296 — resulting in a total of 27 simulations. Here, we present and discuss only the most pertinent results.

The performance of the CF Filter in the Gaussian walk with Gaussian noise scenario is illustrated in Figure 3.31. The first three plots show the cumulative RMSE of each filter across all 50 runs (represented by gray lines). These plots also indicate the upper and lower performance bounds for each filter, denoted by the maximum and minimum lines. The 75^{th} percentile of the RMSE is represented in green to provide further insight into the overall performance. Finally, for ease of comparison, Figure 3.31d shows the average cumulative RMSE at each time step, averaged over the 50 runs.

The CF Filter demonstrates strong performance, closely following the KF, which is expected to yield optimal results in this linear Gaussian setting. This outcome is particularly noteworthy as it highlights the capability of the CF Filter to remain competitive even in scenarios where the KF benefits from assumptions that perfectly align with its design. Given that the CF Filter is also capable of handling non-Gaussian noise distributions, its close performance to the KF in this Gaussian scenario suggests robustness and adaptability across a wide range of noise types. Moreover, the CF Filter exhibits consistent behavior across the 50 runs, as evidenced by the tight clustering of error bounds in Figure 3.31a. This consistency implies that the CF Filter maintains reliability and robustness even under varying conditions.

In contrast, the PF demonstrates relatively weaker performance. While the PF theoretically has the capability to manage non-Gaussian noise and nonlinearities, its effectiveness in this scenario is limited by the relatively small particle count (256). This constraint likely results in higher variance in the estimation results and a greater overall RMSE, as evidenced in Figure 3.31c. The reliance of the PF on a sufficiently large number of particles for accurate state estimation thus becomes a significant bottleneck, especially when compared to the CF Filter.

Figure 3.31d provides a direct comparison of the average cumulative RMSE for each filter across 50 runs. While the KF yields the best performance, the CF



(a) CF Filter with 256 particles.

(b) KF.

(c) PF with 256 particles

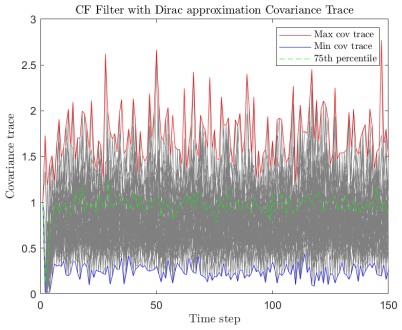
(d) Average of the cumulative RMSE over the 50 runs.

Fig. 3.31: Cumulative RMSE of the position of Gaussian walk with Gaussian noise.

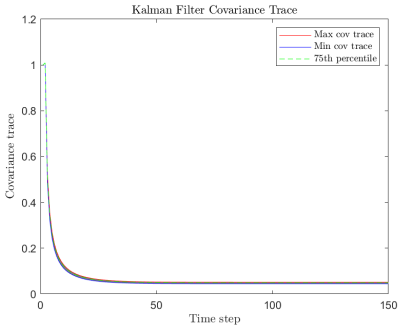
Filter achieves results that are nearly indistinguishable from those of the KF, further validating its applicability as a viable alternative, particularly in environments where the Gaussian noise assumption might not hold. The PF, on the other hand, falls significantly behind, underscoring the limitations of particle-based methods when computational resources are constrained.

The covariance trace results in Figures 3.32a, 3.32b, and 3.32c illustrate key differences in how each filter manages estimation uncertainty. The CF Filter, while effective in state estimation, does not prioritize minimizing estimation uncertainty to the same degree as the KF. The KF had lower covariance trace values in contrast with the CF which presents a greater variability across runs.

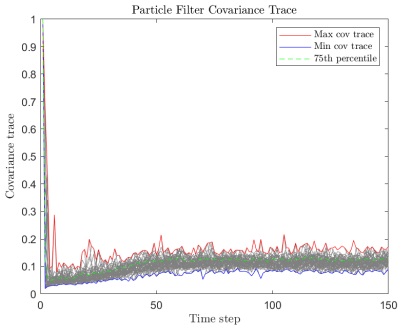
Although the PF displays greater variability compared to the KF, its overall uncertainty remains relatively low. This can be attributed to the inherent nature of particle filters, which represent state distributions by sampling a large number of particles. However, the limited particle count introduces fluctuations in the covariance trace, indicating the need for more particles to achieve more reliable estimation.



(a) CF Filter with 256 particles.



(b) KF.



(c) PF with 256 particles.

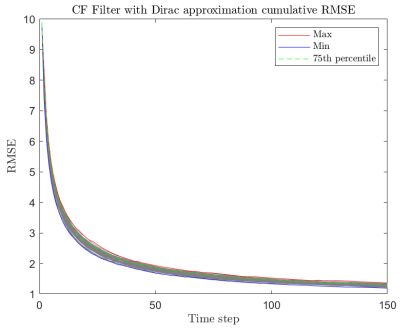
Fig. 3.32: Covariance matrix trace of the position estimate uncertainty in the Gaussian walk with Gaussian noise.

To evaluate the impact of increasing the particle count, we turn to Figures 3.33 and 3.34. These figures demonstrate improved performance for both the CF Filter and the PF. Notably, the cumulative RMSE is reduced, and the error bounds in Figures 3.33a and 3.33b are narrower, indicating increased consistency across runs.

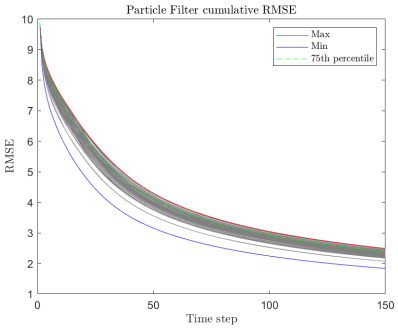
A similar trend is observed in Figures 3.34a and 3.34b, where the average covariance trace decreases slightly and the uncertainty bounds become tighter, further highlighting the benefits of increasing the particle count.

In the scenario where the CF Filter is tested with a Gaussian mixture walk and exponential noise using 1296 particles, the results show the ability of the CF filter to cope with non-gaussian signals. The results are provided in Figures 3.35a, 3.35b, and 3.35c for the behavior over the 50 runs.

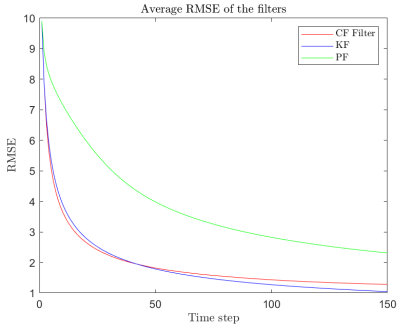
The CF Filter accuracy surpasses the KF due to the introduction of exponential noise. The tighter error bounds and reduced variability across runs (as observed in Figure 3.35a) underscore the robustness of the CF Filter, which manages to maintain consistent accuracy across a more complex noise distribution.



(a) CF Filter with 1296 particles.

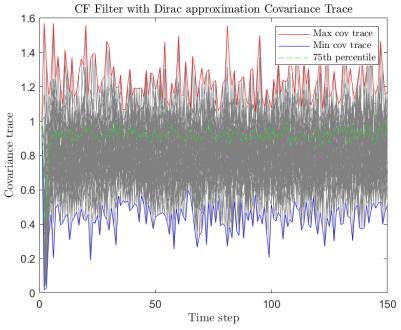


(b) PF with 1296 particles.

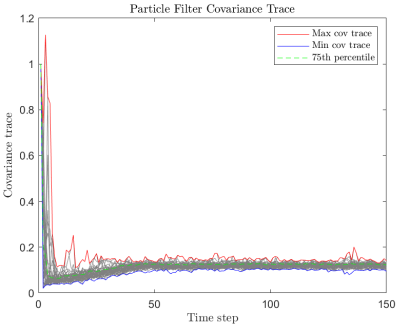


(c) Average of the cumulative RMSE over the 50 runs.

Fig. 3.33: Cumulative RMSE of the position of Gaussian walk with Gaussian noise with 1296 particles.



(a) CF Filter with 1296 particles.

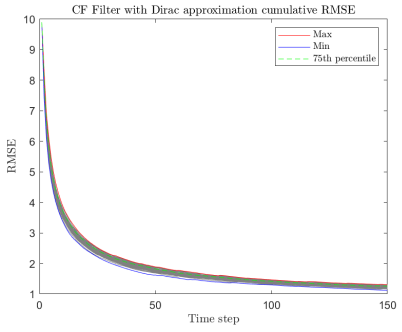


(b) PF with 1296 particles.

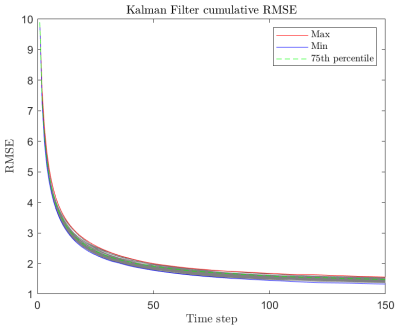
Fig. 3.34: Covariance matrix trace of the position estimate in the Gaussian walk with Gaussian noise.

In comparison, the KF, which is designed primarily for Gaussian noise environments, exhibits a decline in relative performance. This outcome is consistent with theoretical expectations, as the KF struggles under non-Gaussian conditions where its underlying noise assumptions are no longer valid.

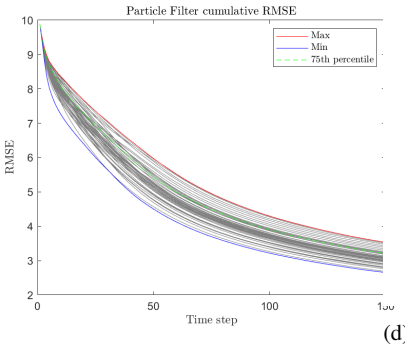
The PF, despite being theoretically suitable for handling non-Gaussian noise, continues to show relatively weaker performance. This can be attributed to the number of particles (1296), which still seems insufficient for accurately capturing the underlying noise characteristics and system dynamics. The higher variance and broader error bounds (as depicted in Figure 3.35c) suggest that the PF struggles to consistently match the performance of the CF Filter. The overall comparison of the average cumulative RMSE in Figure 3.35d shows a better accuracy even in steady-state.



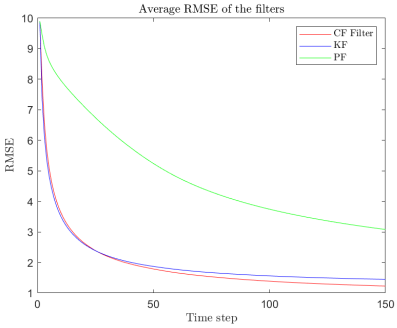
(a) CF Filter with 1296 particles.



(b) KF.



(c) PF with 1296 particles.



(d) Average of the cumulative RMSE over the 50 runs.

Fig. 3.35: Cumulative RMSE of the position of Gaussian Mixture walk with Exponential noise.

To finalize the analysis of the results from the Dirac approach, we refer to Table 3.1, which provides a summary of the performance of each filter across different

scenarios. The table reports both the mean RMSE and its standard deviation over the 50 runs, with the best results for each scenario highlighted in bold.

From the table, it is evident that the CF Filter consistently achieves the best performance in scenarios involving exponential noise. This is further underscored by its low standard deviation, suggesting that the filter maintains robust performance across different runs. Such consistency can be attributed to the inherent flexibility of the CF Filter, which is well-adapted to handle non-Gaussian noise types. Additionally, the results illustrate the beneficial impact of increasing the particle count on the CF Filter's performance. As the number of particles grows, the RMSE decreases, indicating an improved capacity of the filter to approximate the underlying system dynamics more accurately.

One notable observation from the table is that, in Gaussian mixture noise scenarios, the KF continues to outperform the CF Filter. This outcome can be explained by the fact that the Gaussian mixture noise used in these simulations is not highly complex and can be effectively approximated by a zero-mean Gaussian distribution.

The PF, on the other hand, continues to exhibit weaker performance across most scenarios. Although the PF is theoretically well-suited to handle non-Gaussian noise and complex system dynamics, its effectiveness here is constrained by the relatively low particle count. Even with an increased number of particles, the PF fails to match the performance of either the CF Filter or the KF, suggesting that it struggles to capture the full spectrum of system behaviors under these conditions. The higher variance in RMSE across the runs further highlights the limitations of the PF, as it requires a significantly larger particle count to achieve a more consistent and accurate estimate.

		Simulation Scenario	CF Filter w/ 256 particles	CF Filter w/ 625 particles	CF Filter w/ 1296 particles	KF	PF w/ 256 particles	PF w/ 625 particles	PF w/ 1296 particles
Mean RMSE	**Average over 50 simulations**	G. Walk w/ G. Noise	1.3635	1.3155	1.2808	**1.0579**	2.5559	2.4408	2.3117
		G. Walk w/ GM Noise	1.3643	1.3262	1.2915	**1.0871**	2.5236	2.4160	2.2951
		G. Walk w/ Exp. Noise	1.2754	1.2395	**1.2263**	1.4449	3.4771	3.2391	3.0742
		GM w/ 2 peaks Walk w/ G. Noise	1.3635	1.3081	1.2729	**1.0410**	2.5611	2.4158	2.2951
		GM w/ 2 peaks Walk w/ GM Noise	1.3712	1.3182	1.2789	**1.0940**	2.5742	2.3881	2.3176
		GM w/ 2 peaks Walk w/ Exp. Noise	1.2917	1.2400	**1.2210**	1.3831	3.4365	3.1862	3.0585
		GM w/ 3 peaks Walk w/ G. Noise	1.3626	1.3166	1.2885	**1.0488**	2.5717	2.3875	2.2942
		GM w/ 3 peaks Walk w/ GM Noise	1.3763	1.3213	1.2875	**1.0934**	2.5918	2.4245	2.3195
		GM w/ 3 peaks Walk w/ Exp. Noise	1.2868	1.2464	**1.2284**	1.4446	3.4472	3.2467	3.0810
	Standard deviation over 50 simulations	G. Walk w/ G. Noise	0.0400	0.0348	**0.0344**	0.0418	0.1572	0.1125	0.1225
		G. Walk w/ GM Noise	0.0360	0.0404	0.0373	**0.0358**	0.1555	0.1487	0.1312
		G. Walk w/ Exp. Noise	0.0471	**0.0352**	0.0362	0.0454	0.2897	0.2810	0.2048
		GM w/ 2 peaks Walk w/ G. Noise	0.0462	0.0376	**0.0356**	0.0361	0.1927	0.1397	0.1238
		GM w/ 2 peaks Walk w/ GM Noise	0.0405	0.0408	**0.0311**	0.0422	0.1248	0.1433	0.0935
		GM w/ 2 peaks Walk w/ Exp. Noise	0.0417	0.0420	**0.0395**	0.0474	0.2594	0.2451	0.1845
		GM w/ 3 peaks Walk w/ G. Noise	0.0451	0.0389	**0.0296**	0.0380	0.1490	0.1338	0.1143
		GM w/ 3 peaks Walk w/ GM Noise	0.0337	0.0328	**0.0298**	0.0485	0.1575	0.1295	0.1108
		GM w/ 3 peaks Walk w/ Exp. Noise	0.0470	**0.0350**	0.0394	0.0400	0.2687	0.2353	0.2083

Table 3.1: RMSE of all filters along different simulation scenarios, Dirac approach.

3.14.2 Hybrid Filter via Gaussian Mixture Approximation

For the Gaussian mixture approach, nine distinct scenarios were tested with the number of Gaussians used in the approximation varying between three and four, resulting in a total of 18 simulations. Here, we present and discuss only the most pertinent results.

The results of the CF Filter via the Gaussian mixture approximation with three Gaussians, tested in a scenario involving both a Gaussian mixture walk and Gaussian mixture noise, are illustrated in Figures 3.36a, 3.36b, 3.36c, and 3.36d. The focus here is on assessing how the CF Filter, based on the Gaussian mixture approximation, performs relative to the KF and PF across these 50 simulation runs.

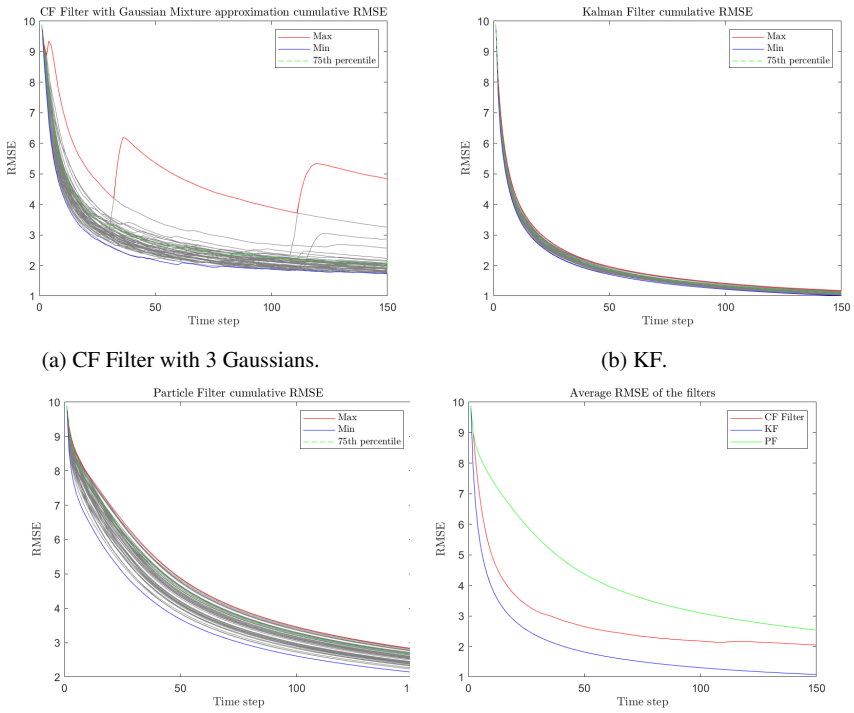
As seen in Figure 3.36a, the CF Filter employing the Gaussian mixture approximation with three Gaussians demonstrates a more erratic performance compared to the results obtained using the Dirac approach. The cumulative RMSE reveals fluctuations across different runs, as indicated by the larger spread between maximum and minimum error bounds. These variations are the result of the non-convex nature of the optimization problem involved in determining the parameters.

Despite these fluctuations, the CF Filter still demonstrates a strong average performance, as highlighted by the comparison in Figure 3.36d. The average cumulative RMSE for the CF Filter remains competitive, particularly in comparison to the PF. This indicates that while individual runs may exhibit some instability due to the complexity of the Gaussian mixture approximation, the filter still approximates the state effectively over the 50 simulations.

In contrast, the KF, as shown in Figure 3.36b, maintains a more stable performance. This is expected given that the Gaussian mixture noise used in these simulations can still be reasonably approximated by a zero-mean Gaussian distribution. The PF remains the worst option in terms of accuracy. As depicted in Figure 3.36c, the PF shows slower convergence and a higher overall RMSE. The performance of the particle filter in this scenario may be hindered by the relatively low number of particles (256).

When comparing the average cumulative RMSE across all filters in Figure 3.36d, the CF Filter, despite exhibiting fluctuations, still performs competitively with the KF. This suggests that while the Gaussian mixture approximation introduces additional complexity, the CF Filter retains a solid overall performance.

Increasing the number of Gaussians in the approximation from 3 to 4, as shown in Figures 3.37a and 3.37b, results in a noticeable improvement in the stability of the CF Filter. The spread between the maximum and minimum error bounds tightens, and the overall convergence behavior becomes smoother compared to the 3-Gaussian case. This suggests that increasing the number of Gaussian components enhances the filter's ability to capture the underlying complexity of the system dynamics and noise distributions, leading to more consistent state estimates.



(a) CF Filter with 3 Gaussians.

(b) KF.

(c) PF with 256 particles.

(d) 50-run average of the cumulative RMSE.

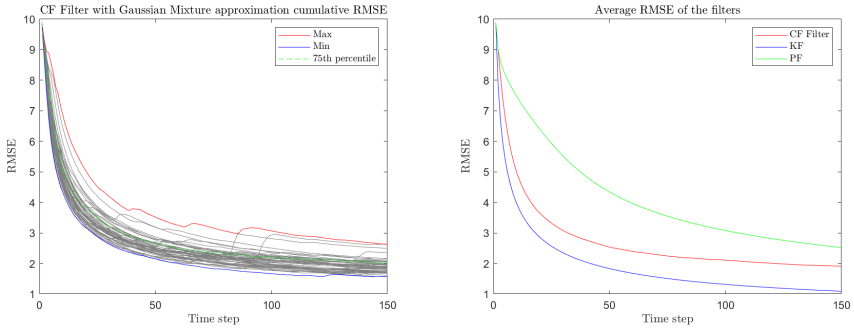
Fig. 3.36: Cumulative RMSE of the position of Gaussian Mixture w/ 2 peaks walk with Gaussian Mixture noise.

In terms of the average cumulative RMSE, the CF Filter with 4 Gaussians shows a modest reduction in error relative to the 3-Gaussian scenario, as illustrated in Figure 3.37b. This improvement helps to narrow the performance gap between the CF Filter and the KF, further emphasizing the value of a more refined Gaussian mixture approximation for accurately capturing the underlying noise characteristics.

Figures 3.38a and 3.38b present the covariance matrix trace for the CF Filter with 3 and 4 Gaussians, respectively.

In the 3-Gaussian case, significant spikes in the covariance trace indicate periods of elevated uncertainty. Although the 75^{th} percentile remains stable, the pronounced peaks in the maximum trace suggest that the filter occasionally struggles to accurately capture all modes of the noise distribution, resulting in increased uncertainty at specific time steps.

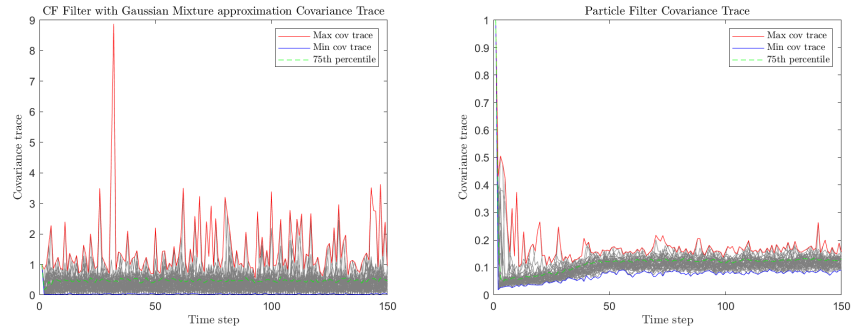
With 4 Gaussians, these spikes are less frequent and less pronounced. The overall maximum covariance trace is reduced, and the filter's uncertainty is more consistently controlled. This reduction in uncertainty and the more consistent performance



(a) CF Filter with 4 Gaussians. (b) 50-runs average of the cumulative RMSE.

Fig. 3.37: Cumulative RMSE of the position of Gaussian Mixture w/ 2 peaks walk with Gaussian Mixture noise.

suggest that increasing the number of Gaussian components improves the filter's ability to handle noise complexity effectively.



(a) CF Filter with 3 Gaussians. (b) CF Filter with 4 Gaussians.

Fig. 3.38: Covariance matrix trace of the position estimate uncertainty in the Gaussian Mixture w/ 2 peaks walk with Gaussian Mixture noise.

To conclude the analysis of the Gaussian mixture approach, Table 3.2 provides a summary of the performance of each filter across the different simulation scenarios. As shown in the table, the KF consistently outperforms the CF Filter across all noise scenarios, including those with exponential noise. Although the CF Filter demonstrates competitive performance, particularly when the number of Gaussian components is increased from 3 to 4, it is still less accurate than the KF.

		Simulation Scenario	CF Filter w/ 3 Gaussian peaks	CF Filter w/ 4 Gaussian peaks	KF	PF w/ 256 particles
Mean RMSE	**Average over 50 simulations**	G. Walk w/ G. Noise	1.9380	1.8374	**1.0595**	2.5732
		G. Walk w/ GM Noise	1.9860	1.8288	**1.1021**	2.5816
		G. Walk w/ Exp. Noise	2.5700	2.6360	**1.4468**	3.3789
		GM w/ 2 peaks Walk w/ G. Noise	1.9336	1.8633	**1.0522**	2.5832
		GM w/ 2 peaks Walk w/ GM Noise	2.0531	1.9073	**1.0907**	2.5199
		GM w/ 2 peaks Walk w/ Exp. Noise	2.5323	2.4467	**1.3974**	3.4502
		GM w/ 3 peaks Walk w/ G. Noise	1.9284	1.8785	**1.0572**	2.5621
		GM w/ 3 peaks Walk w/ GM Noise	2.0566	1.9670	**1.0913**	2.5201
		GM w/ 3 peaks Walk w/ Exp. Noise	2.6267	2.4170	**1.4407**	3.8082
	Standard deviation over 50 simulations	G. Walk w/ G. Noise	0.1905	0.2026	**0.0450**	0.1712
		G. Walk w/ GM Noise	0.2991	0.1551	**0.0427**	0.1573
		G. Walk w/ Exp. Noise	0.4495	0.8652	**0.0488**	0.2524
		GM w/ 2 peaks Walk w/ G. Noise	0.1473	0.1929	**0.0423**	0.1609
		GM w/ 2 peaks Walk w/ GM Noise	0.4822	0.2389	**0.0386**	0.1531
		GM w/ 2 peaks Walk w/ Exp. Noise	0.4619	0.4532	**0.0480**	0.2143
		GM w/ 3 peaks Walk w/ G. Noise	0.1880	0.1878	**0.0457**	0.1836
		GM w/ 3 peaks Walk w/ GM Noise	0.2693	0.4473	**0.0448**	0.1809
		GM w/ 3 peaks Walk w/ Exp. Noise	0.4756	0.3312	**0.0512**	0.3872

Table 3.2: RMSE of all filters along different simulation scenarios, Gaussian Mixture approach.

Chapter 4
Optimized Surveillance Trajectory Generation

The last objective of project FirePuma was the design of optimized trajectories over the computed risk map to prevent forrest fires through autonomous surveillance of the most problematic areas. Various problems were target during this task:

- Path following with collision avoidance using splines [86], soft constraints [87] [88], hard constraints [89] [90], Control Barrier Functions (CBFs) [91];
- Comparison of the various methods in [92];
- Trajectory generation for surveillance assuming local measurements of the risk map [93], [94];
- Trajectory generation for surveillance assuming global knowledge of the risk map [95];
- Vehicle design for surveillance [96];
- Cooperative control for a formation of vehicles [97];
- Theoretical study of which vehicles must receive commands from the base station for the overall system to remain observable and controllable [98];
- Techniques to guarantee feasibility of the controllers [99];
- Optimized solutions to include binary decisions within the controllers [100] [101].

4.1 Surveillance Trajectory as an Optimal Control Problem

Given the main problem introduced in Chapter 1 regarding the autonomous vehicle control to inspect the area, it is required to present how the problem was simplified

and modeled concerning the uncertainty map and the sensing model of the Unmanned Aerial Vehicle (UAV).

4.1.1 Uncertainty Map

The first topic to be defined is the uncertainty map. The uncertainty map is mathematically described by a nonnegative function $h : \mathbb{R}^2 \to \mathbb{R}_0^+$ that represents the *a priori* level of uncertainty about the existence of fire at each position $\mathbf{p} \in \mathbb{R}^2$. Since the original structure of the uncertainty map typically may not follow common and well-known models, we assume that the uncertainty function can be arbitrarily well approximated by a Gaussian mixture, which is a weighted sum of Gaussian components. Consequently, for a model with M components, the uncertainty function is given by

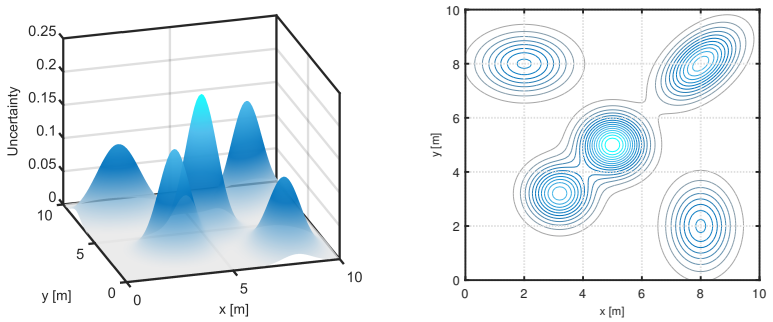
$$h(\mathbf{p}) = \sum_{i=1}^{M} w_i \, \mathcal{N}(\mathbf{p}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{4.1}$$

where each component is a two-dimensional Gaussian distribution defined by

$$\mathcal{N}(\mathbf{p}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \triangleq \frac{1}{\sqrt{4\pi^2 |\boldsymbol{\Sigma}_i|}} \exp\left\{ -\frac{1}{2} (\mathbf{p} - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1} (\mathbf{p} - \boldsymbol{\mu}_i) \right\}. \tag{4.2}$$

The parameters $w_i > 0$, $\boldsymbol{\mu}_i \in \mathbb{R}^2$, and $\boldsymbol{\Sigma}_i \in \mathbb{R}^{2\times 2}$ are, respectively, the weight, the mean vector, and the covariance matrix of the i^{th} Gaussian component.

Additionally, we clarify that the uncertainty map is not originally a PDF, so the volume of uncertainty in the map is not necessarily one. However, it is convenient to assume that this function is normalized, meaning that the *a priori* volume of uncertainty is one, and, therefore, the weights verify $\sum_{i=1}^{M} w_i = 1$. A plausible instance of an uncertainty map is shown in Figure 4.1.



(a) Graph (b) Level curves

Fig. 4.1: Example of an uncertainty function composed of five Gaussian distributions.

4.1.2 Sensing Model

In this work, we assume that the UAV flies at a constant altitude and is equipped with a gimbal camera, which always aims straight down even when the UAV performs pitch or roll maneuvers. Consequently, at each time instant t, we assume that the drone analyzes a given area, $\mathcal{B}_r(\mathbf{p}_c)$, defined as a circle centered in the UAV's horizontal position, $\mathbf{p}_c$, and with a radius of observation r, i.e.,

$$\mathcal{B}_r(\mathbf{p}_c) \triangleq \left\{\mathbf{p} \in \mathbb{R}^2 : \|\mathbf{p} - \mathbf{p}_c\| < r\right\}, \tag{4.3}$$

as illustrated in Figure 4.2. Additionally, the vehicle is assumed to have a perfect quality of exploration, meaning that all points within the observation radius are analyzed perfectly. This assumption implies that, immediately after the UAV analyses a given area, the uncertainty becomes zero for all points inside the area covered by the UAV, and, therefore, there is no reward in revisiting the same region.

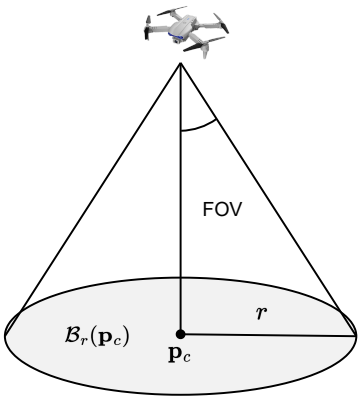


Fig. 4.2: Illustration of sensor Field of View (FOV) and visibility region.

4.1.3 Optimal Control Problem

The trajectory generation problem addressed in this chapter can be defined as finding optimal trajectories that guide the UAV. The trajectories should maximize an objective functional regarding the mission goals while satisfying constraints accounting for their dynamic feasibility. These features prevent the use of flocking algorithms [102], [103] or based on gradient information as in [93]. Consequently, this problem can be formulated as the following optimal control problem

$$\begin{aligned} \underset{\mathbf{x}(.),\mathbf{u}(.)}{\text{maximize}} \quad & J[\mathbf{x}(.),\mathbf{u}(.)] \\ \text{subject to} \quad & \mathbf{x}(0) = \mathbf{x}_0, \\ & \dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t),\mathbf{u}(t)),\ t \in [0,T], \\ & \mathbf{x}(t) \in \mathcal{X},\ t \in [0,T], \\ & \mathbf{u}(t) \in \mathcal{U},\ t \in [0,T], \end{aligned} \tag{4.4}$$

where T denotes the total flight time, the functions $\mathbf{x}(.) : [0,T] \rightarrow \mathbb{R}^{n_x}$ and $\mathbf{u}(.) : [0,T] \rightarrow \mathbb{R}^{n_u}$ denote the state and input of the vehicle's model described by an Ordinary Differential Equation (ODE), and $\mathbf{x}_0$ is the initial value of the state. Moreover, the sets $\mathcal{X}$ and $\mathcal{U}$ constitute the admissible states and inputs for the vehicle, which are derived from limits imposed by vehicle dynamics and the surrounding environment.

Let $\boldsymbol{\varphi} : [0,T] \rightarrow \mathbb{R}^2$ denote the vehicle's trajectory on the horizontal plane, which is related to the state of the vehicle by

$$\boldsymbol{\varphi}(t) = \mathbf{C}\mathbf{x}(t), \ t \in [0, T], \tag{4.5}$$

where $\mathbf{C} \in \mathbb{R}^{2\times n_x}$ is an auxiliary matrix that extracts the horizontal position of the vehicle from the state. The mission objective is to maximize the uncertainty reduction, i.e., the difference between the uncertainty volume in the map before and after the surveillance mission. Therefore, considering the previously mentioned assumptions, the objective functional J is given by

$$J[\boldsymbol{\varphi}] = \int_{C_r[\boldsymbol{\varphi}]} h(\mathbf{p})\, d\mathbf{p}, \tag{4.6}$$

where the set $C_r[\boldsymbol{\varphi}]$ is defined as the union of all circles of observation along the trajectory of the vehicle,

$$C_r[\boldsymbol{\varphi}] \triangleq \bigcup_{t=0}^{T} \mathcal{B}_r(\boldsymbol{\varphi}(t)), \tag{4.7}$$

as illustrated in Figure 4.3. The usefulness of the set $C_r[\boldsymbol{\varphi}]$ arises from the fact that each position is only taken into account once to increase the uncertainty integration since we are assuming that the UAV has a perfect quality of exploration.

(a) 3D illustration (b) Top view

Fig. 4.3: Illustration of the set $C_r[\boldsymbol{\varphi}]$.

Solving the problem in (4.4) is very complex since the objective functional J, as defined in (4.6), does not have a closed-form expression. Therefore, a relaxed formulation needs to be considered. In addition, to make the problem computationally tractable, it also needs to be discretized. However, even after relaxing and discretizing the problem, solving the problem globally for a relatively large time horizon T is computationally challenging. Consequently, we consider a local approach based on Model Predictive Control (MPC) to approximate the solutions of (4.4) while adding the possibility for feedback to the control law.

4.2 Model-Predictive Approach

In order to address the problem defined in the previous section, we follow an MPC-based approach. MPC involves the solution of an open-loop discrete-time optimal control problem at each sampling time k. Each of these optimizations results in a sequence of future optimal control actions and a sequence of corresponding future states. The first control action in the sequence is applied to the plant, and then the optimization is solved again at the next sampling time.

More specifically, at every discrete-time instant k, for a given initial state $\mathbf{x}_k$ of the system, the control policy is defined by solving a discrete-time optimal control problem of the form

$$\begin{aligned} \underset{\hat{\mathbf{X}}_k, \hat{\mathbf{U}}_k}{\text{maximize}} \quad & J_k(\hat{\mathbf{X}}_k, \hat{\mathbf{U}}_k) \\ \text{subject to} \quad & \hat{\mathbf{x}}_{k,0} = \mathbf{x}_k, \\ & \hat{\mathbf{x}}_{k,j+1} = \mathbf{f}(\hat{\mathbf{x}}_{k,j}, \hat{\mathbf{u}}_{k,j}), \ j = 0, \ldots, N-1, \\ & \hat{\mathbf{x}}_{k,j} \in \mathcal{X}, \ j = 0, \ldots, N, \\ & \hat{\mathbf{u}}_{k,j} \in \mathcal{U}, \ j = 0, \ldots, N-1, \end{aligned} \tag{4.8}$$

where N is the horizon length, the sets $\mathcal{X}$ and $\mathcal{U}$ constitute the admissible states and inputs for the vehicle, and the function $\mathbf{f}$ represents a discrete-time version of the vehicle dynamics. The matrices $\hat{\mathbf{X}}_k$ and $\hat{\mathbf{U}}_k$ are the optimization variables and represent the predicted state and control sequences over the time horizon at time instant k, i.e.,

$$\begin{aligned} \hat{\mathbf{X}}_k &\triangleq \left[\hat{\mathbf{x}}_{k,0} \ \hat{\mathbf{x}}_{k,1} \ \ldots \ \hat{\mathbf{x}}_{k,N-1} \ \hat{\mathbf{x}}_{k,N} \right], \\ \hat{\mathbf{U}}_k &\triangleq \left[\hat{\mathbf{u}}_{k,0} \ \hat{\mathbf{u}}_{k,1} \ \ldots \ \hat{\mathbf{u}}_{k,N-1} \right]. \end{aligned} \tag{4.9}$$

The input applied to the system at the discrete-time instant k, $\mathbf{u}_k$, is given by

$$\mathbf{u}_k = \hat{\mathbf{u}}^*_{k,0}, \tag{4.10}$$

where $\hat{\mathbf{u}}^*_{k,0}$ is the first sample of the predicted optimal control sequence at time instant k. The optimization problem in (4.8) may be solved efficiently using available Nonlinear Program (NLP) solvers.

4.2.1 Objective Function

In order to approximate the problem described in the previous section, we define the MPC objective function as a combination of two objectives as

$$J_k(\hat{\mathbf{\Phi}}_k) = \tilde{J}_k(\hat{\mathbf{\Phi}}_k) - \lambda P_k(\hat{\mathbf{\Phi}}_k), \tag{4.11}$$

where $\hat{\mathbf{\Phi}}_k \triangleq [\hat{\boldsymbol{\varphi}}_{k,0}\ \hat{\boldsymbol{\varphi}}_{k,1}\ \dots\ \hat{\boldsymbol{\varphi}}_{k,N}] = \mathbf{C}\hat{\mathbf{X}}_k$ is the predicted discrete-time trajectory of the vehicle at the discrete-time instant k and $\lambda > 0$ is a scaling coefficient.

The first term in (4.11), $\tilde{J}_k$, represents the objective of prioritizing the most uncertain areas and is defined as

$$\tilde{J}_k(\hat{\mathbf{\Phi}}_k) = \sum_{j=0}^{N} \int_{\mathcal{B}_r(\hat{\boldsymbol{\varphi}}_{k,j})} h(\mathbf{p})\, d\mathbf{p}. \tag{4.12}$$

However, this term does not consider the previously covered areas neither the intersections between the areas of observation within the prediction horizon. Consequently, if the objective function was defined by this term alone, the trajectories would converge to an uncertainty maximizer and remain at the maximizer. Therefore, to encode the previously covered areas and the intersections between the areas of observation within the prediction horizon, we add a penalization term $P_k(\hat{\mathbf{\Phi}}_k)$ to the objective function.

The penalization term is constructed by penalizing intersections between the circles of observation along the trajectory of the vehicle. Two types of intersections need to be considered: intersections between the predicted circles and circles corresponding to positions that were already covered, and intersections between the predicted circles of observation. Thus, the penalization term has the form

$$P_k(\hat{\mathbf{\Phi}}_k) = P_k^B(\hat{\mathbf{\Phi}}_k) + P_k^H(\hat{\mathbf{\Phi}}_k), \tag{4.13}$$

where $P_k^B(\hat{\mathbf{\Phi}}_k)$ penalizes intersections between the predicted circles and previously covered circles, and $P_k^H(\hat{\mathbf{\Phi}}_k)$ penalizes intersections within the prediction horizon. Hence, assuming that $p : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}_0^+$ is a function that penalizes the intersection between two circles and that $\boldsymbol{\varphi}_i$ is the actual position of the vehicle at the discrete-time instant i, $P_k^B(\hat{\mathbf{\Phi}}_k)$ is defined by

$$P_k^B(\hat{\mathbf{\Phi}}_k) = \sum_{j=1}^{N} \sum_{i=0}^{k} p(\hat{\boldsymbol{\varphi}}_{k,j}, \boldsymbol{\varphi}_i), \tag{4.14}$$

and $P_k^H(\hat{\mathbf{\Phi}}_k)$ is defined as

$$P_k^H(\hat{\mathbf{\Phi}}_k) = \sum_{j=2}^{N} \sum_{i=1}^{j-1} p(\hat{\boldsymbol{\varphi}}_{k,j}, \hat{\boldsymbol{\varphi}}_{k,i}). \tag{4.15}$$

To conclude the definition of the objective function, it remains to define how the penalty function p is constructed, which is addressed in the following subsection.

Prior to moving forward, it is worth noting that the integrals presented in (4.12) still do not have a closed-form expression. Nevertheless, since the integrals are now computed over circular domains, they may be approximated through numerical methods such as quadrature rules [104] or simply by discretizing the area of observation using a grid. However, we will typically consider examples where the radius of observation is small when compared to the structure of the uncertainty map, and, therefore, (4.12) may be well approximated by

$$\tilde{J}_k(\hat{\mathbf{\Phi}}_k) \simeq \pi r^2 \sum_{j=0}^{N} h(\hat{\boldsymbol{\varphi}}_{k,j}). \tag{4.16}$$

4.2.2 Penalty Function

A plausible definition for the penalty function p would be the intersection area between two circles, as detailed in Figure 4.4.

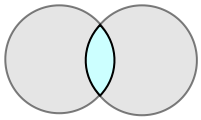


Fig. 4.4: Overlap between two circles.

The area of intersection between two circles centered at the positions $\mathbf{c}_1$ and $\mathbf{c}_2$, both with the same radius r, can be computed analytically by

$$a(\mathbf{c}_1, \mathbf{c}_2) = \begin{cases} 2r^2 \arccos\left(\frac{1}{2r}\|\mathbf{c}_1 - \mathbf{c}_2\|\right) - \|\mathbf{c}_1 - \mathbf{c}_2\| \sqrt{r^2 - \|\mathbf{c}_1 - \mathbf{c}_2\|^2}, & \text{if } \|\mathbf{c}_1 - \mathbf{c}_2\| \leq 2r \\ 0, & \text{if } \|\mathbf{c}_1 - \mathbf{c}_2\| > 2r \end{cases}. \tag{4.17}$$

However, an expression of such complexity would be a computational bottleneck. In addition, the function in (4.17) is piecewise defined, posing additional implementa-

tion difficulties. For instance, the logic condition would have to be addressed through the Big-M notation from YALMIP [105], which serves to convert the logic condition into a set of constraints using auxiliary binary variables and logic constraints.

Nevertheless, it is not necessary to precisely calculate the overlap area between two circles to penalize the intersection between them. Such penalization might be achieved by constructing a function that simply penalizes the condition of existing intersection. Consequently, we design the penalty function by applying an exponential penalty to the violation of the condition $\|\mathbf{c}_1 - \mathbf{c}_2\| > 2r$ as

$$p(\mathbf{c}_1, \mathbf{c}_2) = \exp\left\{\gamma\left((2r)^2 - \|\mathbf{c}_1 - \mathbf{c}_2\|^2\right)\right\} - 1, \tag{4.18}$$

where $\gamma > 0$ is a parameter that can be tuned. Additionally, the subtraction of 1 is included so that the function has a value of zero when $\|\mathbf{c}_1 - \mathbf{c}_2\| = 2r$, but it has no effect on the optimization since it is a constant term. Figure 4.5 illustrates the evolution of the penalization as a function of the distance between the centers of the two circles.

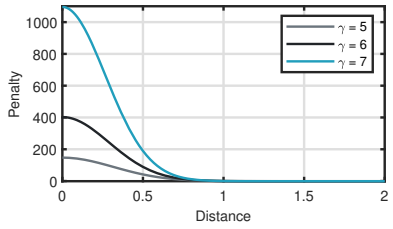


Fig. 4.5: Illustration of the penalty function for some values of γ while considering $r = 0.5$.

4.2.3 Computational Complexity

From a computational standpoint, it is essential to assess the complexity of the proposed algorithm. Besides the inherent complexity of the problem, determined by the structure of the uncertainty map and the imposed restrictions, it is crucial to examine the number of terms comprising the objective function, which directly impacts the number of evaluations that the solver must carry out. In particular, it is worth noting that the number of terms comprising $\tilde{J}_k$ and P_k^H is determined by the prediction horizon length. More specifically, the number of terms in $\tilde{J}_k$ increases linearly with the horizon length, while P_k^H comprises $N(N-1)/2$ terms and, therefore, grows quadratically with the horizon.

Besides the quadratic growth of P_k^H as the horizon length increases, a significant computational burden arises from P_k^B. At each time instant k, the number of terms in

P_k^B increases by N, meaning that P_k^B grows linearly with the flight time assigned for the surveillance mission. One apparent solution could involve defining a maximum backward horizon length N_B, thereby limiting P_k^B to a maximum number of terms. Consequently, in such a case, P_k^B would be given by

$$P_k^B(\hat{\mathbf{\Phi}}_k) = \sum_{j=1}^{N} \sum_{i=k-N_B+1}^{k} p(\hat{\boldsymbol{\varphi}}_{k,j}, \boldsymbol{\varphi}_i). \tag{4.19}$$

Nevertheless, if the backward time horizon is not sufficiently long, the vehicle would possibly revisit previously covered areas. Therefore, a better future approach revolves around developing a subroutine that can progressively reduce the number of components in the penalization term while retaining the information about all the previously explored regions.

Additionally, it is essential to clarify that despite the notion that the objective function grows at each time step, the optimization solvers are constructed by allocating the necessary resources for the entire mission duration. This decision follows from the substantial additional overhead that there would be in building a solver at each time instant k. Hence, the number of terms in the objective function is actually constant throughout the whole mission, with the terms regarding future time steps in P_k^B being attributed a null weight. As a result, despite potential fluctuations introduced by the problem, the computational times are expected to remain approximately constant throughout the surveillance mission.

4.2.4 Evaluation Metric

It is necessary to establish an overall metric to evaluate the performance of the algorithm and perform comparisons. In this context, a reliable method of assessing the quality of the generated trajectories is computing the time evolution of the volume of uncertainty covered by the vehicle. By disregarding the coverage between sampling times, this metric can be approximated as

$$H_k(\mathbf{\Phi}_k) = \int_{\bigcup_{i=0}^{k} \mathcal{B}_r(\boldsymbol{\varphi}_i)} h(\mathbf{p})\, d\mathbf{p}, \tag{4.20}$$

where $\mathbf{\Phi}_k \triangleq [\boldsymbol{\varphi}_0 \ \boldsymbol{\varphi}_1 \ \dots \ \boldsymbol{\varphi}_k]$ is the discrete-time trajectory of the vehicle until time instant k. The numerical approximation of (4.20) is accomplished by discretizing the map into a grid.

4.3 Quadrotor Motion Control

We are particularly interested in multirotor aerial vehicles because of their agility, hovering performance, low cost, and production ease. Moreover, in this project, a quadrotor is available for performing experimental tests of the proposed MPC algorithm. Therefore, this section presents the control architecture used to implement the proposed MPC algorithm on a quadrotor aerial vehicle, as well as the dynamic model of the quadrotor.

4.3.1 Control Architecture

We consider a dual-layer structure of motion control, as illustrated in Figure 4.6. The proposed MPC algorithm serves as a high-level controller (trajectory planner) that generates high-level references for the UAV. The inner-loop controller (trajectory tracker) directly applies control inputs to the vehicle to accurately track the references provided by the MPC algorithm. For the purpose of efficiency, the MPC algorithm considers a simplified model of the vehicle, while the lower-level controller takes into account the full dynamic model of the quadrotor.

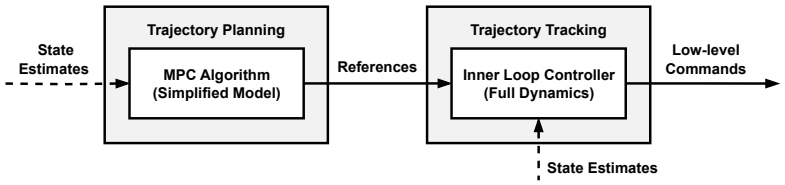


Fig. 4.6: Full motion control scheme of the UAV.

4.3.2 Full Dynamics

For completeness, we start by presenting the full nonlinear dynamics of a quadrotor. The nonlinear dynamics of the quadrotor are described in the body $\{B\}$ and inertial $\{I\}$ frames depicted in Figure 4.7, while assuming that the origin of $\{B\}$ is coincident with the center of mass of the quadrotor.

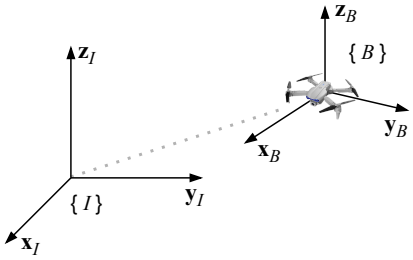


Fig. 4.7: Quadrotor reference frames.

Let $\mathbf{p} \in \mathbb{R}^3$ denote the position of the quadrotor's center of mass in the inertial frame. Let $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^\top$ describe the orientation of the of the body frame with respect to the inertial frame, where ϕ, θ and ψ are the roll, pitch and yaw angles, respectively. Let $\mathbf{v}$ denote the linear velocity of $\{B\}$ with respect to $\{I\}$ expressed in $\{I\}$. Let $\boldsymbol{\omega} = [p\ q\ r]^\top$ denote the angular velocity of $\{B\}$ with respect to $\{I\}$, this time expressed in $\{B\}$. Finally, let m be the mass of the rigid object, $\mathbf{I} \in \mathbb{R}^{3\times3}$ the inertia matrix expressed in $\{B\}$, and g the gravitational acceleration. The quadrotor equations of motion, based on the Newton-Euler formalism [106], are given by

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v}, \\ m\dot{\mathbf{v}} &= -mg\mathbf{e}_3 + {}^I\mathbf{R}_B(\boldsymbol{\eta})\, F_T\mathbf{e}_3, \\ \dot{\boldsymbol{\eta}} &= \mathbf{T}(\boldsymbol{\eta})\,\boldsymbol{\omega}, \\ \mathbf{I}\dot{\boldsymbol{\omega}} &= -\boldsymbol{\omega}\times\mathbf{I}\boldsymbol{\omega} + \boldsymbol{\tau}, \end{aligned} \tag{4.21}$$

where F_T is the net thrust and $\boldsymbol{\tau} = [\tau_\phi\ \tau_\theta\ \tau_\psi]^\top$ is the vector of moments applied to the UAV described in $\{B\}$. Additionally, ${}^I\mathbf{R}_B(\boldsymbol{\eta}) \in SO(3)$ is the rotation matrix from $\{B\}$ to $\{I\}$ and $\mathbf{T}(\boldsymbol{\eta}) \in \mathbb{R}^{3\times3}$ is a matrix that converts the angular velocity to angle rates.

Assuming that the Euler angles follow the sequence of rotation *Z-Y-X* that is described in [107], ${}^I\mathbf{R}_B(\boldsymbol{\eta})$ is given by

$${}^I\mathbf{R}_B(\boldsymbol{\eta}) = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}, \tag{4.22}$$

and $\mathbf{T}(\boldsymbol{\eta})$ is

$$\mathbf{T}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{4.23}$$

The steady-state thrust and the yaw moment generated by rotor i are modeled as

$$\begin{aligned} F_{T_i} &= K_i\Omega_i^2, \\ \tau_{\psi_i} &= C_i F_{T_i}, \end{aligned} \tag{4.24}$$

where Ω_i is the rotation speed of rotor i, and the constants K_i and C_i may be determined experimentally. The roll and pitch moments, τ_ϕ and τ_θ, result from the generated rotor thrusts and their arrangement relative to the quadrotor's center of mass. Therefore, the net thrust and moments, for a quadrotor with an X-configuration, are computed through

$$\begin{bmatrix} F_T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ L & -L & -L & L \\ -L & -L & L & L \\ C_1 & -C_2 & C_3 & -C_4 \end{bmatrix} \begin{bmatrix} F_{T_1} \\ F_{T_2} \\ F_{T_3} \\ F_{T_4} \end{bmatrix}, \tag{4.25}$$

where L denotes the perpendicular distance of the rotors to the x or y axis of the body frame.

To conclude, we point out that there are additional aerodynamic effects, which would increase the complexity of the model. However, a model with such a level of precision is typically not required.

4.3.3 Simplified Model

At the trajectory planning level, considering that the UAV flies at a constant altitude and ignoring the fast rotational dynamics of the vehicle, the UAV might be modeled as a two-dimensional point-mass system that follows double-integrator dynamics. Thus, the state vector $\mathbf{x}$ is composed of the positions and velocities on the horizontal plane, $\mathbf{x} = [\boldsymbol{\varphi}^\top \ \dot{\boldsymbol{\varphi}}^\top]^\top$, and the control input $\mathbf{u}$ consists of the acceleration on the

horizontal plane. Consequently, at the planning level, the quadrotor dynamics take the linear form

$$\begin{bmatrix} \dot{\boldsymbol{\varphi}} \\ \ddot{\boldsymbol{\varphi}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{2\times2} & \mathbf{I}_{2\times2} \\ \mathbf{0}_{2\times2} & \mathbf{0}_{2\times2} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varphi} \\ \dot{\boldsymbol{\varphi}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{2\times2} \\ \mathbf{I}_{2\times2} \end{bmatrix} \mathbf{u}, \tag{4.26}$$

where $\mathbf{0}_{2\times2}$ denotes a matrix of zeros and $\mathbf{I}_{2\times2}$ denotes the identity matrix, both with dimension 2×2.

The MPC algorithm considers this simplified model of the UAV, which reduces the computational cost of the optimization process, while the inner-loop controller processes the remaining dynamics of the UAV. This mismatch is not critical for obtaining good performance in real conditions as long as the generated trajectories are not extremely aggressive so that the inner-loop dynamics become visible.

4.3.4 Actuation Limits

As any other vehicle, a quadrotor is subject to limitations imposed by its actuators. In this case, the maximum thrust magnitude of the quarotor is limited. At any time, the quadrotor should have a vertical force to balance its weight, i.e., a force of magnitude mg. Then, it must be able to maneuver around this equilibrium. A descent can be achieved by decreasing the vertical force that balances the weight. However, to ascend it must be able to produce a higher thrust on the vertical direction.

In the vertical direction, the quadrotor should have available a thrust force of magnitude $m(g+a_z^{\max})$, where $a_z^{\max} \in \mathbb{R}^+$ represents the maximum acceleration along the vertical axis. Let $F_T^{\max} \in \mathbb{R}^+$ be the maximum thrust magnitude that the quadrotor can produce, and $a_{xy}^{\max} \in \mathbb{R}^+$ be the maximum acceleration on the horizontal plane. The maximum acceleration on the horizontal plane can be computed by

$$a_{xy}^{\max} = \sqrt{(F_T^{\max}/m)^2 - (g + a_z^{\max})^2}. \tag{4.27}$$

The saturation along the vertical direction must first be chosen, to ensure that the multirotor is capable of maintaining its altitude. Then, the horizontal saturation is a result of the choice made about $a_z^{\max}$.

4.3.5 Implementation Details

In practical terms, the proposed motion control scheme is implemented using a PX4 Autopilot [108]. The PX4 Autopilot provides both the inner-loop controller and an Extended Kalman Filter (EKF) to process sensor measurements. In this implementation, the MPC algorithm generates references to be tracked by the inner-loop controller provided by the PX4 Autopilot, while both controllers receive the corresponding state estimates provided by the EKF algorithm. Figure 4.8 illustrates the described implementation.

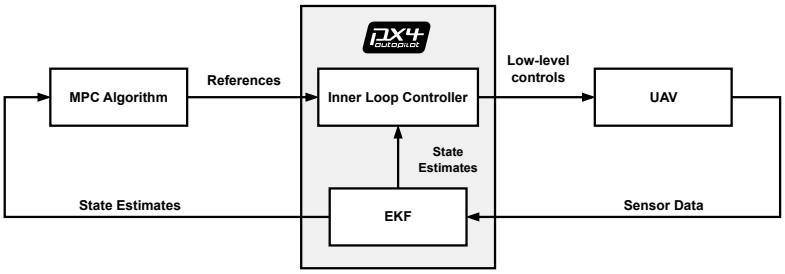


Fig. 4.8: Implementation of the proposed motion control scheme.

As detailed in Figure 4.9, the controller supplied by the PX4 Autopilot follows a standard cascaded architecture with several stages. Each stage is composed of a proportional or Proportional-Integral-Derivative (PID) controller that generates references for the upcoming stage based on references provided by the previous stage. From a general perspective, the PX4 controller consists of two main control loops: position and attitude. The position control loop commands accelerations, which are then converted into attitude and net thrust references. The attitude control loop receives attitude and net thrust references and commands low-level thrust references for the vehicle motors.

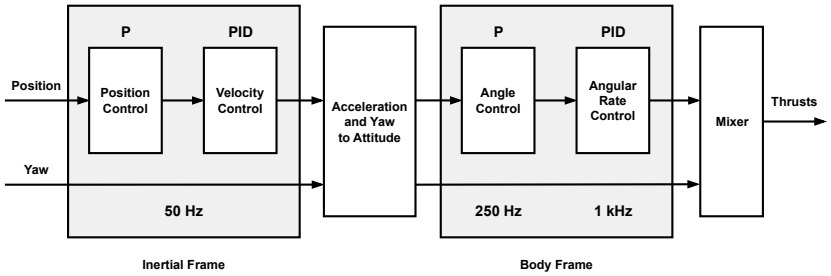


Fig. 4.9: PX4 controller architecture (adapted from PX4 documentation).

With such an architecture, the PX4 controller is able to receive different kinds of references from the MPC algorithm, from high-level references to low-level ones. In our case, we resort to high-level references such as position, velocity, or acceleration references, while keeping the altitude constant. The type of references used may then be adjusted given the experimental results obtained.

4.4 Simulation Results

In this section, the efficacy of the proposed MPC algorithm is assessed through different simulation examples obtained within a MATLAB environment. The objective is to analyze the behavior and performance of the algorithm, as well as the quality of the generated trajectories. We begin by presenting some simulations that illustrate the trajectories that the algorithm is able to produce. Subsequently, we study the influence of some parameters on the algorithm.

4.4.1 Simulation Setup

The goal of this section is to perform an initial analysis of the behavior of the proposed MPC algorithm. Therefore, the simulations presented in this section are performed assuming that the UAV follows ideal double-integrator dynamics. The full nonlinear dynamics of the UAV and the PX4 inner-loop controller are then included in the simulations and experiments of Section 4.5. At each discrete-time instant k, the MPC algorithm is based on the following optimization problem

$$\begin{aligned} \underset{\hat{\mathbf{X}}_k, \hat{\mathbf{U}}_k}{\text{maximize}} \quad & J_k(\hat{\mathbf{X}}_k, \hat{\mathbf{U}}_k) \\ \text{subject to} \quad & \hat{\mathbf{x}}_{k,0} = \mathbf{x}_k, \\ & \hat{\mathbf{x}}_{k,j+1} = \mathbf{A}\hat{\mathbf{x}}_{k,j} + \mathbf{B}\hat{\mathbf{u}}_{k,j}, \ \ j = 0, \ldots, N-1, \\ & \left\| \mathbf{C}'\hat{\mathbf{x}}_{k,j} \right\| \leq v_{xy}^{\max}, \ \ j = 0, \ldots, N, \\ & \left\| \hat{\mathbf{u}}_{k,j} \right\| \leq a_{xy}^{\max}, \ \ j = 0, \ldots, N-1, \end{aligned} \tag{4.28}$$

where the objective function J_k is obtained as described in Section 4.2, and the matrices $\mathbf{A}$ and $\mathbf{B}$, corresponding to the discrete-time double-integrator dynamics (zero-order hold), are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{2\times 2} & T_s\,\mathbf{I}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} T_s^2/2\,\mathbf{I}_{2\times 2} \\ T_s\,\mathbf{I}_{2\times 2} \end{bmatrix}. \tag{4.29}$$

The auxiliary matrix $\mathbf{C}'$, given by

$$\mathbf{C}' = \begin{bmatrix} \mathbf{0}_{2\times 2} & \mathbf{I}_{2\times 2} \end{bmatrix}, \tag{4.30}$$

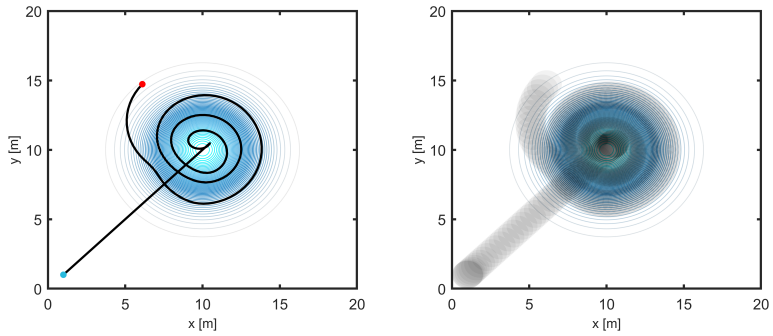
extracts the velocity from the state. The parameters $v_{xy}^{\max}$ and $a_{xy}^{\max}$ are, respectively, the maximum velocity and acceleration that the vehicle may achieve on the horizontal plane.

The simulation results presented in this section were obtained in MATLAB [109] using the CasADi [110] optimization modeling toolbox, along with the IPOPT [111] numerical solver. At each sampling time, the solution obtained at the previous step is used to set the initial guess for the current step by performing the shifting warm-start method. All computations were executed on a single desktop computer equipped with an Intel Core i7-6700K @ 4.00 GHz processor and 32.00 GB of RAM.

In the following examples, the drone starts at the position $\mathbf{p} = [1\ 1]^\top$ with no initial velocity, and the radius of observation is assumed to be $r = 1$ m. The sampling period is $T_s = 0.1$ s, the horizon length is $N = 15$, and the vehicle has a maximum velocity of 4 m/s and a maximum acceleration of 4 m/s^2.

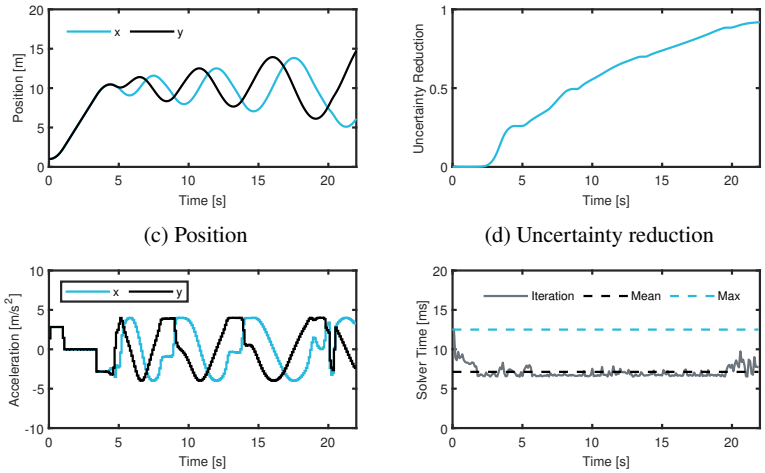
4.4.2 Example 1

We begin by presenting a simple simulation where the uncertainty function is composed of only one Gaussian component with a circular shape. The simulation results are shown in Figure 4.10.



(a) Trajectory (b) Sensor footprint

Fig. 4.10: Simple simulation with one Gaussian component with a circular shape.



(c) Position (d) Uncertainty reduction

(e) Acceleration (f) Solver times (100 runs mean)

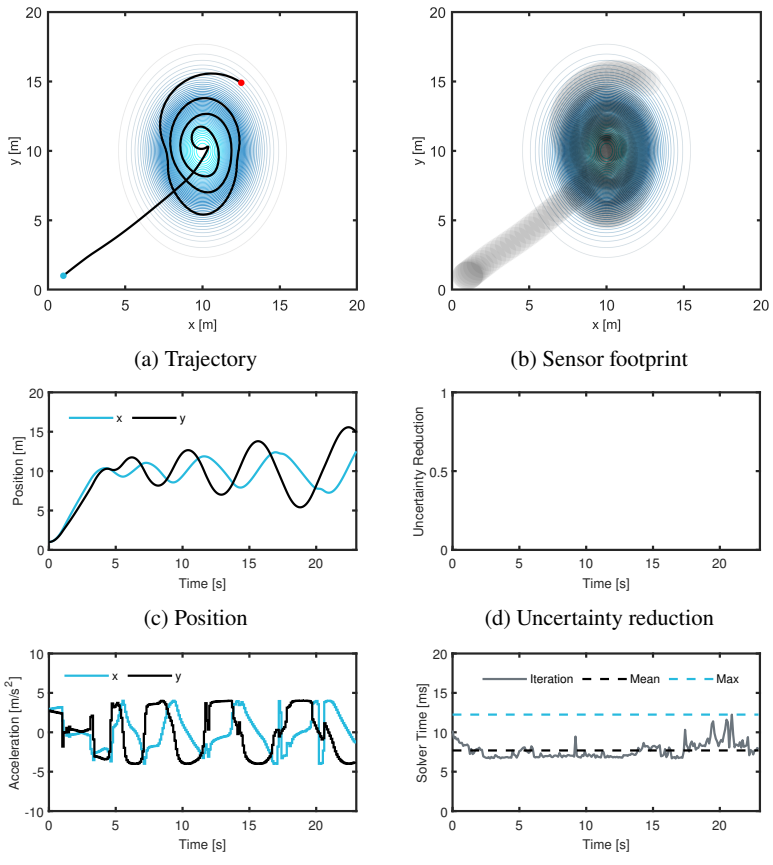
Fig. 4.10: Simple simulation with one Gaussian component with a circular shape.

As shown in Figure 4.10 (a), initially the vehicle moves towards the maximum of the Gaussian component. Subsequently, as a result of the penalizations applied by the algorithm, the vehicle goes to wider areas by executing a spiral curve. The evolution of the x and y components of the position and acceleration of the vehicle is depicted in Figures 4.10 (c) and 4.10 (e). Moreover, Figure 4.10 (b) illustrates the sensor footprint of the UAV, and Figure 4.10 (d) shows the accumulation of the uncertainty volume covered by the vehicle. In addition, we draw attention to Figure

4.10 (f), which presents the mean solver times acquired through 100 simulations, with each iteration taking approximately 7 ms on average.

4.4.3 Example 2

We introduce another simple simulation, in which the uncertainty function consists of a Gaussian component with an elliptical shape. As depicted in Figure 4.11 (a), the trajectory adjusts itself to the elliptical shape of the Gaussian component. Moreover, as shown in Figures 4.11 (d) and 4.11 (f), the resulting uncertainty reduction profile and the mean solver times are similar to those from the previous example.



(a) Trajectory (b) Sensor footprint

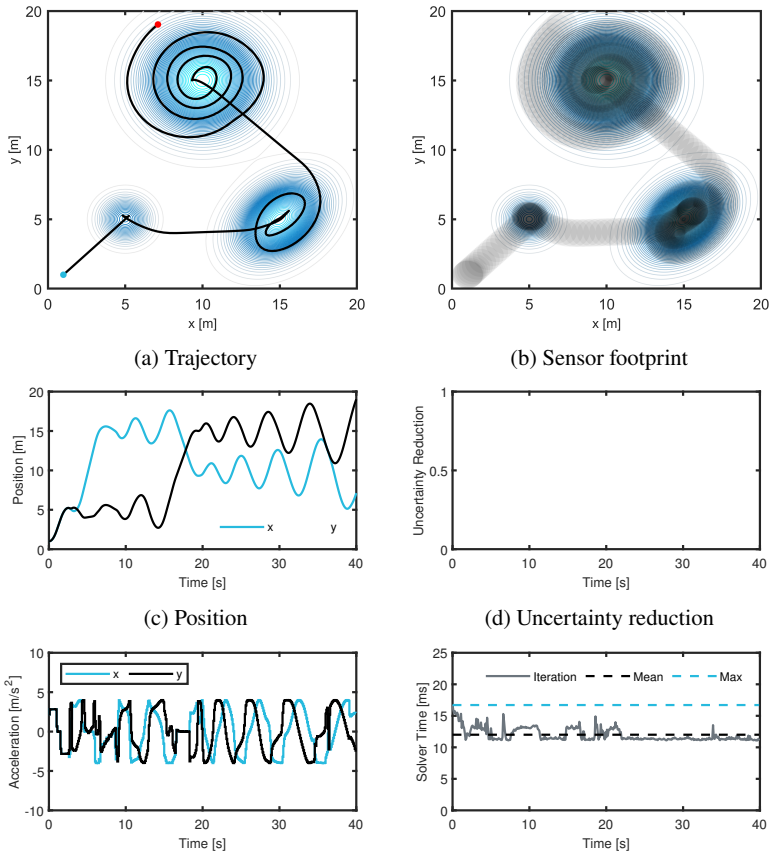
(c) Position (d) Uncertainty reduction

(e) Acceleration (f) Solver times (100 runs mean)

Fig. 4.11: Simple simulation with one Gaussian component with an elliptical shape.

4.4.4 Example 3

Now we introduce a more complex example where the uncertainty map comprises three Gaussian components, with the simulation results displayed in Figure 4.12. As depicted in Figure 4.12 (a), the drone analyzes each component individually. In particular, it is worth noting that the components with means at the positions $\mathbf{p} = [15\ 5]^\top$ and $\mathbf{p} = [10\ 15]^\top$ are similar to those from the previous examples, and the flight paths observed when the drone analyzes such components are also similar to the previous ones. However, the third component located at $\mathbf{p} = [5\ 5]^\top$ has a smaller variance when compared to the observation radius of the UAV. Consequently, when the drone analyzes this component, it simply remains at the maximum of the component. Additionally, it should be noticed that the solver times are slightly higher in this example, with each iteration averaging approximately 12 ms.



(a) Trajectory (b) Sensor footprint

(c) Position (d) Uncertainty reduction

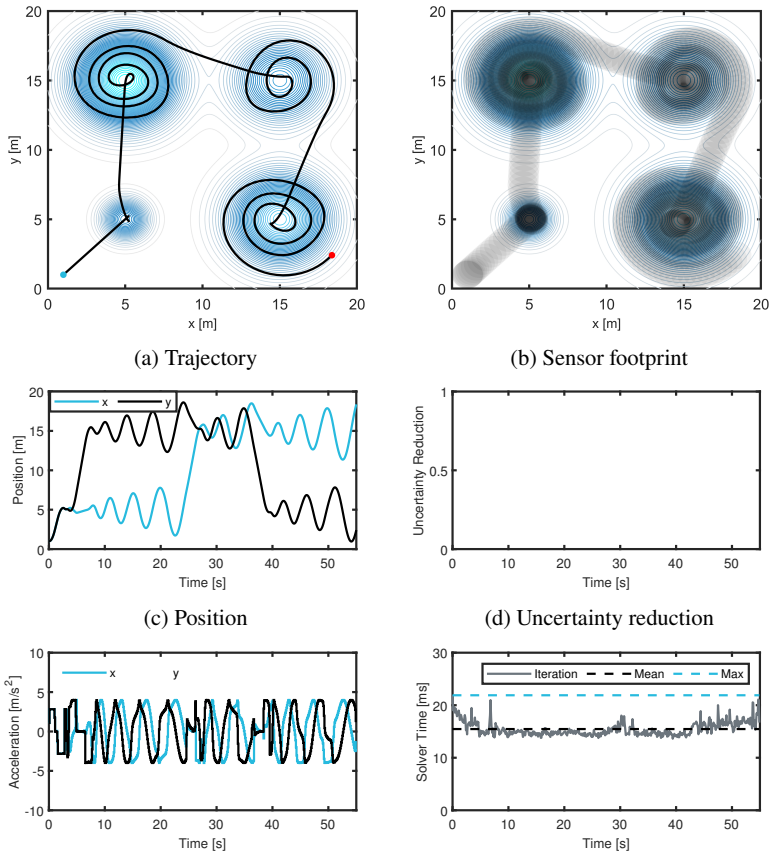
(e) Acceleration (Input) (f) Solver times (100 runs mean)

Fig. 4.12: Example where the uncertainty map comprises three Gaussian components.

4.4.5 Example 4

We present another example in which the uncertainty map is now composed of four radially-symmetric Gaussian components, with the simulation results displayed in Figure 4.13. As it can be observed in Figure 4.13 (a), the component with mean at $\mathbf{p} = [5\ 5]^\top$ is similar to the one from the previous example, and the vehicle exhibits a similar behavior when analyzing this specific component. The remaining components with means at $\mathbf{p} = [5\ 15]^\top$, $\mathbf{p} = [15\ 15]^\top$, and $\mathbf{p} = [15\ 5]^\top$ all have similar covariance matrices but different associated weights. By observing Figures 4.13 (a) and 4.13 (b), one can notice that, as the weights of the components increase,

the spiral curves become more tightly concentrated, and there is a greater overlap of the vehicle's observation circles. In this example, each solver iteration averages approximately 16 ms, as shown in Figure 4.13 (f).



(a) Trajectory (b) Sensor footprint

(c) Position (d) Uncertainty reduction

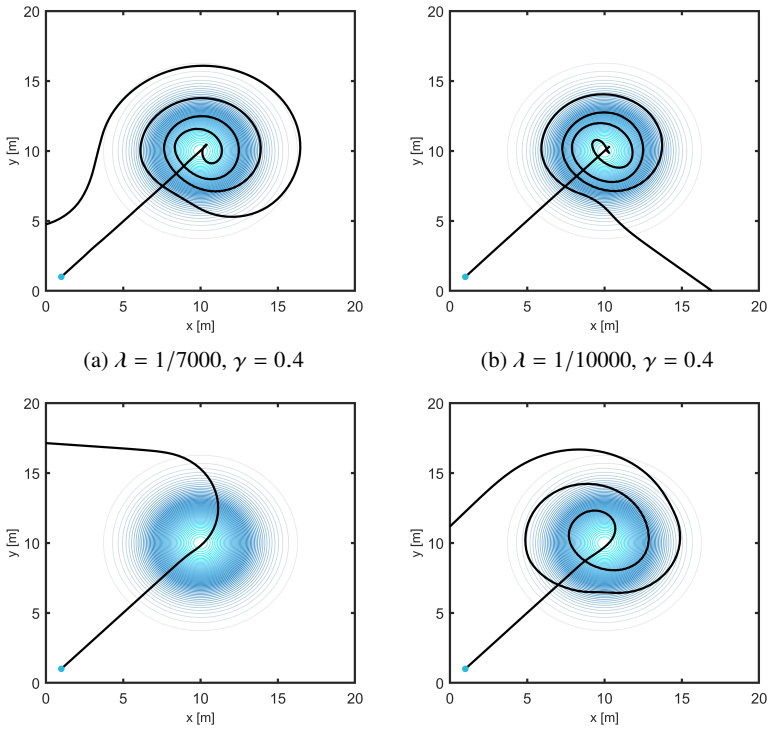
(e) Acceleration (Input) (f) Solver times (100 runs mean)

Fig. 4.13: Example where the uncertainty map comprises four radially-symmetric components.

4.4.6 Effect of the Weights

Given that the objective function of the proposed algorithm relies on the exponent γ to penalize intersections and the scaling coefficient λ, it is essential to assess how these two parameters influence the algorithm. In this context, we consider the

conditions of the initial example, where the uncertainty map consists of a single radially-symmetric Gaussian component, and we manipulate the weights λ and γ. Figure 4.14 displays the resulting trajectories for four distinct combinations of values of λ and γ.



(a) $\lambda = 1/7000$, $\gamma = 0.4$ (b) $\lambda = 1/10000$, $\gamma = 0.4$

(c) $\lambda = 1/7000$, $\gamma = 1$ (d) $\lambda = 1/7000$, $\gamma = 0.8$

Fig. 4.14: Trajectories obtained for different values of λ and γ.

As λ decreases in value, less emphasis is placed on the penalization term. Consequently, the trajectories are expected to become more tightly concentrated, resulting in a greater overlap of the vehicle's observation circles. This effect is evident in the examples depicted in Figures 4.14 (a) and 4.14 (b), and it becomes more pronounced when examining Figure 4.15, which illustrates the evolution of the uncertainty volume covered by the vehicle for the various scenarios presented in Figure 4.14. As shown in Figure 4.15, in the case of Figure 4.14 (b), the vehicle remains closer to the peak of the Gaussian component for an extended duration compared to Figure 4.14 (a), resulting in a slower initial convergence. However, note that at $t = 30$ s, both trajectories exhibit a similar coverage.

A similar impact can be anticipated when examining the variation of γ. As the value of γ increases, the penalizations become more pronounced, leading to the expectation of a reduced overlap in the resulting trajectories. This effect is clearly observable in the examples presented in Figures 4.14 (a) and 4.14 (d). In particular, as shown in Figure 4.15, note that the trajectory in Figure 4.14 (d) initially exhibits a quicker convergence when compared to the trajectory in Figure 4.14 (a). However, at the final simulation instant $t = 30$ s, the uncertainty volume covered by the trajectory in Figure 4.14 (a) is greater than that achieved by the trajectory in Figure 4.14 (d). Additionally, Figure 4.14 (c) illustrates a more extreme case where γ is sufficiently high to prevent the vehicle from executing a spiral curve.

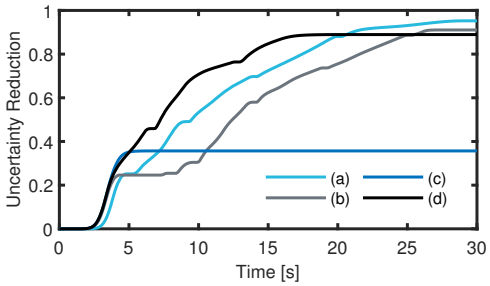


Fig. 4.15: Uncertainty volume accumulation for the different values of λ and γ.

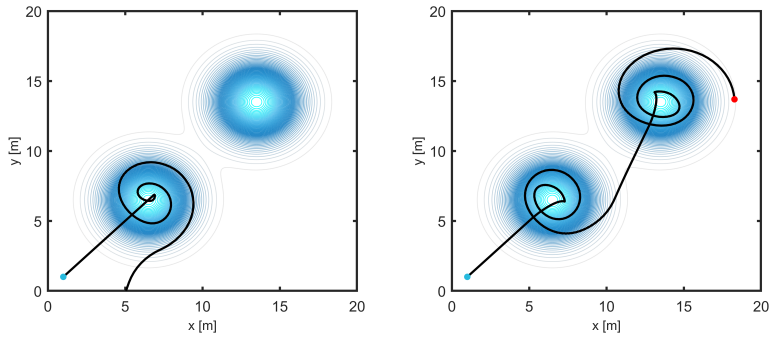
In light of the previous discussion, it is evident that there exists some parameter tuning associated with the proposed algorithm. Nevertheless, it should be acknowledged that the algorithm has the potential to be extended and generalized through the incorporation of variable weights. For instance, one could consider assigning higher penalizations in regions where the uncertainty function has higher values, and lower penalizations in regions where the uncertainty is lower. Moreover, one could employ decaying weights in the term $\tilde{J}_k$ of the objective function to prioritize earlier prediction instants, potentially resulting in a faster convergence. Such variations of the algorithm could be easily incorporated, and a more exhaustive analysis could be performed. However, the decision to implement these variations is left as a user choice and may be a subject of consideration in future research.

4.4.7 Effect of the Horizon

It is also important to evaluate how the performance of the proposed MPC algorithm is impacted by varying the length of the prediction horizon. In this context, we begin our analysis by considering an uncertainty map comprising two Gaussian

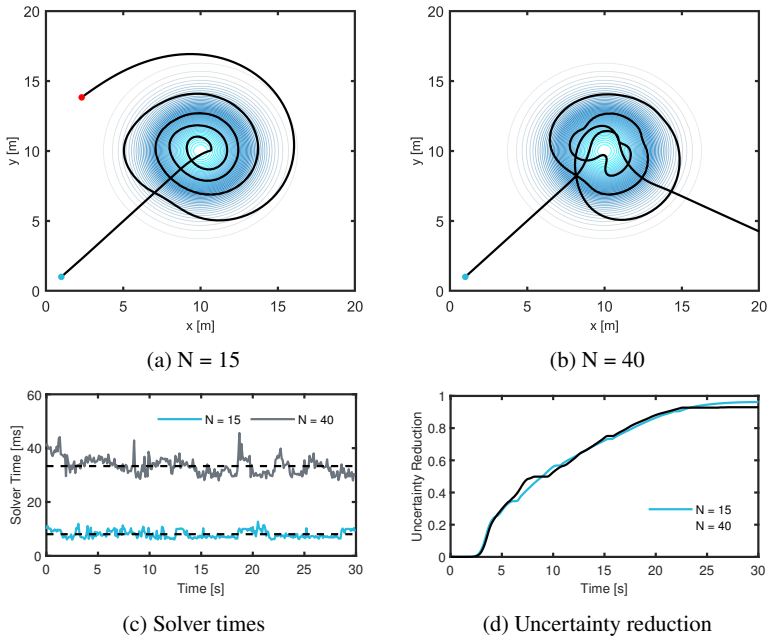
components, with Figure 4.16 depicting the generated trajectories for two different prediction horizon lengths. As depicted in Figure 4.16 (a), for a horizon length of $N = 5$, the vehicle's predictive ability falls short and it is not able to predict the second Gaussian component. In contrast, when a longer horizon length is employed, as illustrated in Figure 4.16 (b), the vehicle is able to predict the second Gaussian component, resulting in a trajectory with a higher coverage.

However, an extended horizon does not necessarily result in higher-quality trajectories, as illustrated in Figure 4.17. Specifically, as highlighted in Figure 4.17 (d), it can be noticed that the trajectories depicted in Figures 4.17 (a) and 4.17 (b) exhibit similar coverage profiles, and, in fact, the trajectory from Figure 4.17 (b) achieves a lower final coverage than the trajectory in Figure 4.17 (a). Furthermore, the trajectory from Figure 4.17 (a) exhibits a smoother profile than that in Figure 4.17 (b). A longer prediction horizon also increases the computational load, as shown in Figure 4.17 (c). In the context of this simple simulation with a duration of 30 seconds and featuring a single Gaussian component, for a horizon of $N = 40$, each solver iteration already takes an average of approximately 35 ms, while for a horizon of $N = 15$ the average iteration time is about 8 ms.



(a) N = 5 (b) N = 15

Fig. 4.16: Trajectories obtained for two distinct prediction horizon lengths.



(a) N = 15 (b) N = 40

(c) Solver times (d) Uncertainty reduction

Fig. 4.17: Results obtained for different prediction horizon lengths.

4.5 Experimental Validation

In this section, the efficacy of the proposed MPC algorithm is assessed through simulations in a higher-fidelity simulation software and by conducting actual experiments. We begin by providing a concise overview of the software architecture employed for simulating and carrying out tests in the physical drone. Subsequently, we showcase the results achieved from these evaluations.

4.5.1 Software Architecture

The software used to perform simulations and conduct actual experiments in the drone follows from the previous work done by Oliveira et al. [112] and Jacinto [113]. The employed software architecture is illustrated in Figure 4.18. The operating system consists of the Ubuntu 18.04 LTS version along with the melodic variant of the Robot Operating System (ROS). In the remainder of this section, we briefly describe each block shown in Figure 4.18.

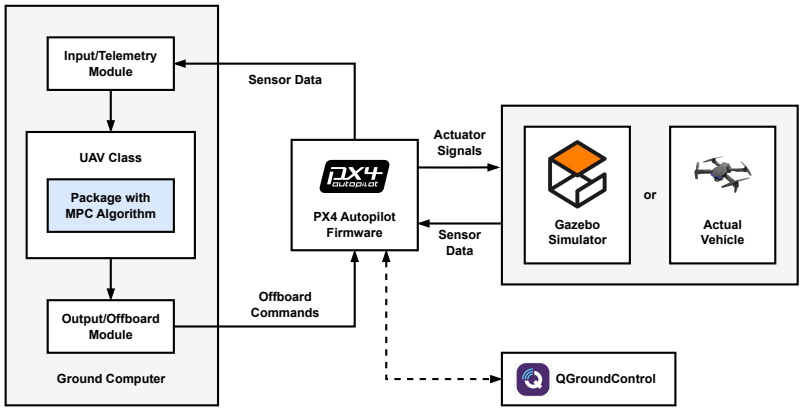


Fig. 4.18: Simplified scheme of the employed software architecture.

4.5.2 Gazebo Simulator

The simulations conducted in this section are performed using the Gazebo simulator [114]. Gazebo stands out as a high-fidelity robotics simulator featuring a physics engine that accurately models the dynamic and kinematic characteristics of vehicles. Moreover, it offers the flexibility to incorporate various sensors and actuators via plugins. In essence, Gazebo simulates the motion of vehicles and the resulting sensor data, based on the inputs provided to the vehicles. One key advantage of Gazebo lies in its seamless integration with the ROS middleware. Such integration facilitates a modular approach to structure the entire system by defining each entity as a separate package and enabling communication via the publication and subscription of messages to topics and services.

4.5.3 PX4 Autopilot

PX4 [108] is an autopilot firmware that can operate within a vehicle, offering two distinct modes: Hardware In The Loop (HITL) and Software In The Loop (SITL). Its primary function is to act as an intermediary between the offboard modules and vehicle actuators. It provides essential functions such as sensor data acquisition, actuators control, estimators, safety features, and communication with external systems. As shown in Figure 4.18, the PX4 Autopilot provides raw sensor and estimator data and accepts commands to control the vehicle. The PX4 Autopilot software also includes a collection of available quadrotor models, in particular the Iris quadrotor [115] that is used in the simulations.

4.5.4 Ground Computer Stack

The Ground Computer stack, as illustrated in Figure 4.18, enables users to implement control algorithms and execute missions using the vehicles. The UAV class represents a UAV and is accessible thanks to the input and output modules developed by Oliveira et al. [112] and adapted by Jacinto [113]. The input module, responsible for subscribing to data published by the PX4 Autopilot, makes this data available to the user in a standardized manner. The output module consists of the methods that allow the user to send offboard commands and control references to the PX4 Autopilot. The proposed MPC algorithm is implemented within this framework using the C++ CasADi Application Programming Interface (API). Additionally, QGroundControl is an application that serves as a graphical user interface offering comprehensive flight control and vehicle configuration capabilities.

4.5.5 Launching Simulations

In order to conduct simulations, a set of launch files is employed to initiate all the fundamental services. The launch files facilitate the initiation of services like Gazebo within the ROS environment, thereby granting further access to variables pertaining to the simulation and the state of the world. The hierarchical organization of the launch files is illustrated in Figure 4.19. Firstly, the control algorithm to execute is developed within a ROS package, and then the drone_sim.launch file is updated to call this package. Subsequently, the simulation may be initiated using the command "roslaunch drone_sim_bringup simulator_bringup.launch."

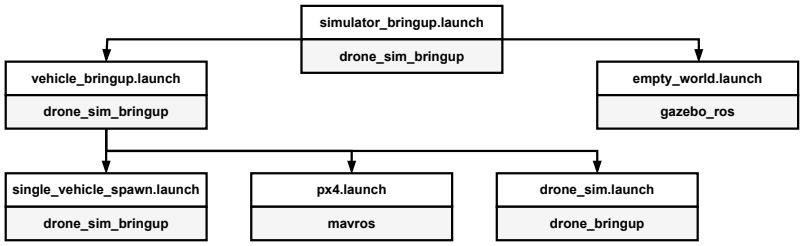


Fig. 4.19: Hierarchical organization of the launch files.

4.5.6 Experimental Setup

In the upcoming subsections, we present the results obtained from different Gazebo simulations and the corresponding experimental tests performed in an outdoor environment (field trials). The Gazebo simulations were executed using the Iris quadrotor [115], available through the PX4 Autopilot SITL plugin. Meanwhile, the field trials were conducted using the M690B drone from a joint effort between FirePuma and Capture projects [116]. Additionally, to ensure validation prior to the experimental tests, we also conducted simulations using the Typhoon H480 hexacopter model, which shares more analogous characteristics with the M690B drone. However, the results were equivalent to those obtained using the Iris quadrotor. Figure 4.20 depicts the Iris and the M690B quadrotors used for the experimental validation.

(a) Iris quadrotor (Gazebo) (b) M690B (field trials)

Fig. 4.20: Quadrotors used in the Gazebo simulations and field trials.

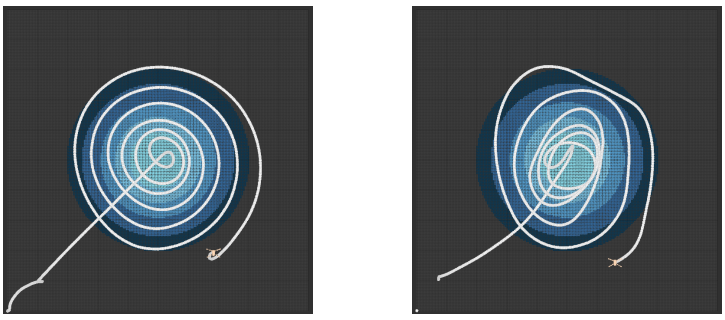
Concerning the Gazebo simulations, our initial approaches involved providing acceleration and, subsequently, velocity references to the PX4 controller. Despite our efforts, these approaches posed challenges in achieving smooth and stable trajectories consistent with those obtained in MATLAB. Nonetheless, when commanding a complete trajectory generated offline under identical vehicle constraints, it yielded the anticipated outcomes, enabling the vehicle to follow the trajectories with minimal error.

Despite the lack of significant advantages in running the algorithm in real-time in this particular scenario, there is a natural desire to enable the real-time execution of the algorithm to accommodate dynamic alterations in the future, like time-varying maps or obstacle avoidance. To enable the real-time execution of the algorithm and overcome the bad results obtained using lower-level references, we opted for a more conservative approach. The approach consists in sending a given slice of the predicted optimal sequence of position waypoints to the PX4 controller. With such an approach, the penalizations of the objective function are still updated at each sampling time k, but the optimization problem is only solved after the application of each sequence of waypoints. This method ultimately produced results similar to those obtained by commanding a complete trajectory generated *a priori*.

In the examples presented in the following subsections, the drone starts at the position $\mathbf{p} = [1\ 1]^\top$ with no initial velocity and the radius of observation is assumed to be $r = 0.5$ m. In the Gazebo simulations, the MPC operates with a sampling period of $T_s = 0.1$ s, a prediction horizon length of $N = 20$, and the first 5 predicted optimal waypoints are commanded to the PX4 controller. Consequently, the optimization problem is only solved from 0.5 s to 0.5 s. The MPC is warm-started using the shifting method but by shifting 5 steps. Moreover, the MPC considers a maximum velocity of 2 m/s and a maximum acceleration of 2 m/s^2 for the vehicle. Regarding the field trials, due to difficulties faced when attempting to execute the algorithm onboard, the experimental tests were carried out by instructing waypoints generated *a priori* under the same conditions.

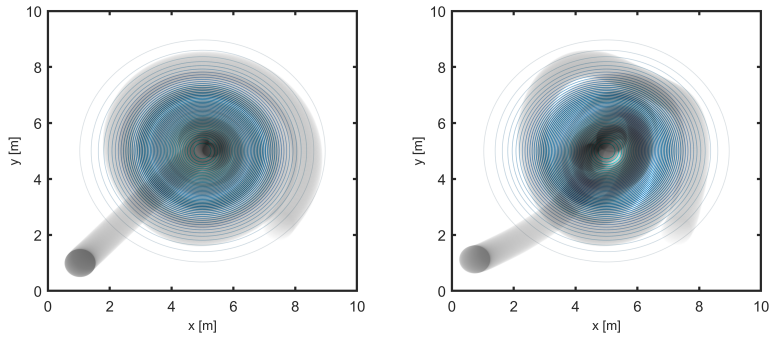
4.5.7 Experiment 1

We begin by considering an example where the uncertainty map is composed of one radially-symmetric Gaussian component, with the corresponding results being displayed in Figure 4.21.



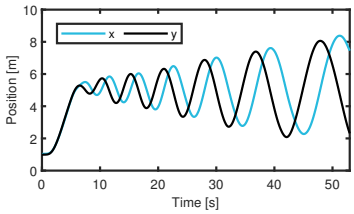
(a) Gazebo - Trajectory (RViz) (b) Field trial - Trajectory (RViz)

Fig. 4.21: Gazebo and field trial results for a simple uncertainty map.

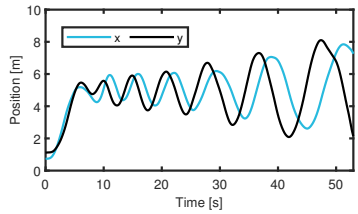


(c) Gazebo - Sensor footprint

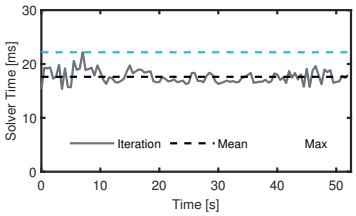
(d) Field trial - Sensor footprint



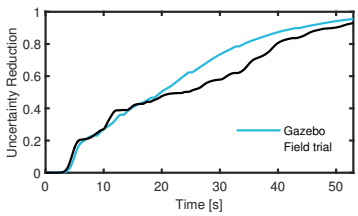
(e) Gazebo - Position



(f) Field trial - Position



(g) Gazebo - Solver times



(h) Uncertainty reduction

Fig. 4.21: Gazebo and field trial results for a simple uncertainty map.

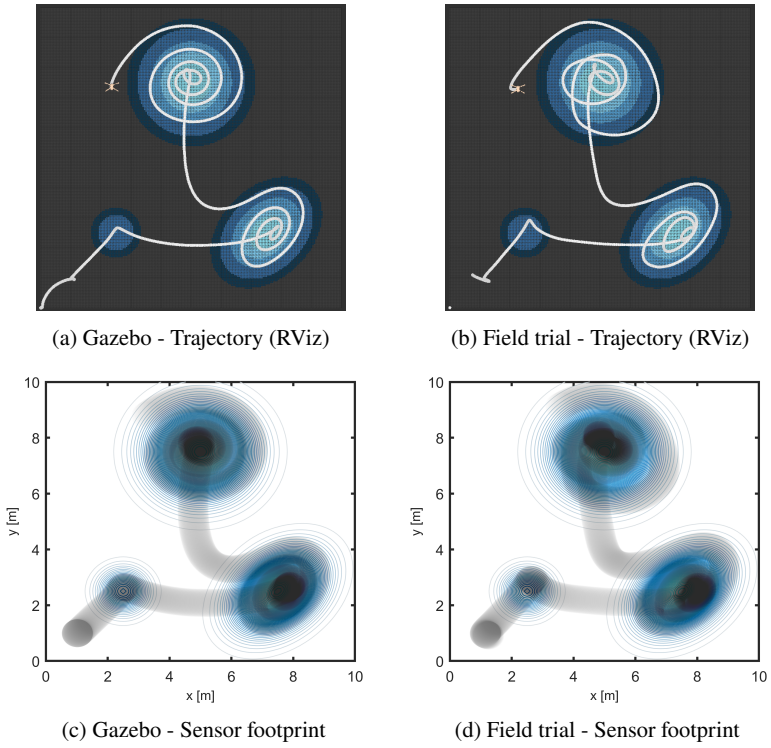
As illustrated in Figure 4.21 (a), the drone exhibits the expected behavior in the Gazebo simulation, executing a smooth spiral curve, as also reflected in the position profiles displayed in Figure 4.21 (e). Such behavior is obviously possible due to the appropriately tuned parameters of the MPC objective function and the selection of the limits for the acceleration and velocity of the vehicle. These choices allow the generation of sufficiently smooth trajectories that the PX4 controller can track efficiently. In addition, as depicted in Figure 4.21 (g), the computational times are also sufficiently fast to allow for a good performance, with each solver iteration taking approximately 18 ms on average.

Concerning the field trial, as shown in Figure 4.21 (b), it is possible to observe that the resulting trajectory is not as consistent as the one from Gazebo. This discrepancy primarily arises from the influence of wind disturbances encountered in

the outdoor experimental environment. In addition, there are also some inaccuracies associated with the Global Positioning System (GPS) of the drone. Nevertheless, for the designated observation radius, both trajectories show a similar coverage by the final time instant, as demonstrated in Figure 4.21 (h).

4.5.8 Experiment 2

In the second example, we consider an uncertainty map composed of three Gaussian components with different shapes. The Gazebo and field trial results are displayed in 4.22.



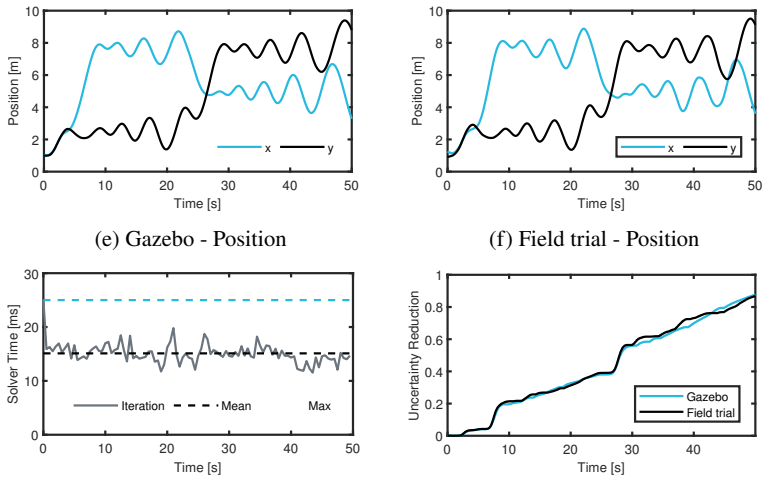
(a) Gazebo - Trajectory (RViz)

(b) Field trial - Trajectory (RViz)

(c) Gazebo - Sensor footprint

(d) Field trial - Sensor footprint

Fig. 4.22: Results for an uncertainty map composed of three Gaussian components.



(e) Gazebo - Position

(f) Field trial - Position

(g) Gazebo - Solver times

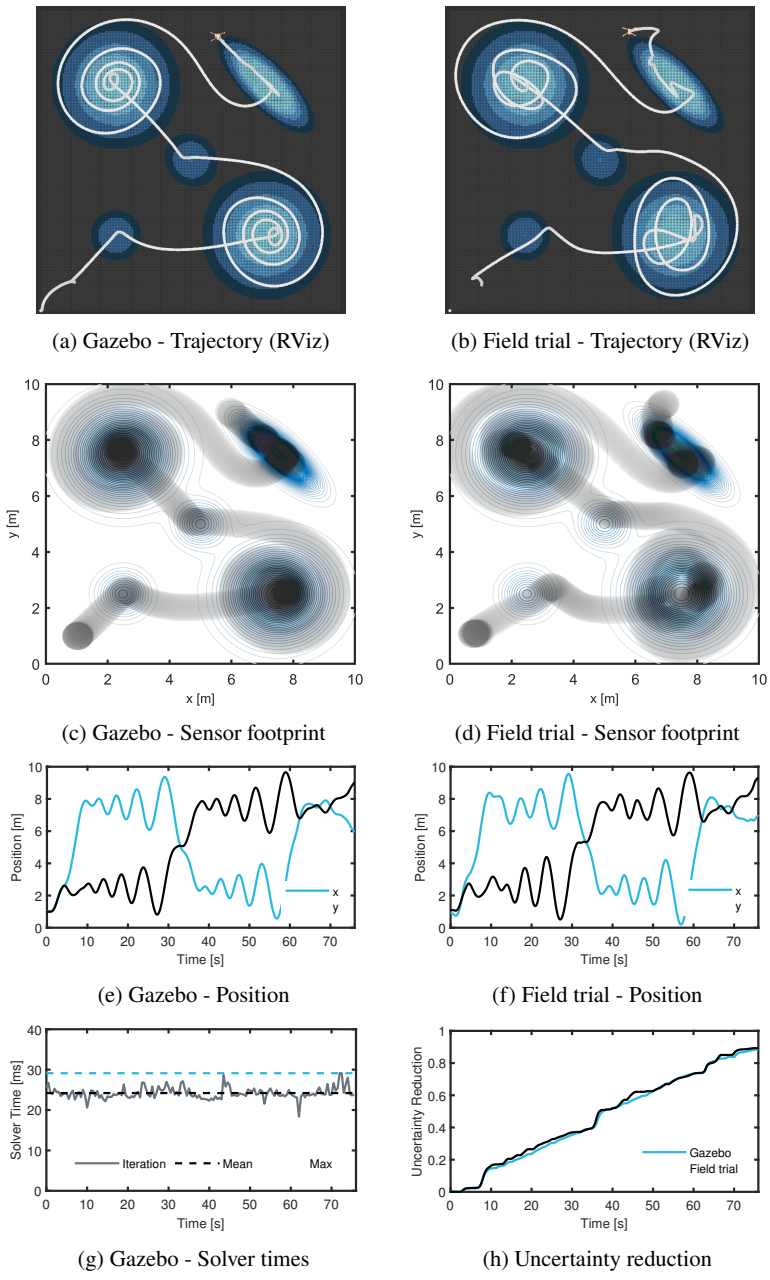
(h) Uncertainty reduction

Fig. 4.22: Results for an uncertainty map composed of three Gaussian components.

As shown in Figure 4.22 (a), in the Gazebo simulation, the drone executes the expected behavior, and the trajectory adapts to the characteristics of each Gaussian component. Concerning the field trials, besides the wind and the GPS errors, we also draw attention to the structural differences between the drones used in Gazebo and in the real trials, as well as differences in the tuning of the inner-loop controllers of the PX4 Autopilot. Despite that, in this example, both trajectories exhibit similar coverage profiles, as shown in Figure 4.21 (h), as well as in Figures 4.21 (c) and 4.21 (d).

4.5.9 Experiment 3

Ultimately, we present an example where the uncertainty map is composed of five Gaussian components. Figure 4.23 displays the results obtained in Gazebo and in the corresponding experimental test.



(a) Gazebo - Trajectory (RViz)

(b) Field trial - Trajectory (RViz)

(c) Gazebo - Sensor footprint

(d) Field trial - Sensor footprint

(e) Gazebo - Position

(f) Field trial - Position

(g) Gazebo - Solver times

(h) Uncertainty reduction

Fig. 4.23: Results for an uncertainty map composed of five Gaussian components.

As illustrated in Figure 4.23 (a), the map comprises four circular Gaussian components. Two of these components have relatively small variances in comparison to the UAV's observation radius, while the other two exhibit higher variances. Additionally, there is a fifth component with an elliptical shape, and its variance along one of its axes is small when compared to the UAV's observation radius. In particular, as this scenario had not been previously introduced, we draw attention to the drone's behavior when analyzing the former component. In such a case, the drone follows a straight path along the major axis of the elliptical component, as depicted in Figure 4.23 (a). Additionally, as shown in Figure 4.23 (g), each solver iteration takes approximately 25 ms in average. This duration is slightly longer than the previous examples due to the map having more components and the mission having a slightly extended duration. Ultimately, it can be observed that the outcomes of the field trial display a comparable pattern to the results obtained from the simulation.

4.6 Summary

This chapter addressed the generation of optimal trajectories for autonomous wildfire prevention using a UAV. The main goal was designing a trajectory planning algorithm based on a map characterizing the uncertainty of fire presence in a given region. We began by establishing the mathematical definition of the uncertainty map, described as a weighted sum of Gaussian components. Then, we outlined the assumptions regarding the sensing capabilities of the UAV. In this context, the vehicle was assumed to fly at a constant altitude, equipped with a gimbal sensor with a limited FOV, and capable of perfectly analyzing a circular region around the UAV's horizontal position. Considering these assumptions, the problem was formulated from an optimal control standpoint, with the objective of maximizing the uncertainty volume covered by the vehicle during the surveillance mission.

Due to the complexity of the original problem formulation, we proposed an alternative approach based on discrete-time MPC. To promote the map exploration and to prevent the UAV from revisiting previously covered regions, the proposed algorithm penalizes intersections between the circular observation areas along the trajectory of the UAV. Given the complexity of precisely calculating the overlap area between two circles, we designed an exponential surrogate function to penalize such intersections.

Subsequently, we described the control architecture used to implement the algorithm in a quadrotor UAV. We considered a dual-layer structure of motion control, with the MPC algorithm serving as a high-level trajectory planner and an inner-loop controller responsible for tracking references generated by the MPC. For the purpose of efficiency, at the planning level, the UAV was modeled as a point-mass system following double-integrator dynamics. The motion control scheme was im-

plemented using a PX4 Autopilot, providing both the inner-loop controller and an EKF to process sensor measurements.

The algorithm was initially tested in MATLAB, assuming ideal double-integrator dynamics for the UAV. Different examples were provided to showcase the trajectories that the algorithm is able to produce, and we conducted an analysis to assess the impact of some parameters on the algorithm's performance. Subsequently, the algorithm was assessed through higher-fidelity simulations in a Gazebo environment and by conducting actual experiments in an outdoor setting.

Chapter 5

Outputs

The execution of project FirePuma was very successful with key scientific, pedagogical and societal contributions that are highlighted in this chapter along with a general overview of output metrics with respect to the anticipated at the proposal stage. The key metrics are presented in Table 5.1 that compare the proposed *versus* the achieved outputs.

Project Outputs Summary			
Outputs	**Proposed**	**Achieved**	**% of execution**
A - Publications			
Books	1	2	200%
Articles in international journals	9	16	178%
Articles in national journals	0	0	n.a.
B - Communications			
International Conference Communications	6	9	150%
National Conference Communications	3	0	0%
C - Reports	3	4	133%
D - Organization of seminars and conferences	2	3	150%
E - Advanced formation			
PhD Theses	2	1	50%
MSc Theses	3	19	633%
Other	0	0	n.a.
F - Models	3	3	100%
G - Computational applications	3	3	100%
H - Pilot installations	0	0	n.a.
I - Laboratory prototypes	0	1	n.a.
J - Patents	0	0	n.a.
L - Others	0	2	n.a.

Table 5.1: Outputs generated by project FirePuma.

Considering the numbers presented in Table 5.1, the FirePuma resulted in the publications of 2 books, which was twice the expected output. Apart from the current document, the Principal Investigator also wrote a book directed at the academic community with the main objective of bridging how state estimation is taught at the graduate level in the deterministic and stochastic cases. The publication is currently under use of one of the curricular units in the School of Science and Technology (SST) from the Nova University of Lisbon.

In terms of publications, there was a clear objective on dissemination using international venues. Even scientific conferences and seminars held nationally have their proceedings published internationally. Therefore, conference communications in Table 5.1 should be viewed as a single indicator where the project met the proposed dissemination goals. In journal publications, the team surpassed the initial objective with 6 more journal papers.

A last point of interest from the values presented in Table 5.1 is in advanced formation. Initially, the project faced difficulties in hiring PhD students, especially during the Covid period. For that reason, a major change in the workplan was to allocate a portion of the human resources budget to hiring at the MSc level. In that regard, the project was very successul completing the formation of 19 MSc students, which represents a 633% execution against the planned 3 theses. This also justifies the number of PhD graduates, which was the only metric not achieved as in the proposal.

5.1 Scientific Contributions

Throughout this book, we have highlighted in each chapter other research questions that were addressed due to their proximity with the main challenge in each task. In this section, we highlight in a broader sense what were the main contributions to the scientific community and how they advance the state-of-the-art and enable further developments.

5.1.1 Reputation-based Resilient Methods

The problem of consensus is often used to provide a distributed version of centralized algorithms. Since it is the building block required for distributed optimization, it can be applied to any task in engineering since all problems correspond to optimizing a function while respecting physical constraints. Prior to project FirePuma, there were two main approaches to eliminating data from compromised agents in a multi-agent system that is running a linear consensus protocol: i) viewing the problem as a fault detection and ii) using the Mean Subsequence Reduced algorithm.

Broadly speaking, option i) often provides theoretical guarantees of detection if the change in the values is large enough as the detectors are using the model to check if the communicated values are consistent with past observations. However, there is an inherent combinatorial nature when we allow multiple nodes to be corrupted as we have to build a detector for all possible combinations of compromised agents. On the other extreme, the Mean Subsequence Reduced algorithm is based on the idea of having each node discard minimum and maximum values as to force attackers to reduce their impact on the network.

During the research of FirePuma, the team introduced the idea of maintaining a reputation for each neighbor much like reputation systems for movies and books value differently the opinion of the contributers depending on their past history of ratings. In doing so, the method has a similar complexity to just removing minimum and maximum values but offers a better accuracy as the attackers are more constrained in their optimal strategy. Current research in this topic is going to release versions that are both resilient, private and able to recover from the attacks.

5.1.2 Constrained Convex Generators

Guaranteed state estimation that uses the support set for each of the unknown signals was primarily conducted using set representations that maintained complexity (intervals and ellipsoids) at the expenses of accuracy or using CZ to have an exact representation if all support sets were polytopes but having a complexity that would grow over time thus preventing running the observer for large periods of time without performing order reductions and introducing conservatism. Moreover, the definition of convex hulls for any of those set representations was either too conservative or computationally expensive.

The research of FirePuma led to the introduction of CCGs which were able to combine any type of bounds for the unknown signals, having all operations in exact format, uncluding the convex hull. As a consequence, the team expanded the state-of-the-art as deterministic state estimation can be computed even for systems with uncertainties while being optimal both in the size of the sets and also their descriptions. Moreover, as these novel formulation allowed combining polytopes and ellipsoids, the team was able to provide a method to perform state estimation with complexity that remains constant and voids the need for order reduction procedures. This method bridges the main criticism against guaranteed state estimation when compared to the KF for its computational complexity and opened the possibility to optimally encode worst-case constraints in optimization-based controllers as those arising from MPC.

5.2 Pedagogical Contributions

During the execution of the project, the team was heavily involved in advanced formation, which led to the identification of two main pedagogical innovations that could arise from the development of FirePuma: i) the shift towards optimization-based controllers in the space domain and ii) the need for a target teaching for the basics of Control Theory.

5.2.1 Survey related to Optimization-based Controllers for Spacecraft

The trajectory generation for the project along with the recent approval of various MSc in Aerospace Engineering in Portuguese universities led the team to compile a survey paper on spacecraft rendezvous that was published in the Encyclopedia of Systems and Control Engineering [117]. In that document, it is highlighted the various stages of spacecraft maneuvers and their associated challenges and typical approaches. The article also compiles a list of recent publications that are aiming in bridging the use of optimization-based controllers for space missions that would lead to control strategies with relevant features like fuel efficiency, safety constraints and computationally lightweight.

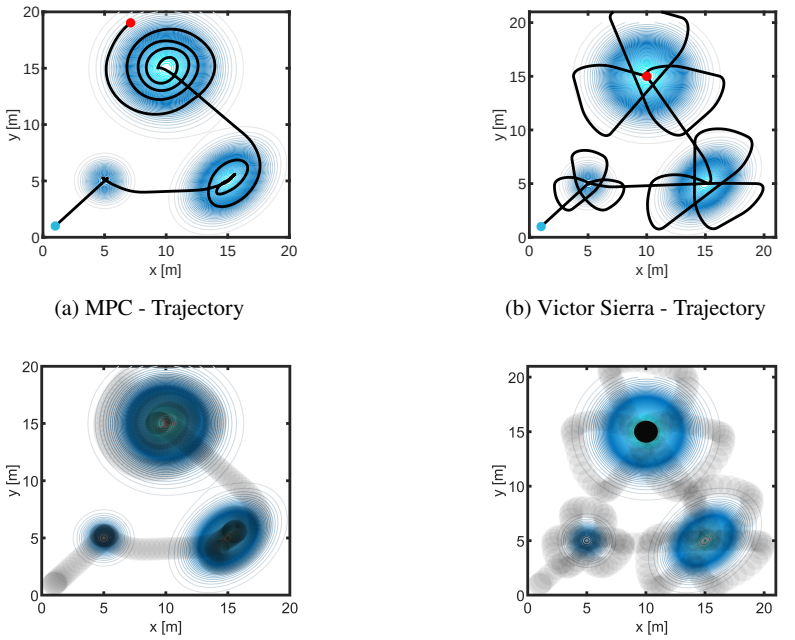
5.2.2 Reformulation of how Control Theory is Taught

Another important aspect that resulted from supervising and being involved with advanced training was realising the need for reformulating how Control Theory is taught at the bachelor degree level. On that note, the team produced a pedagogical article [118] describing how teaching should be organized into two curricular units as opposed to how most universities have 3, if the syllabus is organized in a different way that also lends itself to a study better focused on preparing students to join teams in companies producing controllers.

5.3 Societal Contributions

FirePuma also produced a major societal contribution regarding its trajectory generation algorithm. As the method is quite generic, the team is currently pursuing its application in the area of precision Agriculture and the search for lost people. In the former case, the utility function can translate the need for applying phytopharmaceu-

ticals within a plantation. In the latter, based on the place where the victim was last seen and considering typical movement and environmental conditions, a probability map can be constructed that needs to be checked by the authorities.



(a) MPC - Trajectory

(b) Victor Sierra - Trajectory

(c) MPC - Sensor footprint

(d) Victor Sierra - Sensor footprint

Fig. 5.1: Trajectories and sensor footprints generated by the two methods.

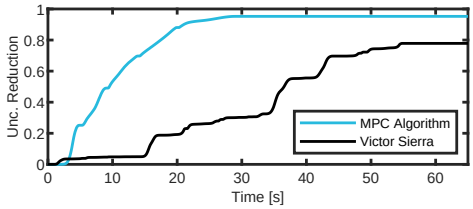


Fig. 5.2: Resulting coverage profiles for the two methods.

The US and Canadian coast guards typically use a pattern named Victor Sierra, also known as Sector Search, that was designed to cover well-defined circular regions centered on a certain position. It consists of straight-line segments that form three equilateral triangular sectors, which are evenly distributed over the circular region. This pattern is popular because in sea operations is straightforward to be implemented

as it only requires the vessel to maitain a constant speed and adjust the angle being followed every multiple of some time period calculated based on the vessel speed. However, such a method is quite inneficient in searching over more complicated maps.

We reproduce in Figure 5.1 the use of the proposed controller by FirePuma (labeled as *MPC*) in comparison with the Victor Sierra pattern that was designed with the 95% confidence ellipse for each gaussian component of the map. As displayed in Figure 5.1 and Figure 5.2, the MPC algorithm has a better coverage profile compared to the Victor Sierra pattern, achieving almost a 100% coverage of the entire volume of the utility function while the Victor Sierra was just below 80%. In search and rescue mission, this difference might have catastrophic impacts and the FirePuma team is working with the Portuguese Guarda Nacional Republicana (GNR) force that is in charge of these missions as to verify the improvements of the proposed method in comparison with the pattern used in real missions.

FirePuma Publications

[6] A. Carvalho, "Automatic Event Detection for Tennis Sport," MSc, School of Science and Technology, University NOVA of Lisbon, Nov. 2023.

[7] D. Fernandes, "Automatic Event Detection for Motorsport," MSc, School of Science and Technology, University NOVA of Lisbon, Nov. 2024.

[8] G. Murta, "Automatic Event Detection for Videogames," MSc, School of Science and Technology, University NOVA of Lisbon, Nov. 2024.

[9] A. Tavares, "Prevention of Forest Fires using Computer Vision," MSc, School of Science and Technology, University NOVA of Lisbon, Nov. 2024.

[10] F. Santos, "Sensor Network using a Mix of Cheap Sensors and Drones," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2022.

[11] F. Santos, D. Silvestre, and R. Cunha, "Development and experimental validation of a lora wireless sensor network for wildfire surveillance," in *2023 IEEE Conference on Control Technology and Applications (CCTA)*, 2023, pp. 911–917. DOI: 10.1109/CCTA54093.2023.10252472.

[12] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Fast desynchronization algorithms for decentralized medium access control based on iterative linear equation solvers," *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 6219–6226, 2022. DOI: 10.1109/TAC.2021.3130888.

[13] F. Pereira, "Six Floor Origin - A Media JIT Packaging and Storage Platform," MSc, School of Science and Technology, University NOVA of Lisbon, Nov. 2024.

[14] R. Pires, "Enhancing Security through the use of Load-Balancing Stochastic Algorithms," MSc, Instituto Superior Técnico, University of Lisbon, Oct. 2022.

[15] A. Yazidi, D. Silvestre, and B. J. Oommen, "Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 650–661, 2023, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2021.3099095.

[16] G. Ramos, D. Silvestre, and A. P. Aguiar, "A resilient continuous-time consensus method using a switching topology," *Systems & Control Letters*, vol. 169, p. 105 381, 2022, ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2022.105381.

[17] G. Ramos, D. Silvestre, and C. Silvestre, "A discrete-time reputation-based resilient consensus algorithm for synchronous or asynchronous communications," *IEEE Transactions on Automatic Control*, vol. 69, no. 1, pp. 543–550, 2024. DOI: 10.1109/TAC.2023.3266982.

[18] P. Matias, "Outlier and Attacker Resilient Methods Based on Rating and Reputation Systems," MSc, School of Science and Technology, University NOVA of Lisbon, Mar. 2023.

[19] D. Silvestre, "Privacy assessment for linear consensus using constrained convex generators," in *62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 8045–8050. DOI: 10.1109/CDC49753.2023.10384271.

[43] D. Silvestre, "Constrained convex generators: A tool suitable for set-based estimation with range and bearing measurements," in *IEEE American Control Conference (ACC)*, Jun. 2022.

[44] D. Silvestre, "Constrained convex generators: A tool suitable for set-based estimation with range and bearing measurements," *IEEE Control Systems Letters*, vol. 6, pp. 1610–1615, 2022. DOI: 10.1109/LCSYS.2021.3129729.

[45] F. Rego and D. Silvestre, "A novel and efficient order reduction for both constrained convex generators and constrained zonotopes," *IEEE Transactions on Automatic Control*, pp. 1–8, 2025. DOI: 10.1109/TAC.2024.3525388.

[46] D. Silva, "Distributed and Centralized Observers for Sensor Fusion with Non-Gaussian Noises," MSc, Instituto Superior Técnico, University of Lisbon, Oct. 2024.

[47] D. Silvestre, "Comparison of recent advances in set-membership techniques: Application to state estimation, fault detection and collision avoidance," *European Journal of Control*, vol. 81, p. 101 161, 2025, ISSN: 0947-3580. DOI: https://doi.org/10.1016/j.ejcon.2024.101161.

[48] D. Silvestre, "Set-valued estimators for uncertain linear parameter-varying systems," *Systems & Control Letters*, vol. 166, p. 105 311, 2022, ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2022.105311.

[49] D. Silvestre, "Accurate guaranteed state estimation for uncertain lpvs using constrained convex generators," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4957–4962. DOI: 10.1109/CDC51059.2022.9993211.

[50] D. Silvestre, "Exact set-valued estimation using constrained convex generators for uncertain linear systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 9461–9466, 2023, 22nd IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2023.10.241.

[51] J. Felizardo, "Passively Safe Trajectory Generation using Model Predictive Control," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2024.

[52] B. Contente, "Formal Safety Verification Using Reachability Analysis for the Koopman Operator," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2024.

[53] L. Silva, "Neural Networks Robustness Verification using Reachability Tools," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2023.

[54] D. Silvestre, "Closed-form approximations for the minimal robust positively invariant set using constrained convex generators," *IFAC-PapersOnLine*, vol. 58, no. 25, pp. 126–131, 2024, 3rd Control Conference Africa CCA 2024, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2024.10.249.

[86] B. Sabetghadam, "Distributed Trajectory Generation for Multiple Autonomous Vehicles Using Bernstein Polynomial-based Methods," PhD, Instituto Superior Técnico, University of Lisbon, Nov. 2023.

[87] B. Lourenço, "Enhancing Truck Platooning Efficiency and Safety," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2023.

[88] B. Lourenco and D. Silvestre, "Enhancing truck platooning efficiency and safety - a distributed model predictive control approach for lane-changing manoeuvres," *Control Engineering Practice*, vol. 154, p. 106 153, 2025, ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2024.106153.

[89] D. Silvestre, J. H. Hespanha, and C. Silvestre, "Model predictive control with collision avoidance for unknown environment," in *62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023.

[90] D. Silvestre and G. Ramos, "Model predictive control with collision avoidance for unknown environment," *IEEE Control Systems Letters*, vol. 7, pp. 2821–2826, 2023. DOI: 10.1109/LCSYS.2023.3288884.

[91] J. Pereira, "Computationally Efficient and Safe Constrained Attitude Control for Spacecraft Slewing with Control Barrier Functions," MSc, Instituto Superior Técnico, University of Lisbon, Jun. 2024.

[92] J. Neves, "A Comparison Study of MPC and Control Barrier Functions Algorithms for Multi-Agent Systems in the presence of Obstacles," MSc, Instituto Superior Técnico, University of Lisbon, Jun. 2021.

[93] J. Coelho, "Surveillance System using Cheap Sensors and Drones," MSc, Instituto Superior Técnico, University of Lisbon, Jun. 2023.

[94] R. Ribeiro, F. Santos, D. Silvestre, and C. Silvestre, "Distributed exploration algorithm for gps-denied areas based on flocking rules," *International Journal of Control*, vol. 0, no. 0, pp. 1–10, 2025. DOI: 10.1080/00207179.2025.2458135.

[95] H. Matias, "Model-Predictive Trajectory Planning for Autonomous Aerial Surveillance Applications," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2023.

[96] H. Policarpo *et al.*, "Conceptual design of an unmanned electrical amphibious vehicle for ocean and land surveillance," *World Electric Vehicle Journal*, vol. 15, no. 7, 2024, ISSN: 2032-6653. DOI: 10.3390/wevj15070279.

[97] F. Rego and D. Silvestre, "Cooperative navigation using constrained convex generators," *European Journal of Control*, vol. 81, p. 101 163, 2025, ISSN: 0947-3580. DOI: https://doi.org/10.1016/j.ejcon.2024.101163.

[98] G. Ramos, D. Silvestre, and C. Silvestre, "The robust minimal controllability and observability problem," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 10, pp. 5033–5044, 2021. DOI: https://doi.org/10.1002/rnc.5527.

[99] R. Palma, "Stability Guarantees for Model Predictive Controllers," MSc, Instituto Superior Técnico, University of Lisbon, Oct. 2023.

[100] P. Taborda, "Model Predictive Control and Allocation with Nonconvex Constraints for Space Rendezvous," MSc, Instituto Superior Técnico, University of Lisbon, Nov. 2023.

[101] P. Taborda, H. Matias, D. Silvestre, and P. Lourenço, "Convex mpc and thrust allocation with deadband for spacecraft rendezvous," *IEEE Control Systems Letters*, vol. 8, pp. 1132–1137, 2024. DOI: 10.1109/LCSYS.2024.3407611.

[117] D. Silvestre and P. Lourenço, "Guidance and control for in-orbit servicing and assembly missions," in *Reference Module in Materials Science and Materials Engineering*, Elsevier, 2024, ISBN: 978-0-12-803581-8. DOI: https://doi.org/10.1016/B978-0-443-14081-5.00027-1.

[118] P. Garrido and D. Silvestre, "A lean path for learning control in undergraduate electronics programmes," *IFAC-PapersOnLine*, vol. 58, no. 25, pp. 78–83, 2024, 3rd Control Conference Africa CCA 2024, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2024.10.241.

References

[1] N. Sönnichsen. "Land burned by wildfires in Europe by country 2022". Statista. Available online: https://www.statista.com/statistics/1260777/area-burned-by-wildfire-in-european-countries/ (accessed on June 10, 2023).

[2] N. Sönnichsen. "Land burned by forest fires in Portugal 2009-2022". Statista. Available online: https://www.statista.com/statistics/1265367/area-burned-by-wildfire-in-portugal/ (accessed on June 10, 2023).

[3] A. Valles, M. Ferrer, K. Poljanšek, and I. Clark (eds.), *Science for disaster risk management 2020: acting today, protecting tomorrow*. Publications Office, 2020.

[4] Departamento de Gestão de Áreas Públicas e de Protecção Florestal, *10º Relatório Provisório de Incêndios Florestais*. Instituto de Conservação da Natureza e das Florestas, 2017.

[5] Presidência do Conselho de Ministros, "Resolução do Conselho de Ministros n.º 159/2017", *Diário da República*, vol. 1, no. 209, pp. 5820-5822, 2017.

[20] S. Kim, H. Lee, and S. Jeon, "An adaptive spreading factor selection scheme for a single channel lora modem," *Sensors*, vol. 20, no. 4, p. 1008, 2020.

[21] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *ieee communications surveys & tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[22] "Sigfox device etsi mode white paper. "[Online]. Available: https://support.sigfox.com/docs/sigfox-device-etsi-mode-white-paper.

[23] A. S. Kurji, A. H. Al-Nakkash, and O. A. Hussein, "Lora in a campus: Reliability and stability testing," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1105, 2021, p. 012 034.

[24] "Semtech sx1276 datasheet. "[Online]. Available: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE.

[25] D. Selent, "Advanced encryption standard," *Rivier Academic Journal*, vol. 6, no. 2, pp. 1–14, 2010.

[26] "Esp32. "[Online]. Available: https://www.espressif.com/en/products/socs/esp32.

[27] "Arduino uno. "[Online]. Available: https://store.arduino.cc/products/arduino-uno-rev3/.

[28] "Arduino spi. "[Online]. Available: https://www.arduino.cc/en/reference/SPI.

[29] N. Deligiannis, J. F. C. Mota, G. Smart, and Y. Andreopoulos, "Fast desynchronization for decentralized multichannel medium access control," *IEEE Transactions on Communications*, vol. 63, no. 9, pp. 3336–3349, Sep. 2015, ISSN: 0090-6778. DOI: 10.1109/TCOMM.2015.2455036.

[30] IEEE, "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, 2012. DOI: 10.1109/IEEESTD.2012.6185525.

[31] D. Antunes, D. Silvestre, and C. Silvestre, "Average consensus and gossip algorithms in networks with stochastic asymmetric communications," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 2088–2093. DOI: 10.1109/CDC.2011.6161444.

[32] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Broadcast and gossip stochastic average consensus algorithms in directed topologies," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 474–486, 2019, ISSN: 2325-5870. DOI: 10.1109/TCNS.2018.2839341.

[33] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016. DOI: 10.1137/15M1009597.

[34] D. Silvestre, J. Hespanha, and C. Silvestre, "Desynchronization for decentralized medium access control based on gauss-seidel iterations," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 4049–4054. DOI: 10.23919/ACC.2019.8814471.

[35] D. Silvestre, J. Hespanha, and C. Silvestre, "A pagerank algorithm based on asynchronous gauss-seidel iterations," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 484–489. DOI: 10.23919/ACC.2018.8431212.

[36] D. Silvestre, "Optool — an optimization toolbox for iterative algorithms," *SoftwareX*, vol. 11, p. 100 371, 2020, ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2019.100371.

[37] D. Silvestre, "Reputation-based method to deal with bad sensor data," *IEEE Control Systems Letters*, vol. 6, pp. 43–48, 2022. DOI: 10.1109/LCSYS.2020.3048098.

[38] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Resilient desynchronization for decentralized medium access control," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 803–808, 2021, ISSN: 2475-1456. DOI: 10.1109/LCSYS.2020.3005819.

[39] Y. Kikuya, S. M. Dibaji, and H. Ishii, "Fault tolerant clock synchronization over unreliable channels in wireless sensor networks," *IEEE Transactions on Control of Network Systems*, pp. 1–1, 2018, ISSN: 2325-5870. DOI: 10.1109/TCNS.2017.2732169.

[40] G. Ramos, D. Silvestre, and C. Silvestre, "Node and network resistance to bribery in multi-agent systems," *Systems & Control Letters*, vol. 147, p. 104 842, 2021, ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2020.104842.

[41] G. Ramos, D. Silvestre, and C. Silvestre, "A general discrete-time method to achieve resilience in consensus algorithms," in *59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 2702–2707. DOI: 10.1109/CDC42340.2020.9304107.

[42] G. Ramos, D. Silvestre, and C. Silvestre, "General resilient consensus algorithms," *International Journal of Control*, vol. 95, no. 6, pp. 1482–1496, 2020. DOI: 10.1080/00207179.2020.1861331.

[55] P. Mali, K. Harikumar, A. K. Singh, K. M. Krishna, and P. Sujit, "Incorporating prediction in control barrier function based distributive multi-robot collision avoidance," in *2021 European Control Conference (ECC)*, 2021, pp. 2394–2399. DOI: 10.23919/ECC54610.2021.9655081.

[56] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz, "Constrained zonotopes: A new tool for set-based estimation and fault detection," *Automatica*, vol. 69, pp. 126 –136, 2016, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2016.02.036.

[57] D. Silvestre, P. Rosa, R. Cunha, J. P. Hespanha, and C. Silvestre, "Gossip average consensus in a byzantine environment using stochastic set-valued observers," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 4373–4378. DOI: 10.1109/CDC.2013.6760562.

[58] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Finite-time average consensus in a byzantine environment using set-valued observers," in *American Control Conference (ACC), 2014*, 2014, pp. 3023–3028. DOI: 10.1109/ACC.2014.6859426.

[59] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Distributed fault detection using relative information in linear multi-agent networks," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 446–451, 2015, 9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 20, Paris, 2-4 September 2015, ISSN: 2405-8963. DOI: http://dx.doi.org/10.1016/j.ifacol.2015.09.567.

[60] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Set-consensus using set-valued observers," in *American Control Conference (ACC), 2015, Chicago, Illinois, USA.*, 2015.

[61] D. Silvestre, J. P. Hespanha, and C. Silvestre, "Fault detection for cyber-physical systems: Smart grid case," in *23rd International Symposium on Mathematical Theory of Networks and Systems (MTNS), Hong-Kong, China.*, 2018.

[62] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Set-based fault detection and isolation for detectable linear parameter-varying systems," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 18, pp. 4381–4397, 2017, rnc.3814, ISSN: 1099-1239. DOI: 10.1002/rnc.3814.

[63] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Fault detection for LPV systems using set-valued observers: A coprime factorization approach," *Systems & Control Letters*, vol. 106, pp. 32 –39, 2017, ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2017.05.007.

[64] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Self-triggered and event-triggered set-valued observers," *Information Sciences*, vol. 426, pp. 61 –86, 2018, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.10.029.

[65] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Stochastic and deterministic fault detection for randomized gossip algorithms," *Automatica*, vol. 78, pp. 46 –60, 2017, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2016.12.011.

[66] M. Althoff and B. H. Krogh, "Zonotope bundles for the efficient computation of reachable sets," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 6814–6821. DOI: 10.1109/CDC.2011.6160872.

[67] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Finite-time convergence policies in state-dependent social networks," in *2015 American Control Conference (ACC)*, 2015, pp. 1041–1046. DOI: 10.1109/ACC.2015.7170870.

[68] D. Silvestre, P. Rosa, J. P. Hespanha, and C. Silvestre, "Sensitivity analysis for linear systems based on reachability sets," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 361–366. DOI: 10.1109/CDC40024.2019.9029765.

[69] D. Silvestre, P. Rosa, and C. Silvestre, "Distinguishability of discrete-time linear systems," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 5, pp. 1452–1478, 2021. DOI: 10.1002/rnc.5367.

[70] M. Conforti, M. Di Summa, and Y. Faenza, "Balas formulation for the union of polytopes is optimal," *Mathematical Programming*, vol. 180, no. 1, pp. 311–326, Mar. 2020, ISSN: 1436-4646. DOI: 10.1007/s10107-018-01358-9.

[71] V. Raghuraman and J. P. Koeln, "Set operations and order reductions for constrained zonotopes," *Automatica*, vol. 139, p. 110 204, 2022, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2022.110204.

[72] D. E. Hernández-Mendoza, G. R. Penaloza-Mendoza, and E. Aranda-Bricaire, "Discrete-time formation and marching control of multi-agent robots systems," in *8th International Conference on Electrical Engineering, Computing Science and Automatic Control*, Oct. 2011, pp. 1–6. DOI: 10.1109/ICEEE.2011.6106618.

[73] M. Goldberg, "Equivalence constants for lp norms of matrices," *Linear and Multilinear Algebra*, vol. 21, no. 2, pp. 173–179, 1987.

[74] S. Kousik, A. Dai, and G. X. Gao, "Ellipsotopes: Uniting ellipsoids and zonotopes for reachability analysis and fault detection," *IEEE Transactions on Automatic Control*, vol. 68, no. 6, pp. 3440–3452, 2023. DOI: 10.1109/TAC.2022.3191750.

[75] D. Bertsekas and I. Rhodes, "Recursive state estimation for a set-membership description of uncertainty," *IEEE Transactions on Automatic Control*, vol. 16, no. 2, pp. 117–128, 1971. DOI: 10.1109/TAC.1971.1099674.

[76] H. Witsenhausen, "Sets of possible states of linear systems given perturbed observations," *IEEE Transactions on Automatic Control*, vol. 13, no. 5, pp. 556–558, 1968. DOI: 10.1109/TAC.1968.1098995.

[77] F. Chernous'ko, "Ellipsoidal estimates of a controlled system's attainability domain," *Journal of Applied Mathematics and Mechanics*, vol. 45, no. 1, pp. 7–12, 1981, ISSN: 0021-8928. DOI: https://doi.org/10.1016/0021-8928(81)90002-2.

[78] F. F. C. Rego, "Distributed state estimation and cooperative path-following under communication constraints," Ph.D. dissertation, Ecole Polytechnique Fédérale de Lausanne, 2018.

[79] M. ApS, *The mosek optimization toolbox for matlab manual. version 9.0.* 2019. [Online]. Available: http://docs.mosek.com/9.0/toolbox/index.html.

[80] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," D, vol. 82, pp. 35–45, 1960.

[81] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002, Conference Name: IEEE Transactions on Signal Processing, ISSN: 1941-0476. DOI: 10.1109/78.978374. Accessed: Oct. 4, 2023.

[82] A. G. Wills, J. Hendriks, C. Renton, and B. Ninness, *A Bayesian Filtering Algorithm for Gaussian Mixture Models*, arXiv:1705.05495 [cs, stat], Jun. 2023. Accessed: Jul. 17, 2023.

[83] A. Runnalls, "Kullback-Leibler Approach to Gaussian Mixture Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 989–999, Jul. 2007, Conference Name: IEEE Transactions on Aerospace and Electronic Systems, ISSN: 1557-9603. DOI: 10.1109/TAES.2007.4383588. Accessed: Nov. 15, 2023.

[84] M. Idan and J. L. Speyer, "An estimation approach for linear stochastic systems based on characteristic functions," en, *Automatica*, vol. 78, pp. 153–162, Apr. 2017, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2016.12.038. Accessed: Jul. 17, 2023.

[85] A. P. Vinod, B. Homchaudhuri, and M. M. K. Oishi, "Forward Stochastic Reachability Analysis for Uncontrolled Linear Systems using Fourier Transforms," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, arXiv:1610.04550 [cs, math], Apr. 2017, pp. 35–44. DOI: 10.1145/3049797.3049818. Accessed: Jul. 17, 2023.

[102] R. Ribeiro, D. Silvestre, and C. Silvestre, "Decentralized control for multi-agent missions based on flocking rules," in *CONTROLO 2020*, J. A. Gonçalves, M. Braz-César, and J. P. Coelho, Eds., Cham: Springer International Publishing, 2021, pp. 445–454, ISBN: 978-3-030-58653-9.

[103] R. Ribeiro, D. Silvestre, and C. Silvestre, "A rendezvous algorithm for multi-agent systems in disconnected network topologies," in *2020 28th Mediterranean Conference on Control and Automation (MED)*, 2020, pp. 592–597. DOI: 10.1109/MED48518.2020.9183093.

[104] W. Press, *Numerical recipes: The art of scientific computing*. Cambridge University Press, 2007.

[105] J. Löfberg, "YALMIP : A Toolbox for Modeling and Optimization in MATLAB", *CACSD Conference*, 2004.

[106] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor", *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[107] B. Siciliano, L. Sciavicco, L.Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[108] "PX4 Autopilot User Guide". PX4 Autopilot. Available online: https://docs.px4.io/main/en/ (accessed on June 10, 2023).

[109] The MathWorks Inc., *MATLAB, 9.11.0 (R2021b)*. The MathWorks Inc., 2021.

[110] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control", *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[111] A. Wätcher and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", *Mathematical programming*, vol. 106, no. 12, pp. 25-57, 2006.

[112] T. Oliveira, P. Trindade, D. Cabecinhas, P. Batista, and R. Cunha, "Rapid development and prototyping environment for testing of unmanned aerial vehicles", *2021 IEEE International Conference on Autonomous Robot Systems and Competitions*, pp. 191–196, 2021.

[113] M. Jacinto, *Cooperative motion control of aerial and marine vehicles for environmental applications*. Master's Thesis, Instituto Superior Técnico, 2021.

[114] "Gazebo simulator". Gazebo. Available online: https://gazebosim.org/ (accessed on June 10, 2023).

[115] "Iris quadcopter uav". Arducopter. Available online: www.arducopter.co.uk (accessed on June 10, 2023).

[116] M690B Wiki. Available online: https://hardtekpt.github.io/M690B-Wiki/ (accessed on June 10, 2023).