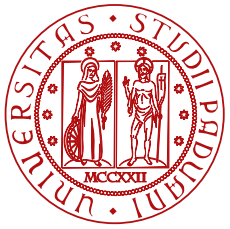


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Bachelor Thesis in Information Engineering

Development and testing of an MPC attitude controller for an ROV

Candidate

Lorenzo Favaron

Student ID 1095508

Supervisor

Prof. Damiano Varagnolo

University of Padova

Academic Year
2022/2023

Contents

1 Abstract

Marine exploration utilizing Remotely Operated Vehicles (ROVs) has become integral to understanding the oceanic environment. Coordinating multiple vehicles concurrently is a critical aspect of enhancing exploration efficiency. This thesis focuses on the development and testing of a Model Predictive Control (MPC) combined controller for an Autonomous Underwater Vehicle (AUV) and an Uncrewed Surface Vehicle (USV) ROV.

Recognizing the importance of coordinated marine exploration, the research investigates the implementation of an MPC algorithm to control both AUVs and USVs simultaneously. The study is conducted in simulated environments to assess the controller's performance under varying conditions.

The development process involves designing a versatile MPC controller capable of effectively coordinating the movements of AUVs and USVs. The goal is to optimize their trajectories and maintain synchronized exploration patterns and a stable optical based connection. The unique dynamics of each vehicle type are considered to ensure adaptability and robust performance in the challenging underwater environment.

This research contributes to the advancement of marine exploration by addressing the challenges of coordinating multiple vehicles. The MPC combined controller offers a possible solution for optimizing exploration missions, providing a foundation for further developments in autonomous marine systems.

2 Introduction

2.1 Marine Exploration

In the latest years marine exploration is finding an increasing representation in research fields, both in naturalistic and technical disciplines. As our knowledge grows, we need more and more powerful tools and techniques to explore marine depths, and underwater robotics is the main technology field that nowadays can help in this expanding area of research.
With great effort, new vehicles and tools are developed to allow us to reach previously impossible locations and, by doing so, more and more challenges are faced every day.

2.1.1 Communications Limitation

The aquatic realm heavily restricts the direct communication capabilities with underwater vehicles, necessitating alternative solutions for remote operability. To overcome this environmental limitation a solution proposed is to employ a surface vessel as an intermediary receiver.
In order to do so is mandatory to ensure that the two vehicles can establish a stable and reliable optic communication line

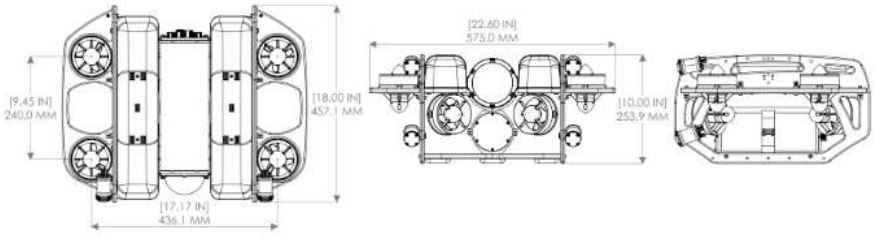
For this purpose two different ROVs are involved: an underwater one (BlueROV2 Heavy) and a surface one (BlueBoat)

The choice on this two vehicles is made due to their large diffusion in this area of research and the broad and reliable computational and mathematical documentation available. Their capability to mount different kind of sophisticated sensors and operativity tools made them a staple in the marine exploration environment. It can be operated both in naturalistic scenarios and technical support and maintainment of complex structures and large vehicles.

This, combined with their small dimensions and weight, sets for the perfect couple for this experimentation.

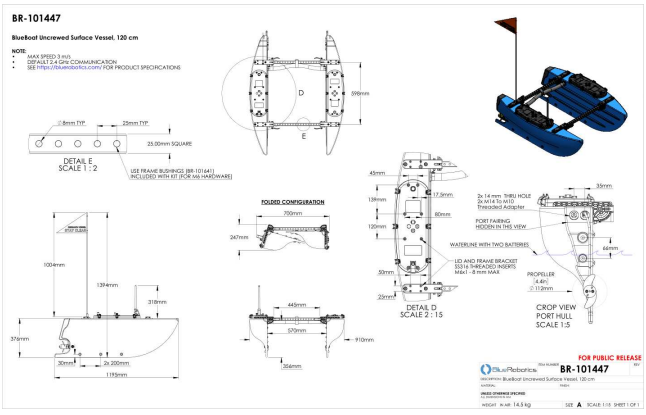
2.1.2 BlueROV2 Heavy - AUV (Autonomous Underwater Vehicle)

The BlueROV2 Heavy is a modular vehicle, customizable with different thrusters and accessories possible configurations. Thanks to the significant number of different thrusters, a variety of movements and rotations can be achieved with speed and precision. The high quality performance of this vehicle is achieved through an high precision set of sensors equipped to it. Pressure, location, orientation, depth and temperature are monitored instant by instant and provided to the processor, that can elaborate and produce the optimal control action. For this project is assumed a setup with eight thrusters, four installed along the z-axis and four in the xy-plane



2.1.3 BlueBoat - USV (Unmanned Surface Vehicle)

The BlueBoat is a surface vehicle equipped with two surge thrusters on the x-axis, that can induce sway and yaw movements by providing different push. It can also be further equipped with different attachments thanks to the several free slots.
As its underwater counterpart, it is equipped with a high precision set of sensors to reliably esteem the vehicle's states and compute the best control inputs.
In the following test it will be assumed the USV's optic communication receiver to be mounted underwater, ad the bottom of its center of mass, pointing downwards



3 Mathematical Model

An accurate mathematical model is the most essential part of the development of an MPC controller. As all the predictions are based on it, a wrong model is often worst than no model at all. So it is crucial to ensure the most accurate depiction of the system.

Every vehicle needs to be described in both its position and orientation, and to do so, we need at least two different reference frames

Also the input vector needs to be mapped in the forces and momentums acting on the vehicles. We want do control the AUV with six degrees of freedom and therefor eight thrusters are equipped on it. The boat, instead, by only having two dimensional dynamics can be maneuvered with only two parallel surge thrusters

To reduce the complexity of the calculations, some assumptions are made on the vehicles

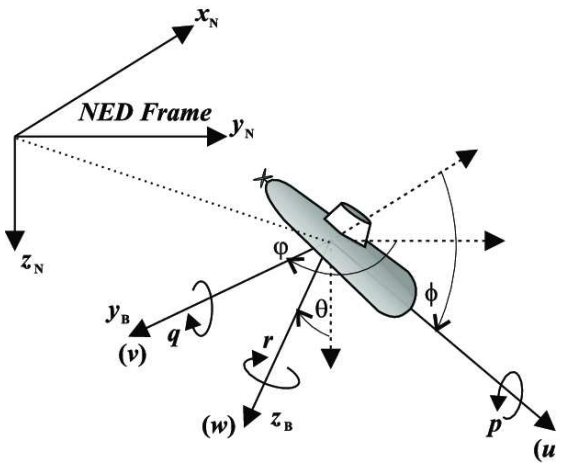
- Symmetry: The vehicles have symmetries that guarantees simpler inertia calculations along determined axes
- Disturbances: The currents affecting the trajectories are considered constant and irrotational for the simulation
- Damping: The AUV damping is quadratic, while the USV one is linear. This can be done thanks to the buoyancy of the boat, where the non linear damping only enhances the stability
- Both vehicles share the same absolute origin for the NED frame, while each one has it's own relative BODY frame
- The angular dimensions are expressed through Euler angles. While quaternion rotations can be implemented to avoid gimbal lock. they also significantly increase the model's coordinate conversion complexity and are not implemented in this occasion.

3.1 Reference frames

In order to describe position and attitude of the rov it is necessary to define two reference frames

- BODY frame: The equations of motion of the ROVs are parametrized in their BODY frame, which is centered on the vehicle and fixed to its orientation.

- NED frame: Its a coordinate reference fixed on the water surface and pointing at North-East-Down positive directions



3.2 AUV Model

The mathematical model of the AUV is derived from T. I. Fossen's Handbook of Marine Craft Hydrodinamics and Motion Control and combines the vehicle's rigid body kinetics, hydrostatics and hydrodinamics

$$M_{\mathrm{RB}}\dot{\nu} + M_{\mathrm{A}}\dot{\nu} + C_{\mathrm{RB}}(\nu)\nu + C_{\mathrm{A}}(\nu)\nu + D_{\mathrm{L}}(\nu)\nu + D_{\mathrm{NL}}(\nu)\nu + g(\eta) = \tau$$

Description	**Parameter**
NED frame position	$p = [x, y, z]^\top$
NED frame Euler angles	$\Theta = [\phi, \theta, \psi]^\top$
NED frame position and orientation	$\eta = [p, \Theta]^\top$
BODY frame linear velocity	$v = [u, v, w]^\top$
BODY frame angular velocity	$\omega = [p, q, r]^\top$
BODY frame velocity vector	$\nu = [v, \omega]^\top$
Gravitational force	W
Buoyancy force	B
Origin of the body frame	CO
Center of Gravity relative to CO	$CG = [x_g, y_g, z_g]^\top$
Center of Buoyancy relative to CO	$CB = [x_b, y_b, z_b]^\top$

$$\begin{bmatrix}\dot{\eta}\\ \dot{\nu}\end{bmatrix} = \begin{bmatrix}\dot{p}\\ \dot{\Theta}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}J_q(\eta)\nu\\ -\left(M_{\mathrm{RB}}+M_{\mathrm{A}}\right)^{-1}\left(C_{\mathrm{RB}}(\nu)\nu + C_{\mathrm{A}}(\nu)\nu + D_{\mathrm{L}}(\nu)\nu + D_{\mathrm{NL}}(\nu)\nu + g(\eta) - \tau\right)\end{bmatrix}$$

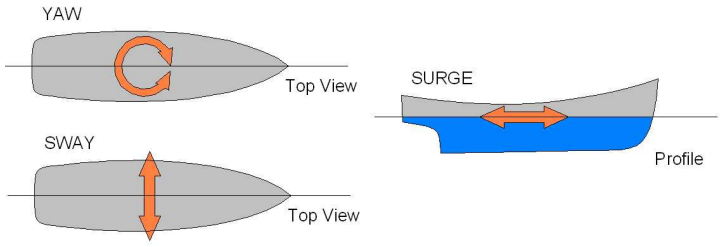
Parameter	**Description**
M_{RB}	Inertia matrix
M_{A}	Added mass
C_{RB}	Rigid body Coriolis centripetal matrix
C_{A}	Coriolis centripetal matrix
D_{L}	Linear damping
D_{NL}	Non-linear damping

The input vector τ must now be mapped into the three forces and momentums acting on the rov through the 8x6 matrix defined by the orientation and position of the eight thrusters

$$T_e = \begin{bmatrix} 0.7071 & 0.7071 & 0.7071 & 0.7071 & 0 & 0 & 0 & 0 \\ -0.7071 & 0.7071 & 0.7071 & -0.7071 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 \\ 0.0601 & -0.0601 & -0.0601 & 0.0601 & -0.2180 & -0.2180 & 0.2180 & 0.2180 \\ 0.0601 & 0.0601 & 0.0601 & 0.0601 & 0.1200 & -0.1200 & 0.1200 & -0.1200 \\ -0.1888 & 0.1888 & -0.1888 & 0.1888 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.3 USV Model

Due to the nature of it's behaviour, the BlueBoat can be described by only two-dimensional dynamics. It's other three degrees of freedom, pitch, roll and heave can be approximated as absent in this simulation's scenario.



Description	Parameter
NED frame position	$p = [x, y]^\top$
NED frame angle	$\Theta = [\psi]$
NED frame position and orientation	$\eta = [p, \Theta]^\top$
BODY frame linear velocity	$v = [u, v]^\top$
BODY frame angular velocity	$\omega = [r]$
BODY frame velocity vector	$\nu = [v, \omega]^\top$
Origin of the body frame	CO
NED frame currents	$[V_x, V_y]^\top$
Actuator configuration matrix	$B \in R^{3x2}$
Surge thrusts	$f = [T_1, T_2]^\top$
Inertia matrix	$M = [m_1, m_2, m_3]$
Damping matrix	$D = [d_1, d_2, d_3]$

The following mathematical model of the dynamics of the boat is derived from [1] in the form

$$\dot{\eta} = R(\psi)\nu + V \tag{1}$$

$$M\dot{\nu} + C(\nu)\nu + D\nu = Bf \tag{2}$$

Where M, as the inertia and mass matrix, already includes the hydrodynamics added mass.

The latter model can now be further expanded into

$$\dot{x} = u\cos(\psi) - v\sin(\psi) + V_x \tag{3}$$
$$\dot{y} = u\sin(\psi) + v\cos(\psi) + V_y \tag{4}$$
$$\dot{\psi} = r \tag{5}$$
$$\dot{u} = F_u(u, v, r) + \tau_u \tag{6}$$
$$\dot{v} = F_v(u, v, r) + \tau_v \tag{7}$$
$$\dot{r} = F_r(u, v, r) + \tau_r \tag{8}$$

with

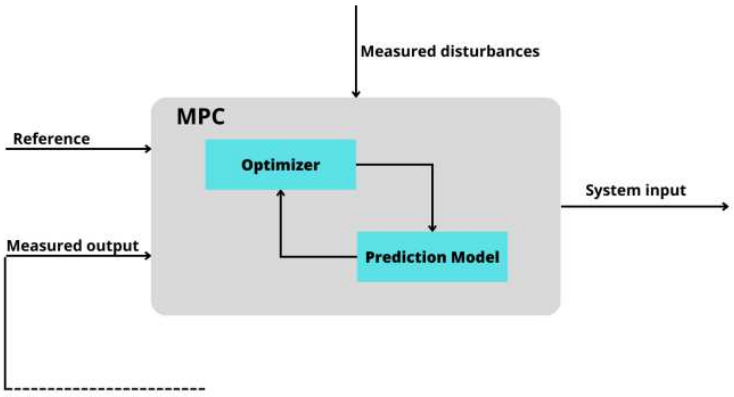
$$F_u(u, v, r) = rv\frac{m_2}{m_1} - u\frac{d_1}{m_1} \tag{9}$$
$$F_v(u, v, r) = -u\frac{m_1}{m_2} - v\frac{d_2}{m_2} \tag{10}$$
$$F_r(u, v, r) = -r\frac{d_3}{m_3} + vu\frac{m_2 - m_1}{m_3} \tag{11}$$

4 Model Predictive Control

To proceed in the experimentation a control approach must be defined. The most widespread control technique, in all scenarios, marine included, is PID (Proportional Integral Derivative) control. It requires no specific knowledge of the system's model and can handle different kind of control problems. Thanks to it's simple and flexible structure it can provide good performance, while having low-level computational requirements.

On the other hand, Model Predictive Control (MPC) is a control technique based on the knowledge of the system's model and through a step by step simulation and solution of the system's equations can have enhanced performances, at the cost of an exponential increase of the computational cost of the control action. It is so necessary to define and solve an optimization problem, to chose the ideal system inputs. MPC can also handle dynamic models of the system, by defining time-varying parameters and incorporating them in the model's equations. This allows for a better description of the transitions of the states and therefor a better adaptation to unexpected (or expected) changes to the model and the environment.

To evaluate the control action a cost function needs to be defined. It is the core of the whole problem and should describe the most unwanted scenarios we might encounter so that the localization of its minima can identify the optimal control inputs. The more we step away from the target trajectory, the more costful it will result for the function. This tool can also be utilized to set bounds for both the states and the inputs. By tuning the coefficients of those weight we can further control precision and smoothness of the control action and the system trajectories.



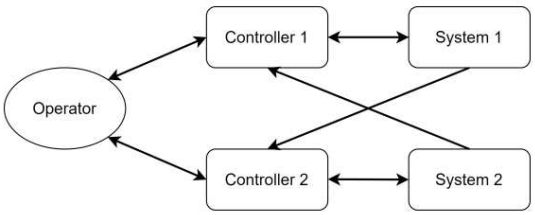
When dealing with two different vehicles it will be required to define two different cost function, embedded in the same framework, and adapting to each

other trajectory

4.1 Control Target

This project has two different target for the two ROVs. The solution adopted was a decentralized control approach. Instead of a single operator tht controls the movement of both vehicles, each one can recive information about other vehicle and autonomously calculate its own optimized inputs. The AUV must reach and follow a target location or path. The USV must follow it on the water surface, while remaining in range to maintain the optical tether.

While the boat has no freedom of vertical orientation, the AUV must also remain within a limited depth and oriented towards the bottom of the USV for the better possible data stream.



4.2 Cost Functions

Do-mpc provides a template for the implementation of the cost functions, to allow a more clear declaration

$J(x,u,z) = \sum_{k=0}^{N} \left(l(x_k, z_k, u_k, p_k, ptv_k) + \Delta u_k^T R \Delta u_k \right) + m(x_{N+1})$
Where

- $l(x_k, z_k, u_k, p_k, ptv_k)$ Lagrangian term: measures the the cost at every time step of the simulation, up to the control horizon N
- $\Delta u_k^T R \Delta u_k$ Mayer term: it's the desired set point and the end of the single time step
- $m(x_{N+1})$ R-term: it's a weight on the input vector and allowes a more smooth control action

4.2.1 AUV

A good definition of the AUV's cost function is

$$J_1(x_1, u_1) = K_{1_1} \cdot (x_{ref} - x_1)^2 + K_{1_2} \cdot u_1^2$$

Where K_1 weighs the difference between the state and the target and K_2 weighs the input amount, penalizing larger values.

To ensure the right orientation is preserved, the target values for ϕ and θ are set to be

$$\phi_{ref} = \text{atan}\left(\frac{z_1}{x_2 - x_1}\right)$$
$$\theta_{ref} = \text{atan}\left(\frac{z_1}{y_2 - y_1}\right)$$

4.2.2 USV

The USV has a similar cost function, as it only needs to follow the AUV's path. It's main challenge will be a good path-following given it's limited degrees of freedom

$$J_2(x_2, u_2) = K_{2_1} \cdot (x_1 - x_2)^2 + K_{2_2} \cdot u_2^2$$

4.3 do-mpc

This project is developed via python 3 and primarily *do_mpc* package, which is an open source toolbox for model predictive control, that provides easier solutions for control, esteem and simulation

Here are included the main and more explicatory parts of the code

4.3.1 Main simulation iterator

```
for i in range(n_sim):

    u0_1 = mpc1.mpc.make_step(x0_1)
    u0_2 = mpc2.mpc.make_step(x0_2)

    y_next_1 = simulator1.make_step(u0_1)
    y_next_2 = simulator2.make_step(u0_2)

    x0_1 = estimator1.make_step(y_next_1)
    x0_2 = estimator2.make_step(y_next_2)

    if (mpc1.x_setp < 5):
        linea = line(i)


    phi_0 = arctan(x0_1[2] / (x0_2[0] - x0_1[0]))
    theta_0 = arctan(x0_1[2] / (x0_2[1] - x0_1[1]))

    mpc1.x_setp, mpc1.y_setp, mpc1.z_setp = linea
    mpc1.phi_setp = phi_0
    mpc1.theta_setp = theta_0
    mpc2.x_setp = x0_1[0]
    mpc2.y_setp = x0_1[1]
```

4.3.2 AUV Controller

```
mterm = (_x_rov['x']**2 +  _x_rov['y']**2 +
         _x_rov['z']**2 + _x_rov['phi']**2 +
         (_x_rov['theta'])**2 + _x_rov['psi']**2)

lterm = ((_tvp_rov['x_sp'] - _x_rov['x'])**2 +
         (_tvp_rov['y_sp'] - _x_rov['y'])**2 +
         (_tvp_rov['z_sp'] - _x_rov['z'])**2 +
         (_tvp_rov['phi_sp'] - _x_rov['phi'])**2 +
         (_tvp_rov['theta_sp'] - _x_rov['theta'])**2 +
         (_tvp_rov['psi_sp'] - _x_rov['psi'])**2
)
self.mpc.set_rterm(
    u_1 = 0.1,
    u_2 = 0.1,
    u_3 = 0.1,
    u_4 = 0.1,
    u_5 = 0.1,
    u_6 = 0.1,
    u_7 = 0.1,
    u_8 = 0.1,
)
```

4.3.3 USV Controller

```
mterm = (_x_rov['x']**2 + _x_rov['y']**2 + _x_rov['psi']**2)

lterm = ((tvp['x_sp'] - _x_rov['x'])**2 +
         (tvp['y_sp'] - _x_rov['y'])**2 +
         (tvp['psi_sp'] - _x_rov['psi'])**2
)
Vx_array = np.array([10, 10*1.3, 10*0.7])
Vy_array = np.array([10, 10*1.3, 10*0.7])

self.mpc.set_uncertainty_values(Vx = Vx_array, Vy = Vy_array)
self.mpc.set_tvp_fun(self.tvp_fun)

self.mpc.set_rterm(
    u1 = 0.1,
    u2 = 0.1,
)
```

4.3.4 AUV Model

```
x = self.model.set_variable('_x', 'x')
y = self.model.set_variable('_x', 'y')
z = self.model.set_variable('_x', 'z')
#euler angles
phi = self.model.set_variable('_x','phi')
theta = self.model.set_variable('_x','theta')
psi = self.model.set_variable('_x','psi')
#lin vel
u = self.model.set_variable('_x', 'u')
v = self.model.set_variable('_x', 'v')
w = self.model.set_variable('_x', 'w')
#lin acc
u_dot = self.model.set_variable('_z', 'u_dot')
v_dot = self.model.set_variable('_z', 'v_dot')
w_dot = self.model.set_variable('_z', 'w_dot')
#ang vel
p = self.model.set_variable('_x', 'p')
q = self.model.set_variable('_x', 'q')
r = self.model.set_variable('_x', 'r')
#ang acc
p_dot = self.model.set_variable('_z', 'p_dot')
q_dot = self.model.set_variable('_z', 'q_dot')
r_dot = self.model.set_variable('_z', 'r_dot')
#input
u_1 = self.model.set_variable('_u', 'u_1')
u_2 = self.model.set_variable('_u', 'u_2')
u_3 = self.model.set_variable('_u', 'u_3')
u_4 = self.model.set_variable('_u', 'u_4')
u_5 = self.model.set_variable('_u', 'u_5')
u_6 = self.model.set_variable('_u', 'u_6')
u_7 = self.model.set_variable('_u', 'u_7')
u_8 = self.model.set_variable('_u', 'u_8')

x_sp = self.model.set_variable('_tvp', 'x_sp')
y_sp = self.model.set_variable('_tvp', 'y_sp')
z_sp = self.model.set_variable('_tvp', 'z_sp')
phi_sp = self.model.set_variable('_tvp', 'phi_sp')
theta_sp = self.model.set_variable('_tvp', 'theta_sp')
psi_sp = self.model.set_variable('_tvp', 'psi_sp')
```

```
self.model.set_rhs('x', cos(psi)*cos(theta)*u +
    (-sin(psi)*cos(phi) + cos(psi)*sin(theta)*sin(phi))*v +
    (sin(psi)*sin(phi)+cos(psi)*cos(phi)*sin(theta))*w)
self.model.set_rhs('y', sin(psi)*cos(theta)*u +
    (cos(psi)*cos(phi)+sin(phi)*sin(theta)*sin(psi))*v +
    (-cos(psi)*sin(phi) + sin(theta)*sin(psi)*cos(phi))*w)
self.model.set_rhs('z', -sin(theta)*u + cos(theta)*sin(phi)*v +
    cos(theta)*cos(phi)*w)

self.model.set_rhs('phi', p + sin(phi)*tan(theta)*q +
    cos(phi)*tan(theta)*r)
self.model.set_rhs('theta',  cos(phi)*q - sin(phi)*r)
self.model.set_rhs('psi', (sin(phi)/cos(theta))*q +
    (cos(phi)/cos(theta))*r)

self.model.set_rhs('p', p_dot)
self.model.set_rhs('q', q_dot)
self.model.set_rhs('r', r_dot)

self.model.set_rhs('u', u_dot)
self.model.set_rhs('v', v_dot)
self.model.set_rhs('w', w_dot)

#algebraic equations f_i(u_1...u_8)
dynamics = vertcat(f_1,f_2,f_3,f_4,f_5,f_6)
self.model.set_alg('dynamics',dynamics)
```

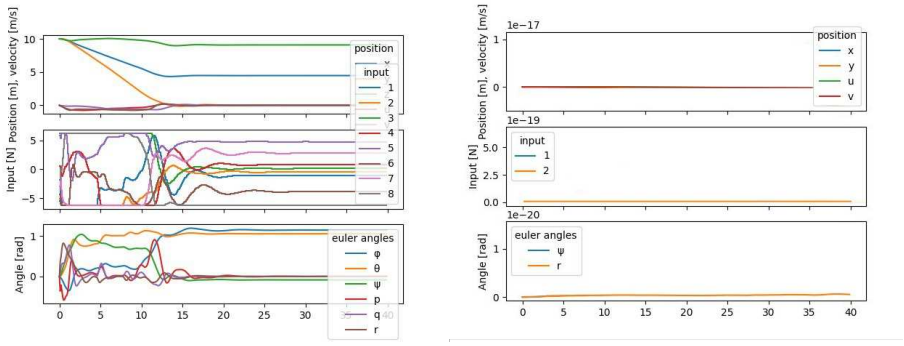
4.3.5 USV Model

```
#position
x = self.model.set_variable('_x', 'x')
y = self.model.set_variable('_x', 'y')

#yaw angle
psi = self.model.set_variable('_x','psi')

#lin vel
u = self.model.set_variable('_x', 'u')
v = self.model.set_variable('_x', 'v')

#lin acc
u_dot = self.model.set_variable('_z', 'u_dot')
v_dot = self.model.set_variable('_z', 'v_dot')

#yaw vel
r = self.model.set_variable('_x', 'r')

#yaw acc
r_dot = self.model.set_variable('_z', 'r_dot')

#input
u1 = self.model.set_variable('_u', 'u1')
u2 = self.model.set_variable('_u', 'u2')

#current in inertial frame
#Vx = self.model.set_variable('_p','Vx')
#Vy = self.model.set_variable('_p','Vy')

#Set point
x_sp = self.model.set_variable('_tvp', 'x_sp')
y_sp = self.model.set_variable('_tvp', 'y_sp')
psi_sp = self.model.set_variable('_tvp', 'psi_sp')

x_2 = self.model.set_variable('_tvp', 'x_2')
y_2 = self.model.set_variable('_tvp', 'y_2')

#dynamics
tu = (u1 + u2)*sin(ang)**2
tv = (u2 - u1)*sin(ang)*cos(ang)
tr = (u2 - u1)*d*cos(ang)
```

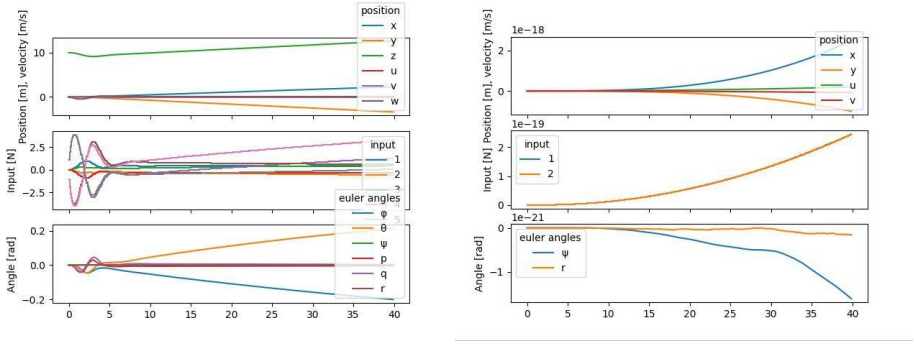
```
self.model.set_rhs('x', u*cos(psi) - v*sin(psi) + Vx)
self.model.set_rhs('y', u*sin(psi) + v*cos(psi) + Vy)
self.model.set_rhs('psi', r)

self.model.set_rhs('u', u_dot)
self.model.set_rhs('v', v_dot)
self.model.set_rhs('r', r_dot)

f_1 = tu + r*v*Iy/Ix - u*Dx/Ix
f_2 = tv - u*Ix/Iy - v*Dy/Iy
f_3 = tr - r*Dz/Iz + v*u*(Iy-Ix)/Iz

dynamics = vertcat(f_1,f_2,f_3)
self.model.set_alg('dynamics', dynamics)
```

5 Results

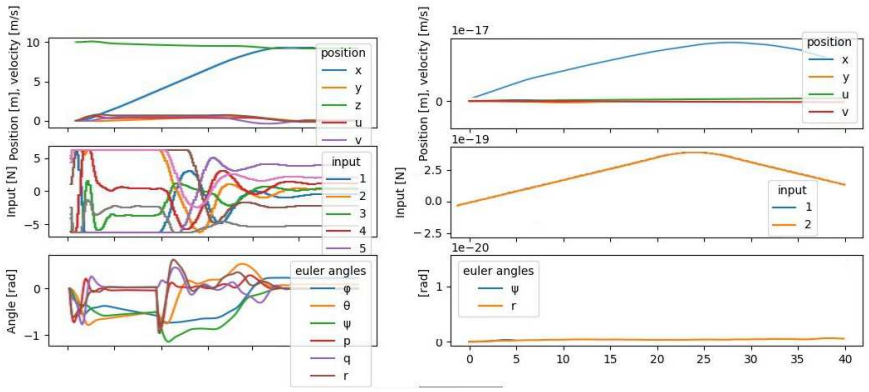
One of the main issues faced during this simulations, was MPC's computationally expensive nature. This scenario is modeled with both linear and non linear functions thus it requires more powerful hardware to elaborate the solutions. Some of the most elaborate trajectories were not possible to solve in a home setup.



In this case the boat is forced to stay still, while the AUV moves away. This was to test it's capability to adjust it's orientation towards the USv. It can be seen from the blue and orange lines, that the angles increase while the AUV moves and stabilize when it reaches it's target coordinates of {5, 0, 10}



Here the AUV is set to have a slow linear trajectory of speed {0.1,-0.1,0.1} and the USV follows it at a much slower rate. Due to this difference the AUV during is path, maintains an increasing inclination, in order to remain aligned with the boat's line of sight.
To properly follow the path, the USV also increasingly steers to the left, to match it's trajectory delay



In this third test, the path for the AUV is a straight line along the x-axis. Provided with a greater input, the USV can now properly follow it. The thrusters provide the same amount of acceleration, so there is no lateral or angular movement, and the USv can place itsalf in the right spot, despite some overshoot caused by the surface wave disturbances.

6 Conclusions

This thesis' goal was to set up and test a simulated version of an MPC controller for two coordinated units, a surface vehicle and an underwater drone. Despite some hardware limitations on the computational side of the project, the more simple scenarios simulated showed satisfying results. The boat was able to follow the drone's reference location with good precision, included situations where it was affected by constant disturbance waves.
The long computational times prevented more structured simulations and also diminished the troubleshooting capabilities on the simpler ones, so some interesting conditions to test were left aside.

6.1 Further Work

The data gathered suggest that a more complex and structured version of this control setup can be developed and tested in a more complex simulated scenarios and later also in real underwater conditions. Both model were provided with approximated physical parameters so a real test should be preceded by a more accurate evaluation of the real values in order to avoid discrepancies between the real dynamics and the simulated ones

7 Bibliography

1. Aurora Haraldsen, Martin S. Wiig, Kristin Y. Pettersen, *Collision Avoidance for Underactuated Surface Vehicles in the Presence of Ocean Currents*, 2022.
https://ffi-publikasjoner.archive.knowledgearc.net/bitstream/handle/20.500.12242/3130/2097155.pdf

2. Lauritz Rismark Fosso, Kristian Aasmundstad Johannesen, Pål Kristoffer Kjærem, Tor Harald Staurnes, *Decentralized Model Predictive Control for Increased Autonomy in Fleets of ROVs*, Norwegian University of Science and Technology, 2023

3. Chu-Jou Wu, *6-DoF Modelling and Control of a Remotely Operated Vehicle*, College of Science and Engineering, Flinders University, 2018
https://flex.flinders.edu.au/file/27aa0064-9de2-441c-8a17-655405d5fc2e/1/ThesisWu2018.pdf

4. Thor I. Fossen *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons, Ltd, 2011 ISBN:9781119994138
DOI:10.1002/9781119994138

5. Sergio Lucia, Felix Fiedler, Do-MPC python toolbox, https://www.do-mpc.com/

6. Blue Robotics, https://bluerobotics.com/