Seminar

Guidance and control of marine vehicles

1 Introduction

The goal of this seminar is to introduce you to the processes of designing navigation, guidance and control (NGC) algorithms for marine vehicles. The seminar consists of a *simulation* and *practical* part. The target vehicle, shown in Figure 1, is called **H2OmniX**. It is an overactuated autonomous surface vehicle (ASV) that weighs around 20 kg and is easily deployable in the LABUST pool.

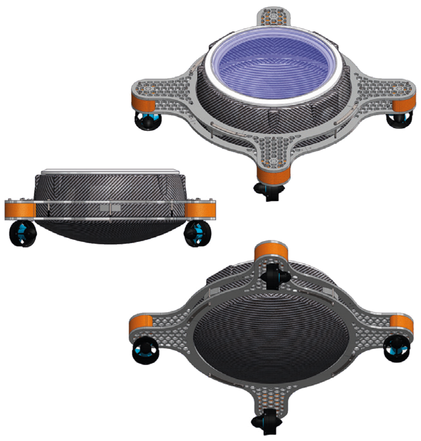


Figure 1: The H2OmniX autonomous surface vehicle (ASV).

Your reponsibility is to design, configure and run a dynamic positioning controller that will allow to move between waypoints in the pool. To achieve this you will need to use the knowledge learned on the Marine robotics classes to implement:

- *Compensation of thruster nonlinearity* ensuring requested thruster forces are correctly provided by the thrusters.
- *Control allocation*, i.e., mapping the generalized vector of forces and moments τ into individual thruster force requests.
- *Heading controller* needed to control the heading angle of **H2OmniX**
- *Positioning controller* to position **H2OmniX** at the desired location in the pool.

The first part of the document contains the simulation tasks that need to be done. The second part of document will contain the details about moving your simulation to the real vehicle. **Hint:** The theoretical and background supplements can be found below in the Appendix of this document.

1.1 Reporting

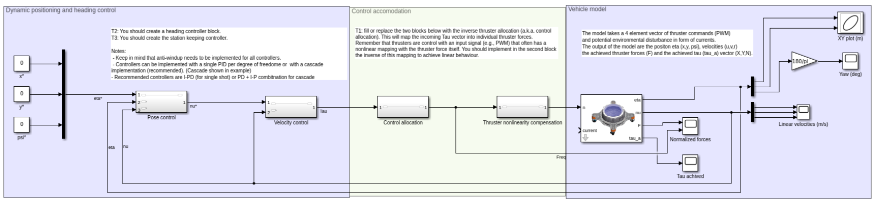
The reporting consists of two parts: a **written report**, and an **visual oral presentation**. Both of these are handed in at the end of the seminar (final week of seminar) and both contribute to the final grade equally.

During the *simulation part* of the exercise each task will specify the required plots and comments that need to be appended in the written report. These include general demonstration of functionality and basic details about the implementation of the NGC algorithms. These plots can also be used during the final presentation for purpose of introduction and/or comparison.

The *partical part* of the exercise will apply concepts developed and validated in the simulator on a real-vehicle in the pool. The "code" itself should be reused and only the parameters should be adjusted to achieve expected performance. In the tasks related to this part some additional plots and questions are specified which need to be addressed in the visual oral presentation. The presentation should also include any issues encountered during execution.

2 Development of the NGC algorithms in simulation

The simulation part is done in MATLAB/Simulink (version R2022b+) and starts by using the model of the **H2OmniX** autonomous surface vehicle (ASV) shown in Figure 1. It is possible to implement the same in a Python or C++ programing language by using ROS1/ROS2, but support will be limited. Contact the lecturer for details.



T2: You should create a heading controller block.
T3: You should create the station keeping controller.

Notes:
- Keep in mind that anti-windup needs to be implemented for all controllers.
- Controllers can be implemented with a single PID per degree of freedome or with a cascade implementation (recommended). (Cascade shown in example)
- Recommended controllers are I-PD (for single shot) or PD + I-P combitnation for cascade

T1: fill or replace the two blocks below with the inverse thruster allocation (a.k.a. control allocation). This will map the incoming Tau vector into individual thruster forces. Remember that thrusters are control with an input signal (e.g., PWM) that often has a nonlinear mapping with the thruster force itself. You should implement in the second block the inverse of this mapping to achieve linear behaviour.

The model takes a 4 element vector of thruster commands (PWM) and potential environmental disturbance in form of currents. The output of the model are the positon eta (x,y, psi), velocities (u,v,r) the achieved thruster forces (F) and the achieved tau (tau_a) vector (X,Y,N).

Figure 2: The main model layout showing the general feedback setup for the seminar tasks.

Provided with this document is a collection of MATLAB scripts and Simulink models. The main model is named `main.slx`. The block diagram of the model, shown in Figure 2, shows the simplified layout which fits the ideal controller design conditions provided within the seminar. The **H2OmniX** model from this block is shown in Figure 3. It consists of the usual components consisting of a propulsion model (thruster model and allocation) and an model for dynamics and kinematics of the vehicle. Often this model is coupled, however, here it is simplified to avoid any coupling issues during your control design.The diagram does not include sensor models in order to provide you with ideal measurements. However, bear in mind that in real-life η and ν states are measured directly or indirectly with sensors that have their own model and noise characteristics. While it is possible to change parameters here it is not necessary for completing the simulation part of the the seminar. The model inputs are PWM signals for the brushless thruster speed controller and environmental effects, namely, currents that will act on the model. The model output are the complete 6 states (horizontal position, yaw angle and corresponding velocities). These states can be used directly as controller feedback signals.

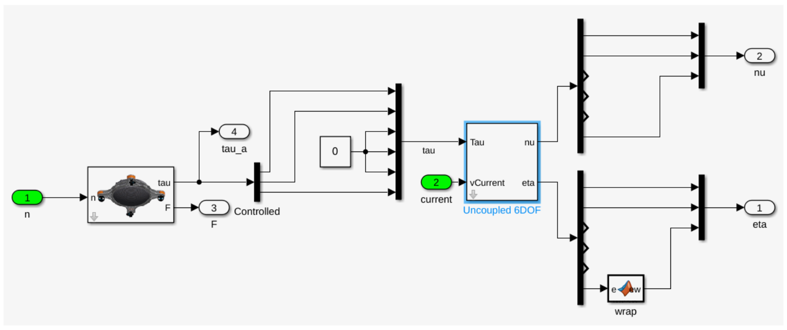


Figure 3: A look at the internals of the **H2OmniX** model. Note that the models reduces the generic 12 states (position, attitude and corresponding velocities) into a 6 state surface output. Note also the wrapping functiong on the yaw angle to ensure the output is within $< -\pi, \pi]$.

2.1 Task 1: Control accomodation

Control accomodation is the important first step that ensures near linear behaviour of the vehicle actuation. It consists of thruster nonlinearity compensation and control allocation.

2.1.1 Thruster nonlinearity compensation

Your first step is to create a dataset mapping input PWM commands to achieved force. In the `single_thruster.slx` Simulink file, also shown in Figure 4, the thruster model is isolated for individual testing. The same thruster model is used for the H2OmniX simulation.

You can access the used data by looking under the mask of the `T200` Simulink block. There you will find two constant vectors for `PWM` and corresponding force. You can use these vectors as the dataset. Once you extracted the dataset use the MATLAB `cftool` to find the corresponding mapping of control commands into PWM signals for the thruster.

Put in the report the following:

1. **PWM vs. Force plot of your dataset. Comment on the shape. Is it already linear or not?**

2. **Comment which function can approximate Force vs. PWM curve and why you selected it. Specify the function parameters received using the `cftool` or similar tool.**

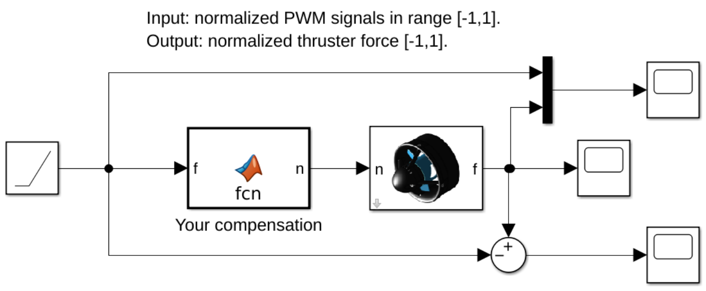

Input: normalized PWM signals in range [-1,1].
Output: normalized thruster force [-1,1].
f
fcn
n
Your compensation
n
f
+
-

Figure 4: The single thruster Simulink model. You need to implement the nonlinear compensation function in the matlab block. You can easily compared if requested and received values are correct in the scope. While there will be always some error between requested and received, it should be withing few precent of the total range $[-1, 1]$.

Hint: you can either find the polynomial function f which gives you forward mapping $F = f(PWM)$ and then find an inverse solution. Alternatively, you can map directly the inverse and find g which gives you $PWM = g(F)$. Either way, keep in mind that: a) it is not necessary to overfit, b) when no force is requested you should output zero, and c) you should always limit the entry into your function to the range of values that you fitted data on (to avoid bad extrapolations).

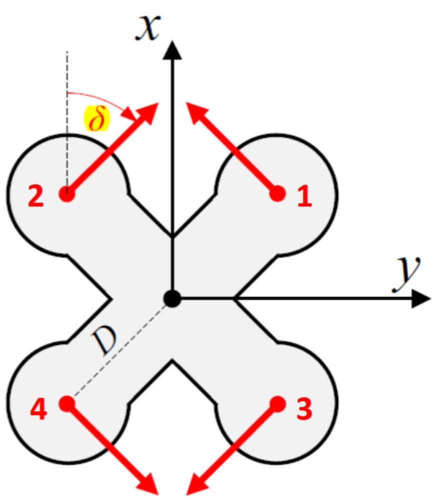


Figure 5: The thruster configuration of **H2OmniX** counting the thrusters from starboard fore towards port aft. The thruster input PWM is normalized between $[-1, 1]$ as is the force output.

2.1.2 Control allocation

Your second step is to determine the control allocation which maps the generalized forces and torques (τ) into thruster specific commands. In case of the H2OmniX vehicle these are four PWM commands. The H2OmniX thruster configuration is shown in Figure 5.

Put in the report the following:

1. **The actuator allocation B determined from Figure 5.**
2. **Describe the control allocation (inverse of actuator allocation) method you selected to calculate $\mathbf{B}^{\dagger}$.**
3. **Compare the coverage of your method in the X-Y space using the `control_allocation_coverage.m` script and attach the corresponding image. Comment on the coverage.**
   Note: you can use the function description to see the usage.

Hint: Implement the control allocation into a matlab function rather than directly into Simulink. You can then use the same function from within Simuluink and within the control_allocation_coverage.m script.

2.1.3 Integrating into the main simulation

Once you add the control allocation and nonlinearity compensation into the main simulation you can test the correct operation in manual control. By providing the X, Y and N commands individually to your control accomodation the vehicle should dominantly move in the corresponding directions along surge, sway and yaw. Due to coupling and thruster imperfection you might notice that the vehicle also moves little in other degrees of freedom. However, if you notice a large movement in sway while providing only surge force X then likely you have an error in your control accomodation setup.

2.2 Task 2: Heading controller

The second task is to implement a heading controller for **H2OmniX**. You can choose to either design the controller with a single PID or with a cascade of two PID controllers. Depending on your choice implementation might differ. Consider also the layout of the PID controller itself. Finally, consider that you need to implement integrator anti-windup mechanism to avoid bad behaviour.

For modeling use the uncoupled version of the yaw-rate dynamic model:

$$\dot{r} = -\frac{\beta(r)}{\alpha_r} r + \frac{1}{\alpha_r} N \tag{1}$$

$$\beta(r) = \beta_r + \beta_{r|r|}|r| \tag{2}$$

where parameters for α_r, β_r and $\beta_{r|r|}$ can be found by double clicking the **H2OmniX** Simulink model.

The output response should have a small overshot (less than 5%) and the transient should be according to the specifications below. **Put in the report the following:**

1. **Decribe and argument your decision for choosing a cascade or single PID design. Argument your choice of PID controller and why you did or did not use the proportional, integral and/or derivative parts.**
2. **Derive the closed loop transfer function $G_{cl}(s) = \frac{\psi}{\psi^*}$**
3. **Describe how you tuned the controller? Which parameters were used?**
4. **Target a settling time of 5 seconds for reference $\psi^* = \pi/2$ Plot the normalized step response ($\frac{\psi}{\psi^*}$) for references $\psi^* \in (\pi/4, \pi/3, \pi/2, \pi)$. Show the all the yaw responses on the same plot. What do you observe?**
5. **Increase the PID tuning to achieve settling time of 3 seconds and repeat the plot from before. What is the difference? Why?**
6. **Implement and use anti-windup mechanism by taking into account the saturation for torque which is ~ 0.85. Record the reponses for settling time of 3 seconds using the anti-windup mechanism.**

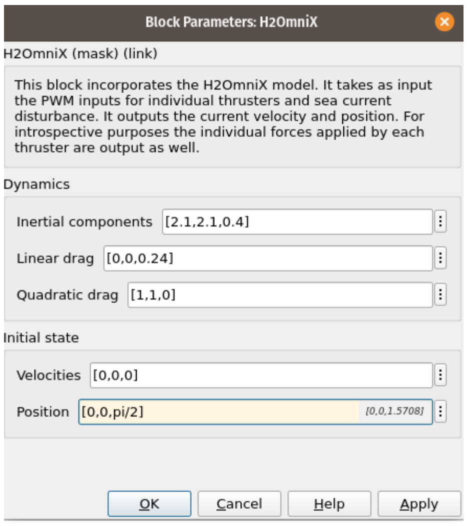
Hint: The yaw degree of freedom is often tracked between $< -\pi, \pi]$ and due to this wraping you will have a discontinuity on the border. This problematic as the error between $-\pi$ and π will appear to be 2π while in reality these represent the same heading. To avoid funky behaviour use `wrap` block already existing in the **H2OmniX** model. When using $I - P$ or similar control structure the wrapping on the state itself will present an issue. To avoid this you can unwrap by using the Simulink `Unwrap` block from DSP System Toolbox. For the PID itself, you can use the `PID` and `PID(2dof)` Simulink blocks, but implementing your own PID controller with simpler block will help confidence and understanding. The PID parameters should not be tuned by hand but rather by a model function or similar approach.

2.3 Task 3: Position controller

In the final task a similar approach to the heading control design can be undertaken. Choose your PIDs and prepare the model. Use the same tuning approach as before, but remember to change the α, β parameters for the corresponding degree of freedom. It is important to remember that the error in position is measured in the navigation coordinate frame while the velocities and forces are measured in the body frame.

Put in the report the following:

1. **Decribe and argument your decision for choosing a cascade or single PID design. Argument your choice of PID controller and why you did or did not use the proportional, integral and/or derivative parts.**
2. **Describe how you tuned the controller? Which parameters were used?**



Block Parameters: H2OmniX

H2OmniX (mask) (link)

This block incorporates the H2OmniX model. It takes as input the PWM inputs for individual thrusters and sea current disturbance. It outputs the current velocity and position. For introspective purposes the individual forces applied by each thruster are output as well.

Dynamics

Inertial components [2.1,2.1,0.4]

Linear drag [0,0,0.24]

Quadratic drag [1,1,0]

Initial state

Velocities [0,0,0]

Position [0,0,pi/2] [0,0,1.5708]

Figure 6: The location of model parameters.

3. **Implement the anti-windup from the previous step. What saturation boundaries did you set?**

4. **Set the heading $\psi^* = 0$ and north (x) reference to $x^* = 1\,m$ and record the time plot. Add the time plot and the $x - y$ plots to the report.**

5. **Set the initial heading of the model $\psi = pi/2$, see Figure 6. And repeat the previous point. Validate that the platform moves in the good direction?**

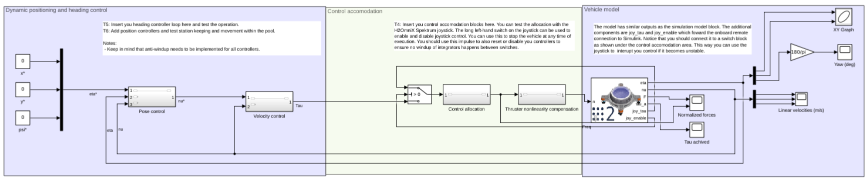
6. **Reset the initial heading to zero and set the references as follows $x^* = 2$, $y^* = 5$. Attach the $x - y$ plot. Is the vehicle moving in a straight line? Why?**

7. **Keep the same refrences as before. Add $0.5m/s$ of current in any desired direction and describe what is happening? Show the plot for τ as well as $x - y$.**

Hint: current input in the model is a two element vector specifying current in northerly and easterly directions.

3 Moving from simulation to real vehicle

Through the previous section you implemented the guidance and control chain needed to control the **H2OmniX** vehicle. The next step is to exchange simulation with the real vehicle. This is done by replacing the model block with the ROS2 enabled block, as shown in Figure 7. The `h2omnix.slx` model already contains the block integrated within the dummy control loop. The ROS2 enabled block subscribes and publishes three topics as seen in the block mask shown in Figure 8. The `pwm_out` topic is the actual PWM command you will send to the vehicle, while the `odometry filtered` is the topic from the onboard EKF filter. The filter is required due to multiple reasons. First, it fuses the IMU and camera location sensor into a single filtered topic. Second, it compensates the delay of the RTSP pool camera which is around $800ms$. Without this compensation the controller performance would be directly degraded due to this delay.

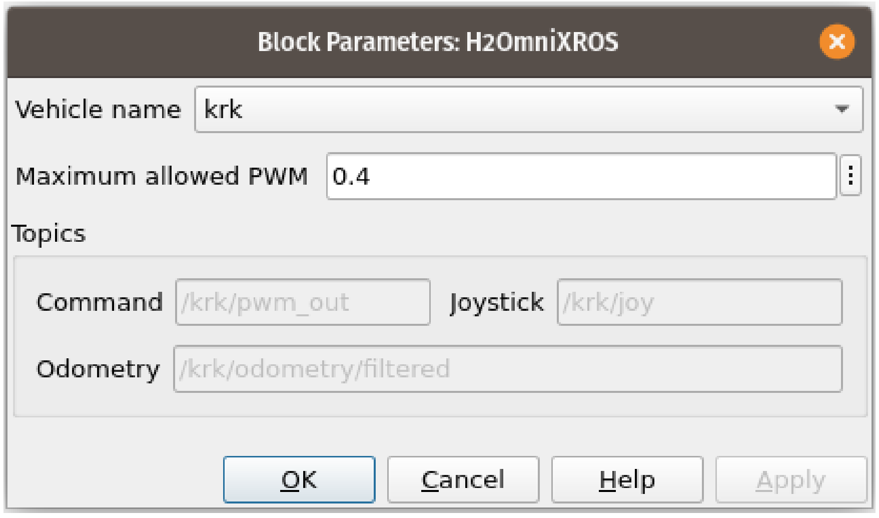

T5: Insert you heading controller loop here and test the operation.
T6: Add position controllers and test station keeping and movement within the pool.

Notes:
- Keep in mind that anti-windup needs to be implemented for all controllers.

T4: Insert you control accomodation blocks here. You can test the allocation with the H2OmniX Spektrum joystick. The long left-hand switch on the joystick can be used to enable and disable joystick control. You can use this to stop the vehicle at any time of execution. You should use this impulse to also reset or disable you controllers to ensure no windup of integrators happens between switches.

The model has similar outputs as the simulation model block. The additional components are joy_tau and joy_enable which foward the onboard remote connection to Simulink. Notice that you should connect it to a switch block as shown under the control accomodation area. This way you can use the joystick to interupt you control if it becomes unstable.

Figure 7: The main model layout with the ROS2 vehicle block.


Block Parameters: H2OmniXROS

Vehicle name: krk

Maximum allowed PWM: 0.4

Topics

Command: /krk/pwm_out
Joystick: /krk/joy
Odometry: /krk/odometry/filtered

Figure 8: The mask of the **H2OmniX** block.

Apart from the topics, the mask has two parameters: vehicle selector (drop-down), and power limiter. Depending on which vehicle you are working, you should select it here from the drop-down. In case of the seminar the *Vis* and *Krk* vehicles are available. The maximum allowed PWM can be selected in range $[0 - 1]$ for safety purposes. The default value for the pool is 0.4 which means that thruster PWM will be capped on that value.

One additional block in Figure 7 should be added to your simulation. This is the switching block between controllers and allocation. The block can be activated using the MODE switch of the vehicle's Spektrum joystick (see Figure 9). This is an additional safety feature that allows you to disconnect your control from the vehicle and take over control using the joystick.

The rest of the blocks should be taken from your simulation and transferred here. What should not be transferred are your controller gains. This is the reason you were encouraged to write a tuning method that will calculate the

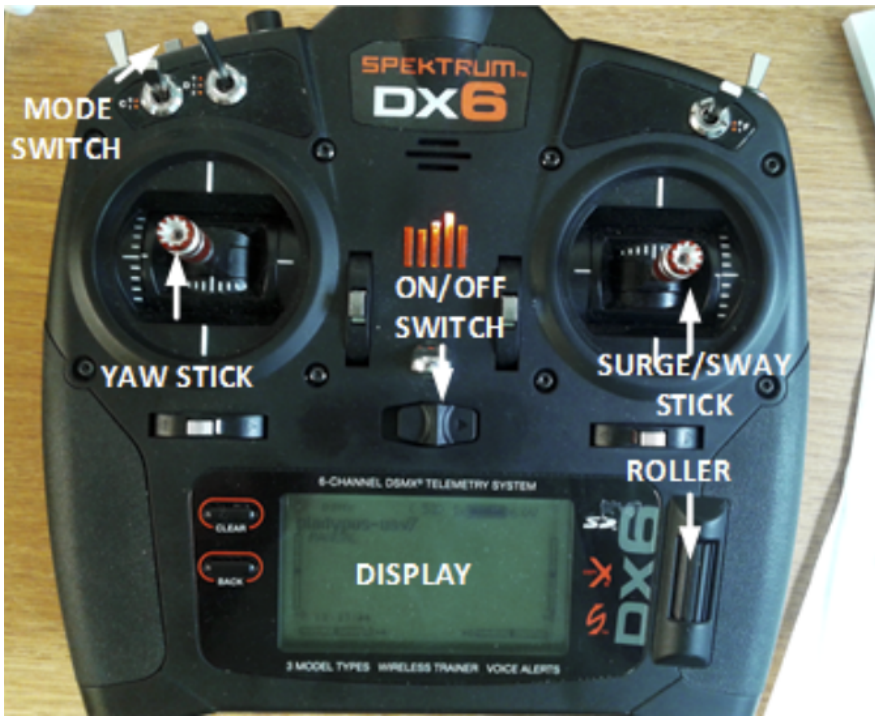

SPEKTRUM
DX6
MODE SWITCH
ON/OFF SWITCH
YAW STICK
SURGE/SWAY STICK
ROLLER
DISPLAY
6-CHANNEL DSMX TELEMETRY SYSTEM
3 MODEL TYPES WIRELESS TRAINER VOICE ALERTS

Figure 9: The Spektrum joystick control layout. The MODE switch is used to turn on/off automatic control.

Table 1: The inertial and constant drag parameter for each DoF of the **H2OmniX** vehicle. Notice that these are rounded for your convenience.

DoF	α	β
surge	0.4	0.4
sway	0.4	0.4
yaw	0.1	0.1

gains based on the dynamic model parameters. You should use the parameters in table 1 instead the ones provided in the simulation model.

3.1 Connecting from Simulink

To enable connection with the ROS2 network two steps are required. First, you need to ensure that `DEFAULT_FASTRTPS_PROFILES.xml` included with the materials is in the directory with your Simulink model. Second, in MATLAB prompt you need to run the command:

```
setenv("ROS_DISCOVERY_SERVER", "10.0.10.20:11811");
```

Validity of connection can be tested directly from the MATLAB console with command `ros2 topic list` which should provide a larger list of topics including the name of your vehicle.

3.2 Task 4: Manual control

Before applying any controllers you should check that manual control is operational. Manual control requires only your control accommodation blocks and the physical Spektrum joystick.

Turn the joystick into manual mode (written on the screen of the joystick) using the MODE switch. Operate the vehicle using the joystick around the pool. Observe the behaviour and consider the following questions:

- **Are the thrusters quiet when you leave the joystick in neutral mode?**
- **Is the vehicle moving in the correct direction when you actuate individual degrees of freedom (surge, sway, yaw)?**
- **Is the vehicle force increasing near linear with the movement of the joystick (estimate empirically)?**
- **Is the simulation running in real-time? Is the Simulink timer running similar to your clock?**

Assuming the general behaviour is acceptable, it can be concluded that the control accommodation is correct.

3.3 Task 5: Heading controller

With control accommodation validated transfer your controllers into the block. If you used a cascade you can perform checking of velocity controllers first and then add the pose controllers. However, if performance was valid in the simulation you can start directly with the whole heading control chain. **Note:** Disable the surge and sway controllers and ensure that controller gains are tuned for the model in table 1.

Record the data from the vehicle and put your report/presentation the following plots:

1. **Target a sensible settling time for reference $\psi^* = \pi/2$ Plot the normalized step response ($\frac{\psi}{\psi^*}$) for references $\psi^* \in (\pi/4, \pi/3, \pi/2, \pi)$. Show all the yaw responses on the same plot. Is the behaviour similar?**
2. **Use the boat hook (mezzo-marinaio) to disturbe the vehicle, i.e., hook the metal loop and try to turn H2OmniX. Show the disturbance rejection as a separate plot. Note: you can use also hands if no boat hook is available but do not get wet.**

3.4 Task 6: Position controller

Finally, the position controller should be tested. Since two vehicles are operated and space is limited it is necessary to plan well the movements to avoid hitting the wall. The pool references should be $x \in [1.5, 7.5]$ and $y \in [1, 3]$. With two vehicles you can select the origins as $(x, y) = (3, 2)$ or $(x, y) = (6, 2)$, depending where you are working. **Note:** The point $(x, y) = (0, 0)$ is located on the south-west edge of the pool so definetly do not command that position.

Put in the report the following:

1. **Set the commanded heading $\psi^* = 0$ and north (x) reference to your selected origin. Let the vehicle converge to the origin. Once in the origin increase or decrease the x^* reference by $1\,m$ and record the time plots for x and y. Add the time plot and the $x - y$ plots to the report.**
2. **Set the commanded heading of the model $\psi^* = pi/2$ and repeat the previous point. Validate that the platform moves in the good direction?**
3. **Reset the initial heading to zero and set the references to make the vehicle move across the pool, i.e., from $T_1 = (1.5, 2)$ to $T_2 = (7, 2)$. Record the transition and comment. What is the overshoot?**

3.5 Task 7: Combined movements (optional)

To additionally test your control try to repeat the last point from the previous task but command a ramp change in the heading. Do the following:

- **The heading reference ramp should have a slope of $0.5 rad/s$ to make the vehicle turn.**
- **Keep it first at T_1 and ensure it is rotating, the command it to go to point T_2. Observe and describe what is happening.**
- **Provide the plots of the movement x-y, time plots of η, ν and τ vector values.**
- **Is the vehicle going in a straight line?**
- **Is the vehicle reaching point T_2?**
- **Is the vehicle rotating continuously during the transit? Why?**

One other additional task is:

- **leave the vehicle rotating and maintaining your origin point $(x, y) = (3, 2)$ or $(x, y) = (6, 2)$**
- **using the hook pull the vehicle far away from the origin and maintain it there for 20-30 seconds.**
- **Release the vehicle. Is it converging straight back to its position? If not, why not?**

A. Mathematical modelling supplemental materials

Most definitions in this seminar rely on terms already established in vehicle modelling literature and the marine robotics course.

Vehicle model vectors, from the nomenclature, are given in Table 2 where {B} and {N} frames are defined in subsection on kinematics. Measurements of ν, η and τ are defined relative to these reference frames, see Figure 11. Vehicle linear and angular velocities (ν) are defined in {B} while the position and orientation (η) are relative to {N}.

Table 2: Rigid body degrees of freedom and their SNAME notation.

Description	Symbol	Surge	Sway	Heave	Roll	Pitch	Yaw	Frame
velocity	$\boldsymbol{\nu}$	u	v	w	p	q	r	{B}
position	$\boldsymbol{\eta}$	x	y	z	ϕ	θ	ψ	{N}
force and torque	$\boldsymbol{\tau}$	X	Y	Z	K	M	N	{B}

Consider the NGC subsystem shown in Figure 10. Additional notation is introduced were Φ and $h(n)$ are the thruster allocation matrix and non-linear thruster characteristics, respectively. RPM and thrust force are denoted with $\mathbf{n}$ and $\mathbf{F}$, respectively. Subscript a indicates actual RPM and thrust forces, while superscript * denotes those desired by the NGC system. Indirect measurements of vehicle states are denoted with $\mathbf{o}_m$.

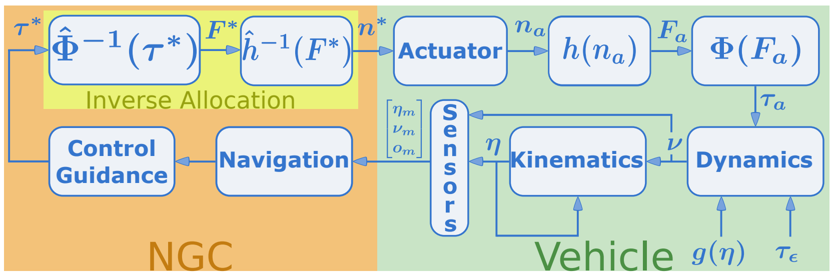


Figure 10: The NGC system and the internal model of the vehicle.

There is a clear separation between NGC and the real vehicle. Inverse allocation is connected to a specific thruster layout of a vehicle. Navigation and control are loosely coupled to vehicles and sensor suites. Due to this loose coupling, the control and navigation can be designed for easy scaling between different vehicles, i.e., avoiding NGC system redesign. Additionally, a clean interface is produced for plug and play replacement of simulated vehicles with their real counterparts. From the NGC standpoint a vehicle is a black box with thruster commands n^* as input and a vector of measurements as output.

Detailed simulation requires analysing each element of the "real" vehicle model for correct emulation. Next sections cover these elements starting from vehicle kinematics and dynamics. Dynamics requires model parameters, specific for each vehicle, which have to be identified. Allocation covers the mapping of generalized/virtual forces and moments ($\boldsymbol{\tau}$) to multiple thrusters. Finally, the

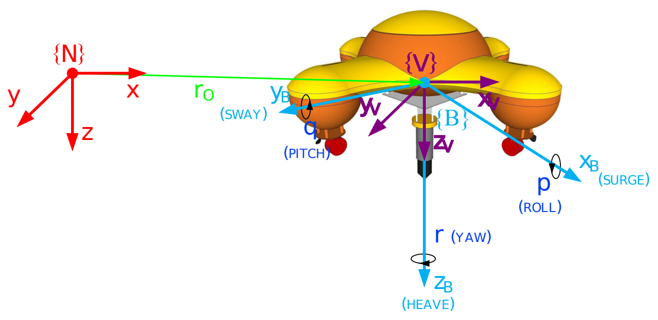


Figure 11: Representation of {N} and {B}.

transformation between desired forces and produced thrust is covered under thruster mapping analysis.

Kinematics

Vehicle kinematics analyses geometrical aspects of motion relating vehicle velocities between frames to calculate vehicle position within a desired navigation frame. Generally, reference frames can be divided into two groups; the first group georeferences the vehicle's Cartesian navigation frame while the second group defines individual sensor frames and vehicle pose.

Vehicle position and orientation are specified within the navigation frame as shown in Table 2. Alternatively, position in {N} is indicated with the vector

$$\mathbf{q}^n = \begin{bmatrix} n & e & d \end{bmatrix}^\intercal = \begin{bmatrix} x & y & z \end{bmatrix}^\intercal . \tag{3}$$

Vehicle frames are defined with their origin at some specific point, usually, the centre of gravity or rotation. The vehicle carried navigation frame is denoted with {V} and its axes coincide with {N}. The vehicle or body frame {B} shares its origin with {V} but its axes coincide with the vehicle as shown in Figure 11. The abscissa is placed along the longitudinal axis (aft to fore), the ordinate is pointing towards starboard (right), and the applicate points downwards. Body frame transformation relative to {N} is defined by roll, pitch and yaw Euler angles as

$$\mathbf{R_b^n} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\cos\phi\sin\theta \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix} . \tag{4}$$

The matrix is used for transforming position and linear velocities while angular velocities are transformed using

$$\mathbf{T_b^n} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} . \tag{5}$$

Both transformation matrices are combined forming the vehicle kinematics shown in (6).

$$\dot{\boldsymbol{\eta}} = \underbrace{\begin{bmatrix} \mathbf{R_n^b} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{T_n^b} \end{bmatrix}}_{\mathbf{J}(\boldsymbol{\nu})} \tag{6}$$

where $\mathbf{J}(\boldsymbol{\nu})$ is known as the Jacobian transformation matrix. In general, the kinematics equation relates {B} velocities to {N} velocities. The positions and angles $\boldsymbol{\eta}$ can be attained by integration of $\dot{\boldsymbol{\eta}}$. Consider that in the seminar we use only the yaw (ψ) angle and horizontal position (x, y) so equations (4)–(6) reduce considerably.

Dynamics

Dynamics, also referred to as kinetics, analyses the relationship between vehicle acceleration and velocity. The vehicles are analysed as regular rigid bodies; however, due to their immersion in a dense fluid environment additional hydrostatic and hydrodynamic effect are exerted on the vehicle.

Rigid-body dynamics is defined as

$$\mathbf{M_{RB}}\dot{\boldsymbol{\nu}} + \mathbf{C_{RB}}\boldsymbol{\nu} = \boldsymbol{\tau_{RB}} \tag{7}$$

where $\boldsymbol{\nu}$ and $\boldsymbol{\tau}_{RB}$ are defined in Table 2. The rigid-body system inertia ($\mathbf{M_{RB}}$) and Coriolis-centripetal matrix ($\mathbf{C_{RB}}$) are expressed as

$$\mathbf{M_{RB}} = \begin{bmatrix} m\mathbf{I}_{3x3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix}, \ \mathbf{I}_b = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & I_{zy} & I_z \end{bmatrix} \tag{8}$$

$$\mathbf{C_{RB}} = \begin{bmatrix} m\mathbf{S}\left(\boldsymbol{\nu}_2\right) & -m\mathbf{S}\left(\boldsymbol{\nu}_2\right)\mathbf{S}\left(\mathbf{r}_g^b\right) \\ m\mathbf{S}\left(\mathbf{r}_g^b\right)\mathbf{S}\left(\boldsymbol{\nu}_2\right) & -\mathbf{S}\left(\mathbf{I}_b\boldsymbol{\nu}_2\right) \end{bmatrix} \tag{9}$$

where m represents the system mass and $\boldsymbol{\nu}_2$ are the rigid-body angular velocities. The system inertial matrix has the property of being symmetrical and positive definite. The matrix equation $\mathbf{S}(\mathbf{x})$ is an antisymetric matrix constructed from vector $\mathbf{x}$. Skew-symmetric (antisymmetric) matrices are square matrices whose transpose is also its negative, i.e., $\mathbf{S}^\intercal = -\mathbf{S}$. The centre of gravity (CoG) lever-arm, represented in {B}, is denoted as $\mathbf{r}_g^b$.

The rigid-body dynamics can be accurately estimated as it depends on design parameters, e.g. mass, inertia around axis, which are provided from most 3D modelling software.

Hydrostatic effects

Gravity is the main environmental force acting on the rigid-body outside of water. When immersed in water, the buoyancy force becomes relevant. Together they are often refereed to as restoring forces. Define, in addition to CoG, the centre of buoyancy (CoB) as $\mathbf{r}_b^b$. The gravity and buoyancy forces are defined as

$$\mathbf{f}_g^n = \begin{bmatrix} 0 & 0 & W \end{bmatrix}^\intercal, \ W = mg \tag{10}$$

$$\mathbf{f}_b^n = \begin{bmatrix} 0 & 0 & B \end{bmatrix}^\intercal, \ B = \rho g V \tag{11}$$

where V represents the volume of the immersed part and ρ the fluid density. The general restoring forces and moments in body frame can be calculated as

$$\mathbf{g}(\eta) = -\begin{bmatrix} \mathbf{R}_n^b\left(\mathbf{f}_g^n + \mathbf{f}_b^n\right) \\ \mathbf{r}_g^b \times \mathbf{R}_n^b \mathbf{f}_g^n + \mathbf{r}_b^b \times \mathbf{R}_n^b \mathbf{f}_b^n \end{bmatrix} \tag{12}$$

where the rotation matrix from {N} to {B} is used to transform the restoration forces. For small angles, i.e. $\phi \approx 0$ and $\theta \approx 0$, the hydrostatic effects reduce to

$$\mathbf{g}(\eta) = \begin{bmatrix} 0 \\ 0 \\ -(W-B) \\ y_b B - y_g W \\ x_g W - x_b B \\ 0 \end{bmatrix} \tag{13}$$

showing that only heave, roll and pitch DOF are affected. Designing $\mathbf{r}_b^b$ adequately reduces the effect solely to heave DOF. Such decoupling is useful during model simplification. Although the heave restoration cannot be neglected, it can be incorporated in the decoupled disturbance vector $\boldsymbol{\tau_\epsilon}$. In this seminar the heave degree of freedom is ignored and small angle assumptions hold, therefore, the restoring forces do not impact the simulation.

Hydrodynamic effects

The model is represented by

$$\underbrace{\mathbf{M_{RB}}\dot{\boldsymbol{\nu}} + \mathbf{C_{RB}}\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\mathbf{M_A}\dot{\boldsymbol{\nu}}_r + \mathbf{C_A}\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \mathbf{g}(\eta) = \boldsymbol{\tau} + \underbrace{\boldsymbol{\tau_{wind}} + \boldsymbol{\tau_{wave}}}_{\text{distrubances}} \tag{14}$$

where $\boldsymbol{\nu}_r$ is the relative speed. For *constant* and *irrotational* currents in {N} the rigid-body and hydrodynamic terms are combined as $\mathbf{M} = \mathbf{M_{RB}} + \mathbf{M_A}$ and $\mathbf{C}\left(\boldsymbol{\nu}_r\right) = \mathbf{C}_{RB}\left(\boldsymbol{\nu}_r\right) + \mathbf{C_A}$.

The added mass $\mathbf{M}_A$ introduces coupling and non-linearities into the underwater vehicle model. Generally, it consists of 36 parameters. As is the case with $\mathbf{M}_{RB}$, it is symmetrical and positive definite which reduces the number of parameters required for identification. The added mass Coriolis-centripetal terms ($\mathbf{C}_A$) are the consequence, similarly to $\mathbf{C}_{RB}$, of {B} rotation relative to {N}.

The hydrodynamic damping matrix, $\mathbf{D}(\boldsymbol{\nu}) = \mathbf{D}_l + \mathbf{D}_q\left(\boldsymbol{\nu}_r\right)$, is often approximated as a sum of a constant linear part and non-linear damping part. It is suggested to model the nonlinear part with quadratic terms, e.g., $X_{|u|u}|u_r|$ for a surge component. The damping matrix is positive definite but, generally, non-symmetric.

The manoeuvring model is already suited for general control and state estimation purposes as well as for parameter identification. However, compared to rigid-body parameters, added mass and damping terms are more difficult to measure, specifically the off-diagonal coupling terms. While realistic simulations benefit from increased model complexity, most control and estimation designs are performed with a simplified, uncoupled model. Assuming that

(i) the added mass diagonal parameters are dominant,

(ii) roll and pitch motion is negligible, i.e. $\phi, \theta \approx 0$,

(iii) the observed marine vehicle is small,

(iv) operational speeds are small so quadratic terms can be neglected,

(v) the hydrodynamic drag can be approximated with a first order speed dependant term,

the model is reduced to

$$\mathbf{M}\dot{\boldsymbol{\nu}}_r + \left(\mathbf{D}_l + \mathbf{D}_q\left(\boldsymbol{\nu}_r\right)\right)\boldsymbol{\nu}_r = \boldsymbol{\tau} + \boldsymbol{\tau}_{\boldsymbol{\epsilon}} \tag{15}$$

$$\alpha_\nu \dot{\nu}_r + \left(\beta_\nu + \beta_{|\nu|\nu}\nu_r\right)\nu_r = \tau + \tau_{\epsilon_\nu} \tag{16}$$

where $\mathbf{M}$, $\mathbf{D}_l$ and $\mathbf{D}_q$ are diagonal matrices. Often, the scalar representation is analysed for individual DOF and scalars α_ν, β_ν and $\beta_{|\nu|\nu}$ represent elements of matrices $\mathbf{M}$, $\mathbf{D}_l$ and $\mathbf{D}_q$, respectively, with the ν subscript representing individual velocities.

The first assumption, ideally requiring the vehicle to be symmetric, is found suitable for most applications. The second assumption voids most coupling terms and decouples vertical and horizontal DOF. Point approximation of a vehicle is possible using the third assumption. Finally, Coriolis effects and off-diagonal damping terms are removed using the fourth assumption. The restoring force in heave is directly included in $\boldsymbol{\tau}_{\boldsymbol{\epsilon}}$. For the seminar it is advised to approximate the vehicle with this simplified model. Model parameters α_ν, β_ν, and $\beta_{|\nu|\nu}$ can be easily found by double clicking the Simulink model.

Allocation

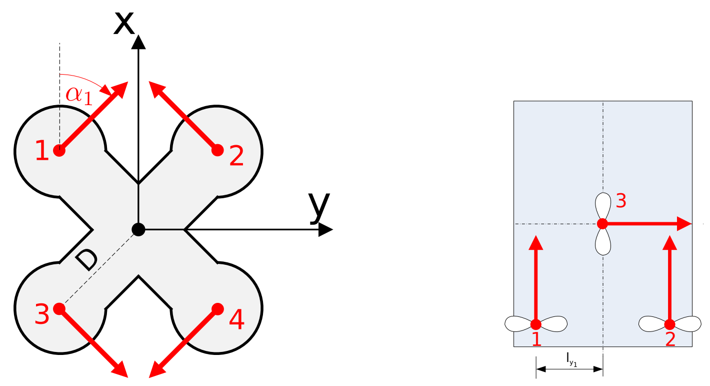
Last section introduced the dynamic model of underwater vehicles and defined the virtual forces and moments $\boldsymbol{\tau}$ as the model input. However, actuators are distributed differently based on vehicle design and $\boldsymbol{\tau}$ is achieved by the combination of forces they exert. Distribution of actuator forces $\mathbf{F}$ and moments to virtual commands $\boldsymbol{\tau}$ is called *allocation.* This section analyses thruster allocation since that is most important for the seminar.

Define $\mathbf{F}$ as the vector of thruster forces. Thrust allocation is any function $\boldsymbol{\Phi}\left(\mathbf{F}\right) : \Re^n \rightarrow \Re^m$, where n and m are the number of thrusters and controllable DOF, respectively. The allocation is often approximated with the linear function

$$\boldsymbol{\tau} = \mathbf{BF} \tag{17}$$

where $\mathbf{B}$ is the allocation matrix which is constant for fixed thruster configurations. To determine required thruster forces $\mathbf{F}$, based on the commanded virtual forces $\boldsymbol{\tau}$, the allocation matrix has to be invertible. Generally, the allocation matrix is not square and the inverse is calculated using the Moore-Penrose pseudo-inverse $\mathbf{B^{-1}} = \mathbf{B}^\intercal(\mathbf{B}\mathbf{B}^\intercal)^{-1}$ or $\mathbf{B^{-1}} = (\mathbf{B}^\intercal\mathbf{B})^{-1}\mathbf{B}^\intercal$.

Thruster actuated marine vehicles often have uncoupled control of vertical and horizontal DOF, i.e., separate thruster groups are employed for vertical and horizontal actuation. Two horizontal configurations, namely, the X-configuration and the standard differential configuration shown in Figure 12. The analysis focuses mainly on the X-configuration since this configuration is used by **H2OmniX** vehicle.



(a) (b)

Figure 12: (a) The X-configuration is used for surge, sway, yaw actuation. The other common configuration is the differential configuration, (b), which can be used to actuate only surge and yaw.

The X configuration is used often in inspection and work-class ROV due to the ability of controlling surge, sway and yaw separately. Minimum of three thrusters is enough to achieve the goal, but a fourth thruster, in addition to redundancy, allows spatially symmetric thruster distribution.

Let $\left(F_i, \alpha_i, \mathbf{l}_i^b\right)$ be a triplet defining a single thruster at location $\mathbf{l}_i^b$ with vectored thrust force $\mathbf{F}_i = F_i \begin{bmatrix}\cos\alpha_i & \sin\alpha_i & 0\end{bmatrix}^\intercal$. The angle α_i is defined in $\{\mathrm{B}\}$ and surge and sway forces are directly defined by the thrust vector. The yaw momentum is defined as a cross product of the lever-arm and the thrust vector and it can be shown that the surge, sway forces and yaw torque are defined as

$$X = \sum_{i=1}^{4} F_i \cos\alpha_i \tag{18}$$

$$Y = \sum_{i=1}^{4} F_i \sin\alpha_i \tag{19}$$

$$N = \sum_{i=1}^{4} F_i |l_i^b| \sin \underbrace{\left(\alpha_i + \operatorname{arctan2}\left(-l_{x_i}, l_{y_i}\right)\right)}_{\alpha_i'} \tag{20}$$

where l_{x_i} and l_{y_i} are the horizontal components $\mathbf{l}_i^b$ and the yaw torque is defined as zero when $\mathbf{l}_i^b = \mathbf{0}$. Notice that the maximum surge and sway force are exclusive, i.e., for maximum surge with $\alpha_i = 0$ the sway force is minimum. Often α_i is selected as $(2k+1)\frac{\pi}{4}$ to supply symmetric force in both surge and sway. In addition, for evenly distributed thrusters, maximum torque is achieved since $\alpha_i' \to k\frac{\pi}{2}$. Using (18)–(20) such an allocation matrix is defined as

$$\mathbf{B} = \begin{bmatrix} \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \sin\frac{\pi}{4} & -\sin\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -1 & 1 & 1 & -1 \end{bmatrix} \tag{21}$$

where an identical and constant lever-arm $l = 1$ is assumed for all thrusters. As noted earlier, the configuration is over-actuated, visible from the fact that there are more columns than rows. This indicates multiple solutions exist for the inverse allocation problem.

Inverse **allocation methods**

Ideally, thrusters are symmetrical in both positive and negative thrust directions and have the same maximum and minimum saturation value. Within the achievable thrust region, the *inverse* allocation is straightforward and the pseudo-inverse gives good results. However, in real-life scenarios the thruster are usually asymmetric offering more thrust in positive directions, due to physical characteristics of thrusters. Additionally, individual thrusters may not provide identical performance, have different saturation limits or require different energy for actuation. Therefore, equal distribution of virtual forces and moments, provided by the pseudo-inverse, is undesired. The generalized weighted inverse is proposed allowing custom force distribution among thrusters. Optimization of the quadratic cost function $J = \frac{1}{2}\mathbf{F}^\intercal\mathbf{W}\mathbf{F}$ with constraint $\boldsymbol{\tau} - \mathbf{B}\mathbf{F} = \mathbf{0}$ yields the generalized inverse

$$\mathbf{F} = \underbrace{\mathbf{W}^{-1}\mathbf{B}^\intercal\left(\mathbf{B}\mathbf{W}^{-1}\mathbf{B}^\intercal\right)^{-1}}_{\mathbf{B_W}}\boldsymbol{\tau} \qquad (22)$$

where $\mathbf{W}$ is the weight function used for tuning the force distribution. It is worth observing that the pseudo-inverse is the special case of the generalized inverse where $\mathbf{W} = \mathbf{I}$.

Thruster saturation imposes limits on attainable virtual force regions when using the described *inverse* allocation methods. Two methods are proposed in literature for dealing with thruster saturation issues: the cascading generalized inverse and the daisy-chain inverse. The daisy-chain solution divides thrusters into groups and the *inverse* allocation, starting from the first group, sequentially includes new groups until either the desired virtual forces are achieved or all groups are exhausted. Similarly, the cascading generalized inverse performs *inverse* allocation iteratively by excluding saturated thrusters from the calculation. These methods are only useful in over-actuated problems. The attainable virtual force region may be concave which can yield undesired effects on region boundaries. Even though both methods are computationally heavier than the pseudo-inverse, the implementation is straightforward. Additional methods exist, such as fixed point optimization as a computationally simple algorithm, as most computation is done upfront, before iterations. The algorithm also provides an exact solution and guarantees converges. Finally, a quadratic programming methods can be used for allocation, offering optimal solutions for convex problems.

The convex virtual force region, for the observed X configuration, is easily computed, however, *inverse* allocation within the complete region, independent of thruster characteristics, requires optimization methods. Quadratic programming, while numerically more involving than other approaches, can easily find the global solution to $\boldsymbol{\tau} - \mathbf{B}\mathbf{F} = \mathbf{0}$.

Table 3: The maximum and minimum thrust for each thruster of both Buddy and Pladypos vehicles. Forces are normalized towards the maximum provided thrust.

Vehicle		F_1	F_2	F_3	F_4
Buddy	+	0.986	0.984	0.998	1.000
	-	-0.128	-0.128	-0.410	-0.410
Pladypos	+	1.000	1.000	1.000	1.000
	-	0.600	0.600	0.600	0.600

Comparison of *inverse* allocation methods

The described *inverse* allocation methods are compared on Buddy and Pladypos thrust data given in Table 3. The latter exhibits asymmetric, but nearly identical, thruster characteristics, while the former has two different pairs of asymmetric thruster characteristics.

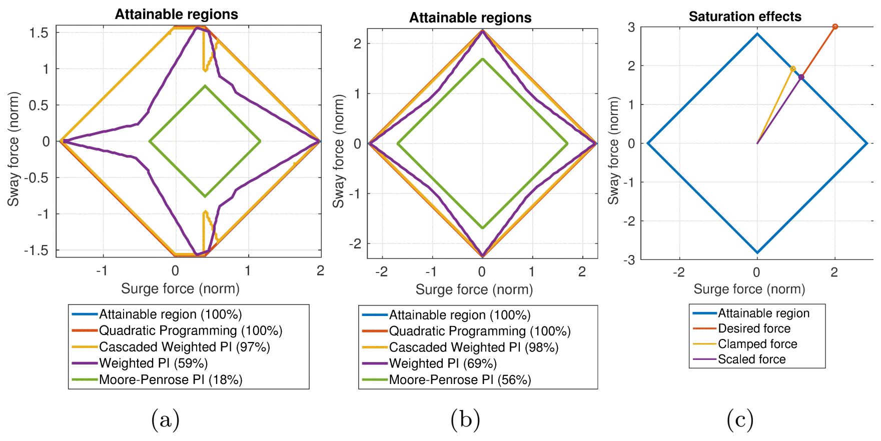
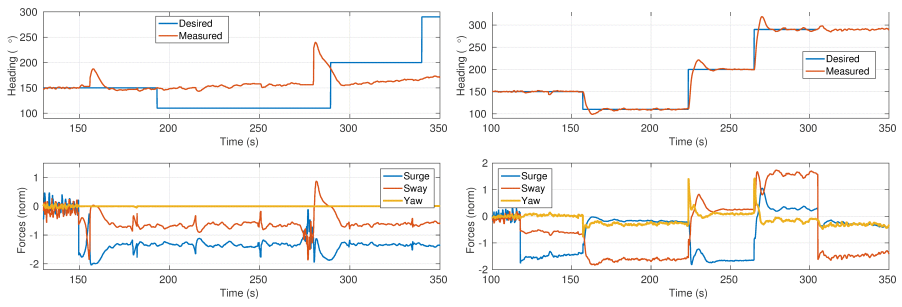


Figure 13: Attainable virtual forces regions using different *inverse* allocation methods for a Buddy and b Pladypos. The coverage percentile relative to the maximum region is noted in plot legends. Force outside attainable regions are c saturated using different clamping or scaling.

Taking into account thruster characteristics a polyhedron is constructed that contains the operation domain. Ignoring the yaw torque and analysing only the surge and sway forces reduces the problem to two dimensions. Operating domain slices for both vehicles are shown in Figure 13. Four *inverse* allocation methods are analysed: *a)* Moore-Penrose pseudo-inverse (PI); *b)* generalized weighted PI; *c)* cascaded weighted PI, and *d)* quadratic programming inverse. Each inverse covers a subset of the attainable domain with quadratic programming covering the complete set. The cascaded weighted pseudo-inverse has high coverage, however, the concave “bites” seen in Figure 13a can cause control degradation and actuator jitter. This occurs for control signals within the “bites” which are clamped discontinuously between boundaries. Asymmetric thrusters with the same thruster characteristics are handled better by the cascaded approach and high coverage is achieved without concavities. The weight matrix is constant and selected to distribute more thrust on stronger thrusters. Adaptive weight selection schemes can improve the coverage but raise complexity. The direct approach with the Moore-Penrose inverse has low coverage especially for complex characteristics.

Handling saturation

The straightforward approach for handling thruster saturation is clamping. Consider an ideal thruster with normalized force in $[-1, 1]$. Clamping considers thrusters individually and saturates request into the operating domain, i.e., $F_{i,achieved} = sat\left(F_1, -1, 1\right)$ where sat denotes the saturation function. Figure 13c shows the achieved thrust vector after clamping and a change in direction is noticeable. The resulting surge and sway vectors produce the vehicle course over ground resulting in unwanted movement and a wrong direction. During open-loop position control, once saturation is entered, the transient path is no longer a straight line. Therefore, a scaling approach is preferred, i.e. $F_{i,achieved} = \frac{F_i}{max\{F_1,...,F_4\}}$, since it maintains the desired vector direction as shown in Figure 13c. However, when disproportional forces and torques are requested, e.g. large surge with small sway and yaw requests, the scaling saturation additionally reduces the small values. These can fall into the thruster dead-band failing to produce the required forces.



(a) (b)

Figure 14: Heading control and achieved thrust with a scaling saturation and b conservative scaling saturation. The heading control fails when scaling is applied. However, when a minimum torque allowed the heading control becomes functional.

Consider the example shown in Figure 14 depicting Pladypos heading control during dynamic position control. The vehicle is commanded towards a new point $25\,m$ away resulting in large surge request. The desired heading changes, however, during movement Pladypos requires a small restoring torque, due to manufacture asymmetries, for maintaining constant heading. Observe that the desired restoring torque is not achieved and heading control fails. Specifying a minimum torque which is not affected by scaling resolves the issue. Basically, torques below the minimum value are allocated and the remaining power is distributed and scaled accordingly. Results of this modified scaling approach are shown in Figure 14b where the heading control is restored and yaw torque becomes competitive.

Consider playing with some techniques described here to see if you can improve the dynamic positioning behaviour in the seminar. E.g., if you saturate the control signal already at the PID controller, rather than looking at the τ_a values the oportunity to apply these techniques is lost.

The propulsion system

Thrusters are the most common propulsion system for underwater vehicles. Small scale vehicles use electrical brushed/brushless thrusters while larger scale ROV employ hydraulics. The concept between the two is similar since the voltage/pressure and current/flow generate torque and a certain propeller RPM which in turn generates thrust force. Since a small scale vehicle is used in the Seminar, the focus is on electrical thrusters.

The thrust produced by a fixed variable speed actuator can be described with

$$F_i = K_i n_i |n_i| \tag{23}$$

where F_i, K_i and n_i are the thrust force, thruster coefficient and RPM of the i-th thruster. K_i is related to the propeller characteristics rather than motor characteristics. This common affine model is often used for approximating the generated thrust. However, linear dependency on RPM as well as combined linear and quadratic relationships are possible. Observe that this model holds for static thrust, i.e., where a thruster is not moving through water. The forward speed of the actuator can be incorporated in a bilinear thruster model

$$F_i = K_i n_i |n_i| - K_{vi} |n_i| v_i \tag{24}$$

where K_{vi} is the reduction coefficient due to water flow around the thruster and v_i is the advance speed of the thruster. Nevertheless, the simpler affine model is often preferred as identifying K_{vi} can be problematic. In general, depending on propeller design, the forward and reverse thrust are asymmetrical and this is taken into account in equation (25).

$$F_i = \begin{cases} K_{ipos} n_i |n_i| & \text{for } n_i \geq 0 \\ K_{ineg} n_i |n_i| & \text{for } n_i < 0 \end{cases} \tag{25}$$

Estimation of the thrust force requires closed loop RPM control. Small scale vehicles often have only open-loop thruster control and do not measure rotation speed. The closest approximation is assuming $n_i \propto U_s p_i$ where U_s and p_i are the supply voltage and normalized PWM, respectively. The thrust force can be expressed as

$$F_i = K_i U_s^2 p_i |p_i|, p_i \in [-1, 1] \tag{26}$$

where p_i becomes the lowest NGC input into the system. when voltage stabilization is not available the supply voltage has to be taken into account during thrust force estimation.

Consider that similar concepts can be applied for the torque Q_i which is generated due to thruster rotation. This torque is especially noticeable on single propeller vehicle, e.g., the torpedo shaped AUV. The thesis deals mostly with multi-propeller vehicles that are equipped with counter-rotating propellers to mitigate the influence of Q_i.

Thruster mapping

Thruster mapping is a procedure where the achieved static thrust F_i is mapped against one control input, e.g., n_i or p_i. Based on this mapping the thrust coefficient K_i can be estimated in order to compensate the static thruster non-linearity.

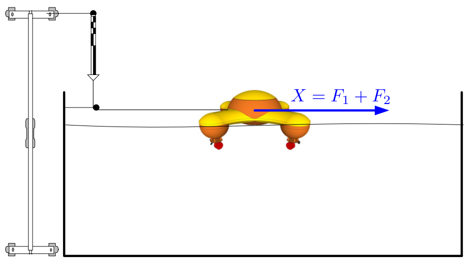


Figure 15: The thruster mapping apparatus.

The apparatus connects a force sensor to the vehicle and performs measurements for a set of input values. The mapping can be performed for individual thrusters; however, the location on the vehicle influences the thruster characteristics. Therefore, mapping is done directly on the vehicle. Thrusters are mapped pairwise and the resulting force is measured. To avoid rotation the effective thrust on each thruster pair must be identical. Therefore, the individual thrust force can be measured indirectly.

Two mappings, for Buddy and Pladypos vehicles, are presented in this section. Common mapping results are shown with the Pladypos dataset. The Buddy dataset shows the mounting position influence on the thruster characteristics. The mapping results are summarized in Table 4 and Table 5 while fitted measurements are shown in Figure 16.

Table 4: Pladypos thrust coefficients for a standard model fit. Separate K_i are provided for forward (+) and reverse (-) directions. The fit error is shown as R^2 approximation. All parameters are in normalized units and multiplied by 1000 to show significant decimal places.

Parameter	F_1	F_2	F_3	F_4
$K_i^+ * 10^{-3}$	1.394	1.734	2.060	1.831
$K_i^- * 10^{-3}$	1.075	0.680	0.861	0.870
R^2	0.947	0.985	0.964	0.982

Pladypos thrusters have an emphasized quadratic characteristics and a noticeable difference exists between forward and reverse directions. Although the thruster models are the same, there is difference in provided thrust. This can be attributed to imperfections in the mounting, propeller and shaft seals which introduced additional resistance. Averaging the results among thruster into a single quadratic model can reduce the allocation complexity. Residual errors introduced by a single model are usually compensated by the control feedback.

Buddy horizontal thrusters show good repeatability in the forward direction but the reverse direction is completely different giving small thrust force on

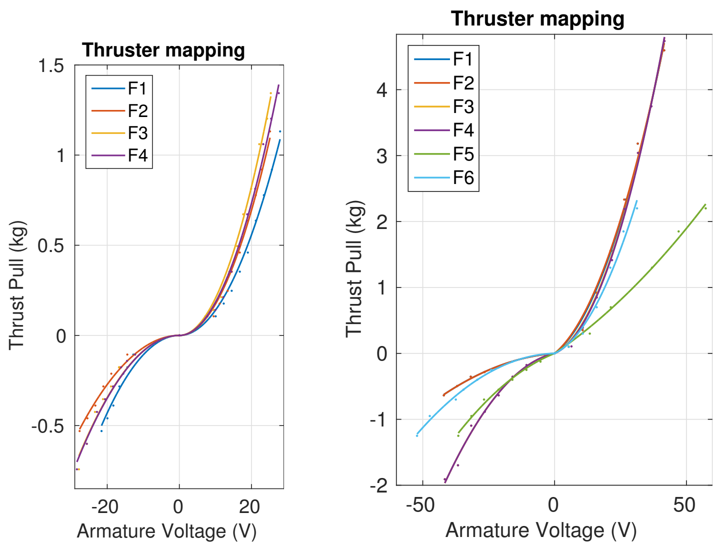


Figure 16: The figure shows the fitted thruster mapping measurement. Pladypos results are shown on the left and for Buddy on the right figure. Notice that thruster F_1–F_4 match well in forward directions but exhibit a larger difference in reverse direction.

forward thrusters. This is caused by the tablet mounted directly in the flow path of frontal thrusters. Similarly, the vertical thrusters show differences due to vehicle body obstructions in the flow path. The unobstructed non-linear characteristics is closest to F_3 and F_4 thrusters. Based on this, it is noticeable that thruster mounting influences the non-linear thruster characteristics making it impossible to describe all thrusters with a single averaged model. Therefore, separate characteristics have to be used for each thruster or the two thruster groups.

Table 5: Buddy thrust coefficients for a polynomial fit $K_i U_s^2 p_i^2 + K_i U_s p_i$. The polynomial fit is an alternative to quadratic only to reduce the overall fitting error. Separate K_i and K_{l_i} are provided for forward (+) and reverse (-) directions. The fit error is shown as R^2 approximation. All parameters are in normalized units and multiplied by 100 to show significant decimal places.

Parameter	F_1	F_2	F_3	F_4	F_5	F_6
$K_i^+ * 10^{-2}$	0.202	0.194	0.230	0.236	0.022	0.178
$K_{l_i}^+ * 10^{-2}$	2.952	3.274	1.896	1.657	2.689	1.883
$K_i^- * 10^{-2}$	0.028	0.028	0.093	0.093	0.047	0.043
$K_{l_i}^- * 10^{-2}$	0.308	0.308	0.839	0.839	1.592	0.109
R^2	0.994	0.993	0.990	0.990	0.994	0.986

For this seminar **H2OmniX** you can consider this type of fit as for the Buddy vehicle. However, it is important to ensure that the R^2 error is within normal bounds, i.e., the error should be above 0.98.