
# AUTOBargeSim: MATLAB® toolbox for the design and analysis of the guidance and control system for autonomous inland vessels

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-22 (2025) 818–823

# AUTOBargeSim: MATLAB® toolbox for the design and analysis of the guidance and control system for autonomous inland vessels ⋆

**Abhishek Dhyani** * **Amirreza Haqshenas Mojaveri** ** **Chengqian Zhang** *** **Dhanika Mahipala** **** **Hoang Anh Tran** **** **Yan-Yun Zhang** ** **Zhongbi Luo** ** **Vasso Reppa** *

* *Delft University of Technology, The Netherlands.*
** *Katholieke Universiteit Leuven, Belgium.*
*** *Chalmers University of Technology, Sweden.*
**** *Norwegian University of Science and Technology, Norway.*

**Abstract:** This paper introduces AUTOBargeSim, a simulation toolbox for autonomous inland vessel guidance and control system design. AUTOBargeSim is developed using MATLAB and provides an easy-to-use introduction to various aspects of autonomous inland navigation, including mapping, modelling, control design, and collision avoidance, through examples and extensively documented code. Applying modular design principles in the simulator structure allows it to be easily modified according to the user's requirements. Furthermore, a GUI interface facilitates a simple and quick execution. Key performance indices for evaluating the performance of the controller and collision avoidance method in confined spaces are also provided.



*Keywords:* Guidance, navigation and control (GNC) of marine vessels; Nonlinear and optimal control in marine systems; Modeling, identification, simulation, and control of marine systems

## 1. INTRODUCTION

The guidance, navigation, and control (GNC) system plays a crucial role in the safe and reliable operation of ASVs. This system comprises technologies ranging from modern sensors to complex algorithms and software that enable the ASV to perceive its environment and have situational awareness and decision-making capabilities. Furthermore, simulation-based testing of the GNC system is an essential step in the design process. A few freely available scientific simulation software/ toolboxes have been proposed for maritime simulation (See e.g., Perez et al. (2006); Sukas et al. (2019); Blindheim and Johansen (2021); Krasowski and Althoff (2022); Tengesdal and Johansen (2023); Clement et al. (2024)). Arguably, the marine systems simulator (MSS) (Perez et al. (2006)) is the most popular and widely used MATLAB® -based toolbox, consisting of various classes of models, transformation functions, guidance and control algorithms, among others. More recently, simulation toolboxes focusing on evaluating collision avoidance algorithms have also been proposed (see Krasowski and Althoff (2022); Tengesdal and Johansen (2023); Clement et al. (2024)).

While the existing simulation platforms are well-equipped with many of the necessary functionalities for autonomous vessel simulation, the majority of these platforms consider open-sea simulation only. However, confined waterways such as inland waterways, ports and canals, which are key use cases for the deployment of autonomous vessels, are not considered. Operating vessels in confined waters is particularly challenging as they are constrained by several factors, such as canal width, infrastructures, dynamic water levels, river currents, and riverbed variations. The existing platforms rely on mathematical models to simulate vessel maneuvers that mimic the characteristics of a seagoing vessel. The hydrodynamic forces generated due to shallow water depth in inland waterways can significantly impact the vessel's motion and maneuverability. By default, these platforms do not offer quantitative performance indicators for the evaluation of guidance and control algorithms. These metrics provide useful benchmarking data for comparing various algorithms to state-of-the-art methods.

In this work, we address these gaps by introducing AUTOBargeSim, a MATLAB®-based simulation toolbox for the design and evaluation of guidance and control algorithms for autonomous inland navigation. AUTOBargeSim has been created with a focus on modularity, reproducibility, and ease of use as the key design principles and is freely available for research and educational purposes. It allows the users to visualize inland map features, set up scenarios

⋆ Corresponding author's e-mail: a.dhyani-1@tudelft.nl
The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 955.768 (MSCA-ETN AUTOBarge). This publication reflects only the authors' view, exempting the European Union from any liability. Project website:http://etn-autobarge.eu/.
All authors contributed equally.


10.1016/j.ifacol.2025.11.736

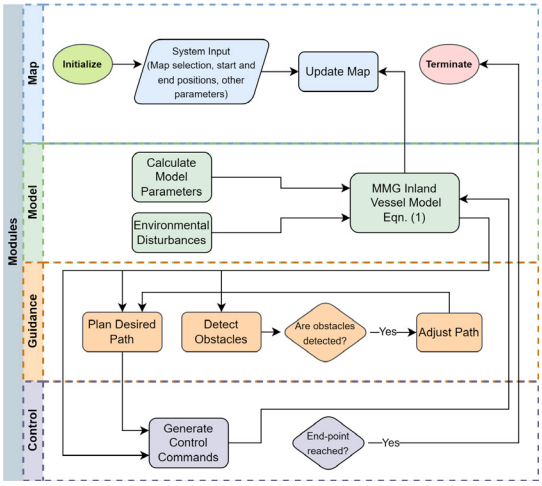


Fig. 1. Flowchart depicting the various modules and the flow of control

for vessel navigation, select various parameters, simulate the vessel motion, and evaluate the performance of vessel path following and collision avoidance algorithms. Various examples provide an introduction to applying specific classes and methods from the toolbox based on the user's requirements.

The remainder of the paper is organised as follows: in Section 2, instructions for installing and using the simulator toolbox are given. Its design and structure as well as the comprising methods and parameters, are detailed in Section 3. Further, qualitative metrics for performance evaluation are presented and described in Section 4. Finally, the conclusions and scope for future development are discussed in Section 5.

## 2. INSTALLATION AND USAGE

AUTOBargeSim can be downloaded from its GitHub repository [1]. The toolbox includes several tutorials for easily customizing the methods included in the modules. These tutorials can be found in the respective module directories. Furthermore, the GUI allows a simple and fast execution of the toolbox, with the possibility to adjust some critical inputs, such as the map area, controller gains, etc. It can be executed by running the script `main_gui.m`, selecting between various options in the GUI window and pressing the *Execute* button. It should be noted that the GUI currently has a limited number of inputs; however, the underlying methods allow far more input flexibility.

## 3. DESIGN AND STRUCTURE

The simulator follows a modular design. Based on functionality, it is divided into several modules, and under each module, various algorithms and demos are provided. The overall structure and the connection between various modules are illustrated in Fig. 1. The mathematical model describing the IWV dynamics and the actuators is defined in the Model module while considering environmental disturbances, including currents and shallow-water effects. Static environmental data, including waterway boundaries and the locations of static objects along the path, is extracted from chart files within the Map module. The map module also provides a set of initial waypoints and, subsequently, a desired path between two selected points on the map, which serve as the inputs to the Guidance module. The Guidance module employs a guidance law to compute the desired/ reference course angles for vessel navigation. Next, the reference course angles are provided to the Control module, and a control law computes the desired rudder angle for steering the vessel (Zhang et al. (2025)). In the case of a collision avoidance scenario, the motion of a *target vessel* is also simulated, and the *own vessel* performs a collision avoidance maneuver if required. The simulation terminates when the vessel successfully reaches the provided endpoint. Finally, all the processed data is used to update the map visualization, and the various performance metrics are displayed. Each module is explained in detail in the subsections that follow.

### 3.1 Model Module

Inland vessels frequently operate on confined waterways in the presence of dynamic traffic and hydraulic structures such as bridges and locks. Therefore, a reliable maneuvering model for accurately predicting the vessels' dynamics is critical for safe navigation.

The maneuvering model follows the popularly known Manoeuvring Modelling Group (MMG) model (Ogawa and Kasai (1978)) architecture, where the hydrodynamic forces and moments are derived into individual components. The original MMG model was developed for classic commercial vessels in open water. Due to its flexible and modular structure, the model can be extended by incorporating shallow water effects to account for the influencing factors of inland waterways. The rigid body dynamics can be represented as:

$$\begin{aligned}(m+m_x)\dot{u}-(m+m_y)vr-x_G m r^2 &= X_H+X_P+X_R,\\ (m+m_y)\dot{v}-(m+m_x)ur+x_G m\dot{r} &= Y_H+Y_R,\\ \left(I_z+x_G^2 m+J_z\right)\dot{r}+x_G m(\dot{v}+ur) &= N_H+N_R,\end{aligned} \tag{1}$$

where the left side contains the mass $(m, m_x, m_y)$ and inertia terms $(I_z, J_z)$; $(u, v, r)$ represent the surge, sway velocity and the yaw rate; $(X, Y, N)$ denote the summation of the surge force, the sway force, and the yaw moment. The subscripts $(H, P, R)$ represent the hydrodynamic force of the individual components acting on the hull, propeller, and rudder, respectively. Therefore, the model is divided into two classes: `modelClass.m` calculates the hydrodynamic forces acting on the hull, and `actuatorClass.m` calculates the propeller thrust and rudder forces (see Fig. 2). It should be noted that the shallow water effect is included by the modified terms acting on the ship hull $(X_H, Y_H, N_H)$, including the added resistance and sway force due to the reduced under-keel clearance.

*Hydrodynamic Forces on the Hull*
The force and moment acting on the hull can be calculated as:

$$\begin{aligned}X_H &= 0.5\rho LTU^2 X'_H,\\ Y_H &= 0.5\rho LTU^2 Y'_H,\\ N_H &= 0.5\rho L^2 TU^2 N'_H,\end{aligned} \tag{2}$$

where $\rho$ is the fresh water density, $L$ is the vessel length, $T$ is the vessel draught, $U$ is the total speed, and

[1] https://github.com/AUTOBarge/autobargesim

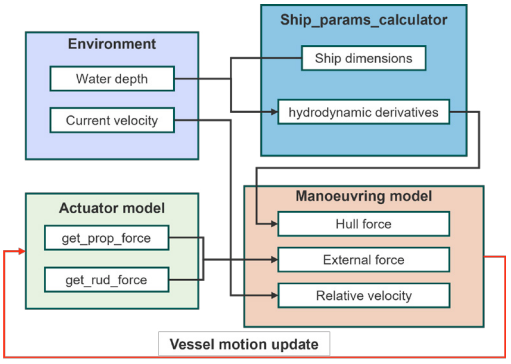


Fig. 2. Architecture of manoeuvring and actuator module.

$(X'_H, Y'_H, N'_H)$ are non-dimensional forces and moment, given as:

$$X'_H = -R'_0 \cos^2 \beta_m + X'_{\beta\beta}\beta_m^2 + X'_{\beta r}\beta_m r' + X'_r r'^2 + X'_{\beta\beta\beta\beta}\beta_m^4, \quad (3)$$

$$Y'_H = Y'_\beta \beta_m + Y'_r r' + Y'_{\beta\beta\beta}\beta_m^3 + Y'_{\beta\beta r}\beta_m^2 r' + Y'_{\beta rr}\beta_m r'^2 + Y'_{rrr} r'^3, \quad (4)$$

$$N'_H = N'_\beta \beta_m + N'_r r' + N'_{\beta\beta\beta}\beta_m^3 + N'_{\beta\beta r}\beta_m^2 r' + N'_{\beta rr}\beta_m r'^2 + N'_{rrr} r'^3, \quad (5)$$

where $-R'_0$ is the resistance coefficient including shallow water effect (Zhang et al. (2023)), and $(X'_{\beta\beta}, ..., N'_{rrr})$ are the so called hydrodynamic derivatives which can be calculated from the function `ship_params_calculator` using empirical formulas based on ship dimensions.

*Propeller and Rudder Forces*

The actuator module (`actuatorClass`) is based on a conventional propeller-rudder configuration. The thrust of a ducted propeller can be calculated using the function (`get_prop_force`) based on the equation:

$$X_P = (1-t)\rho n_P^2 D_P^4 K_T(J), \quad (6)$$

where $t$ is the thrust deduction, $n_P$ is the propeller rpm, $D_P$ is the propeller diameter and $K_T(J)$ is the thrust coefficient as a function of advanced ratio $J$. In this work, the open water coefficient is derived from the open-source propeller design tool OpenProp (Epps et al. (2009)). The rudder steering force and moment are calculated using function (`get_rud_force`) as follows:

$$\begin{aligned} X_R &= -(1-t_R)(F_N^P + F_N^S)\sin\delta, \\ Y_R &= -(1+\alpha_H)(F_N^P + F_N^S)\cos\delta, \\ N_R &= -(x_R + \alpha_H x_H)(F_N^P + F_N^S)\cos\delta, \end{aligned} \quad (7)$$

where $F_N$ is the rudder normal force, the superscript $P$ and $S$ denote the rudder at port side and starboard side, $t_R$ denotes the steering resistance deduction factor, $\alpha_H$ represents the rudder force increase factor, $x_R$ is the relative location of rudders and keeping identical at each side, $x_H$ is the relative acting point of the additional lateral force, and $\delta$ is the rudder angle. Here, $F_N$ is given as:

$$F_N = 0.5\rho A_R U_R^2 \left(\frac{6.13\Lambda}{\Lambda + 2.25}\sin\alpha_R\right), \quad (8)$$

where $A_R$ is the rudder area, $U_R$ is the incoming flow velocity at the rudder ($U_R = \sqrt{u_R^2 + v_R^2}$), $\Lambda$ is the rudder aspect ratio and $\alpha_R$ is the effective inflow angle at the rudder during manoeuvring (see Ogawa and Kasai (1978)).

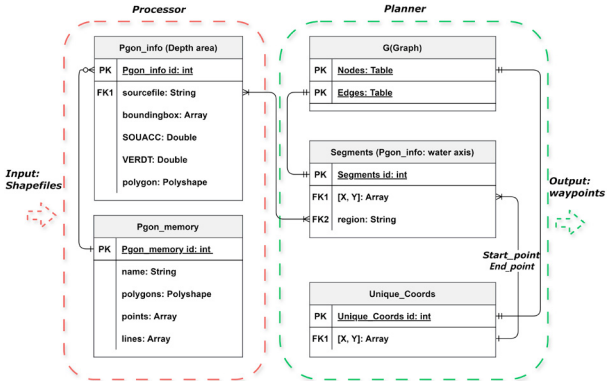


Fig. 3. Entity Relationship Diagram (ERD) shows the relationship between different data in each submodule of the map module.

### 3.2 Map Module

The Map module processes Inland Electronic Navigational Chart (IENC) data to provide essential environmental information for path planning and collision avoidance. It comprises two main submodules: the Processor and the Planner. The Processor converts input shapefile (`.shp`) data into a structured format called `pgon_memory`, while the Planner uses this data to generate waypoints with associated depth information, as shown in Fig. 4. Fig. 3 illustrates the relationships between data entities in these modules. Due to MATLAB's inability to decode S-57 standard ENC files (`.000` files), which contain standardized navigational data with semantic tags and attributes, we employ a preprocessing step using (GDAL/OGR contributors (2020)) Python library. By extracting specified regions and attributes from the IENC and structuring them into spatial data, compatibility with the simulator is ensured.

*Processor*

The Processor reads shapefiles generated from the `.000` ENC files and categorizes them based on predefined navigational features. Users are allowed to add predefined features from the S-57 standard as needed. The following are

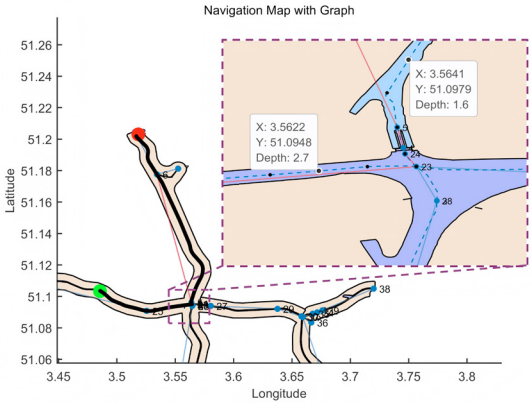


Fig. 4. The navigation map for the Ghent area includes the starting point (green), end point (red), the waypoints (black), and the graph composed of nodes and edges. An attached, localized, zoomed-in view of the navigation map, displaying the coordinates and depth information of waypoints within the waterway.

the main features: Depth Areas (depare): Polygons with attributes such as Sounding Accuracy (SOUACC), Vertical Datum (VERDAT), and polygon extent (boundingbox), providing essential depth and positional data. Waterway Axes (wtwaxs): Polylines indicating central navigational paths within waterways. Bridges (bridge): Polygons representing bridge locations and dimensions, important for height restrictions. Land Areas (lndare): Polygons denoting non-navigable regions.

For each category, the Processor reads geometric data and attributes from the shapefiles, constructing the pgon_memory structure with fields: name: Category name (e.g., depare, wtwaxs). points: Coordinates of point features. lines: Coordinate arrays of line features. polygons: polyshape objects of polygon features. info: Attributes for each geometric entity.

For depth areas, the Processor constructs polyshape objects and extracts depth attributes like SOUACC and VERDAT, as well as the boundingbox and the unique identifier of the source .000 file (sourcefile or region), storing them in depare.info. For processing waterway axes, it extracts coordinates, removes redundant points to maintain data integrity, and splits the data into multiple segments, associating each line with relevant metadata (region).

*Planner*
The Planner uses pgon_memory to perform path planning by constructing a navigational graph denoted as $G$ illustrated in Fig. 4. This graph comprises nodes and edges derived from wtwaxs. Nodes are derived from unique coordinates (unique_coords) at segment endpoints of waterway axes, and edges represent connectivity between nodes. Disconnected components are linked by connecting the nearest nodes.

To ensure the graph $G$ is fully connected, the Planner checks for disconnected components using functions conncomp and connects them by identifying the closest nodes between components. Depth information from depth areas (depare) is associated with the nodes and edges of the graph by spatial queries. Specifically, the Planner performs a hierarchical search from region to boundingbox to polygon to determine if nodes or path points are within depth polygons and assigns depth values from attributes SOUACC and VERDAT to the corresponding nodes and edges.

Path planning is performed by finding the shortest path in the graph $G$ that satisfies depth constraints using the Dijkstra algorithm (Dijkstra (2022)). The Planner considers the given starting point (given_point1) and ending point (given_point2), locating the nearest nodes in unique_coords and planning a path between them. The resulting path is stored in path_points, with the corresponding depth information saved as path_depths.

To refine the path for practical navigation, the Planner removes duplicate points and smooths the trajectory. The final output is a series of waypoints in path_points with corresponding depth information in path_depths, ensuring the planned path is safe and efficient given the vessel's draft and environmental constraints.

### 3.3 Guidance Module

The main goal of the Guidance module is to fulfil two tasks: Track Keeping and Collision Avoidance. The Track Keeping submodule ensures that the ship sails toward the next waypoint. The Collision Avoidance submodule adjusts the speed and course angles provided by the Track Keeping submodule of the ship to avoid obstacles.

*Track Keeping*
The Track Keeping submodule receives the waypoint list from the map class and provides a course angle and speed reference for Collision Avoidance to ensure that the ship follows the desired waypoints. In this submodule, we define the track-keeping class as a superclass containing a common function that can be used by subclasses created for specific track-keeping controllers. In this class, we provide a function to find the current active waypoint in the waypoint list and the position of the ship. The key properties are the radius of acceptance, $R_a$, and pass_angle_threshold, which are used to identify if the ship passed a waypoint.

The default track-keeping controller is the Line-of-Sight (LOS) steering algorithm (Fossen (2011)). The only property of this subclass is the proportional gain of the LOS steering law $K_{p_{los}} = 1/D_{los}$, with $D_{los}$ being the look-ahead distance. The main function is compute_LOSRef, which receives the current state of the vessel, a waypoint_list containing the coordinates of the waypoints, and the expected nominal speeds at each waypoint. It then returns the reference course angle, $\chi_d$.

*Collision Avoidance*
The collision avoidance submodule is responsible for modifying the course and speed commands provided by the Track Keeping submodule to avoid collisions. This submodule contains two main classes: a superclass named colav and a subclass named sbmpc. The colav class includes common methods necessary to implement any collision avoidance algorithm, such as internal kinematic models for trajectory prediction. Users are encouraged to use these methods to implement custom collision avoidance algorithms in this simulator. End-users interact with the application layer of the module, which includes classes related to specific collision avoidance algorithms. Currently, only one such class is available in this simulator: the class implementing the Scenario-based Model Predictive Control (SBMPC) algorithm (Johansen et al. (2016)). The SB-MPC algorithm presents a proactive collision avoidance strategy based on simulation and receding horizon optimization. It guarantees compliance with COLREGS Rules 6, 8, and 13-19 and mitigates collision risks by evaluating a cost function that also accounts for maneuvering effort. The exact details and components of the algorithm are provided in Mahipala and Johansen (2023).

The sbmpc class contains the methods and properties related to the use of the SBMPC algorithm. Its key properties include T and dt, which represent the prediction time horizon and the algorithm sample time, respectively. Furthermore, a property named tuning_param contains the constant tuning parameters listed in Johansen et al. (2016) as subproperties, which can be modified using optional arguments when initializing the sbmpc object. The

class only has one public method that is accessible to the user, which is named `run_sbmpc()`. The function takes the current state of the vessel, the course and the speed commands from the Track Keeping submodule, course, and speed modifications from the previous time step, and the states of one or more target vessels as input. The outputs of the function are the course and speed modifications for the current time step.

### *3.4 Control Module*

The Control module provides two low-level controllers, namely Proportional-Integral-Derivative (PID) control and Model Predictive Control (MPC), to calculate the required rudder angle for navigating the vessel and ensuring that it tracks the desired heading angle. The Control module is defined as a class containing several properties and methods. The properties include `num_st`, `num_ct`, and `Flag_cont`, which represent the number of state variables, the number of control variables, and the selected control method, respectively. It also includes `pid_params`, a structure data type containing the PID controller's parameters including $K_p$: proportional gain, $T_i$: integral time-constant, $T_d$: derivative time-constant, and `psi_d_old`, `error_old` for storing the desired heading angle and heading angle error from the last iteration. Finally, the structure `mpc_params` contains the MPC controller's parameters including $T_s$: sampling time, $N$: prediction horizon, `headingGain`: weighting gain for heading angle, `rudderGain`: weighting gain for rudder angle, `max_iter`: maximum number of iterations for the MPC and `deltaMAX`: maximum value for rudder angle. The methods within this class manage tasks such as handling the state variables and updating the control parameters. The property `states` represents the controlled state variables of the system, $s = [r, \psi]^T$. Further, the variables $\psi_d$ and $r_d$ stand for the desired heading angle and turning rate, which are the outputs of the Guidance module, and form the reference input vector, $s_{\text{ref}} = [r_d, \psi_d]^T$. A detailed explanation follows in the respective subsections below.

*PID Controller*
The PID controller is a classical control technique popular for its simplicity and ease of implementation. Within the Control module, the PID controller is the default controller choice. It determines the rudder angle by minimizing the error between the desired and actual heading angles. The control law can be stated as:

$$\delta_c(t) = K_p\psi_e(t) + T_d(\psi_e(t) - \psi_e(t-1)) + \frac{1}{T_i}\left(\sum_{i=0}^{t}\psi_{e_i}\right), \quad (9)$$

where $\psi_e(t) = \psi(t) - \psi_d(t)$ is the heading angle error, and $K_p$, $T_i$ and $T_d$ are the controller's proportional gain, the integral and derivative time constants, respectively. The method `LowLevelPIDCtrl` computes and outputs the desired rudder command $\delta_c$ by using the $\psi$ variable from `states` and the reference heading angle $\psi_d$ as inputs, respectively.

*Model Predictive Control (MPC)*
This subsection describes the implementation of MPC within the Control module. The MPC determines the rudder control input for the vessel based on the desired heading angle and turning rate. The Control module includes multiple methods for formulating the MPC for rapid implementation. These methods include `init_mpc`, `initial_guess_creator`, `constraintcreator` and `LowLevelMPCCtrl`. The `init_mpc` method employs CasADi (Andersson et al. (2019)) as the backbone to formulate a graph stored in `mpc_nlp` for solving the constrained Nonlinear Programming (NLP) problem defined within the MPC. The `initial_guess_creator` method requires two inputs, the initial states and the initial control input, to construct the initial guess vectors and store them in an internal structure. The `constraintcreator` obtains its necessary information from `mpc_params` and generates a built-in structure for storing all the needed arguments to be passed on to the NLP solver created by the `init_mpc`. The method for building the MPC controller is `LowLevelMPCCtrl`, which uses `states`, $s_{\text{ref}}$, `args`, `initial_guess` and `mpc_nlp` as its inputs. The variable `args` is the output of the `constraintcreator` method and presents the NLP arguments. Note that this method is required to be called at each iteration of the simulation and solves the following optimization problem:

$$\begin{aligned} &\min_{\delta_c} J(s, s_{\text{ref}}, \delta_c, k) \\ \text{subject to } &s[k+1] = f(s[k], \delta_c[k]), \\ &s[0] = s_0, \\ &s \in U_{\mathbf{s}}, \end{aligned} \quad (10)$$

where $f(s[k], \delta_c[k])$ denotes the prediction model describing the relation between the states and the input, and $U_s \subset \mathbb{R}^2$ represents the set of permissible states (Fossen (2011)). Moreover, $J$ represents the cost function and can be formulated as $J(s, s_{\text{ref}}, \delta_c, k) = \sum_{i=1}^{N}\Big[(s - s_{\text{ref}})^T_{(k+i)}Q(s - s_{\text{ref}})_{(k+i)} + \left(\delta_{c(k+i-1)}\right)^2 p\Big]$, where $Q \in \mathbb{R}^2$ and $p \in \mathbb{R}$ are the state- and control- weights, respectively, and they are chosen by the designer. Solving the optimization problem (10) yields the optimal rudder angle for the time instance $k+1$.

## 4. PERFORMANCE EVALUATION

The guidance and control module includes functions used to evaluate the controllers, namely the track-keeping and path-following controllers.

### *4.1 Track-keeping Control*

The `perf` function under `track-keeping` class provides the following indices:

(1) **Nominal distance ($D_{\text{nom}}$)**: The cumulative distance between waypoints from the start point to the endpoint, and is computed as:
$$D_{\text{nom}} = \sum_{i=1}^{N-1}\sqrt{(x_{i+1}^{wp} - x_i^{wp})^2 + (y_{i+1}^{wp} - y_i^{wp})^2},$$
where $[x_i^{wp} y_i^{wp}]$ is the position of $i^{th}$ waypoint, and $N$ is the number of waypoints.
(2) **Nominal navigation time ($T_{\text{nom}}$)**: The "unreal" time it takes for the vessel to sail from the start point to the endpoint with the nominal speed, $v_{\text{ref}}$. The nominal navigation time is calculated as $T_{\text{nom}} = D_{\text{nom}}/v_{\text{ref}}$.

(3) **Actual navigation distance ($D_{\text{actual}}$)**: The actual navigation distance that the vessel sails from the start point to the endpoint, and is computed as:

$$D_{\text{actual}} = \sum_{i=1}^{\mathcal{I}-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2},$$

where $[x_i, y_i]$ is the position of the ship at time $i$, and $\mathcal{I}$ is the total simulation iteration taken for the vessel to reach the endpoint.

(4) **Actual navigation time ($T_{\text{actual}}$)**: The "unreal" time that it takes for the vessel to sail from the start to the endpoint and is calculated as $T_{\text{actual}} = \Delta T \times \mathcal{I}$, with $\Delta T$ is the sampling period of the simulation.

### *4.2 Path Following Control*

To evaluate the controller's performance for the vessel's path following control, the following key performance metrics have been utilized:

(1) **Cumulative Heading error** ($E_\psi$): The cumulative heading error $\psi_{e,c}$ is calculated as $E_\psi = \int_{t=1}^{N} (\psi(t) - \psi_d(t))dt$, where $\psi_d(t)$ is the desired heading angle at the time instant $t$.

(2) **Cumulative cross-track error** ($E_{\text{xt}}$): The Cumulative cross-track error is calculated by

$$E_{\text{xt}} = \int_{t=1}^{N} \sqrt{(x(t) - x_{cl}(t))^2 + (y(t) - y_{cl}(t))^2} dt,$$

where $(x_{cl}(t), y_{cl}(t))$ are the points of the desired path that are at the closest distance from the vessel's position at the time instant $t$.

In addition, metrics for the rudder angle fluctuation rate and energy consumption can also be incorporated, and are planned for future work.

## 5. CONCLUSION AND FUTURE WORKS

This paper presented AUTOBargeSim, a toolbox for simulating autonomous inland vessels. AUTOBargeSim provides a vital environment for testing various aspects of autonomous vessel navigation in inland waterway environments. Its modular design provides improved flexibility, allowing users to easily modify or replace individual modules without impacting the functionality of others. Further, AUTOBargeSim is extensively documented and openly available, promoting reproducibility in the design and development of marine navigation systems.

The future developments of the simulator will aim to incorporate additional aspects of autonomous vessel operations, such as considering sensor characteristics and abnormal events. A communication module will be developed to allow information exchange between vessels, providing collaborative navigation capabilities. Moreover, the collision avoidance system will be evaluated against metrics suitable for inland waterway scenarios. Currently, the toolbox supports only constant-speed vessel simulations; however, it is planned to include variable-speed maneuvering capabilities to reflect real-world operational characteristics.

## REFERENCES


Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Blindheim, S. and Johansen, T.A. (2021). Electronic navigational charts for visualization, simulation, and autonomous ship control. *IEEE Access*, 10, 3716–3737.

Clement, B., Chaffre, T., Sarhadi, P., and Dubromel, M. (2024). Colsim, a simulator for hybrid navigation acceptability and safety. *IFAC-PapersOnLine*, 58(20), 147–152. 15th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles CAMS 2024.

Dijkstra, E.W. (2022). *A Note on Two Problems in Connexion with Graphs*, 287–290. Association for Computing Machinery, New York, NY, USA, 1 edition.

Epps, B., Chalfant, J., Kimball, R., Techet, A., Flood, K., and Chryssostomidis, C. (2009). Openprop: An open-source parametric design and analysis tool for propellers. In *Proceedings of the 2009 grand challenges in modeling & simulation conference*, 104–111.

Fossen, T.I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

GDAL/OGR contributors (2020). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. URL `https://gdal.org`.

Johansen, T.A., Perez, T., and Cristofaro, A. (2016). Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems*, 17(12), 3407–3422.

Krasowski, H. and Althoff, M. (2022). Commonocean: Composable benchmarks for motion planning on oceans. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 1676–1682. IEEE.

Mahipala, D. and Johansen, T.A. (2023). Model Predictive Control for Path Following and Collision-Avoidance of Autonomous Ships in Inland Waterways. In *2023 31st Mediterranean Conference on Control and Automation (MED)*, 896–903. ISSN: 2473-3504.

Ogawa, A. and Kasai, H. (1978). On the mathematical model of manoeuvring motion of ships. *International shipbuilding progress*, 25(292), 306–319.

Perez, T., Smogeli, O., Fossen, T., and Sorensen, A.J. (2006). An overview of the marine systems simulator (MSS): A simulink toolbox for marine control systems. *Modeling, identification and Control*, 27(4), 259–275.

Sukas, O.F., Kinaci, O.K., and Bal, S. (2019). Theoretical background and application of mansim for ship maneuvering simulations. *Ocean Engineering*, 192, 106239.

Tengesdal, T. and Johansen, T.A. (2023). Simulation framework and software environment for evaluating automatic ship collision avoidance algorithms. In *2023 IEEE Conference on Control Technology and Applications (CCTA)*, 186–193. IEEE.

Zhang, C., Dhyani, A., Ringsberg, J.W., Thies, F., Negenborn, R.R., and Reppa, V. (2025). Nonlinear model predictive control for path following of autonomous inland vessels in confined waterways. *Ocean Engineering*, 334, 121592.

Zhang, C., Ringsberg, J.W., and Thies, F. (2023). Development of a ship performance model for power estimation of inland waterway vessels. *Ocean Engineering*, 287, 115731.