CHALMERS
UNIVERSITY OF TECHNOLOGY

Nonlinear Model Predictive Control for Constant Distance Between Autonomous Transport Robots

Master's thesis in Systems, Control and Mechatronics

KLARA PÅLSSON
EMMA SVÄRLING

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Nonlinear Model Predictive Control for Constant Distance Between Autonomous Transport Robots

KLARA PÅLSSON
EMMA SVÄRLING

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Nonlinear Model Predictive Control for Constant Distance Between Autonomous Transport Robots.

KLARA PÅLSSON
EMMA SVÄRLING

Supervisor: Per-Lage Götvall, Volvo Trucks
Examiner: Knut Åkesson, Electrical Engineering

Master's Thesis 2020
Department of Electrical engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Nonlinear Model Predictive Control for Constant Distance Between Autonomous Transport Robots.
KLARA PÅLSSON
EMMA SVÄRLING
Department of Electrical Engineering
Chalmers University of Technology

Abstract

A possible solution in today's production plants for transporting parts is with autonomous transport robots (ATR), guided by cameras in the ceiling and odometry from the wheels. For larger parts, this thesis proposes a nonlinear model predictive controller (NMPC) for two ATRs to collaborate with carrying the object together from A to B, given two feasible reference trajectories for both ATRs. Three different approaches with NMPC were tested and verified with respect of time, robustness and distance between the ATRs. The first approach was an NMPC using Fmincon built in MATLAB, the second approach was an NMPC using IPOPT built in Python and the third approach was an NMPC using PANOC built in Rust. The NMPC using Fmincon was set up with a numerical framework while NMPC using IPOPT and PANOC were set up in a symbolic framework using CasADi. Both Fmincon and IPOPT solved the problem with interior-point optimizer, while PANOC applied a Newton-method combined with forward-backward iterations. The NMPC using PANOC was implemented with the embedded optimizer, Optimization Engine, which generated an NMPC in Rust code from the setup in Python. Verification of the different approaches was demonstrated through several simulations. The approach with the shortest calculation time, that had satisfied robustness and kept the distance between robots, was the NMPC implemented with Optimization Engine that utilised PANOC for solving.

Keywords: Nonlinear model predictive control, Autonomous transport robots, Optimization engine, PANOC, Fmincon, IPOPT, Rust, MATLAB, Python

Acknowledgements

This master thesis has been carried out under the Department of Electrical Engineering at Chalmers University of Technology, in collaboration with Volvo AB. We want to thank Knut Åkesson and Per-Lage Götvall for the good idea to the master thesis and all the help during the spring and your commitment. Thanks to Bo Egardt and Nikolce Murgovski for the help with our questions around NMPC and to Björn Lindqvist for the help with OpEn. A big thanks to the people who have worked with the previous work in the area, with whom we have been able to ask and get help, Anthony Cavin, Amanda Fransson, Max Reidal and Jonas Herzfeld.

Klara Pålsson & Emma Svärling, Gothenburg, June 2020

Contents

List of Figures

List of Tables

Glossary

AGV	Autonomous Guided Vehicle.
ATR	Autonomous Transport Robot.
MPC	Model Predictive Control
NMPC	Non-linear Model Predictive Control
AprilTag	Visual fiducial system, similar to QR code
OpEn	Optimization Engine

1
Introduction

1.1 Background

The transport industry is changing rapidly where the demands of sustainable fuels, including electrification, are increasing. In the transition, the flexibility of the production plants needs to increase, with the possibility to produce a wide variety of trucks including the traditional diesel trucks and the trucks with new fuels. This will increase storage demands in the production plant for different parts for the various types of trucks. To create a good work environment for the factory workers, a robot fleet could be the solution to collect the right parts, for the right truck, just in time.

Today, there are mostly trucks and human labour that move the parts in the factories, however, it has become more common with AGVs (automated guided vehicles). The first AGV was created in 1953 by A.M. Barret Jr. [2], but was quickly followed up by Barrett Electronics from USA, who manufactured an AGV which was a tow truck that followed a wire, embedded in the floor of a warehouse or factory. Today, there are several different models of AGV and ways of control them, but most are still guided by a wire in the floor.

One way to control autonomous system is with MPC (model predictive control). MPC was founded, according to [3], in the petro-chemical industry in the 1970s. The great advantage of MPC are the possibilities to have constraints on the inputs and outputs, nonlinear models of the system and online update of the controller. In recent years, it has also become more common in electric systems.

1.2 Problem description

To succeed with a just in time system, with a robot fleet, the problem starts with the robots and how to make them collaborate and work together. The proposed just in time robot fleet for this thesis, consist of ATRs (autonomous transport robots), equipped with necessary near-range sensors for a low cost, which are guided by a generic, photogrammetry-based sensor system (GPSS) for cost efficiency. The ATR together with the GPSS, compared to the AGV, has the advantages to move around without the guidance of a wire or a laser. A previous work was done by

Cavin [4] where a path planner, a motion planner and a controller was developed for transporting objects in an environment with static, semi-static and moving objects with the GPSS system. The path planner and controller were developed for the ATRs to carry the load individually.

This thesis aim is to extend the existing system with a cooperative object transport system. More specifically, two ATRs will cooperate with carrying a larger object together in a robust way from A to B, while constant distance between the ATRs are maintained. This thesis will handle the development of the control system, in form of a nonlinear model predictive controller (NMPC). Three different solvers for NMPC will be tested, Fmincon, IPOPT and PANOC.

This thesis will be carried out in collaboration with Volvo Group Trucks Operations and Chalmers University of Technology. This system's purpose is to be used in Volvo Trucks production where an object will be carried by the ATRs with preserved distance between the ATRs.

1.3 Related Works

One of the most relevant work is the master thesis *Obstacle Avoidance for Mobile AGVs Using a Photogrammetry-Based Sensor System* by A. Cavin [4]. Cavin used the photogrammetry-based sensor systems (GPSS) to move one ATR from A to B. He designed the control system which use the position information from the GPSS. A local path planner and motion planner is then used, in combination with a trajectory tracking regulator, to transport the object from A to B while avoiding obstacles. [4] did not address maintained distance between the ATRs.

If there are more than one robot, the system is often referred to cooperative object transport. The article [5] describes how there are different strategies in how the robots are carrying objects together. Three different ways are raised with pushing or pulling the object, carrying the object or combinations.

Vision based methods like in the thesis by A. Cavin with more than one robot is done in the work [6]. There the robots had a planned trajectory to follow and manage to keep the formation. The control is made of several algorithms, but they do not use nonlinear model predictive control.

A continuation of [4] was carried out by [7] at Chalmers University of Technology in the course *Design project in systems, control and mechatronics.* The aim was to create a cooperative object transportation system with two ATRs that could carry an object together. The group made a path planner and a motion planner for the second ATR and an MPC controller for both ATRs. This was done to create a cooperative object transport system, which made it possible to control the ATRs along a given trajectory and also avoid collision with static objects. In the end, the simulations worked good for them, but it did not function in real time due to high computational time.

Typically, nonlinear problems are determined by sequential quadratic programming (SQP) [8], which can cause heavy calculations if the problem is formulated poorly. An algorithm, named PANOC, was developed by [8], there the problems are solved by a Newton-method combined with forward-backward iterations. The algorithm was built to suite well for embedded NMPC applications.

The master thesis from Luleå university by B. Lindqvist [9] used a nonlinear model predictive control (NMPC) for a Micro Aerial Vehicles (MAV's). The thesis's aim was to move the MAV without any collision with obstacles, given start and end positions. To be able to use the NMPC for both obstacle avoidance and path-planning, B. Lindqvist used the optimisation tool, Optimisation Engine (OpEn). Optimization Engine is a relatively new optimisation solver with focus on robotic and autonomous systems. There are several similarities with B. Lindqvist work and this thesis where the need for a fast NMPC is of the utmost importance without compromising safety. The embedded optimizer, OpEn, use the solver PANOC.

1.4 Research Questions

The goal for this work is to control two cooperating ATRs in a robust way from A to B while carrying an object. The main research question to conclude the goal will thus be:

- **Research question 1.** How well will the NMPC using PANOC solve the problem of carrying an object from A to B compared to the NMPC using Fmincon and NMPC using IPOPT, with respect to time, robustness and distance between the ATRs?
- **Research question 2.** Verify the advantages, disadvantages and further development of the NMPCs using Fmincon, IPOPT and PANOC in a production plant with respect of time, robustness and safety aspects.

1.5 Limitations

This thesis is limited to 20 weeks during the spring semester 2020. The work is done in collaboration with Volvo Trucks and Chalmers University of Technology. The idea is to implement an algorithm that, once it is functional for two ATRs, has the potential to be scaled for the usage with a fleet of ATRs, but in this thesis limited to two ATRs. Further, two obstacle-free and feasible trajectories are assumed to be given for the ATRs, and the project is restricted to handle the control of the ATRs. This means that object detection and localisation are not treated in this project. The ATRs are not available and therefore only simulations are possible.

2
Problem Setup

The aim for this thesis was to extend the existing system with a cooperative object transport system. The existing system, that Volvo Trucks has right now is present in this chapter, and is the system which the simulations were based on.

2.1 Setup architecture

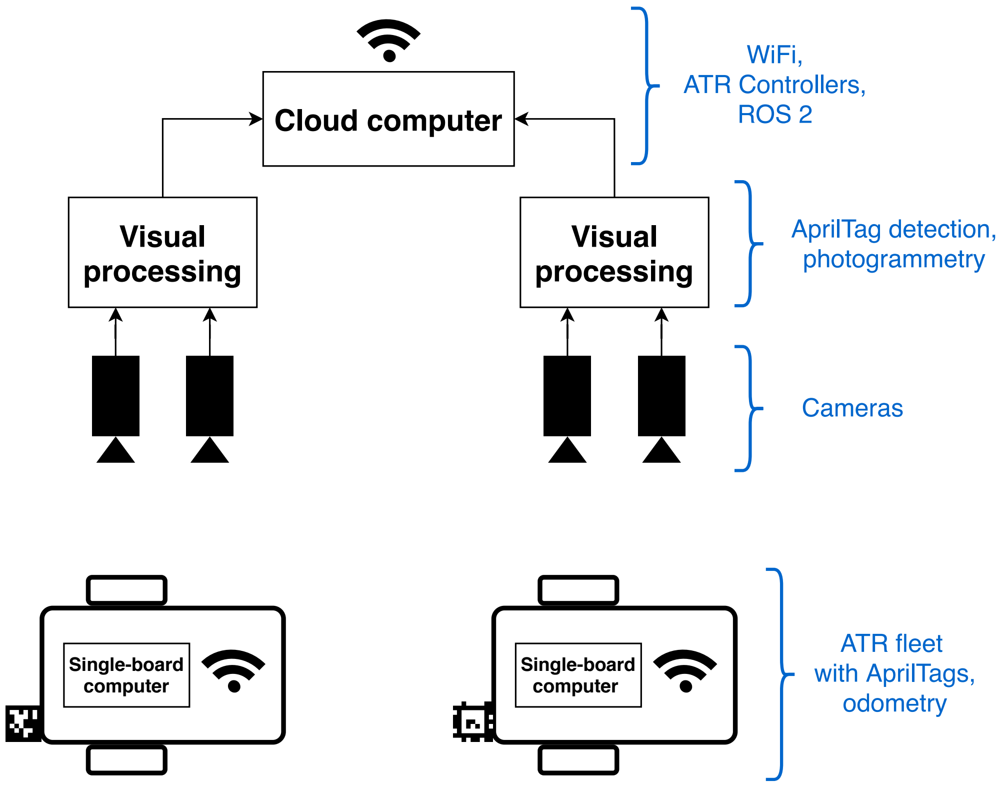


Figure 2.1: The setup architecture.

The setup architecture consisted of the ATR fleet, cameras, AprilTags, WiFi and hardware as seen in Figure 2.1. The cameras were mounted in a grid in the ceiling. The visual processing took the information from the cameras and detected the

AprilTags positions and orientations. This information was used in the photogrammetry, which is explained in details in section 2.4 Photogrammetry. In the cloud computer, the ATR controllers and ROS™, among other things, were implemented. The Robot Operating System (ROS) is an operating system with nodes to enable communication between the different hardware, and ROS is further explained in section 2.5 ROS. The communication between the ATRs and the cloud computer was through WiFi. The ATRs had a single-board computer and a unique AprilTag mounted on each one of them and the ATRs is further explained in section 2.2 Autonomous transport robot. The sampling time of the controller was set to $2[Hz]$, which means that new control inputs to the ATRs were calculated every $500[ms]$.

2.2 Autonomous transport robot

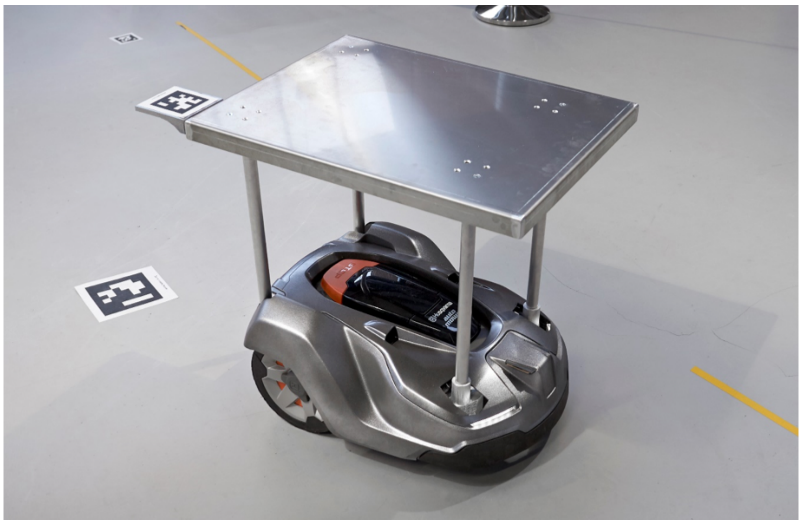


Figure 2.2: A prototype of Volvo Truck's ATR, which is based on a robotic lawnmower from Husqvarna modified with a table and AprilTag. From [1]. Reproduced with permission.

.

The ATR used in this work was a differential drive robot, seen in Figure 2.2. The differential drive robot has two drive wheels mounted on a common axis. Each wheel can be driven independently both backward or forward and the robot had two supporting wheels in the front. The development was done based on a differential drive robot from Husqvarna, slightly modified with a table and an AprilTag, as seen in Figure 2.2. The AprilTag, looks similar to a QR-code, together with the cameras is a visual fiducial system which can be used to compute the exact positions and rotations of the ATRs.

2.3 Odometry

The ATRs were equipped with wheel encoders and the odometry estimated the change in position and orientation based on the wheel encoder sensors.

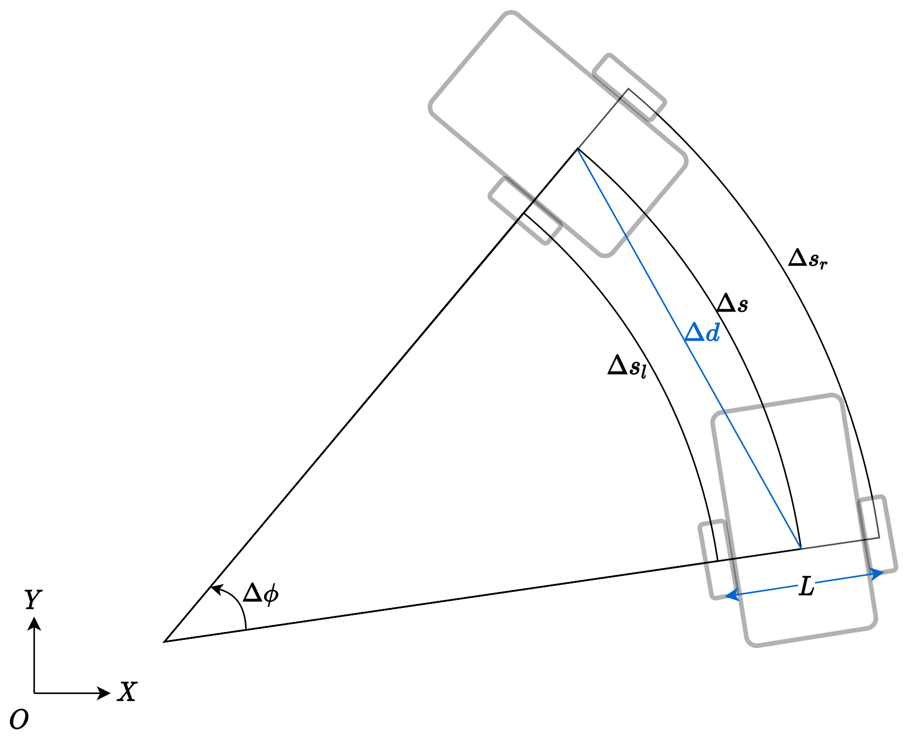


Figure 2.3: Travelled distance and angle for odometry calculations. Δs for arc length, $\Delta\phi$ for angle difference and Δd for distance between two consecutive sensor readings. L is the distance between the wheels.

If the arc length for the left wheel, Δs_l and the right wheel Δs_r was measured between two consecutive sensor readings by the wheel encoder, [10] showed that the change in position and angle could be calculated. The assumptions made to achieve an accurate estimation was that the angle difference $\Delta\phi$ between two readings were small. If $\Delta\phi$ was small, then the distance between the ATRs could be approximated by the arc length, $\Delta d \approx \Delta s$. ΔX, ΔY and $\Delta\phi$ could be calculated according to,

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}, \tag{2.1a}$$

$$\Delta\phi = \frac{\Delta s_r - \Delta s_l}{L}, \tag{2.1b}$$

$$\Delta X = \Delta s \cos(\phi + \Delta\phi/2), \tag{2.1c}$$

$$\Delta Y = \Delta s \sin(\phi + \Delta\phi/2). \tag{2.1d}$$

Due to recursive integration of each wheel rotation, this will propagate estimation error over time. Other source of error could be unequal wheel diameters and slipping due to variable friction and uneven floors. The advantages of the odometry was that it was fast to calculate, and it is a relatively cheap sensor.

2.4 Photogrammetry

To estimate a more accurate position of the ATRs, a grid of cameras was mounted in the ceiling. These cameras detected the AprilTags mounted on the ATRs. In the visual processing, the photos of the AprilTags were recalculated to decide the position and orientation of the ATR. The AprilTags, seen in Figure 2.1, looks similar to a QR code but was specially developed for visual fiducial systems, including robotics. The AprilTags is a fast and robust system according to [11].

2.5 ROS

The Robot Operating System (ROS™) was used for communication between the different hardware in the system, since only simulations was made in this thesis, ROS was not used, but ROS is required for real testing. ROS is a meta operating system, which means that it is built on top of the existing operation system. The ROS architecture allows different processes, called nodes, to communicate during run time. The information was sent between the nodes through Ethernet or WiFi. For example, the node in the visual processing sends the updated position decided by the photogrammetry to the node in the ATR controller in the cloud computer. The ATR controller's node then sends the information about the next control input, through WiFi, to the nodes in the ATRs. The predicted position, decided by the odometry, was sent back to the ATR controller node through WiFi. According to [12], ROS can be used with several different languages, including Python and C++, where the first one was used in this thesis.

2.6 Summary

The system in Volvo Trucks is setup with ATR fleet, cameras, AprilTags, WiFi and hardware. The cameras are mounted in the ceiling and detected the ATR through the AprilTags. The ATRs are differential drive robots, which are slightly modified with a table with an AprilTag on.

The ATRs are equipped with wheel encoders and the odometry estimate the change in position and orientation based on the wheel encoder sensors. The cameras give the photogrammetry and in the hardware the photos from the cameras of the AprilTags are recalculated to decide the position and orientation of the ATRs. To communicate between the different hardware is robot operating system, ROS, used.

3

Controller and Differential Drive Robot

The calculations behind the position, velocity and angle velocity of the ATRs are described in this chapter, to know how the ATRs could move. Three discrete motion models are also presented, where the differences depends on whether the ATRs were driving straight forward, following curves or needed a better approximation. The chapter ends by introducing the control architecture with the trajectory planner, NMPC controller and observer.

3.1 Differential Drive Robot

The ATR used, seen in Figure 2.2, was a differential drive robot. As explained in section 2.2 Autonomous transport robot, the differential drive robot had two drive wheels mounted on a common axis and each wheel could be driven independently both backward or forward.

3.1.1 Motion model

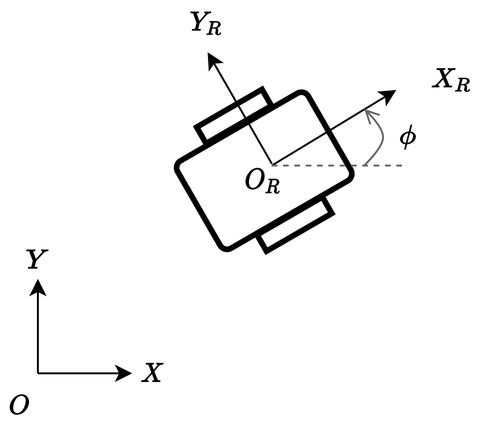


Figure 3.1: The coordinates and frame's of the differential drive robot.

The differential drive robot had three degrees of freedom, $\boldsymbol{x} = [X, Y, \phi]^T$, in position X, position Y and orientation ϕ, as seen in Figure 3.1. The robot's local frame was mention as O_R in the Figure and the global reference frame, O. The connection between the two frames is,

$$\dot{\boldsymbol{x}}_R = R\left(\phi\right)\dot{\boldsymbol{x}}, \tag{3.1}$$

where the rotation matrix $R(\phi)$ in (3.1), was formulated according to [4] and [13],

$$R(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

In the local frame, the speed was controlled by the input $\boldsymbol{u} = [v, \omega]^T$, which moved the ATR in the X_R direction and turned the ATR around in the ϕ angle. The ATR could not move in the Y_R direction, and the ATR could therefore be said to have only two degrees of mobility. The model of the ATR then had the form,

$$\dot{X}_R = v, \tag{3.3a}$$
$$\dot{Y}_R = 0, \tag{3.3b}$$
$$\dot{\phi}_R = \omega. \tag{3.3c}$$

Given (3.2) and (3.3), the Equation (3.1) was used to translate the kinematic model from the local frame O_R to the global frame O. For continuous time, this was expressed as,

$$\dot{X} = v\cos\left(\phi\right), \tag{3.4a}$$
$$\dot{Y} = v\sin\left(\phi\right), \tag{3.4b}$$
$$\dot{\phi} = \omega. \tag{3.4c}$$

The connection between right (r) and left (l) wheel is,

$$v_r = v + \frac{\omega b}{2}, \tag{3.5a}$$
$$v_l = v - \frac{\omega b}{2}, \tag{3.5b}$$

where b stands for the distance between the wheels. Combining equations in (3.5) gave the connections,

$$v = \frac{1}{2}(v_r + v_l), \tag{3.6a}$$

$$\omega = \frac{1}{b}(v_r - v_l). \tag{3.6b}$$

3.1.2 Discrete motion models

There was three different discrete motion models used in this thesis. The first one was the Exact motion model along a curve, which was used when the ATR should follow curves and not drive straight ahead. The second discrete motion model was the first order Runge-Kutta which was a simpler model, and lastly forth order Runge-Kutta. In the Exact motion model, the equation was depending on the turning rate, ω, and divided by it. Since it does not work when the ATR have zero turning rate, one of the Runge-Kutta models was needed.

Exact motion along a curve

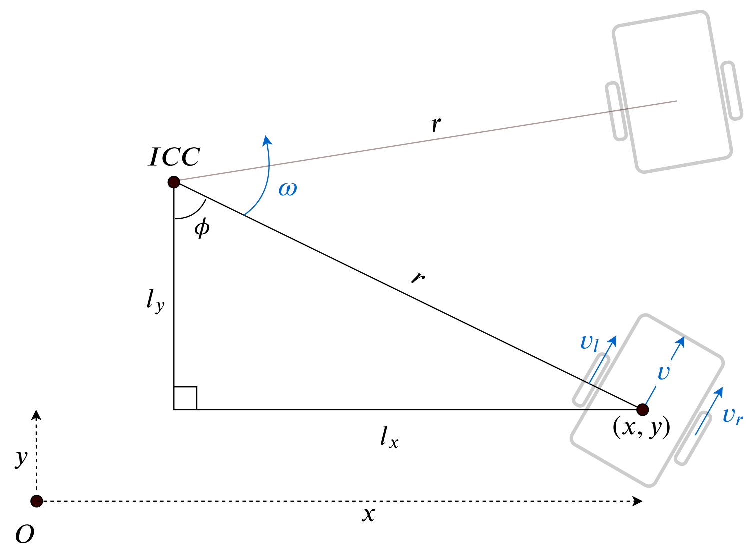


Figure 3.2: The coordinates and frame's of the ATR for calculation of exact motion along a curve. ICC stands for instantaneous center of curvature.

The derivation of the method can be found in Appendix A. The ATRs kinematics in discrete time,

$$X_{k+1} = X_k - \frac{v_k}{\omega_k}\sin\left(\phi_k\right) + \frac{v_k}{\omega_k}\sin\left(\phi_k + \omega_k T_s\right), \tag{3.7a}$$

$$Y_{k+1} = Y_k + \frac{v_k}{\omega_k}\cos\left(\phi_k\right) - \frac{v_k}{\omega_k}\cos\left(\phi_k + \omega_k T_s\right), \tag{3.7b}$$

$$\phi_{k+1} = \phi_k + \omega_k T_s. \tag{3.7c}$$

Runge-Kutta 1st order

Runge-Kutta (RK) solves differential equations with numerical approximation. The 1st order RK (RK1) is the same as forward Euler,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{3.8}$$

The kinematic model for the ATR for continuous time in equation (3.4), together with (3.8) gave the ATRs kinematics in discrete time,

$$X_{k+1} = X_k + T_s v_k \cos\left(\phi_k\right), \tag{3.9a}$$

$$Y_{k+1} = Y_k + T_s v_k \sin\left(\phi_k\right), \tag{3.9b}$$

$$\phi_{k+1} = \phi_k + \omega_k T_s. \tag{3.9c}$$

Runge-Kutta 4th order

Runge-Kutta of the 4th order (RK4) was a better approximation of the discretisation than RK1, and RK4 that was used in this thesis is called the classic RK4, and have the Butcher tableau seen in Table 3.1.

Table 3.1: Butcher tableau classic RK4.

$$\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}$$

The iteration formula for the classic RK4 is,

$$\boldsymbol{k}_1 = T_s \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \tag{3.10a}$$

$$\boldsymbol{k}_2 = T_s \boldsymbol{f}\left(\boldsymbol{x}_k + \frac{1}{2}\boldsymbol{k}_1, \boldsymbol{u}_k + \frac{T_s}{2}\right), \tag{3.10b}$$

$$\boldsymbol{k}_3 = T_s \boldsymbol{f}\left(\boldsymbol{x}_k + \frac{1}{2}\boldsymbol{k}_2, \boldsymbol{u}_k + \frac{T_s}{2}\right), \tag{3.10c}$$

$$\boldsymbol{k}_3 = T_s \boldsymbol{f}\left(\boldsymbol{x}_k + \boldsymbol{k}_3, \boldsymbol{u}_k + T_s\right), \tag{3.10d}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \frac{1}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right). \tag{3.10e}$$

3.1.3 Linearized motion model

The motion model can be linearized around a reference trajectory with the first order Taylor expansion, to speed up the calculation time. First order Taylor expansion in continuous time looks as follows;

$$f(x,u) \approx f(x^{ref}, u^{ref}) + \nabla_x f(x^{ref}, u^{ref})(x - x^{ref}) + \nabla_u f(x^{ref}, u^{ref})(u - u^{ref}). \tag{3.11}$$

This Taylor expansion could be used when the ATRs position was close to the trajectory, $x - x^{ref} \rightarrow 0$. $x - x^{ref}$ refer to the tracking error and $u - u^{ref}$ refer to the input variations. To obtain the linearization for the differential drive robot, Equation (3.4) was utilized to calculate $f(x^{ref}, u^{ref})$,

$$\nabla_x f\left(x^{ref}, u^{ref}\right) = \begin{bmatrix} 0 & 0 & -v^{ref}\sin\left(\phi^{ref}\right) \\ 0 & 0 & v^{r}\cos\left(\phi^{ref}\right) \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.12a}$$

$$\nabla_u f\left(x^{ref}, u^{ref}\right) = \begin{bmatrix} \cos\left(\phi^{ref}\right) & 0 \\ \sin\left(\phi^{ref}\right) & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.12b}$$

The forward Euler method, seen in Equation (3.8), together with Equation (3.12) result in the linearized discrete motion model;

$$x_{k+1} - x_{k+1}^{ref} = A_k(x_k - x_k^{ref}) + B_k(u_k - u_k^{ref}), \tag{3.13a}$$

$$A_k = \begin{bmatrix} 1 & 0 & -T_s v_k^{ref} \sin\left(\phi_k^{ref}\right) \\ 0 & 1 & T_s v_k^{ref} \cos\left(\phi_k^{ref}\right) \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.13b}$$

$$B_k = \begin{bmatrix} T_s \cos\left(\phi_k^{ref}\right) & 0 \\ T_s \sin\left(\phi_k^{ref}\right) & 0 \\ 0 & T_s \end{bmatrix}. \tag{3.13c}$$

As mentioned earlier, this linearization method imposes $x - x^{ref} \to 0$ and therefore, if the ATR followed a trajectory, the reference point was needed to be updated at every time step and a time varying controller was required.

3.2 Controller

The controller selected, to accomplish a cooperative object transporting system with two ATRs, was a nonlinear model predictive controller (NMPC) together with a trajectory planner and a Kalman filter observer. The focus was on the NMPC, which was the central part for achieving a sufficient cooperative object transporting system.

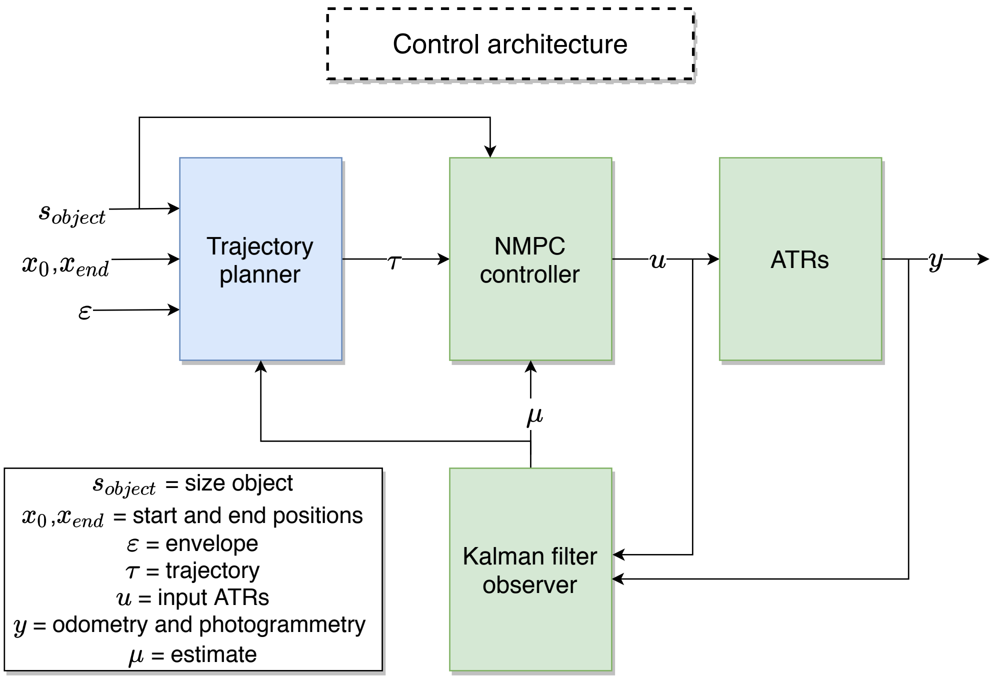


Figure 3.3: General control architecture where the blue block is not covered by this thesis.

The general control architecture for the two ATRs can be seen in the Figure 3.3. The state vectors are defined in Equations (3.14), where elevated denotes which ATR, for instance X^1 refer to ATR_1. Index k indicate discrete time while index l and r refer to left and right wheel respectively.

$$\boldsymbol{s}_{object} = \begin{bmatrix} s_{length} & s_{width} \end{bmatrix} \tag{3.14a}$$

$$\boldsymbol{x}_0 = \begin{bmatrix} X_0^1 & Y_0^1 & \phi_0^1 & X_0^2 & Y_0^2 & \phi_0^2 \end{bmatrix} \tag{3.14b}$$

$$\boldsymbol{\epsilon}_k = \begin{bmatrix} X_k^l & Y_k^l & X_k^r & Y_k^r \end{bmatrix} \tag{3.14c}$$

$$\boldsymbol{\tau}_k = \begin{bmatrix} X_k^1 & Y_k^1 & \phi_k^1 & v_k^1 & \omega_k^1 & X_k^2 & Y_k^2 & \phi_k^2 & v_k^2 & \omega_k^2 \end{bmatrix} \tag{3.14d}$$

$$\boldsymbol{u} = \begin{bmatrix} v_l^1 & v_r^1 & v_l^2 & v_r^2 \end{bmatrix} \tag{3.14e}$$

$$\boldsymbol{y} = \begin{bmatrix} X_{odom}^1 & Y_{odom}^1 & \phi_{odom}^1 & X_{odom}^2 & Y_{odom}^2 & \phi_{odom}^2 \\ X_{photo}^1 & Y_{photo}^1 & \phi_{photo}^1 & X_{photo}^2 & Y_{photo}^2 & \phi_{photo}^2 \end{bmatrix} \tag{3.14f}$$

$$\boldsymbol{\mu} = \begin{bmatrix} X^1 & Y^1 & \phi^1 & X^2 & Y^2 & \phi^2 \end{bmatrix} \tag{3.14g}$$

3.2.1 Trajectory planner

The trajectory planner provided two trajectories which consisted of path- and motion planner for each of the ATRs. Given the size of the carrying object $\boldsymbol{S}_{object}$, start position $\boldsymbol{x}_0$, end position $\boldsymbol{x}_{end}$ and the envelop $\boldsymbol{\epsilon}$, the trajectory planner calculated two feasible trajectories which became the input to the NMPC controller. The trajectory planner was based on the path planner and motion planner from [7] and [4]. This thesis was not developing the trajectory planner further.

3.2.2 Observer

The result from Cavin's [4] various observers was utilised and the most sufficient observer for this implementation was the linear Kalman filter. The linear Kalman filters prediction step used the odometry from the ATR to estimate the position. The Kalman filters update step used the photogrammetry to update the position.

3.2.3 NMPC controller

NMPC was the method behind how the system was controlled. The theory behind MPC and NMPC is in more detail described in the section 4.1 Introduction to Model Predictive Control and section 4.2 Nonlinear MPC. The NMPC carried in the trajectories from the trajectory planner, the envelop for the both ATRs and the reference estimate. The output from the NMPC is the input to the ATR motors.

3.3 Summary

The kinematic model for the ATR was,

$$\dot{X}_R = v, \tag{3.15a}$$
$$\dot{Y}_R = 0, \tag{3.15b}$$
$$\dot{\phi}_R = \omega, \tag{3.15c}$$

and the velocity and angle velocity was calculated from,

$$v = \frac{1}{2}\left(v_r + v_l\right), \tag{3.16a}$$
$$\omega = \frac{1}{b}\left(v_r - v_l\right). \tag{3.16b}$$

There were three different discrete motion models for the ATR. Two for driving straight forward (Runge-Kutta) and one for following curves (Exact motion along a curve).

The motion model could be linearized around a reference trajectory using the first order Taylor expansion. The linearization method imposed $x - x^{ref} \rightarrow 0$ and therefore, if the ATR followed a trajectory, the reference point needed to be updated at every time step and a time varying controller was required.

The system was controlled by a NMPC controller. The NMPC controller got information from the trajectory planner, envelop and Kalman filter observer. The information from the NMPC controller became the input signals to the ATRs.

4

Controller

To control the two ATRs, nonlinear model predictive control (NMPC) was used in this thesis. The theory behind MPC and NMPC is described in this chapter with cost function and constraints. Later in the chapter the nonlinear solvers are explained and real-time NMPC.

4.1 Introduction to Model Predictive Control

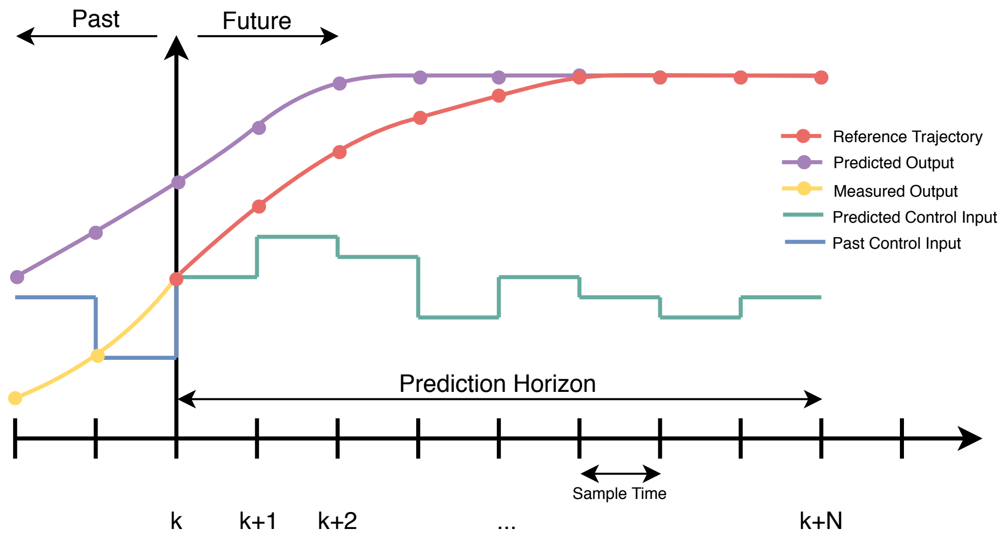


Figure 4.1: General MPC scheme.

Model predictive control (MPC), sometimes called receding horizon control, differ from the common proportional integral derivative (PID) control in many ways. The MPC can, in contradistinction to the PID, handle constraints on both the control input and the output, while predicting both the control input and the output at every time step in a chosen horizon. The MPC can predict how the ATR will behave in the horizon based on possible future control inputs, although this require a reliable model of the system. The MPC optimise the next control input based on future control input and predicted output, see Figure 4.1. This is achieved by optimising the control input for the whole horizon while only implementing the control input for the next time step. This optimisation is done at every time step, and the advantage is that the optimal control will be obtained, even if the conditions change over time.

Cost function

The cost function, that can be formulated, according to [3], as in Equation (4.1), has the purpose to minimise a chosen configuration of states and inputs. The configuration or algorithm is essential to get a good performance of the MPC. The tuning of the cost function consists of giving the different part of the cost function different weights. Higher weight means higher penalty on that part and to obtain the minimum, the parameters need to consider that part more.

$$\min \qquad V_N(x, u(0:N-1)) = \sum_{i=0}^{N-1} l(x(i), u(i)) + V_f(x(N)) \tag{4.1}$$

Constraints

If constraints are required for the MPC system, they can be added to Equation (4.1) according to [3], as seen in Equation (4.2). Constraints can be on both the control input and the output. Constraints on the control input can for example be the maximum velocity that the ATR can achieve and constraints on the output can be forbidden positions of the ATR.

$$\begin{aligned}
\min \qquad & V_N(x, u(0:N-1)) = \sum_{i=0}^{N-1} l(x(i), u(i)) + V_f(x(N)) \\
\text{s.t.} \qquad & x^+ = f(x,u), \quad x(0) = x \\
& x(k) \in \mathbb{X}, \quad u(k) \in \mathbb{U} \quad \text{for all } k \in (0, N-1) \\
& x(N) \in \mathbb{X}_f \subseteq \mathbb{X}
\end{aligned} \tag{4.2}$$

4.2 Nonlinear MPC

The most efficient MPC is the linear MPC and therefore it is common to linearize parts that are nonlinear. To be considered a linear MPC, the system dynamics, cost function and constraints must all be linear and convex. An example of linearizing the system dynamics is the trajectory following MPC, where the dynamics can be linearized around the reference trajectory, with the assumption that the ATR stay close to the reference trajectory. A linear MPC formulation can for example be the quadratic programming with the form

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} \frac{1}{2}\boldsymbol{x}^T(i)Q\boldsymbol{x}(i) + \boldsymbol{u}^T(i)R\boldsymbol{u}(i); \quad i = 0, ..., N-1, && (4.3a) \\ \text{s.t.} \quad & \boldsymbol{x}(i+1) = A\boldsymbol{x}(i) + B\boldsymbol{u}(i), && (4.3b) \\ & F\boldsymbol{u}(i) + G\boldsymbol{x}(i) \leq h. && (4.3c) \end{aligned}$$

In some cases, it is not possible to linearize one or several part of the formulation, or the linearization is too simplified to create a usable controller. In these cases, a nonlinear MPC (NMPC) is necessary. For instance, if a non-convex constraint is required or the dynamics cannot be linearized around a reference trajectory. The NMPC can be formulated as

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & \sum_{i=0}^{N-1} \frac{1}{2}\begin{bmatrix} \boldsymbol{x}(i) \\ \boldsymbol{u}(i) \end{bmatrix}^T W \begin{bmatrix} \boldsymbol{x}(i) \\ \boldsymbol{u}(i) \end{bmatrix}; \quad i = 0, ..., N-1, && (4.4a) \\ \text{s.t.} \quad & \boldsymbol{x}(i+1) = f(\boldsymbol{x}(i), \boldsymbol{u}(i)), && (4.4b) \\ & h(\boldsymbol{u}(i), \boldsymbol{x}(i)) \leq 0. && (4.4c) \end{aligned}$$

4.2.1 Nonlinear constraints

There are many different types of nonlinear constraints. One common nonlinear constraint is the obstacle avoidance constraint. This is obvious non-convex since there are no solutions inside the obstacle's area. Instead, in many applications, the trajectory planner could take care of the obstacle avoidance, but this might not be safe if the ATR does not follow the reference trajectory. Possible scenario is in this case that the ATR might run in to an obstacle if it is not considered as a constraint, which could be unsafe if the obstacle is a human. This is just one of many applications where non-convex constraints would be sufficient to use. Another example of a nonlinear constraint is to keep the distance between two ATRs, and is further explained in subsection 5.1.5 Nonlinear constraint.

4.2.2 Nonlinear dynamics

If the system dynamics cannot be linearized or the linearization is not sufficient enough, NMPC is required. There are several ways of implementing the nonlinear system dynamics in an NMPC formulation. Two common ways are direct single shooting and multiple shooting.

Direct single shooting

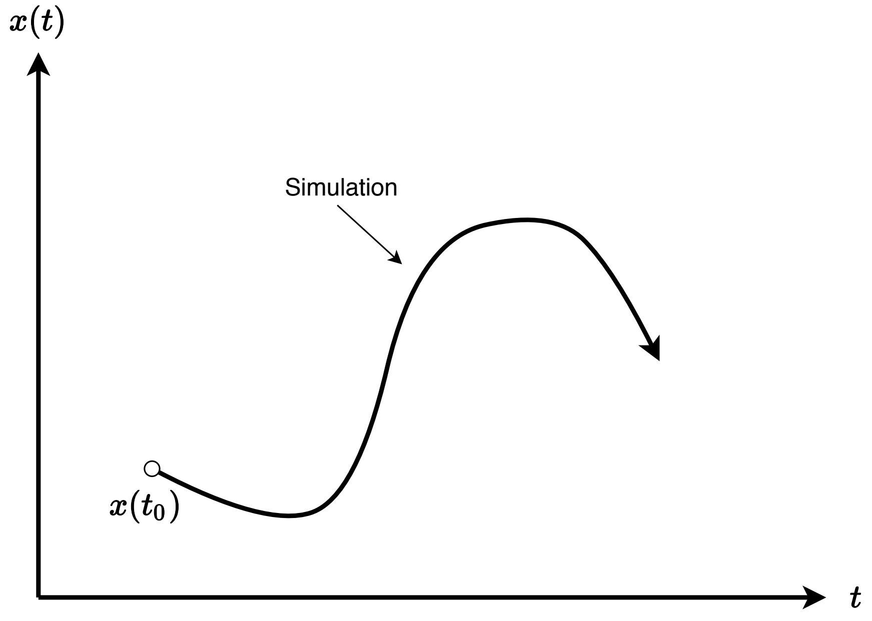


Figure 4.2: An illustration over single shooting.

Direct single shooting implies, according to [14], that the decision variable $\boldsymbol{z}$ only consists of the control input $\boldsymbol{z} = [u_0, \dots, u_{N-1}]$. The states $\boldsymbol{x}(1, \dots, N-1)$ are instead predicted or simulated by the system dynamics,

$$\boldsymbol{x}(i+1) = f(\boldsymbol{x}(i), \boldsymbol{u}(i)), \tag{4.5}$$

see Figure 4.2. The system dynamic simulated in the cost function and constraints, given the initial state, will be nested functions and increase in size depending on horizon length, i.e.

$$\boldsymbol{x}(3) = f(f(f(\boldsymbol{x}(0), \boldsymbol{u}(0)), \boldsymbol{u}(1)), \boldsymbol{u}(2)). \tag{4.6}$$

This means that only the control input $\boldsymbol{u}$ are seen as unknown and the states $\boldsymbol{x}$ are seen as known variables. Direct single shooting require a good model of the dynamics and a good discretization, especially if the horizon is long.

Multiple shooting

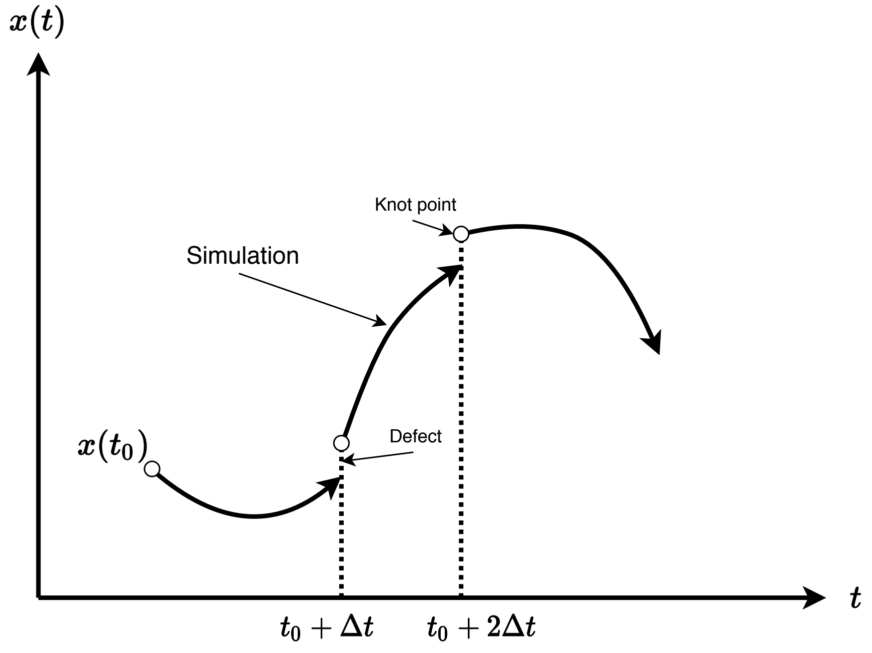


Figure 4.3: An illustration over multiple shooting.

Multiple shooting denotes that the decision variable $\boldsymbol{z}$ includes all, or some, states $\boldsymbol{x}$ in the decision variable, $\boldsymbol{z} = [u_0, \ldots, u_{N-1}, x_1, \ldots, x_{M-1}]$. The states chosen to be unknown are called knot points, see Figure 4.3. This means that both the states chosen as knot points and the control input are seen as unknown. The simulation does only go between the knot points, and the different simulations can be parallelised. To ensure that the system follows the system dynamics, the dynamics is added as a constraint,

$$x(i+1) = f(x(i), u(i)). \tag{4.7}$$

A significant difference between single shooting and multiple shooting, beside that the system dynamics are added as a constraint, is that multiple shooting require an initial guess for all the decision variables and all the state variables, see Figure 4.3, there the initial guesses are marked as *Knot point*, while single shooting only require initial guess for all the decision variables. [14] suggest that multiple shooting could lead to better numerical performances if there exist good initial guess. On the other hand, if there exist non-convex constraints, there are a risk of giving an initial guess that leads to infeasibility. Also, the fact that the simulation only does short intervals will lead to better performance if the system model is not accurate or if the discretization is not that precise.

4.3 Nonlinear solvers

There are different nonlinear solvers, with different grade of specialization in control. In this thesis three different solvers were tested and the solvers will be explained in this section.

4.3.1 Fmincon

Fmincon [15] finds the minimum of constrained nonlinear multivariable functions with start at an initial guess. This is often called constrained nonlinear optimisation or nonlinear programming. Fmincon can solve a variety of different constrained nonlinear optimisation problems, not only NMPC, and minimise an objective function with respect of linear and nonlinear constraints.

Fmincon have the possibility to implement four different algorithms and is a gradient-based method. The different algorithms are: interior point, SQP (sequential quadratic programming), active set, and trust region reflective. If the algorithm is correctly implemented, Fmincon will find the global minimum. If the problem becomes infeasible, then Fmincon will try to minimise the maximum constraint value. In this thesis was the algorithm, *interior point*, elected, which is the default option in Fmincon.

4.3.2 IPOPT

IPOPT (Interior Point OPTimizer) is, according to [16], an open-source software package for large-scale nonlinear optimisation. IPOPT minimise the objective function with respect of constraints and the objective function and the constraints has the opportunity to be nonlinear and non-convex but must be twice continuously differentiable. IPOPT implements, according to [17], an interior-point algorithm for continuous, nonlinear, non-convex, constrained optimisation problems. For faster solving does IPOPT find local minimum which require good starting position to find the global minimum.

4.3.3 PANOC

PANOC stands for Proximal Averaged Newton-type method for Optimal Control [9] and is a solver for nonlinear optimal control. It works for non-convex optimisations problems. PANOC apply a Newton-method combined with forward-backward iterations, which make PANOC to a very fast solver. Like IPOPT, does PANOC find local minimum.

The solver require, according to [8], a single shooting formulation of the problem. In other word, the decision variable consist of only the control input and does not include the states. It is the forward-backward iterations in PANOC that require the single shooting formulation.

PANOC can handle both simple constraint and more complex constraints. If the constraints are more complex, PANOC handle it in two different ways, either with the augmented Lagrangian method which works for constraint on the form,

$$F_1(u, p) \in C, \tag{4.8}$$

or the Penalty method, which work for constraints on the form,

$$F_2(u, p) = 0. \tag{4.9}$$

According to [18] gives the Lagrangian method better result, since the Penalty method handle constraints on more general form, but it is possible to use both the Lagrangian method and the Penalty method simultaneously. The Penalty method can be seen as a soft constraint with high penalty when implemented in PANOC, even if the intention of the constraint can be a hard constraint.

4.4 Real-time NMPC

To implement PANOC in this thesis, was the embedded optimizer, Optimization Engine (OpEn), used. OpEn is a relatively new optimization tool, and as stated in [18] does OpEn focus on optimal control, robotics and autonomous systems. The solver in OpEn is written in Rust code, which is a fast programming language, ideal for embedded applications according to [18]. It is possible to design the problem directly with Rust code, but it is also possible to design the interface in other language such as MATLAB or Python. If MATLAB or Python is used, it will automatically be converted to Rust code by OpEn.

To set up the problem in OpEn, the code needs to be written according to the CasADi guidelines. CasADi is, according to [19], an open-source tool for algorithmic differentiation and nonlinear optimization. It uses symbolic framework instead of numerical framework, which is the big advantage of CasADi and what makes it so fast. In each time step, CasADi construct a symbolic object which is only dependent on the decision variables and initial conditions. After the problem has been set up according to CasADi guidelines, OpEn use the solver PANOC described in subsection 4.3.3 PANOC.

4.5 Summary

Nonlinear model predictive control was chosen as the controller, since a controller that could handle the nonlinear distance constant between the two ATRs was needed. NMPC enable the controller to handle constraints on both the control input and the output, while it predicts both the control input and the output at every time step in a chosen horizon. If the systems decision variable only consists of the control input, the system is a single shooting system. It is a multiple shooting system if the decision variable also includes the states.

There are a lot of different nonlinear solvers today, but with different grade of specialisations in control. Three solvers were evaluated in this thesis, Fmincon, IPOPT and PANOC. PANOC was implemented with an embedded optimizer, Optimization Engine (OpEn), which generated a NMPC in Rust code from the set up in Python.

5

Approach

This chapter describes the three different approaches that were tested. First approach was NMPC with the solver Fmincon, second was NMPC with the solver IPOPT and last was NMPC with the solver PANOC. In this chapter is the different setups for the different approaches described, and the chapter is concluded by a description of the simulation environment.

5.1 NMPC using Fmincon

This thesis further developed the method by [7]. This method was written in MATLAB and the solver was MATLAB's Fmincon. Theory about Fmincon can be seen in subsection 4.3.1 Fmincon. This thesis used the same NMPC formulations and the same setup for the controller as in [7]. The framework for the NMPC was numerical and the shooting method was multiple shooting. Theory about shooting methods is described in subsection 4.2.2 Nonlinear dynamics.

5.1.1 NMPC formulation

The model of the system dynamics was linearized along a reference trajectory using the linearization method described in subsection 3.1.3 Linearized motion model. The state and input vectors for this NMPC controller is seen in the equations below. Elevated denotes ATR_1 and ATR_2. Index in k represent discrete time,

$$\boldsymbol{x}_k = \begin{bmatrix} X_k^1 & Y_k^1 & \phi_k^1 & X_k^2 & Y_k^2 & \phi_k^2 \end{bmatrix}, \tag{5.1a}$$

$$\tilde{\boldsymbol{x}}_k = \Big[X_k^1 - X_k^{1ref} \quad Y_k^1 - Y_k^{1ref} \quad \phi_k^1 - \phi_k^{1ref} \tag{5.1b}$$

$$X_k^2 - X_k^{2ref} \quad Y_k^2 - Y_k^{2ref} \quad \phi_k^2 - \phi_k^{2ref} \Big], \tag{5.1c}$$

$$\tilde{\boldsymbol{u}}_k = \begin{bmatrix} v_k^1 - v_k^{1ref} & \omega_k^1 - \omega_k^{1ref} & v_k^2 - v_k^{2ref} & \omega_k^2 - \omega_k^{2ref} \end{bmatrix}^T, \tag{5.1d}$$

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k^{1^T} & \boldsymbol{x}_k^{2^T} & \tilde{\boldsymbol{x}}_k^{1^T} & \tilde{\boldsymbol{x}}_k^{2^T} & \tilde{\boldsymbol{u}}_k^{1^T} & \tilde{\boldsymbol{u}}_k^{2^T} \end{bmatrix}^T. \tag{5.1e}$$

The linearized motion model for the NMPC consisted of five equations as seen in (5.2). The first equation was the object function, the second the velocity and acceleration constraints, the third the system dynamic constraints and the last two were constraints for keeping the constant distance between the two ATRs. The last two were also the nonlinear constraints, since they were non convex constraints, δ was the smallest distance the ATRs were allowed to distinguish without affecting the object and l_{beam} the length of the object,

$$
\begin{aligned}
\min_{\boldsymbol{z}_k} \quad & \sum_{i=0}^{N} \frac{1}{2}\boldsymbol{z}_k^T(i) W \boldsymbol{z}_k(i); \quad i = 0, ..., N, && (5.2a)\\
\text{s.t.} \quad & A_{eq}\boldsymbol{z}_k = b_{eq}, && (5.2b)\\
& D\tilde{\boldsymbol{u}}_{\boldsymbol{k}} \leq d_k, && (5.2c)\\
& \boldsymbol{z}_k^T G \boldsymbol{z}_k - (l_{object} + \delta)^2 \leq 0, && (5.2d)\\
& -\boldsymbol{z}_k^T G \boldsymbol{z}_k + (l_{object} - \delta)^2 \leq 0. && (5.2e)
\end{aligned}
$$

5.1.2 Cost function

Multiple shooting, see subsection 4.2.2 Nonlinear dynamics was used in this approach and therefore the decision variable $\boldsymbol{z}_k$ consisted of both the control input and the states. W in the cost function, seen in Equation (5.2a), was written in a way to only penalise on $\tilde{\boldsymbol{x}}_k$ and $\tilde{\boldsymbol{u}}_k$,

$$
W = \begin{bmatrix}
\boldsymbol{0}_{2N,2N} & & & & & & \boldsymbol{0} \\
& Q_{N-1,N-1} & & & & & \\
& & Q_f & & & & \\
& & & Q_{N-1,N-1} & & & \\
& & & & Q_f & & \\
& & & & & R_{N,N} & \\
\boldsymbol{0} & & & & & & R_{N,N}
\end{bmatrix}, \tag{5.3}
$$

where

$$
Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \tag{5.4}
$$

and for the terminal cost,

$$
Q_f = \begin{bmatrix} 2000 & 0 & 0 \\ 0 & 2000 & 0 \\ 0 & 0 & 2000 \end{bmatrix}. \tag{5.5}
$$

Control penalty on u was,

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}. \tag{5.6}$$

5.1.3 Linear equality constraint

The NMPC in this method used multiple shooting and this imposed adding the system dynamics as a constraint. The linearized motion model formulated in subsection 3.1.3 Linearized motion model needed to include both ATRs and could be described as

$$Ad_k z = Bd_k, \tag{5.7}$$

where

$$Ad_k = \begin{bmatrix} \mathbf{0} & \mathbf{0} & Ag_k^1 & \mathbf{0} & Bg_k^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & Ag_k^2 & \mathbf{0} & Bg_k^2 \end{bmatrix}. \tag{5.8}$$

Here Ag_k^j looked like

$$Ag_k^j = \begin{bmatrix} -I & & & & \mathbf{0} \\ A_k^j(1) & -I & & & \\ & A_k^j(2) & -I & & \\ & & \ddots & \ddots & \\ \mathbf{0} & & & A_k^j(N-1) & -I \end{bmatrix}, \tag{5.9}$$

and Bg_k^j was described as

$$Bg_k^j = \begin{bmatrix} B_k^j(0) & & & \mathbf{0} \\ & B_k^j(1) & & \\ & & \ddots & \\ \mathbf{0} & & & B_k^j(N) \end{bmatrix}. \tag{5.10}$$

and Bd_k was constructed as

$$Bd_k = [-(A_k^1(0) \cdot \tilde{x}_k^1(0))^T \quad 0 \quad \dots \quad 0 \quad -(A_k^2(0) \cdot \tilde{x}_k^2(0))^T \quad 0 \quad \dots \quad 0]^T. \tag{5.11}$$

To ensure that

$$x_k^j - \tilde{x}_k^j = x_k^{ref,j}, \tag{5.12}$$

is held, it could be added in the equality constraint with

$$\underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}}_{C} \boldsymbol{z} = \begin{bmatrix} \boldsymbol{x}^{ref,1} \\ \boldsymbol{x}^{ref,2} \end{bmatrix}. \tag{5.13}$$

The total equality constraint, Equation (5.2b), was described as

$$\underbrace{\begin{bmatrix} Ad_k \\ C \end{bmatrix}}_{A_{eq}} \boldsymbol{z} = \underbrace{\begin{bmatrix} Bd_k \\ \boldsymbol{x}^{ref,1} \\ \boldsymbol{x}^{ref,2} \end{bmatrix}}_{b_{eq}}. \tag{5.14}$$

5.1.4 Linear inequality constraint

The linear inequality constraint consisted of the velocity constraint on the forward velocity and the angular velocity, also called the yaw rate. The linear inequality constraint also included acceleration and deceleration constraint for the ATRs. To achieve this was the constraint expressed as constraints on the tracking error $\tilde{u}$.

The velocity constraint could be expressed as

$$u_{min} \leq u_k(i) \leq u_{max}; \quad i = 0, ..., N-1, \tag{5.15}$$

and to express this as $\tilde{u}$ it could be extended with the reference as

$$u_{min} - u_k^{ref}(i) \leq u_k(i) - u_k^{ref}(i) \leq u_{max} - u_k^{ref}(i); \quad i = 0, ..., N-1. \tag{5.16}$$

The acceleration/deceleration constraint was expressed in a similar way;

$$a_{min} \leq a_k(i) \leq a_{max}; \quad i = 0, ..., N-1, \tag{5.17}$$

where

$$a_k(i) = \frac{v_k(i) - v_k(i-1)}{T_s}, \tag{5.18}$$

with the extended version as

$$\begin{aligned}
&T_s a_{min} - v_k^{ref}(i) + v_k^{ref}(i-1) \leq \\
&\leq (v_k(i) - v_k^{ref}(i)) - (v_k(i-1) - v_k^{ref}(i-1)) \leq \\
&\leq T_s a_{max} - v_k^{ref}(i) + v_k^{ref}(i-1); \quad i = 0, ..., N-1.
\end{aligned} \tag{5.19}$$

When $i = -1$ $(v_k(-1))$ in above equation, was the velocity set to the first velocity in previous time instance $v_{k-1}(0)$, thus $v_k(-1) = v_{k-1}(0)$. In the absolute first step when $i = -1$ was the velocity set to zero, $v_0(-1) = 0$, since the ATRs were assumed to start as standing still.

The total inequality constraint Equation (5.2c), was described as

$$\underbrace{\begin{bmatrix} I \\ -I \\ M \\ -M \end{bmatrix}}_{D} \tilde{\boldsymbol{u}} = \underbrace{\begin{bmatrix} \tilde{u}_{max} \\ -\tilde{u}_{min} \\ A_{max} \\ -A_{min} \end{bmatrix}}_{d_k}, \tag{5.20}$$

where

$$\tilde{u} = \begin{bmatrix} \tilde{u}_k(0) \\ \vdots \\ \tilde{u}_k(N-1) \end{bmatrix}, \tag{5.21a}$$

$$\tilde{u}_{\max} = \begin{bmatrix} u_{\max} - u_k^{ref}(0) \\ \vdots \\ u_{\max} - u_k^{ref}(N-1) \end{bmatrix}, \tag{5.21b}$$

$$\tilde{u}_{\min} = \begin{bmatrix} u_{\min} - u_k^{ref}(0) \\ \vdots \\ u_{\min} - u_k^{ref}(N-1) \end{bmatrix}, \tag{5.21c}$$

and

$$M = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & -1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & \dots & -1 & 0 & 1 \end{bmatrix}, \tag{5.22a}$$

$$A_{\max} = \begin{bmatrix} a_{\max} \\ \vdots \\ a_{\max} \end{bmatrix} - \begin{bmatrix} v_k^{ref}(1) \\ \vdots \\ v_k^{ref}(N-1) \end{bmatrix} + \begin{bmatrix} v_k^{ref}(0) \\ \vdots \\ v_k^{ref}(N-2) \end{bmatrix}, \tag{5.22b}$$

$$A_{\min} = \begin{bmatrix} a_{\min} \\ \vdots \\ a_{\min} \end{bmatrix} - \begin{bmatrix} v_k^{ref}(1) \\ \vdots \\ v_k^{ref}(N-1) \end{bmatrix} + \begin{bmatrix} v_k^{ref}(0) \\ \vdots \\ v_k^{ref}(N-2) \end{bmatrix}. \tag{5.22c}$$

5.1.5 Nonlinear constraint

The nonlinear constraint consisted of the constant distance between the two ATRs and the distance was maintained within $l_{beam} \pm \delta$ at all time. δ was assumed to be small enough to not damage the carried object. This constraint was non-convex since the constraint for one ATR could be seen as a circumference around the other ATR, with the radius $l_{beam} \pm \delta$. To become convex, the whole circle area is required to be an allowed area. The nonlinear distance constraint could be declared as

$$l_{object} - \delta \leq \sqrt{(X_k^1(i) - X_k^2(i))^2 + (Y_k^1(i) - Y_k^2(i))^2} \leq l_{object} + \delta. \tag{5.23}$$

This could be divided into two constraints,

$$\left(X_k^1(i) - X_k^2(i)\right)^2 + \left(Y_k^1(i) - Y_k^2(i)\right)^2 \leq (l_{object} + \delta)^2, \tag{5.24a}$$

$$-\left(\left(X_k^1(i) - X_k^2(i)\right)^2 + \left(Y_k^1(i) - Y_k^2(i)\right)^2\right) \leq -(l_{object} - \delta)^2. \tag{5.24b}$$

Above constraints could be redefined to be adapted to the Equation (5.2d) and (5.2e) as

$$\boldsymbol{z}_k^T G \boldsymbol{z}_k - (l_{object} + \delta)^2 \leq 0, \tag{5.25a}$$

$$-\boldsymbol{z}_k^T G \boldsymbol{z}_k + (l_{object} - \delta)^2 \leq 0, \tag{5.25b}$$

where

$$G = \begin{bmatrix} 1 & 0 & 0 & \dots & -2 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots & 0 & -2 & 0 & \dots \\ 0 & 0 & 0 & & 0 & 0 & 0 & \\ \vdots & \vdots & & \ddots & & & & \ddots \\ & & & & 1 & 0 & 0 & \dots \\ & & & & 0 & 1 & 0 & \dots \\ & & & & 0 & 0 & 0 & \\ & & & & \vdots & \vdots & & \ddots \end{bmatrix}. \tag{5.26}$$

5.2 NMPC using IPOPT

The code was rewritten from MATLAB to Python to be able to use IPOPT, described in subsection 4.3.2 IPOPT and CasADi, briefly described in section 4.4 Real-time NMPC. CasADi was used to obtain a symbolic framework and the NMPC was constructed according to CasADi guidelines with $\boldsymbol{z}_k$ as symbolic variable. The shooting method was multiple shooting, see subsection 4.2.2 Nonlinear dynamics, and the solving method for IPOPT was interior point method.

5.2.1 NMPC formulation

The state vectors for the NMPC controller were the same as in the NMPC formulation in section 5.1 NMPC using Fmincon. This formulation was a multiple shooting formulation where $\boldsymbol{z}_k$ included both the decision variables and the state variables,

$$\boldsymbol{x}_k = \begin{bmatrix} X_k^1 & Y_k^1 & \phi_k^1 & X_k^2 & Y_k^2 & \phi_k^2 \end{bmatrix}, \tag{5.27a}$$

$$\tilde{\boldsymbol{x}}_k = \Big[X_k^1 - X_k^{1ref} \quad Y_k^1 - Y_k^{1ref} \quad \phi_k^1 - \phi_k^{1ref} \tag{5.27b}$$

$$X_k^2 - X_k^{2ref} \quad Y_k^2 - Y_k^{2ref} \quad \phi_k^2 - \phi_k^{2ref} \Big], \tag{5.27c}$$

$$\tilde{\boldsymbol{u}}_k = \begin{bmatrix} v_k^1 - v_k^{1ref} & \omega_k^1 - \omega_k^{1ref} & v_k^2 - v_k^{2ref} & \omega_k^2 - \omega_k^{2ref} \end{bmatrix}^T, \tag{5.27d}$$

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{x}_k^{1^T} & \boldsymbol{x}_k^{2^T} & \tilde{\boldsymbol{x}}_k^{1^T} & \tilde{\boldsymbol{x}}_k^{2^T} & \tilde{\boldsymbol{u}}_k^{1^T} & \tilde{\boldsymbol{u}}_k^{2^T} \end{bmatrix}^T. \tag{5.27e}$$

Also the NMPC formulation was the same as in section 5.1 NMPC using Fmincon,

$$
\begin{aligned}
\min_{\boldsymbol{z}_k} \quad & \sum_{i=0}^{N} \frac{1}{2} \boldsymbol{z}_k^T(i) W \boldsymbol{z}_k(i); \quad i = 0, ..., N, && (5.28a)\\
\text{s.t.} \quad & A_{eq}\boldsymbol{z}_k = b_{eq}, && (5.28b)\\
& D\tilde{\boldsymbol{u}}_{\boldsymbol{k}} \leq d_k, && (5.28c)\\
& \boldsymbol{z}_k^T G \boldsymbol{z}_k - (l_{object} + \delta)^2 \leq 0, && (5.28d)\\
& -\boldsymbol{z}_k^T G \boldsymbol{z}_k + (l_{object} - \delta)^2 \leq 0. && (5.28e)
\end{aligned}
$$

Equation (5.28a) is described in subsection 5.1.2 Cost function, Equation (5.28b) in subsection 5.1.3 Linear equality constraint, Equation (5.28c) in subsection 5.1.4 Linear inequality constraint and Equation (5.28d) and (5.28e) in subsection 5.1.5 Nonlinear constraint.

5.3 NMPC using PANOC

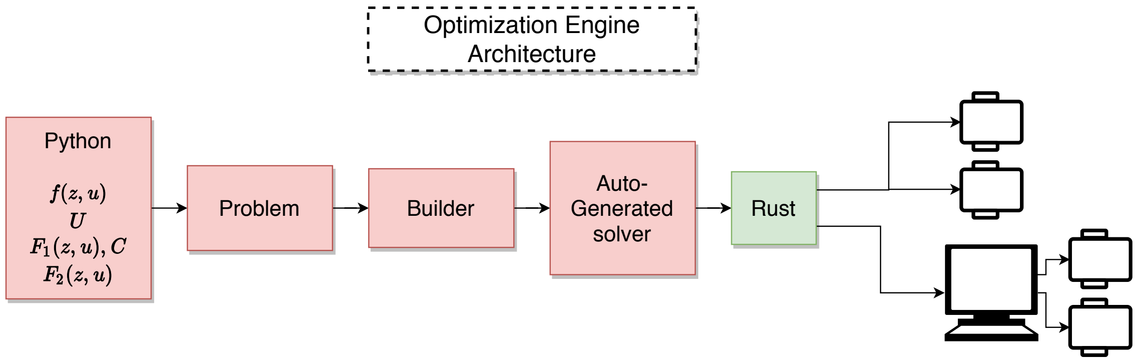


Figure 5.1: Architecture over how OpEn was set up in this thesis.

To attain an NMPC that was faster than NMPC using IPOPT or Fmincon, the proposed solution was an NMPC using Optimization Engine (OpEn). The architecture over OpEn for this thesis can be seen in Figure 5.1. The code for setting up the problem was written in Python and the problem was set up according to CasADi guidelines with symbolic variables on the input, reference trajectory, current position and previous input velocity for the ATRs. OpEn used PANOC as solver and after building and generating the solver, the interface for OpEn automatically generated a Rust code. The Rust code could then be used on embedded devices directly or on a separate computer that only send the input to the ATRs. PANOC used Newton method combined with forward-backward iterations and PANOC requred the setup to be single shooting. Theory about single shooting can be seen in subsection 4.2.2 Nonlinear dynamics.

5.3.1 NMPC formulation

The NPMC using PANOC had the state and input vectors,

$$\boldsymbol{x}_k = \begin{bmatrix} X_k^1 & Y_k^1 & \phi_k^1 & X_k^2 & Y_k^2 & \phi_k^2 \end{bmatrix}, \tag{5.29a}$$

$$\tilde{\boldsymbol{x}}_k = \Big[X_k^1 - X_k^{1ref} \quad Y_k^1 - Y_k^{1ref} \quad \phi_k^1 - \phi_k^{1ref} \tag{5.29b}$$

$$X_k^2 - X_k^{2ref} \quad Y_k^2 - Y_k^{2ref} \quad \phi_k^2 - \phi_k^{2ref} \Big], \tag{5.29c}$$

$$\boldsymbol{u}_k = \begin{bmatrix} v_k^1 & \omega_k^1 & v_k^2 & \omega_k^2 \end{bmatrix}^T, \tag{5.29d}$$

$$\tilde{\boldsymbol{u}}_k = \begin{bmatrix} v_k^1 - v_k^{1ref} & \omega_k^1 - \omega_k^{1ref} & v_k^2 - v_k^{2ref} & \omega_k^2 - \omega_k^{2ref} \end{bmatrix}^T. \tag{5.29e}$$

Direct single shooting, see subsection 4.2.2 Nonlinear dynamics, was used in this approach, and therefore was the NMPC formulations set up different from the approach using Fmincon and IPOPT. The NMPC minimised only over the decision variables $\boldsymbol{u}_k$. The states were instead predicted over the whole horizon with help of the state dynamics. Since the proposed solution used PANOC as solver, the problem had to be set up as single shooting according to [8]. Also, the system dynamics were not linearized in this approach. The reason was that the solver was fast enough to handle the nonlinear dynamics of an ATR and it was advisable to not linearize if not needed for better accuracy. A drawback with single shooting is the need of accurate system dynamics and therefore is the discretisation a critical point, even if the NMPC can handle the nonlinearity. Therefore was 4th order Runge-Kutta (RK) used for the OpEn setup instead of 1st order RK as in the multiple shooting approaches for Fmincon and IPOPT. The problem was set up in OpEn according to,

$$\min_{\boldsymbol{u}_{k,i}} \quad \sum_{i=0}^{N-1} \frac{1}{2}\Big(\tilde{\boldsymbol{x}}_k(i)^T Q \tilde{\boldsymbol{x}}_k(i) + \tilde{\boldsymbol{u}}_k(i)^T R \tilde{\boldsymbol{u}}_k(i) \Big) + \tilde{\boldsymbol{x}}_k(N)^T Q_f \tilde{\boldsymbol{x}}_k(N), \quad i = 0, ..., N-1, \tag{5.30a}$$

$$\text{s.t.} \quad \boldsymbol{x}_k(i+1) = f(\boldsymbol{x}_k(i), \boldsymbol{u}_k(i)), \tag{5.30b}$$

$$U_{min} \le \boldsymbol{u_k}(i) \le U_{max}, \tag{5.30c}$$

$$\Big(X_k^1 - X_k^2\Big)^2 + \Big(Y_k^1 - Y_k^2\Big)^2 \le (l_{object} + \delta)^2, \tag{5.30d}$$

$$-\bigg(\Big(X_k^1 - X_k^2\Big)^2 + \Big(Y_k^1 - Y_k^2\Big)^2\bigg) \le -(l_{object} - \delta)^2. \tag{5.30e}$$

5.3.2 Cost function

With OpEn, the problem was set up with single shooting, in other words, the decision variable in this case, $\boldsymbol{u}_k$, only consisted of the control input. The states were decided by the system dynamics and were implemented in the cost function as prediction of the states instead of unknown variables. The matrices Q, Q_f and R for the cost function, Equation (5.30a), is present below. The control penalty on $\boldsymbol{x}_k$ was divided into the stage cost,

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \tag{5.31}$$

and the terminal cost,

$$Q_f = \begin{bmatrix} 2000 & 0 & 0 \\ 0 & 2000 & 0 \\ 0 & 0 & 2000 \end{bmatrix}. \tag{5.32}$$

The matrix for control penalty on $\boldsymbol{u}_k$,

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}. \tag{5.33}$$

5.3.3 Nonlinear equality constraint

The Equation (5.30b), aims to describe the system dynamics, was indirect a constraint since it was used to predict the states in the cost function, Equation (5.30a), and the nonlinear inequality constraints, Equations (5.30d) and (5.30e). Equation (5.30b) was not expressed as a constraint in the code and that came from the direct single shooting formulation. Instead were the system dynamics from Equation (3.4) in subsection 3.1.1 Motion model described as a CasADi function. The discretisation used was the 4th order Runge-Kutta (RK4) as seen in Equation (3.10). RK4 was coded as a simple equation and the CasADi function of the system dynamics was used to calculate $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$.

5.3.4 Linear inequality constraint

The velocity constraint (5.30c) was set to a Rectangle constraint, which was one of the option of constraints in OpEn. The constraint on the velocity and angle velocity was explained with,

$$U_{max} = \begin{bmatrix} \boldsymbol{v}_{max} \\ \boldsymbol{\omega}_{max} \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \end{bmatrix} \cdot nu \cdot N, \tag{5.34a}$$

$$U_{min} = \begin{bmatrix} \boldsymbol{v}_{min} \\ \boldsymbol{\omega}_{min} \end{bmatrix} = \begin{bmatrix} -1 \\ -10 \end{bmatrix} \cdot nu \cdot N. \tag{5.34b}$$

Where N was the horizon length and nu the number of inputs.

5.3.5 Nonlinear inequality constraint

The distance was set as a penalty constraint, more about penalty constraint can be seen in subsection 4.3.3 PANOC. The distance constraint was,

$$l_{object} - \delta \leq \sqrt{(X_k^1(i) - X_k^2(i))^2 + (Y_k^1(i) - Y_k^2(i))^2} \leq l_{object} + \delta, \tag{5.35}$$

and was split into the two constraints in Equations (5.30d) and (5.30e),

$$\Big(X_k^1(i) - X_k^2(i)\Big)^2 + \Big(Y_k^1(i) - Y_k^2(i)\Big)^2 \leq (l_{objec} + \delta)^2, \tag{5.36a}$$

$$-\bigg(\Big(X_k^1(i) - X_k^2(i)\Big)^2 + \Big(Y_k^1(i) - Y_k^2(i)\Big)^2\bigg) \leq -(l_{object} - \delta)^2. \tag{5.36b}$$

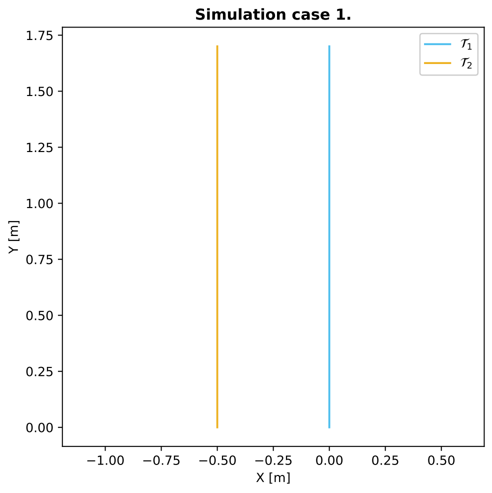
To implement Equations (5.30d) and (5.30e) as penalty constraints in OpEn, they need to be modified into,

$$max(0, ((X_k^1(i) - X_k^2(i))^2 + (Y_k^1(i) - Y_k^2(i))^2 - (l_{object} + \delta)^2)), \tag{5.37a}$$

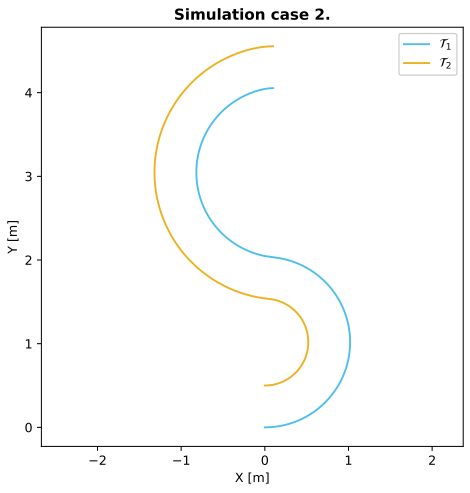
$$max(0, (-(X_k^1(i) - X_k^2(i))^2 - (Y_k^1(i) - Y_k^2(i))^2 + (l_{object} - \delta)^2)). \tag{5.37b}$$

This work, since the *max* in OpEn indicate that the constraint is not allowed to be higher than zero. All value combinations of X and Y that give the right expression in (5.37), a higher value than zero, will be implemented as forbidden, or in other words as a constraint.

5.4 Simulation



(a) Simulation case 1, where the ATRs drove straight ahead.



(b) Simulation case 2, where the ATRs followed an S-shape.

Figure 5.2: Simulation cases for the ATRs where $\mathcal{T}_1$ was the trajectory for ATR_1 and $\mathcal{T}_2$ was the trajectory for ATR_2.

Two different simulation cases were generated to see how well the different approaches worked, the first is seen in Figure 5.2a and the second is seen in Figure 5.2b. The simulations were run on a Ubuntu Linux 18.04 computer with i7 processor at 2.60 GHz up to 4.6 GHz Turbo and with the GPU Nvidia Quadro RTX 5000. In this thesis is trajectory referred to the reference positions, angles, velocities and angle velocities for the whole simulation,

$$\mathcal{T} = \begin{bmatrix} X^{j,ref} & Y^{j,ref} & \phi^{j,ref} & v^{j,ref} & \omega^{j,ref} \end{bmatrix}, \tag{5.38}$$

where j is denotes for what ATR the trajectory is for. If index k is added, it denotes the trajectory point at the k-th discrete time instance. Path is referred to the positions along the trajectory without respect of time,

$$\mathcal{P} = \begin{bmatrix} X^{j,ref} & Y^{j,ref} & \phi^{j,ref} \end{bmatrix}. \tag{5.39}$$

In both cases, the trajectories were predetermined for the simulations and were thus not real trajectories in a real environment. Trajectory 2, shown in yellow in Figure 5.2a and 5.2b, differed in Approach C compared to Approach A and B. For Approach C was trajectory 2 only a copy of trajectory 1 with only an offset of the distance between the ATRs when carrying an object, here set to $0.5[m]$. Approach A and B did require trajectory 2 to start a little earlier, which mean further down for case 1 and further to the left for case 2. This trajectory was hard coded from an already created trajectory from [7] where the trajectory was created, and all positions were optimised to find the best feasible point on the trajectory to meet the requirement from the distance constraint.

The velocity profiles of the trajectories were adapted in both cases for smooth acceleration and deceleration and were set to go from 0 m/s to 0.2 m/s and back to 0 m/s with 100 steps in the start and 200 steps in the end. This mean, with sampling time $T_s = 0.1$, that acceleration in the trajectory was set to $1[s]$ and deceleration was set to $2[s]$ which was enough for the NMPCs to act in a satisfying way.

The horizon length was set to 10 since the ATRs were assumed to be close enough to the trajectories. Sampling time was set to $0.01[s]$ in the simulation and the smallest distance the ATRs were could distinguish without affecting the object, δ, was set to $0.005[m]$. The implemented NMPC controlled the ATRs along the trajectories, to try to keep the relative distance between the robots constant. The distance between the two ATRs was measured in every time step to evaluate the error. To see how well the ATRs followed the trajectories, the Euclidean norm between the ATRs actual positions compared to the trajectories were measured to both reference trajectory point and reference path. The orientation of the ATRs compared to the trajectories reference angle was also measured, to evaluate if the ATRs were keeping the right direction along the trajectories.

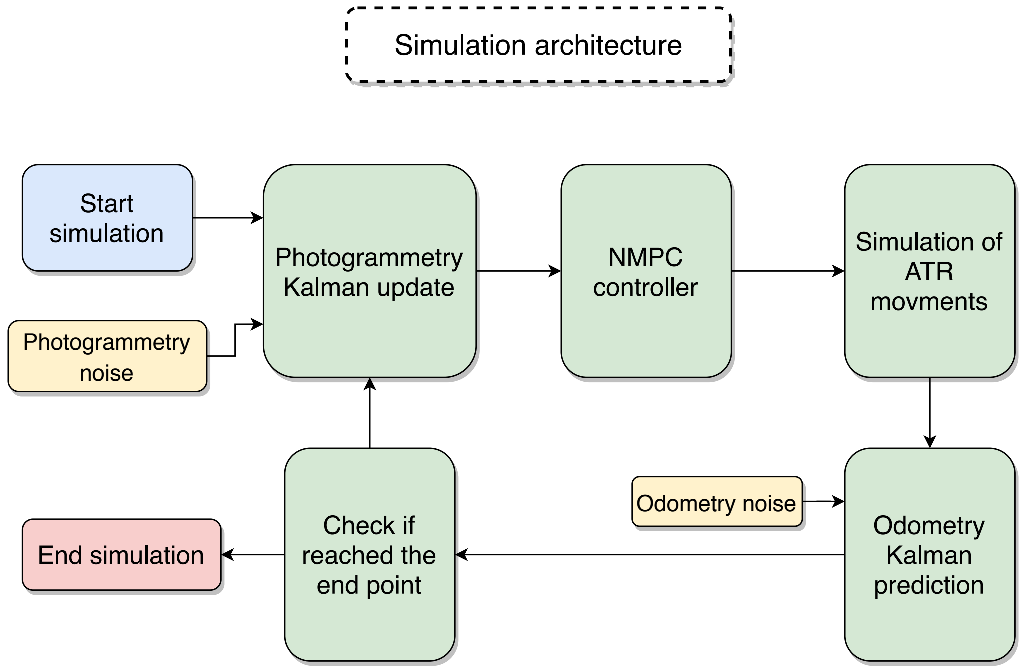

Simulation architecture
Start simulation
Photogrammetry noise
Photogrammetry Kalman update
NMPC controller
Simulation of ATR movments
Odometry noise
Odometry Kalman prediction
Check if reached the end point
End simulation

Figure 5.3: Simulation architecture, to illustrate the steps in the code.

The architecture of the steps in the code for the simulation is seen in Figure 5.3. Noise were added to the odometry, see section 2.3 Odometry, and photogrammetry, see section 2.4 Photogrammetry, to create a more realistic simulation and to verify the robustness of the NMPCs. The noises were attained from [7] and were the same in both cases and all approaches. The photogrammetry noise (R_{photo}) was determined after the worst placed AprilTag in the corner of the floors. The noise covariance for the odometry (Q_{odom}) increases with the distance measured by the odometry. The noise covariance that was adding is present in the matrices,

$$Q_{\text{odom}} = \frac{1}{764^2} \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1745 \end{bmatrix} \text{ and } \tag{5.40a}$$

$$R_{\text{photo}} = \begin{bmatrix} \frac{1}{223^2} & 0 & 0 \\ 0 & \frac{1}{265^2} & 0 \\ 0 & 0 & \frac{1}{569^2} \end{bmatrix}. \tag{5.40b}$$

The estimation of the positions by the odometry and photogrammetry was calculated by a linear Kalman filter and is described in subsection 3.2.2 Observer. In the Kalman filter was the model parameters A, B and C all defined as,

$$A, B, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.41}$$

since the updated position was only one time step away and the different between the readings can be seen as Δx. The state covariance matrix, P, attained the values from [7],

$$P = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.0001745 \end{bmatrix}. \tag{5.42}$$

After calculation of the control input from the NMPC, was the states updated in the simulation, see *Simulation of ATR movments* in Figure 5.3. This step would have been done by the ATRs itself but needed to be added in the simulation environment. To create reliable movements of the ATRs, were Exact motion along a curve or 1st order Runge-Kutta used, both described in subsection 3.1.2 Discrete motion models, depending on if the ATRs was moving straight or along a curve. The updated position was then used in the simulation of the odometry. When both ATRs were $1[cm]$ or less near the end of the trajectories, was the simulation terminated.

5.5 Summary

Three different solvers (Fmincon, IPOPT and PANOC) were tested with NMPC. Fmincon and IPOPT had the same solving method, but the problems were set up in different ways. Both IPOPT and PANOC used CasADi's symbolic framework, there Fmincon instead had a numerical framework, see table 5.1.

Table 5.1: Summary of the different solvers.

Solver:	**Fmincon**	**IPOPT**	**PANOC**
Framework:	Numerical	Symbolic	Symbolic
Solving method:	Interior-point	Interior-point	N.F.B.*
Language:	MATLAB	Python	Rust
Shooting method:	Multiple	Multiple	Single
Finding optimum:	Global	Local	Local

* *Newton method combined with forward-backward iterations*

The simulations were set up in two different cases, one where the ATRs drove straight ahead and one where the ATRs followed an S-shape. To evaluate the ATRs performance was the distance between the ATRs measured at every time step. Also, the ATRs divergence from the trajectories in distance and orientation were measured.

The values used for simulation in the three different approaches can been seen in Table 5.2.

Table 5.2: Values used in simulation.

	Values:
δ **[m]**	0.005
Distance [m]	0.5
Sampling time, T_s [s]	0.01
Horizon	10

6 Results

To evaluate the controller, different cases with trajectories were set up. The different approaches were compared in how robust the system was and the computing time. Four different errors were also set up to see how well the simulations worked for every approach. First the position error with respect to trajectory point, there the plot shows how much the two ATRs separately differ from the trajectory points. Second plot shows error in relative distance of the ATRs, to make sure that the ATRs keep the constant distance between them. Third, the angle error with respect to reference path angle for both ATRs separately, this is to ensure that the robots keep their orientation in comparison with the trajectory. Last error plot is the position error of the ATRs with respect to reference path, to see how well the ATRs followed the path. For more details on the simulations, see section 5.4 Simulation.

6.1 Approach A - NMPC using Fmincon

The results for approach A are present below. First case 1, where the ATRs drove in a straight line and second, case 2, where the ATRs drove in a S-curve.

6.1.1 Case 1

Figure 6.1 shows when the ATRs drove straight ahead.

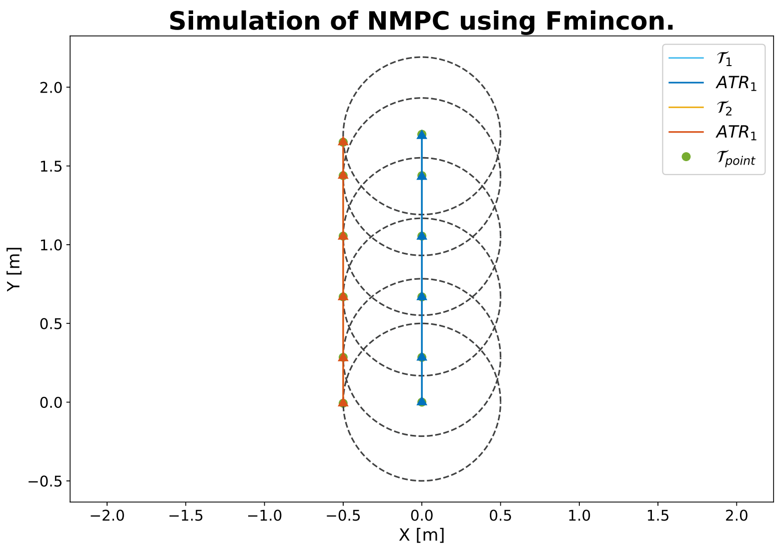


Figure 6.1: Simulation over case 1 for approach A. The black stretched circle around ATR_1, shows where ATR_2 was allowed to run to follow the constraint with constant distance between the ATRs. The triangles point the directions of the ATRs and the green dot show the corresponding trajectory point, where the triangle should be.

In Figures 6.2, 6.3, 6.4 and 6.5 are the different errors present. The max errors and computing time are present in a table in subsection 6.1.3 Summary. There was 1000 time steps for case 1.

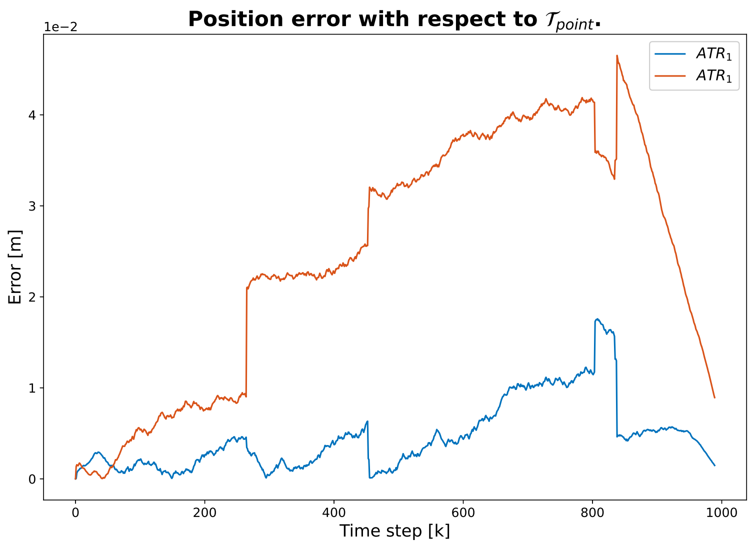


Figure 6.2: Position error with respect to trajectory point, case 1 - approach A.

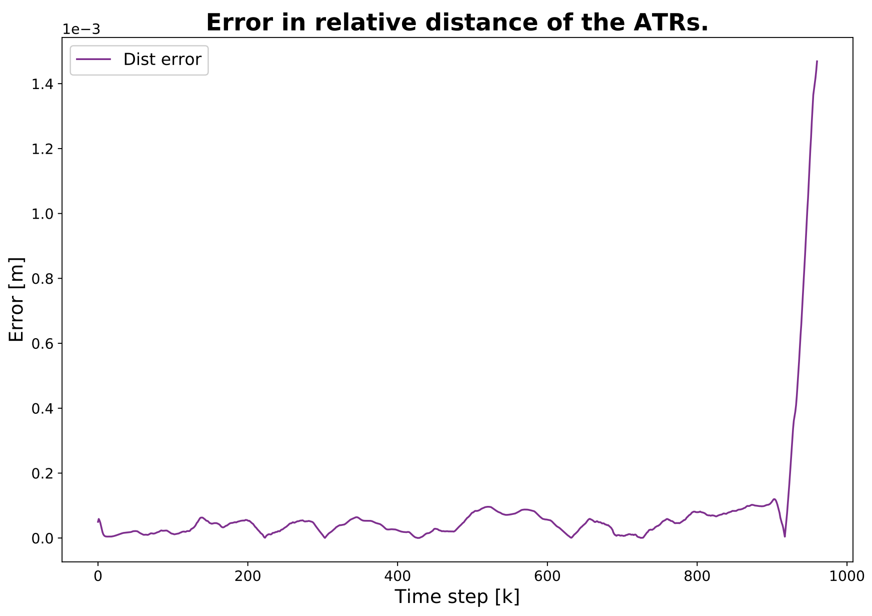


Figure 6.3: Error in relative distance of the ATRs, case 1 - approach A.

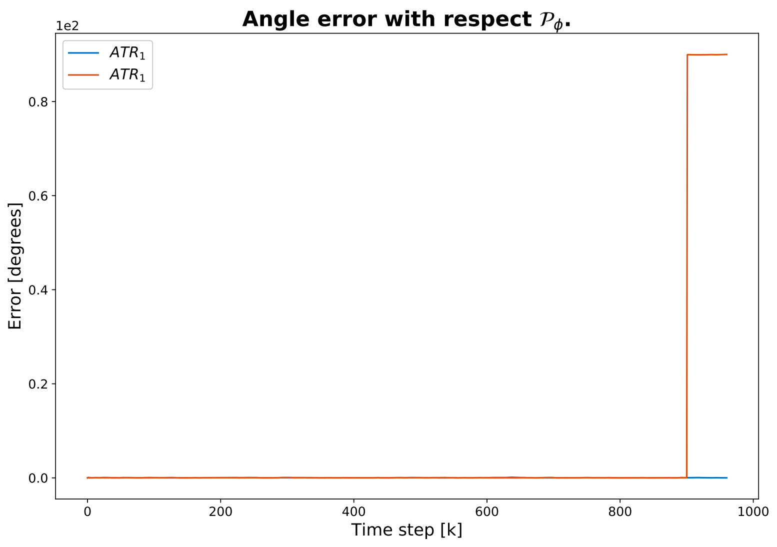


Figure 6.4: Angle error with respect to reference path angle, case 1 - approach A.

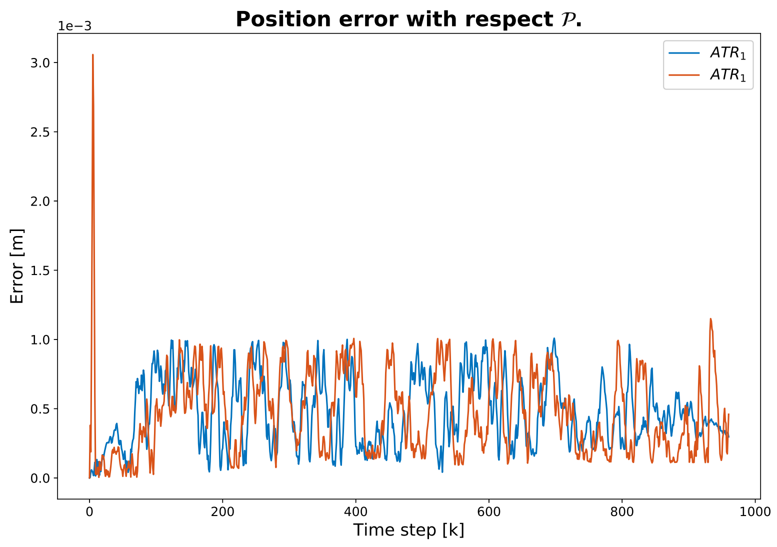


Figure 6.5: Position error with respect to reference path, case 1 - approach A.

6.1.2 Case 2

The Figure below shows when the ATRs drove in a S-curve for Approach A.

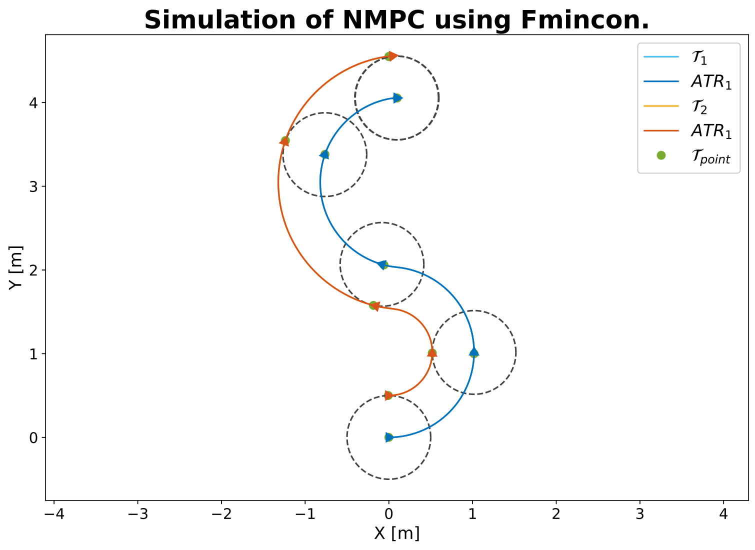


Figure 6.6: Simulation over case 2 for approach A.

In Figures 6.7, 6.8, 6.9 and 6.10 are the different errors present. The max errors and computing time are presented in Table 6.1 Approach A - NMPC using Fmincon, with noise. There was 3200 time steps for case 2.

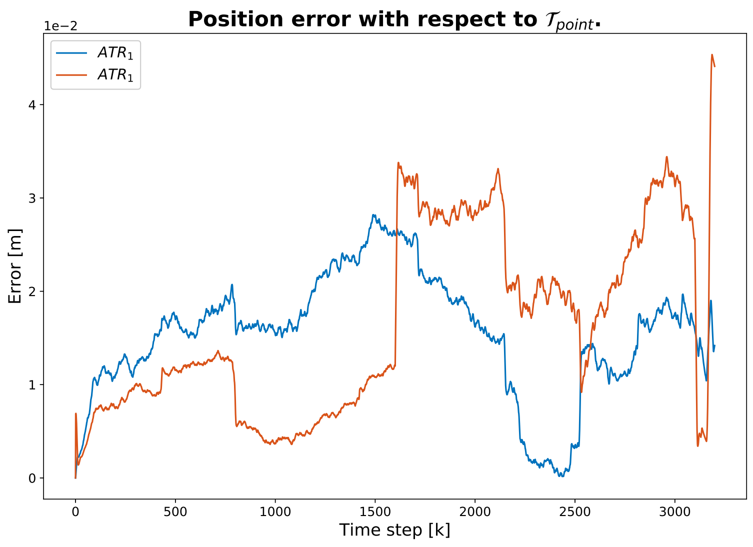


Figure 6.7: Position error with respect to trajectory point, case 2 - approach A.

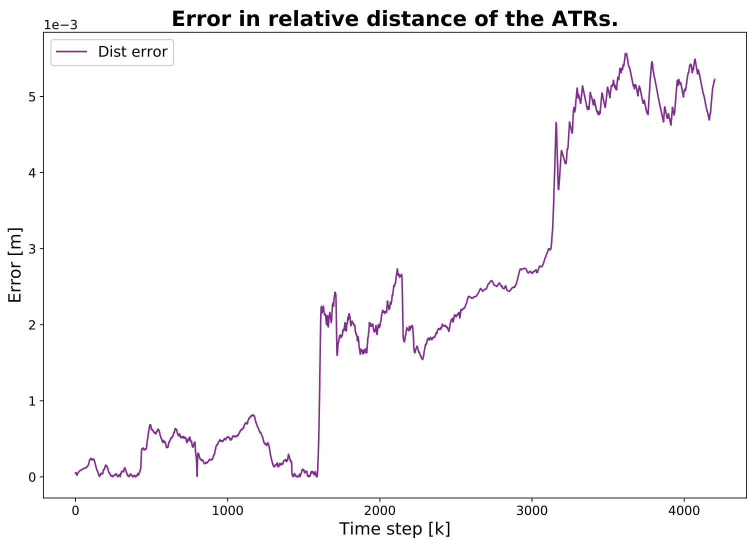


Figure 6.8: Error in relative distance of the ATRs, case 2 - approach A.

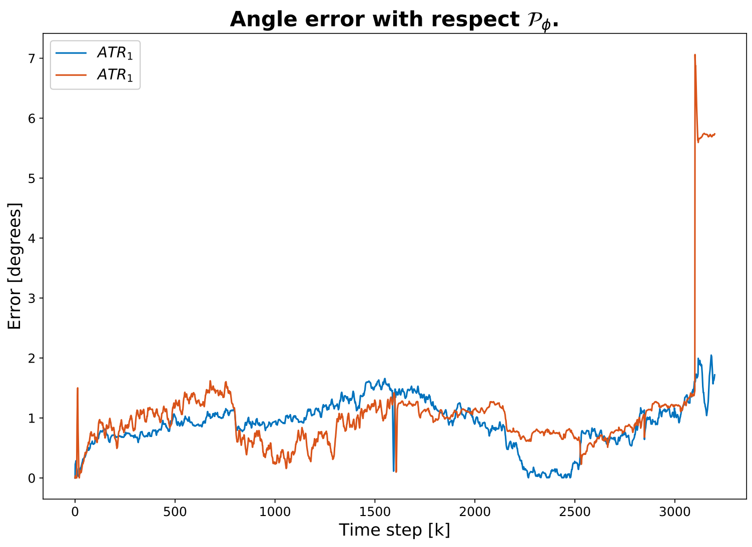


Figure 6.9: Angle error with respect to reference path angle, case 2 - approach A.

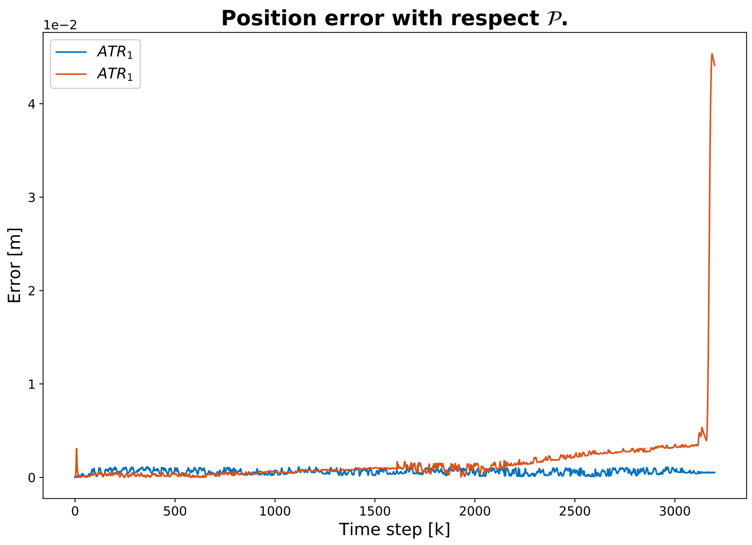


Figure 6.10: Position error with respect to reference path, case 2 - approach A.

6.1.3 Summary

The results over Approach A are summarized in Table 6.1.

Table 6.1: Approach A - NMPC using Fmincon, with noise.

Approach A		
	Case 1	Case 2
Max error ATR_1 from $\mathcal{T}_1$ $[m]$	0.0071	0.0282
Max error ATR_2 from $\mathcal{T}_2$ $[m]$	0.0075	0.0454
Max error ATR_1 from $\mathcal{P}_1$ $[m]$	0.0010	0.0011
Max error ATR_2 from $\mathcal{P}_2$ $[m]$	0.0031	0.0454
Max distance error $[m]$	0.0015	0.0056
Total simulation time $[s]$	723	3150
Average time one time step $[ms]$	753	749

6.2 Approach B - NMPC using IPOPT

6.2.1 Case 1

The figure below shows when the ATRs drives straight ahead.

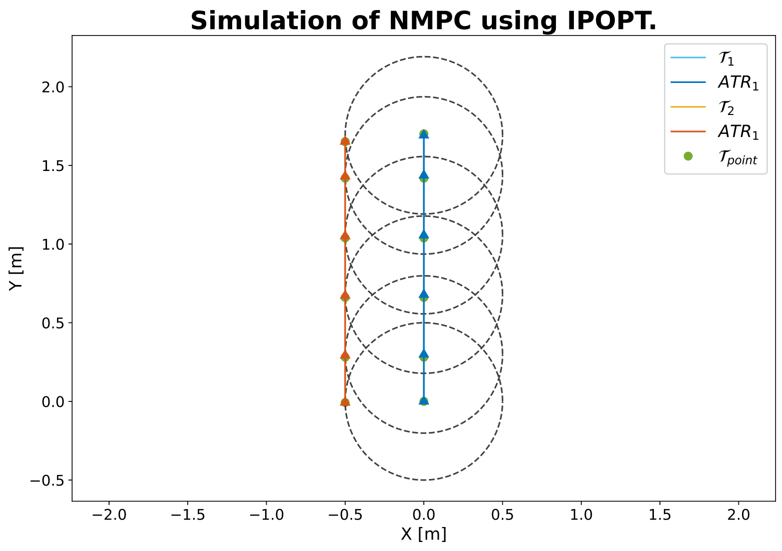


Figure 6.11: Simulation over case 1 for approach B. The black stretched circle around ATR_1, shows where ATR_2 was allowed to run to follow the constraint with constant distance between the ATRs. The triangles point the directions of the ATRs and the green dot show the corresponding trajectory point, where the triangle should be.

In Figures 6.12, 6.13, 6.14 and 6.15 are the different errors present. The max errors and computing time are presented in Table 6.2 Approach B - NMPC using IPOPT, with noise. There was 1000 time steps for case 1.

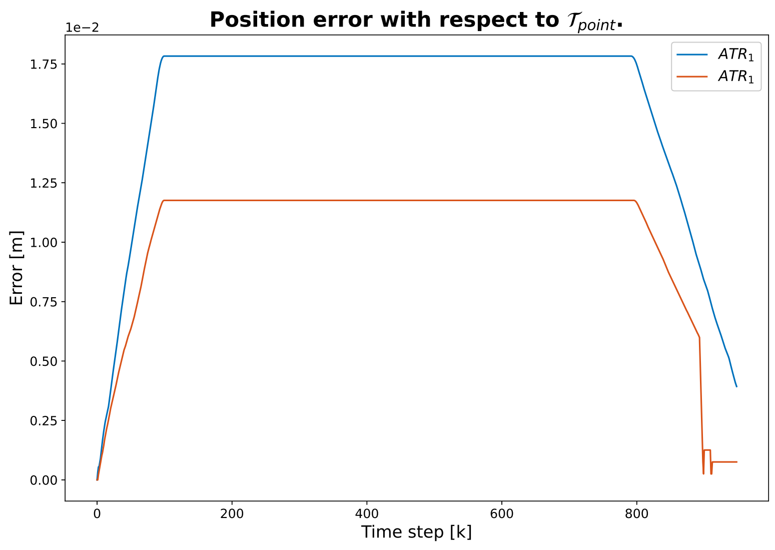


Figure 6.12: Position error with respect to trajectory point, case 1 - approach B.

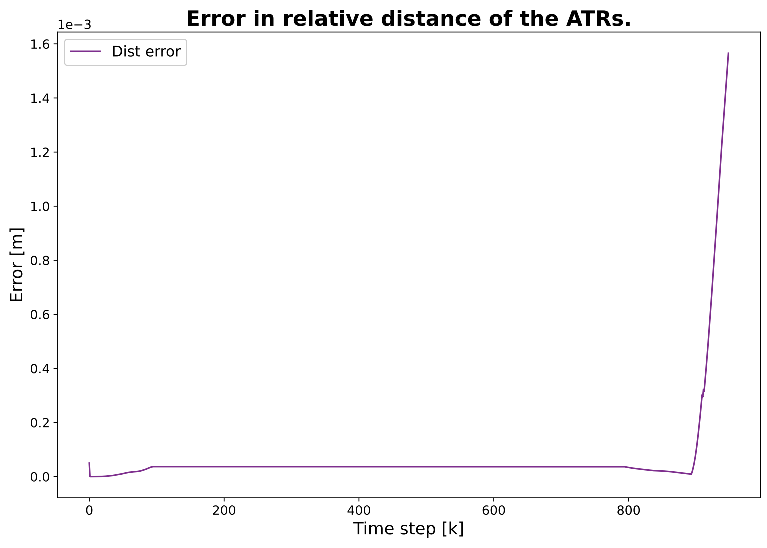


Figure 6.13: Error in relative distance of the ATRs, case 1 - approach B.

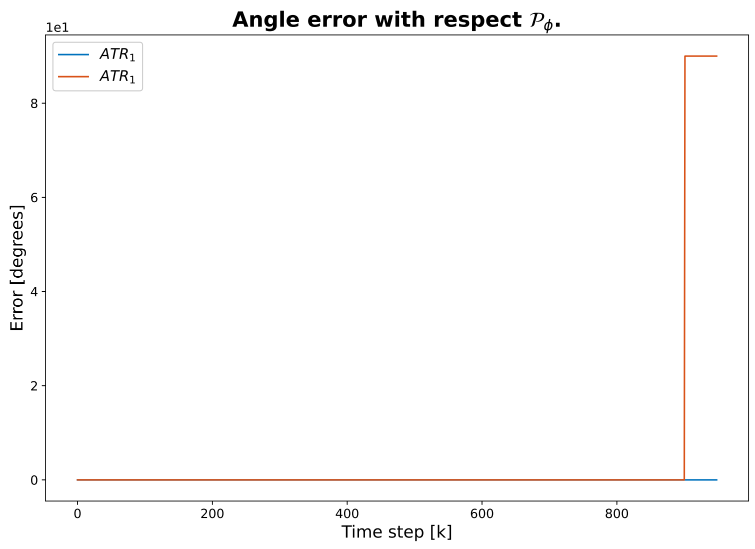


Figure 6.14: Angle error with respect to reference path angle, case 1 - approach B.

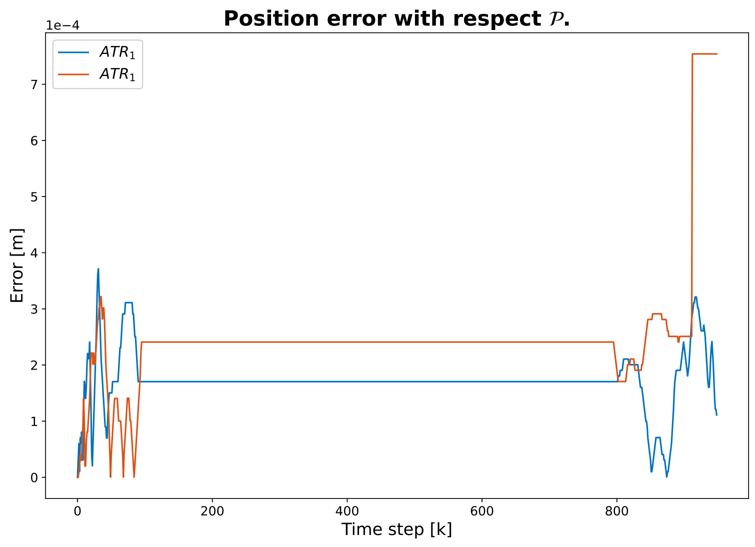


Figure 6.15: Position error with respect to reference path, case 1 - approach B.

6.2.2 Case 2

Figure 6.16 shows when the ATRs drove in a S-curve for Approach B.

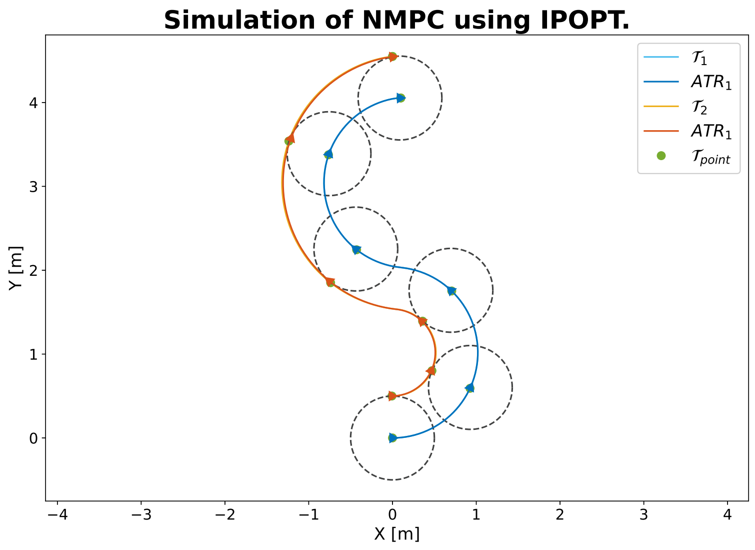


Figure 6.16: Simulation over case 2 for approach B. The black stretched circle around ATR_1, shows where ATR_2 was allowed to run to follow the constraint with constant distance between the ATRs. The triangles point the directions of the ATRs and the green dot show the corresponding trajectory point, where the triangle should be.

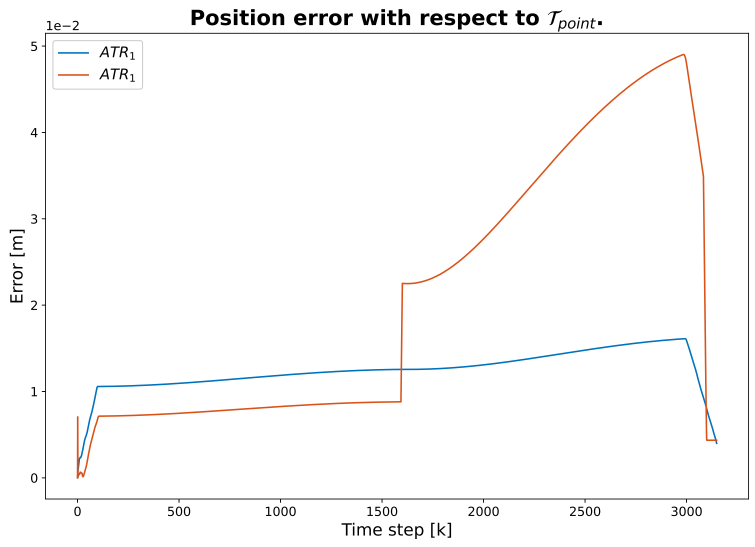


Figure 6.17: Position error with respect to trajectory point, case 2 - approach B.

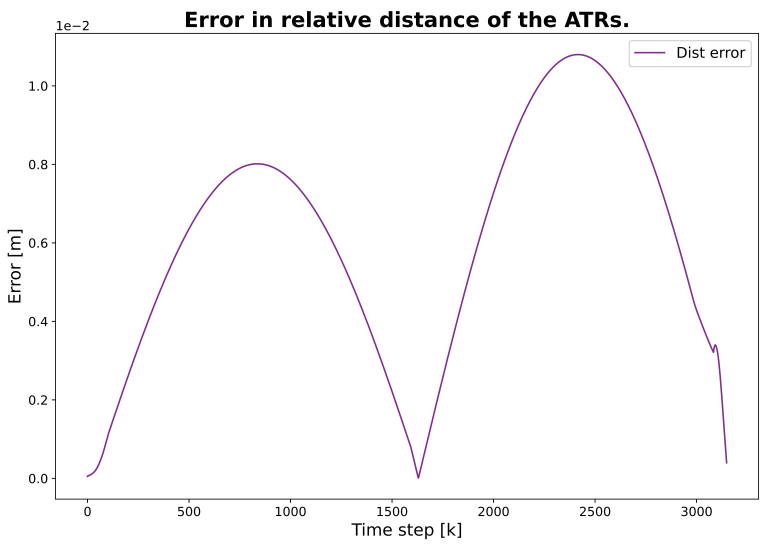


Figure 6.18: Error in relative distance of the ATRs, case 2 - approach B.

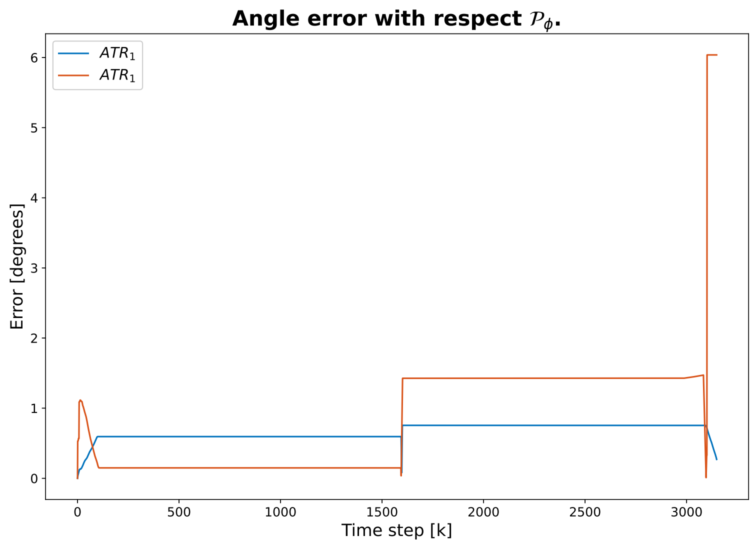


Figure 6.19: Angle error with respect to reference path angle, case 2 - approach B.

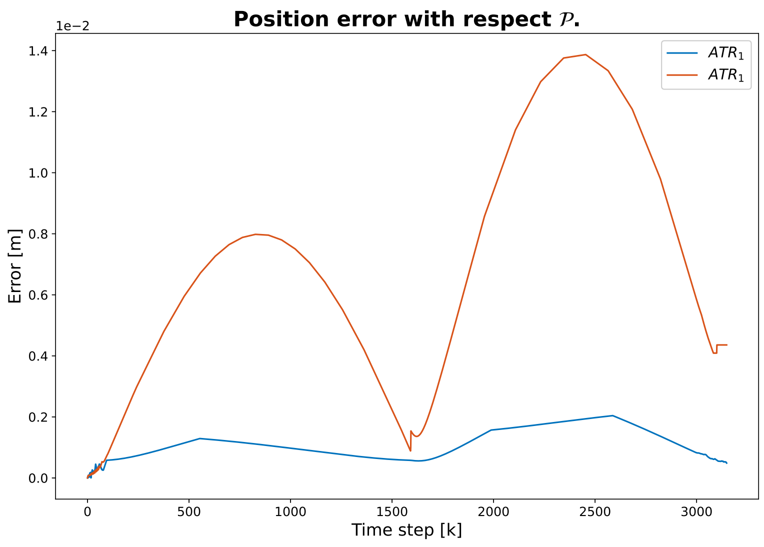


Figure 6.20: Position error with respect to reference path, case 2 - approach B.

6.2.3 Summary

The results over Approach B are summarized in Table 6.2.

Table 6.2: Approach B - NMPC using IPOPT, with noise.

Approach B		
	Case 1	Case 2
Max error ATR_1 from $\mathcal{T}_1$ $[m]$	0.0178	0.0161
Max error ATR_2 from $\mathcal{T}_2$ $[m]$	0.0118	0.0490
Max error ATR_1 from $\mathcal{P}_1$ $[m]$	0.0004	0.0020
Max error ATR_2 from $\mathcal{P}_2$ $[m]$	0.0008	0.0139
Max distance error $[m]$	0.0016	0.0108
Total simulation time $[s]$	965	2914
Average time one time step $[ms]$	1017	925

6.3 Approach C - NMPC using PANOC

The results for approach C are present below. First case 1, where the ATRs drove in a line and second, case 2, where the ATRs drove in a S-curve.

6.3.1 Case 1

Figure 6.21 shows when the ATRs drove straight ahead.

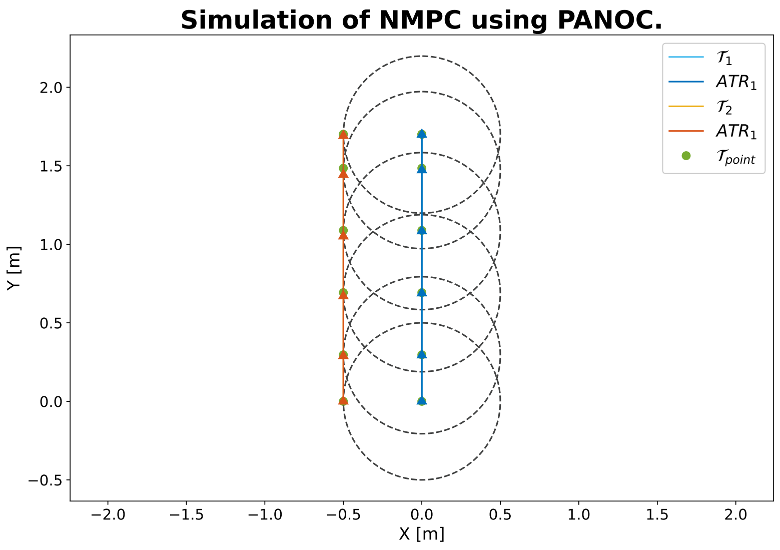


Figure 6.21: Simulation over case 1 for approach C. The black stretched circle around ATR_1, shows where ATR_2 was allowed to run to follow the constraint with constant distance between the ATRs. The triangles point the directions of the ATRs and the green dot show the corresponding trajectory point, where the triangle should be.

In Figures 6.22, 6.23, 6.24 and 6.25 are the different errors present. The max errors from every graph and computing time are presented in Table 6.3 Approach C - NMPC using PANOC, with noise. There was 1000 time steps for case 1.

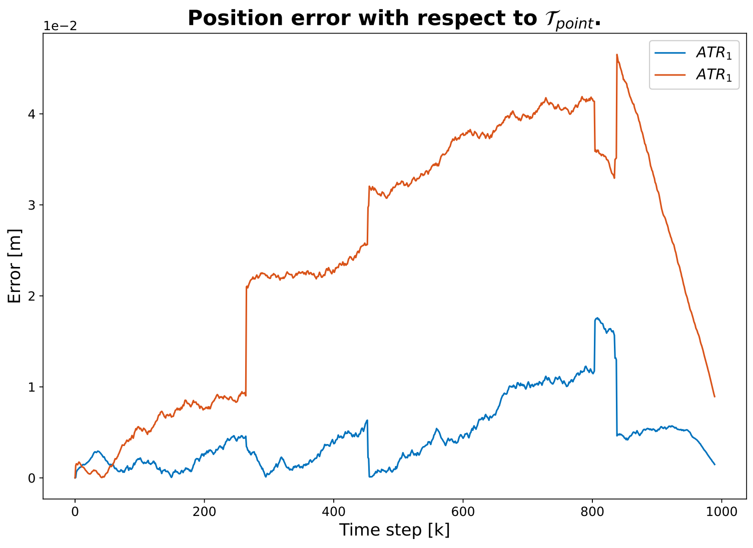


Figure 6.22: Position error with respect to reference trajectory point.

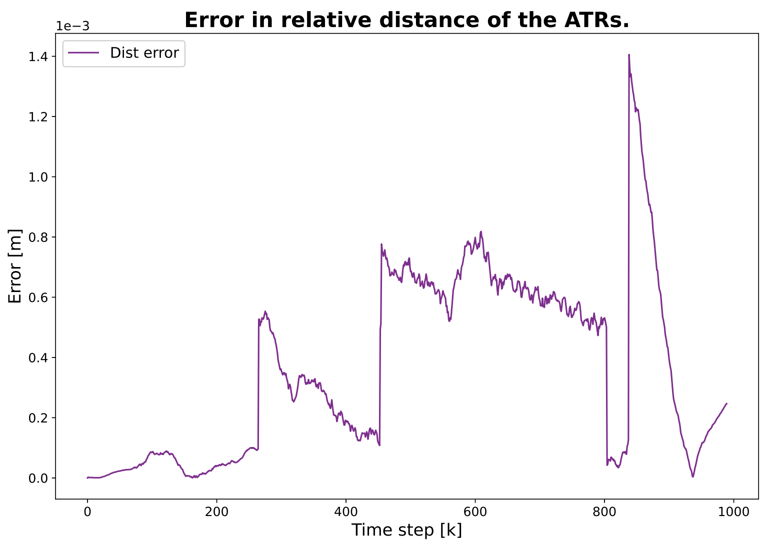


Figure 6.23: Error in relative distance of the ATRs.

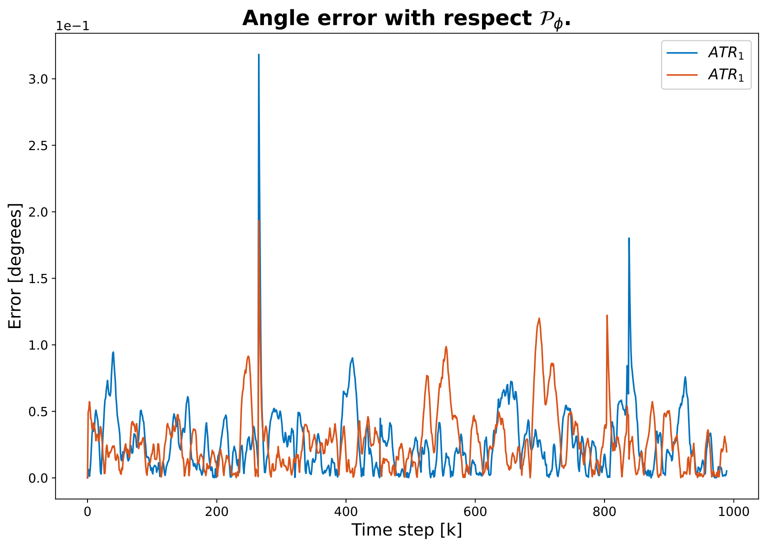


Figure 6.24: Angle error with respect to reference path angle.

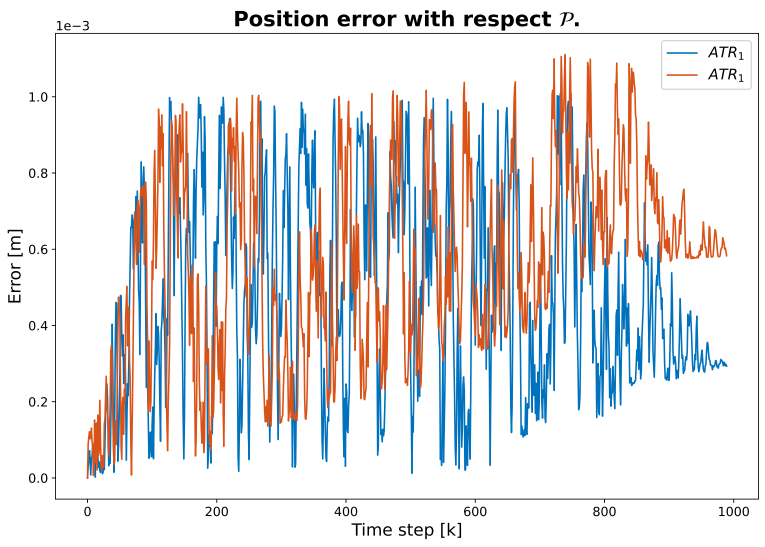


Figure 6.25: Position error with respect to reference path angle.

6.3.2 Case 2

Figure 6.26 shows when the ATRs in Approach C drives in a S-curve.

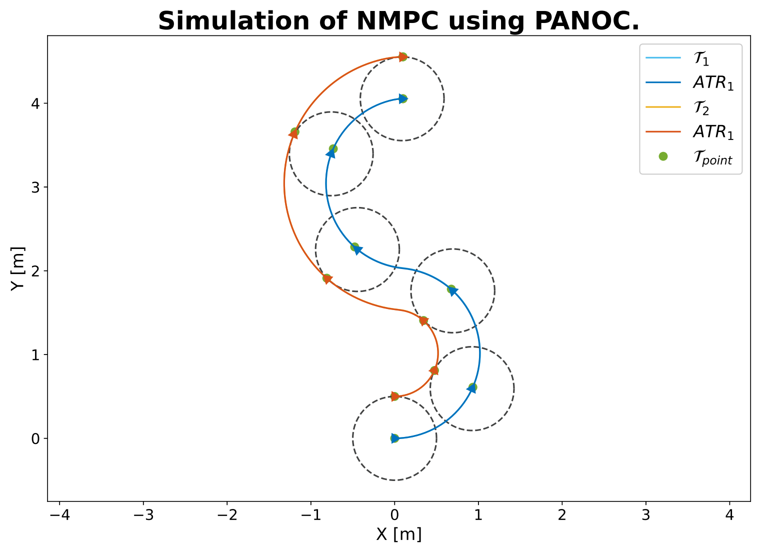


Figure 6.26: Simulation over case 2 for approach C. The black stretched circle around ATR_1, shows where ATR_2 was allowed to run to follow the constraint with constant distance between the ATRs. The triangles point the directions of the ATRs and the green dot show the corresponding trajectory point, where the triangle should be.

In Figures 6.27, 6.28, 6.29 and 6.30 are the different errors present. The max errors from every graph and computing time are present in Table 6.3 Approach C - NMPC using PANOC, with noise.. There was 3200 time steps for case 1.

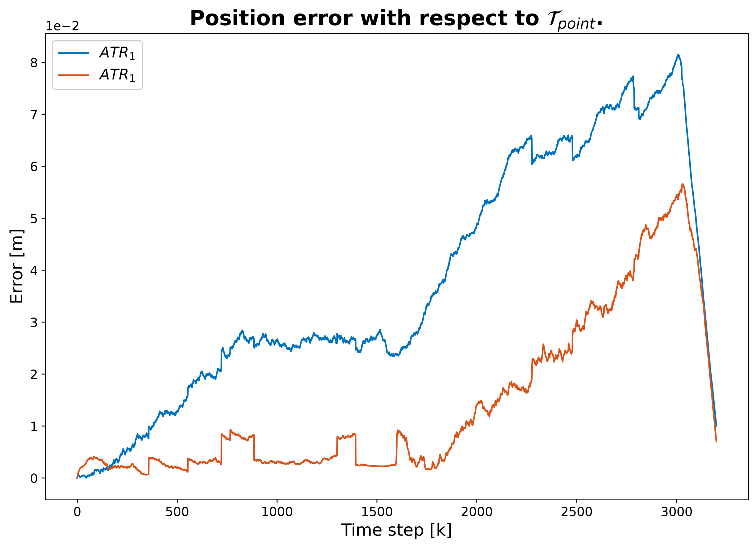


Figure 6.27: Position error with respect to reference trajectory point.

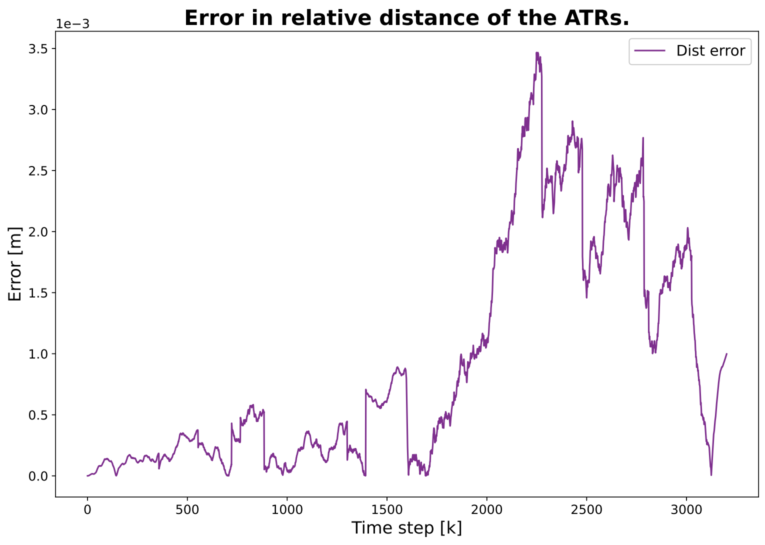


Figure 6.28: Error in relative distance of the ATRs.

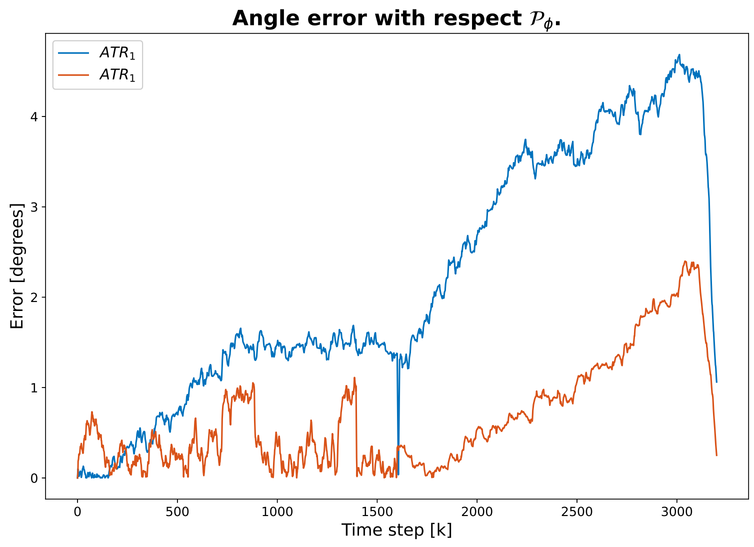


Figure 6.29: Angle error with respect to reference path angle.

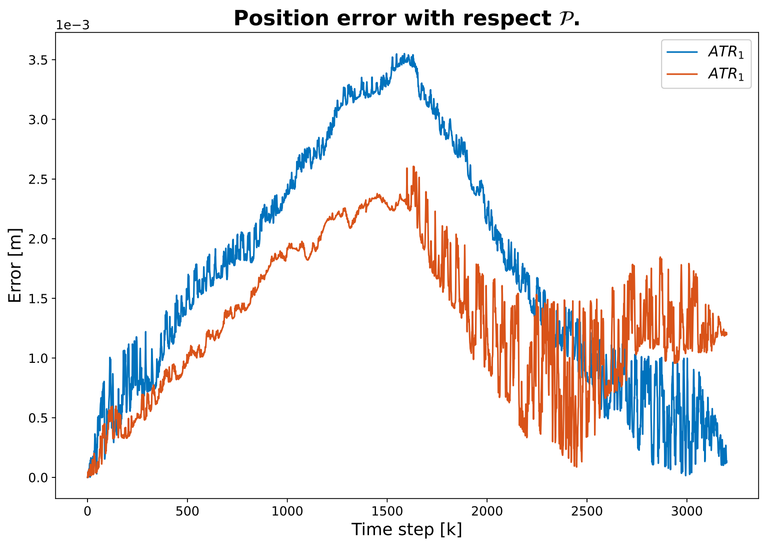


Figure 6.30: Position error with respect to reference path.

6.3.3 Summary

The result for Approach C is summarized in Table 6.3.

Table 6.3: Approach C - NMPC using PANOC, with noise.

Approach C		
	Case 1	Case 2
Max error ATR_1 from $\mathcal{T}_1$ $[m]$	0.0176	0.0815
Max error ATR_2 from $\mathcal{T}_2$ $[m]$	0.0465	0.0566
Max error ATR_1 from $\mathcal{P}_1$ $[m]$	0.0010	0.0036
Max error ATR_2 from $\mathcal{P}_2$ $[m]$	0.0011	0.0026
Max distance error $[m]$	0.0014	0.0035
Total simulation time $[s]$	6.51	16.35
Average time one time step $[ms]$	3.77	3.73

6.4 Summary

In the Tables 6.4 and 6.5, are the results from the three different approaches compared in max errors and computing time.

Table 6.4: Case 1 with noise for all approaches.

Case 1			
	Approach A	Approach B	Approach C
Max error ATR_1 from $\mathcal{T}_1$ $[m]$	0.0071	0.0178	0.0176
Max error ATR_2 from $\mathcal{T}_2$ $[m]$	0.0075	0.0118	0.0465
Max error ATR_1 from $\mathcal{P}_1$ $[m]$	0.0010	0.0004	0.0010
Max error ATR_2 from $\mathcal{P}_2$ $[m]$	0.0031	0.0008	0.0011
Max distance error $[m]$	0.0015	0.0016	0.0014
Average time one time step$[ms]$	753	1017	3.77
Total time $[s]$	723	965	6.51

Table 6.5: Case 2 with noise for all approaches.

Case 2			
	Approach A	Approach B	Approach C
Max error ATR_1 from $\mathcal{T}_1$ $[m]$	0.0282	0.0161	0.0815
Max error ATR_2 from $\mathcal{T}_2$ $[m]$	0.0454	0.0490	0.0566
Max error ATR_1 from $\mathcal{P}_1$ $[m]$	0.0011	0.0020	0.0036
Max error ATR_2 from $\mathcal{P}_2$ $[m]$	0.0454	0.0139	0.0026
Max distance error $[m]$	0.0056	0.0108	0.0035
Average time one time steps$[ms]$	749	925	3.67
Total time $[s]$	3150	2914	16.16

7 Discussion

The aim for this thesis was to control two ATRs in a robust way from A to B, with NMPC. How well it went is discussed in this chapter.

7.1 Results

The results for the three different approaches are discussed in this section.

7.1.1 Results Approach A - NMPC using Fmincon

Approach A has the average time for one elapse of $753[ms]$ for case 1 and $749[ms]$ for case 2, which is longer than the limit of $500[ms]$ that come from the controllers sampling frequency of $2[Hz]$. The Table 6.1 shows the maximum errors for both ATRs in position to trajectory point and trajectory path. The maximum error for the path for ATR_2 have one outlier error, where the errors are lower for all other time steps. This happens either at the beginning or at the end of the simulations, which is probably because ATR_2 is adjusting close to the edges of the reference path due to the added noise. Otherwise, both ATRs only differ from the path with a few millimetres in both cases.

From the trajectory points are the errors a bit higher, around $7[mm]$ for case 1 and $3-4[cm]$ for case 2. Why the error is higher for the trajectory point compared to the path depends on that the NMPC is set to punish more to follow the reference path, $Q = 100$, than to follow the reference velocities, $R = 0.1$. A few centimetres error is consider low enough to be seen as a good result.

The angle error, with respect to reference path angle, shows how good the ATR's orientation is in relation to the trajectory. For case 1 does the ATRs follow the trajectories angles well, and in case 2 does the ATRs angles varies only a few degrees from the reference, which is reasonable given the turns.

The maximum distance between the two ATRs are $1.5[mm]$ in case 1 and $5.6[mm]$ in case 2. The distance constraint in the NMPC have the tolerance of $5[mm]$ in distance between the ATRs. Why it differs more in case 2 is probably due to a more difficult trajectory and the noise imposed. A stricter constraint could resolve this, with the drawback of a slower solver.

The NMPC using Fmincon calculates the global optimum and will not be confined in a local minimum. In the Figures 6.1 and 6.6, it can be seen that this approach succeed to follow the trajectories and keep the distance constraint throughout the whole simulation. The multiple shooting method that was implemented in this approach is often seen as a more robust shooting method than the single shooting method. Multiple shooting does require good initial guesses of the states that does not lead to infeasibility, and since there are reference trajectories with reference points, can the initial guess be seen as enough accurate. This all depend on that the ATRs start very close to the trajectories and that the ATRs does not deviate too much from the reference trajectories, which was the case in this thesis's simulations. The chosen horizon length of only 10, also require that the ATRs are close to the reference trajectories. The ability to have the constant distance constraint between the ATRs as a hard constraint will secure that the object carried will not be damage. This create the problem where the trajectory must be very good with feasible points where the ATR can follow the trajectories and at the same time follow the distance constraint. The trajectory in this approach is therefore different from the trajectory used in the NMPC using PANOC and is further described in section 5.4 Simulation. To simplify the calculation and thereby achieve shorter calculation time, is the system dynamics in Approach A linearized. This does not help to achieving the goal of calculation time under 500[ms].

7.1.2 Results Approach B - NMPC using IPOPT

Approach B has the average time of 1017[ms] for one elapse for case 1 and for case 2, 925[ms], which is higher than the limit of 500[ms]. IPOPT calculate the local optimum and might be confined in a local minimum while there is a better global minimum. Finding local optimum can speed up the calculations, but might not be as robust as a global optimum solver. Otherwise is the set up nearly the same as for Approach A and therefore could it be expected that the result would be nearly the same, but the results of Approach B is not that accurate as in Approach A, see Table 6.2 for results of Approach B. The calculation time is too high, which led to the decision that another solver will be a better choice, and Approach B was not further optimised. The consequences is that Approach B has some minor errors in the code since this approach was not fully explored.

7.1.3 Results Approach C - NMPC using PANOC

Approach C has the average time for one elapse of 3.77[ms] for case 1 and 3.67[ms] for case 2, which is significantly shorter than the limit of 500[ms]. This means that this approach could be implemented in a hardware with less capacity than the hardware used in these simulations. Therefore, the NMPC using PANOC could be implemented directly on the ATRs single-board computers, but this has not been tested in this thesis.

The errors from Approach C are present in Table 6.3. The error with respect to the reference trajectories are around $5-8[cm]$, and with respect to the reference paths are around $1[mm]$. This show that the ATRs followed the path but does not totally follow the reference velocities in the trajectories. That is reasonable since the penalty is higher for following the path, than to keep the right velocities. The errors are still very low and are considered as very good results. Approach C keep the distance between the ATRs well below the $5[mm]$ limit for the maximum distance error in both cases, with maximum error around $1-3.5[mm]$. The angle errors with respect to reference path angle is small for case 1 and with some degrees error in case 2, but the orientations are still considered good.

Approach C calculates the local optimum and there is a risk that the solver will be stuck in a local minimum. The simulation show that the solver probably find global minimum in both simulations since the errors are very low and the calculation time is short, but there might be problems if there will be added obstacles or other nonlinear constraints. In this thesis are the trajectories for both ATRs assumed to be feasible and that means that the trajectories do not run into obstacles.

The shooting method is single shooting since PANOC require this. Single shooting might be problematic if the solver get close to the border of a constraint considering that the precision of the ATRs position further ahead in the horizon strongly depends on the accuracy of the system dynamic model and the discretization method. Shorter horizon is therefore preferred and the horizon of 10 steps is beneficial. Also, the discretization method chosen is the Runge-Kutta of order four that give reasonable precision.

7.2 Research questions

If the research questions are answered is discussed in this section.

7.2.1 Discussion research question 1

- **Research question 1.** How well will the NMPC using PANOC solve the problem of carrying an object from A to B compared to the NMPC using Fmincon and NMPC using IPOPT, with respect to time, robustness and distance between the ATRs?

The absolute most significant difference between the approaches was the average time for one elapse, where Approach C's average time was as short as $3.67-3.77[ms]$. The other two approaches were over the time limit ($500[ms]$), with Approach A at $749-753[ms]$ and Approach B at $925-1017[ms]$. The maximum error with respect to the reference trajectory and path was low for all approaches. The maximum distance error was low for all approaches in case 1 but have some differences between the approaches for case 2. In case 2 is only Approach C under the limit of $5[mm]$. The distance constraint is utmost relevant to achieve that two ATRs collaborate in carrying an object together. Approach A pass the constraint with $0.5[mm]$, probably due to the added noise but. Approach B has 1$[cm]$ as maximum error in distance between the ATRs, which is too high and show that there is an error in the implementation. The distance constraint is set as a hard constraint in Approach B and should not exceed with 1$[cm]$ only due to noise.

Approach A could be seen as the most robust approach with the solver Fmincon that find the global minimum, while the solvers for Approach B and C only finds the local minimum. On the other hand has global optimum solver longer calculation time, so local solver could optimise the calculation time, with the drawback that the minimisation problem needs to be formulated to eliminate the risk of reaching the wrong minimum.

The discretization method is the same for Approach A and B where it is done with Runge-Kutta (RK) of order 1, also called the forward Euler, while it differs from Approach C where it is done with the classic RK of order 4. This development is done to improve the accuracy of the discretization to improve the result with single shooting method.

The difference in robustness of shooting methods, where Approach A and B have multiple shooting while Approach C have single shooting, seems not to be that critical in the simulations of this thesis. This is probably due to the short horizon of only 10 steps, the RK4 as discretization methods and that Approach C does not need to linearize the dynamics, which make single shooting good to use. Also, multiple shooting works well for trajectory following NMPCs since initial guess of positions can be seen as accurate enough.

The single shooting together with not linearizing the dynamics in Approach C, cause the formulation and the code to be more compact and perspicuous expression than for Approach A or B. This is an advantage for further development of the NMPC in the future.

7.2.2 Discussion research question 2

- **Research question 2.** Verify the advantages, disadvantages and further development of the NMPCs using Fmincon, IPOPT and PANOC in a production plant with respect of time, robustness and safety aspects.

This thesis only covered simulations, thus might the answers differ from tests in a real production plant. The simulation results, show that calculation time was reduced with PANOC to the extent that it seems to be a reasonable time to get the ATRs to run without delay, considering the calculation time well under the limit of 500$[ms]$. Fmincom has too high calculation time for real applications and further optimisation in calculation time is needed if further developed. The simulation results show that Fmincon and PANOC is robust, since they follow the trajectories with low error and keep the distance constraint. To improve robustness for PANOC, could an acceleration constrain be implemented and further tests are desired to verify if the penalty constraint is safe in extreme situations. The development is not enough on IPOPT to make a fair comparison. [9] show that it is robust to implement PANOC, through OpEn, as the controller on a drone. It is reasonable that real testing of this thesis approach of NMPC using PANOC could be fast and robust enough for a production plant.

7.3 Future research and development

There are several possibilities to continue with this master thesis. This thesis does not cover the connection to work with ROS and test it on the ATRs in a real application.

There are still a lot left to discover with PANOC. [9] show that PANOC is fast enough to also include the trajectory planning in the NMPC. It could either be done in the same NMPC or in a separate one, where the first NMPC find the trajectory with a longer horizon and the second NMPC work as in this thesis, or that both trajectory planning and controller are in the same NMPC. The later might be the best option for simplicity. Another continuation is to see if it is possible to minimize over Δu instead of minimizing over u, this to make it easier to implement an acceleration constraint. Another way is to implement acceleration constraint as penalty constraints or include Δu in the cost function. The latter is not an acceleration constraint but will instead penalise on high acceleration. [9] also show that obstacle avoidance can be included in the NMPC as constraints in PANOC.

8

Conclusion

In this thesis is the control of two ATRs collaborate carrying an object between them, presented. The final control system is an NMPC solved in PANOC and written in Rust code. The comparison has been made between the solvers, Fmincon, IPOPT and PANOC. With the NMPC used PANOC it was possible to reduce the computation time in simulation drastically to an average on 3.67 milliseconds for one elapse, for a case when the ATRs was driven in a S-curve, and 16.16 seconds for the total time. This was under the aim for the calculation time of one elapse under 500 milliseconds. This compared to NMPC used Fmincon or IPOPT, there the average time per elapse was at least 700 milliseconds, which means around 200 time longer than for PANOC. The total time was around 190 times longer for Fmincon and IPOPT compared to PANOC. The goal of this thesis was that two ATRs should carry an object between them and be able to keep that distance constant between them, without dropping the object, from A to B. In the simulation did all three approaches with the different solvers achieve this, but PANOC showed best results, by not diffusing more than 4 millimetres from the constant distance between the ATRs, while being computationally efficient.

PANOC showed best result in the computing time since PANOC solves the problem in a completely different way than Fmincon and IPOPT. PANOC apply a Newton-method combined with forward-backward iteration, while Fmincon and IPOPT using interior-point optimization. PANOC require a symbolic problem formulation of the NMPC according to CasADi guidelines instead of numerical framework. This reduced the calculation time even further.

Bibliography

[1] P.-L. Götvall, May 2020. Private communication.

[2] G. Ullrich *et al.*, “Automated guided vehicle systems,” *Information Systems Frontiers*, vol. 17, 2015.

[3] B. Egardt, *SSY208 Model Predictive Control, Lecture Notes.* Department of Signals and Systems Chalmers University of Technology, 2017.

[4] A. Cavin, “Obstacle avoidance for mobile robots using a photogrammetry-based sensor system,” Master’s thesis, Automatic Control Laboratory, Swiss Federal Institute of Technology Lausanne, Switzerland, 2019.

[5] E. Tuci, M. H. Alkilabi, and O. Akanyeti, “Cooperative object transport in multi-robot systems: A review of the state-of-the-art,” *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[6] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, “A vision-based formation control framework,” *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 813–825, 2002.

[7] A. Fransson, M. Reidal, E. Svärling, and J. Herzfeld, “Trajectory planning and control of two connected robots,” *Design project in systems, control and mechatronics, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden*, 2020.

[8] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, “A simple and efficient algorithm for nonlinear model predictive control,” in *IEEE Conference on Decision and Control (CDC)*, pp. 1939–1944, Dec 2017.

[9] B. Lindqvist, “Combined control and path planning for a micro aerial vehicle based on non-linear mpc with parametric geometric constraints,” Master’s thesis, Department of Computer Science, Electrical and Space Engineering,Luleå University of Technology, Sweden, 2019.

[10] C. Clark, “ARW – lecture 01 odometry kinematics.” `http://www.hmc.edu/`

`lair/ARW/ARW-Lecture01-Odometry.pdf`, 2016. Accessed on: May 04, 2020.

[11] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3400–3407, IEEE, May 2011.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[13] S. K. Malu and J. Majumdar, "Kinematics, localization and control of differential drive mobile robot," *Global Journal of Research In Engineering*, vol. 14, no. 1, 2014.

[14] T. A. Johansen, "Introduction to nonlinear modelpredictive control and moving horizonestimation," in *Selected Topics on Constrained and Nonlinear Control*, ch. 5, pp. 187–239, Bratislava, Slovakia and Trondheim, Norway: STU Bratislava and NTNU Trondheim, 2011. ISBN: 978–80–968627–4–0.

[15] The MathWorks Inc., *Fmincon*. Natick, Massachusetts, United State, 2020. Accessed on: May 08, 2020.

[16] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," 2006. Accessed on: May 08, 2020.

[17] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[18] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," in *IFAC World Congress*, (Berlin, Germany), 2020.

[19] J. Andersson, J. Åkesson, and M. Diehl, "CasADi: A symbolic package for automatic differentiation and optimal control," in *Recent advances in algorithmic differentiation*, pp. 297–307, Springer, 2012.

A

Appendix 1

The derivation to exact motion a long a curve.

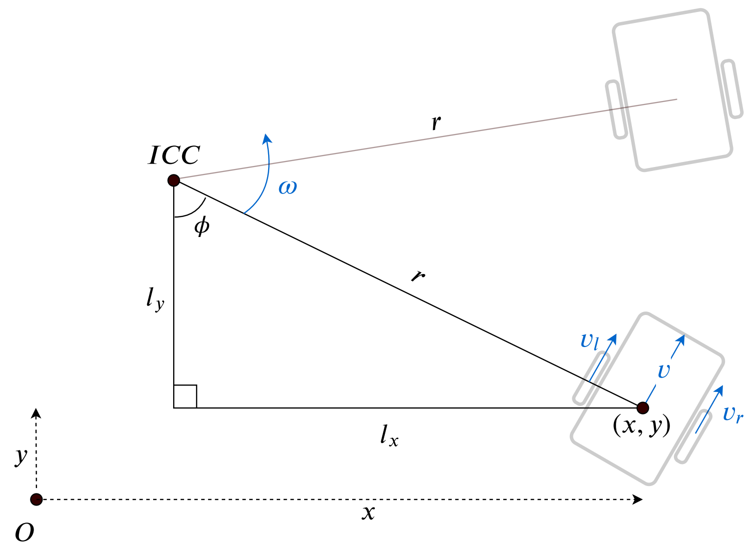


Figure A.1: The coordinates and frame's of the ATR for calculation of exact motion along a curve. ICC stands for instantaneous center of curvature.

First, the radius in discrete time was calculated between the centre of the ATR and ICC, according to figure A.1, as following,

$$r_k = \left|\frac{v_k}{\omega_k}\right|. \tag{A.1}$$

The Forward Euler method for the angle ϕ,

$$\phi_{k+1} = \phi_k + \omega_k Ts. \tag{A.2}$$

Equations (A.3) to (A.7) gave X_{k+1},

$$\sin(\phi_k) = \frac{l_{x_k}}{r_k} = \frac{X_k - ICC_{x_k}}{r_k}, \tag{A.3}$$

$$\Rightarrow ICC_{x_k} = x_k - r_k \sin(\phi_k), \tag{A.4}$$

$$ICC_{x_k} = X_k - r_k \sin(\phi_k) = X_{k+1} - r_{k+1} \sin(\phi_{k+1}). \tag{A.5}$$

If the radius of rotation was the same, $r_k = r_{k+1}$, it together with forward Euler gave the following equation,

$$X_k - r_k \sin(\phi_k) = X_{k+1} - r_{k+1} \sin(\phi_k + \omega_k Ts). \tag{A.6}$$

Finally this applied, together with (A.1),

$$X_{k+1} = X_k - \frac{v_k}{\omega_k} \sin(\phi_k) + \frac{v_k}{\omega_k} \sin(\phi_k + \omega_k T_s). \tag{A.7}$$

Equations (A.8) to (A.12) gave Y_{k+1},

$$\cos(\phi_k) = \frac{l_{y_k}}{r_k} = \frac{ICC_{y_k} - Y_k}{r_k}, \tag{A.8}$$

$$\Rightarrow ICC_{y_k} = Y_k + r_k \cos(\phi_k), \tag{A.9}$$

$$ICC_{y_k} = Y_k + r_k \cos(\phi_k) = Y_{k+1} + r_{k+1} \cos(\phi_{k+1}). \tag{A.10}$$

If rotation radius remained the same, $r_k = r_{k+1}$, then it gave, together with forward Euler, following equation,

$$Y_k + r_k \cos(\phi_k) = Y_{k+1} + r_{k+1} \cos(\phi_k + \omega_k Ts). \tag{A.11}$$

Finally this applied, together with (A.1),

$$Y_{k+1} = Y_k + \frac{v_k}{\omega_k} \cos(\phi_k) - \frac{v_k}{\omega_k} \cos(\phi_k + \omega_k T_s). \tag{A.12}$$

All this together gave the exact motions along a curve,

$$X_{k+1} = X_k - \frac{v_k}{\omega_k} \sin\left(\phi_k\right) + \frac{v_k}{\omega_k} \sin\left(\phi_k + \omega_k T_s\right), \tag{A.13a}$$

$$Y_{k+1} = Y_k + \frac{v_k}{\omega_k} \cos\left(\phi_k\right) - \frac{v_k}{\omega_k} \cos\left(\phi_k + \omega_k T_s\right), \tag{A.13b}$$

$$\phi_{k+1} = \phi_k + \omega_k T_s. \tag{A.13c}$$