NTNU

Norwegian University of
Science and Technology

Identification of failure modes in autonomous marine vehicles using Adaptive Stress Testing

Hanna Waage Hjelmeland

Project thesis
Supervisor: Anastasios Lekkas
Co-supervisor: Ole Jacob Mengshoel
Industrial co-supervisors: Bjørn-Olav Holtung Eriksen and Øyvind Smogeli
Submission date: December 20, 2021

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Abstract

Emission-free autonomous marine transportation systems are expected to contribute to positive societal development. However, due to the safety critical nature of such transportation systems, measures have to be made in order to ensure their safety. As new complex technologies are introduced to these systems, many of which are hard for humans to understand, thorough simulation based safety validation methods are needed. One such method is called the Adaptive Stress Testing (AST) method, which was proposed in 2015 (Lee, Kochenderfer, et al., 2015).

AST is a reinforcement learning-based approach which optimizes towards the most likely failure modes of a system. Optimizing towards failure leads to an efficient search as the need to perform exhaustive searches over all possible outcomes is eliminated. The AST method has been applied to air craft and car systems, where it has searched for near mid air collisions of air crafts and collisions between car and pedestrian, respectively. The AST method has shown promising results by efficiently identifying more failures compared to other search methods (Lee, Kochenderfer, et al., 2015).

The AST method has yet to be applied to marine systems. Therefore, this thesis aims to demonstrate the relevance of the AST method for marine vehicles by implementing the method in two cases of marine vessel simulators. In the first case, the control system of a simple marine vessel is tested by attempting to identify scenarios where disturbances has a significant impact on the vessel. In the second case, the collision avoidance system of the milliAmpere ferry is tested in order to find failure scenarios where the ferry collides with an obstacle. The AST method is in both cases able to identify scenarios that contain failure modes.

The results presented in this thesis demonstrate how the AST method successfully identifies failure modes of different parts of marine vehicle systems. Simultaneously, some drawbacks of the AST method are presented. The system tends to find scenarios that are very similar or where failure is unavoidable.

The results can provide insights to system developers as it exposes potentially dangerous scenarios that can be taken into consideration during development. However, the drawbacks of the method makes the results less relevant. In order to improve the results of the method in the marine domain, more domain knowledge can be introduced in the system such as the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) framework.

Preface and Acknowledgements

This project thesis was written in the fall of 2021 at the Norwegian University of Science and Technology (NTNU) at the department of Engineering Cybernetics. The thesis is written in an industrial collaboration with Zeabuz, a Norwegian company that springs out from NTNU (ZEABUZ, 2021).

The reader is expected to have basic knowledge about data science, mathematics, physics and control theory. The theory behind the methods used in the thesis will be shortly introduced. The figures are made by the author unless specified otherwise.

The aim of the thesis is to implement the AST method in simulation of autonomous marine vehicles. The implementation of the AST method rely on the open source AST Python Toolbox (Koren, Ma, et al., n.d.) and the Python milliAmpere Collision Avoidance (COLAV) simulator developed by Bjørn-Olav Holtung Eriksen at Zeabuz. Computer equipment used to perform the experiments was provided by the department of Engineering Cybernetics.

Writing the thesis has been a challenging, yet exciting and valuable process. It could not have been done without the guidance and support from my supervisor Anastasios Lekkas, who has been excited for the thesis on my behalf even when I have not been.

I also want to direct a thanks to the Zeabuz crew for showing great interest and curiosity and letting me work on their systems with the thesis and as a part time employee. A special thanks to my industrial co supervisors Øyvind Smogeli and Bjørn-Olav Holtung Eriksen whom have taken time from their busy schedule and assisted me with guidance, simulator details and industrial perspectives.

I have also had the pleasure of working with Ole Jacob Mengshoel from the Department of Computer Science, who has contributed to the development of the AST method and gave me a good introduction on the subject early on. Ole Jacob also put me in contact with two of the core delevopers of the AST method, Ritchie Lee and Mark Koren, whom I also owe a big thanks, with an extra emphasis on Mark Koren for taking time to debug my code with me all the way from California.

Additionally, I want to direct a big thanks to my family for supporting me and my friends for getting my mind out of the debugging-rabbithole by being great lunch mates.

Hanna Waage Hjelmeland December 20, 2021

Contents

List of Figures

Nomenclature

Reinforcement learning

X_t State at time t

U_t Action at time t

X State Space

U Action Space

$P_u(X_t, X_{t+1})$ Transition function

$R_u(X_t, X_{t+1})$ Reward function

$\pi(X_t)$ Determinnistic policy

$\pi(U_t \mid X_t)$ Stochastic policy

$\boldsymbol{\theta}$ Parameter vector

$\pi_{\boldsymbol{\theta}}(U_t \mid X_t)$ Parameterized policy

$J(\boldsymbol{\theta})$ Cost function

$\nabla J(\boldsymbol{\theta})$ Cost function gradient

$V^{\pi}(X_t)$ State-value function

$Q^{\pi}(X_t, U_t)$ Action-value function

γ Discount factor

Adaptive Stress Testing

E Event space

$\mathcal{S}$ Simulator

$\mathcal{E}$ Environment

$\mathcal{M}$ System under Test

$\mathcal{A}$ Reinforcement Learning Agent

Marine vessel control

$\boldsymbol{\eta}$ Marine vessel state vector

$\boldsymbol{\nu}$ Marine vessel velocity vector

$\boldsymbol{\tau}$ Vector of generalized forces and torques

$\boldsymbol{K}_p$ Proportional gain vector

$\boldsymbol{K}_d$ Derivative gain vector

$\boldsymbol{K}_i$ Integral gain vector

K_{ψ} Heading gain constant

Acronyms

AST Adaptive Stress Testing. i, iii, vii, viii, 5–7, 17–19, 25–27, 29, 31, 32, 34–37, 39–46, 49, 54, 57, 58

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. i, 54, 58

NTNU Norwegian University of Science and Technology. iii

COLAV Collision Avoidance. iii, 5–7, 25, 30–32, 34, 35, 37, 39, 46, 53–55, 57, 58

RL Reinforcement Learning. vii, 6, 12, 13, 15, 17, 18, 31, 32, 39, 40, 42, 43, 46, 50, 57

LOS Line Of Sight. vii, 21, 22, 26, 27, 32

PD Proportional–Derivative. vii, 26–28, 30, 40, 43, 45, 54, 57

PID Proportional–Integral–Derivative. vii, 26–30, 43, 45, 54, 57

SCS Safety Critical Systems. 4

ACAS X Airborne Collision Avoidance Systems. 5

TCAS Traffic Collision Avoidance Systems. 5

MCTS Monte Carlo Tree Search. 6, 31

DRL Deep Reinforcement Learning. 6, 17, 31, 37

ANN Artificial Neural Networks. 9

MDP Markov Decision Process. 13, 17

GAE Generalized Advantage Estimation. 16, 17, 31

TRPO Trust Region Policy Optimization. 16, 17, 31, 54

SUT System Under Test. 18

DOF Degrees of Freedom. 19–21

NED North East Down. 19, 21

SP-VP Single Path Velocity Planner. 22, 23, 30, 54

DAST Differential Adaptive Stress Testing. 45, 57, 58

PPO Proximal Policy Optimization. 54

Chapter 1

Introduction

1.1 Motivation

Industry 4.0, also known as the fourth industrial revolution, is a term describing the ongoing change into computerized manufacturing (Tang et al., 2019). New technologies are being developed, among them technology to enable autonomous vehicles. It is expected that autonomous vehicles will be part of the logistic function in the Industry 4.0 era (Tang et al., 2019). With the fourth industrial revolution approaching, these new technologies are providing companies with competition advantages. One of the world leading technology consultant firms, Accenture, claims that:

> "Recently, AI-controlled, autonomous robots have become economically viable. Driven by a combination of falling prices and the rising practicality cases, "smart" robots can now be deployed at scale - and harnessed for an ever-increasing number of ever more complex tasks" (Accenture, *Autonomous Robotics Systems | Accenture* (2021))

Autonomous solutions are making many industries rethink the way to perform tasks, among these the marine industry, where research is being conducted in order to develop fully autonomous ships. An example of such a project is the YARA Birkeland ship developed by Kongsberg and Yara, which is a fully electric autonomous container vessel (*Yara Birkeland | Yara International* 2021), see Figure 1.1.

The greenhouse gas emissions from international shipping is estimated to account for 2.89% of global emissions (*Fourth Greenhouse Gas Study 2020* 2021), meaning that electric ships could have a significant effect on the worlds emissions. The market for autonomous ships is expected to grow significantly, as the market size was estimated to be valued at $85.84 billion in 2020 and is expected to reach approximately $165 billion by 2030 (*Autonomous Ships Market Growth, Companies, Trends by 2030* 2021).

Figure 1.1: The YARA Birkeland electric autonomous ship. Image obtained from (*Yara Birkeland | Yara International* 2021).

Firms and governments are exploring these new technologies in order to obtain more environmentally friendly solutions. One of the approaches that is being examined is to reduce emissions via smart transportation (Tang et al., 2019). Emissions from transportation has been recognized by the UN as a sustainable development challenge towards 2030. Due to the rising urbanization of the world, the UN has also put emphasis on sustainable development in the worlds cities, by defining one out of their 17 sustainable development goals as

> "Goal 11: Make cities and human settlements inclusive, safe, resilient and sustainable" (UN, *Goal 11 | Department of Economic and Social Affairs* (2021))

In a report released in 2021 the UN further states that

> "There is an urgent need for transformative action that will accelerate the transition to sustainable transport at the global level." (UN, *Interagency Report for Second Global Sustainable Transport Conference | Department of Economic and Social Affairs* (2021))

Many large cities are situated near waterways, as this was important when the economy depended on trade conducted on the waterways. Today, these waterways are causing challenges when cities are becoming more densely populated due to the ongoing urbanization. With more people, the demand increases for more flexible mobility solutions. Thus, there is an ongoing challenge to provide sufficient mobility solutions which also reduce emissions.

Zeabuz is a norwegian startup company that attempts to address the challenge of enabling smart, emission-free transportation in urban cities by applying autonomy technology to small marine ferries (ZEABUZ, 2021). The Zeabuz ferry is supposed to offer seamless transportation of human passengers in waterways and thus avoiding the need to expand mobility solutions by e. g. building bridges. The Zeabuz technology builds upon decades of research from NTNU, especially a project called Autoferry (*Autoferry - NTNU* 2021), where the Zeabuz ferry predecessors

milliAmpere and milliAmpere 2 were developed. This thesis is written in colaboration with Zeabuz, and a simulator of milliAmpere will later be used.

(a) Zeabuz ferry design example. Image obtained from (ZEABUZ, 2021)

(b) The milliAmpere. Image obtained from (*Autoferry - NTNU* 2021).

Figure 1.2: The Zeabuz ferry design example and its predecessor milliAmpere.

Similar projects as the Zeabuz ferry have gained attention worldwide, e.g. in Amsterdam, where researchers from the Massachusetts Institute of Technology and the Amsterdam Institute for Advanced Metropolitan Solutions have developed the *Roboat*, an autonomous boat that can carry up to five people, collect waste, deliver goods and provide on-demand infrastructure in the Amsterdam canals (*One Autonomous Taxi, Please* 2021), see Figure 1.3.

Figure 1.3: The Roboat, an autonomous boat that was set sea in Amsterdam in october 2021.

These new technologies that enable autonomous vehicles can thus potentially lead to both reduced emissions and a more mobile urban design. But, these new technologies also raises new concerns. **How do we make sure these systems are safe?**

Evaluating these new and complex algorithms with respect to safety is a complex task, as the reasoning behind decisions is often hidden or hard to understand. These kinds of systems are often referred to as *black box*, as the internal processes of the system are non-intuitive to humans. A typical example are systems involving artificial neural networks, in which humans typically understand the input and the output, but have little idea about how the network has reasoned.

There is a rising concern for the implications of implementing such algorithms into decision making systems,

> "No one really knows how the most advanced algorithms do what they do. That could be a serious problem as computers become more responsible for making important decisions." (Will Knight, *The Dark Secret at the Heart of AI* (2021))

This issue raises several concerns, especially regarding cases where the algorithms are implemented in Safety Critical Systems (SCS)s, which are systems where failure can lead to loss of life or significant damage to people or property. When black-box concepts are introduced into these systems, this challenges the safety assessment as traditional methods and industry standards are not adjusted to such complex methods (Zhao et al., 2020). The Zeabuz ferry is an example of a SCS where the autonomy system incorporates several different complex systems with varying levels of black-box-factor, which can be hard to analyze for failures. As the internal workings of the system becomes hard to analyze, simulation- and operational based testing becomes more relevant. Zhao et al. (2020) states that an increased reliance on empirical demonstrations of safety and reliability via simulated and operational testing seems inevitable.

Therefore, as these new and complex methods rise in popularity, the demand for simulation based testing methods that can provide insight regarding safety is likely to rise as well. This thesis aims to contribute to the field of safety validation for new and complex technologies with respect to the marine industry, in order to provide a useful safety assessment tool that can help bring new, green marine technologies to life in a safe manner.

1.2 Related work and literature review

1.2.1 Safety verification, validation and falsification

There are several approaches to ensure safety. The most robust approach is to perform formal safety verification. This is done by conducting e. g. mathematical proofs or exhaustive swipes over all possible scenarios in order to prove or demonstrate whether a safety property holds (Lee, Mengshoel, et al., 2020). These methods provide completeness guarantees over the entire model, which of course is desirable to system producers.

Formal safety verification is suitable for systems that operate with in a comprehensible environment with respect to size and complexity. However, most cyberphysical systems operate in large and stochastic environments. In these environments, formal verification methods becomes too inefficient or even impossible due to complexity. In addition, the stochastic nature of the environments leads to the fact that failures cannot be completely excluded. Another approach to ensure system safety is to perform *safety validation*, which is a process where the system is rigorously tested and it shows *sufficient* results in regards to safety.

As mentioned in section 1.1, one proposed way to perform safety validation of complex systems is by thorough simulation-based testing. This can be done by simulating many scenarios and recording the number and details of resulting failure modes. However, this approach can be inefficient and demand a high number of simulations in order to identify failure modes, especially in systems where failure modes are rare.

A technique that address the problem of rare failure modes is the AST method, which will be applied in this thesis. This is a method that performs *falsification*, which means that the method optimizes directly towards failures (Corso, Moss, et al., 2021). By introducing optimization towards failures in the simulations, the search for failures can become more efficient and the number of simulations required is reduced (Lee, Mengshoel, et al., 2020).

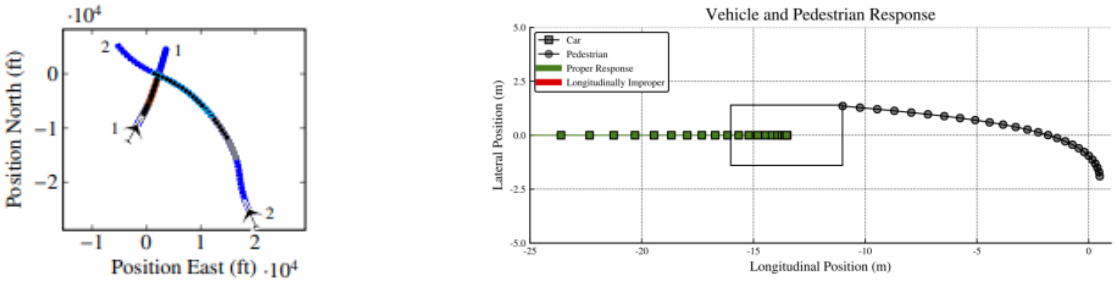
1.2.2 Adaptive Stress Testing (AST)

AST is a reinforcement learning approach to finding the most likely failure states of a system, provided a simulator of the system is available. AST was first proposed in 2015 (Lee, Kochenderfer, et al., 2015), where the method was applied to an airborne COLAV system called Airborne Collision Avoidance Systems (ACAS X), which was supposed to replace the existing airborne COLAV system called Traffic Collision Avoidance Systems (TCAS). This work contributed to the approval of ACAS X to replace TCAS, which took place in 2018 (Lee, Mengshoel, et al., 2020).

Since AST was first proposed, augmented versions of the method has been tested and proposed. A general formulation of the AST method was formulated in 2020 (Koren, Corso, et al., 2020). In the general formulation, it is evident that there are several parts of the method that can be replaced by alternative values or methods in order to adapt the method to the test domain. The AST method is thus flexible with regards to the system under test, and can be applied to many systems as long as a simulator is provided. There are mainly two systems that appear frequently in the AST literature:

- Airborne COLAV systems, with other planes as obstacles. Papers that incorporate this system include: (Lee, Kochenderfer, et al., 2015), (Lee, Mengshoel, et al., 2020), (Moss et al., 2020)
- Autonomous automobile approaching crosswalk, with pedestrians as obstacles. Papers that incorporate this system include: (Koren, Alsaif, et al., 2018), (Corso, Du, et al., 2019), (Koren and Kochenderfer, 2019), (Koren and Kochenderfer, 2020)

The two systems are illustrated in Figure 1.4



(a) Airborne COLAV systems (Lee, Mengshoel, et al., 2020)

(b) Automobile approaching crosswalk with pedestrian (Corso, Du, et al., 2019)

Figure 1.4: Examples of common systems under test in Adaptive Stress Testing papers.

Another augmentation found in the AST literature is the augmentation of the reinforcement learning solver algorithm and the reward function. These augmentations are performed in order

to better adapt the AST method to the test domain, or to obtain a more efficient algorithm. In the first proposed version of the AST method, a reinforcement learning solver called Monte Carlo Tree Search (MCTS) was used, but other versions have shown to outperform the MCTS for certain tasks. Koren, Alsaif, et al. (2018) proposed the use of Deep Reinforcement Learning (DRL) as the reinforcement learning solver and reported better performance in the automobile test case.

An identified problem of the AST method is that the failure scenarios it finds are often very similar (Koren and Kochenderfer, 2020). When the AST method finds a strategy to lead the system to failure, it often stops exploring other possible strategies. Koren and Kochenderfer (2020) proposed that the problem formulation of the AST method makes it a so called *hard-exploration* field as the system is not pointed towards failure during the simulation, only when the simulation has finished. Koren and Kochenderfer (2020) proposed a way to counteract the effects of this by implementing a method called Go-Explore, which is a method meant to improve performance in hard-exploration fields.

Another known issue of the method is that the identified failure modes often contain unavoidable failures (Corso, Du, et al., 2019). An example of this is the autonomous automobile case, where the method would find a high number failures where the majority included a scenario where the pedestrian suddenly decides to walk right into the vehicle. This is a scenario that is practically impossible to prevent for car system designers, and thus it is less relevant in order to improve the system design specifications.

Corso, Du, et al. (2019) successfully obtained more realistic results for the autonomous vehicle case, by augmenting the reinforcement learning method used. The augmentation also urged the RL agent to tend towards less likely failures. The augmentations that were implemented were based on two measures:

- The degree of which the trajectories were compliant with Responsible-Sensitive Safety rules, which is a set of driving rules motivated by common-sense driving practises. Thus, the system was incentivized to find the most likely failures trajectories that also exhibited rational driving behaviour.
- The degree of which the obtained failure trajectories are dissimilar. The system was incentivized to obtain less similar trajectories.

The technical details of the AST method is elaborated on in Section 2.4.

1.3 Research Questions and Objectives

The research questions examined in this thesis are:

RQ1: Research question 1: Can the AST Toolbox be applied to find failure modes of the milliAmpere collision avoidance (COLAV) simulator?

RQ2: Research question 2: Can the information obtained from the AST method generate valuable insights to the Zeabuz ferry design team?

1.4 Contributions

In this thesis, the last version of the AST Python toolbox is implemented to uncover failure modes in the control system of a simple marine vehicle and in the COLAV system of the milliAmpere ferry through simulations. The main contributions of the thesis are:

1: The AST method is demonstrated first using a simple marine vessel simulator. It is shown to provoke the system into failure more efficiently than uniformly distributed disturbances, and manages to identify efficient disturbance patterns in order to do so.

2: The AST method is then demonstrated in use for the milliAmpere COLAV system. The method is able to uncover failure modes in a situation which the collision avoidance system would have successfully navigated without AST intervention.

3: A heuristic is constructed to model the behaviour of the driver of the obstacle in the milliAmpere collision avoidance scenario, in order to obtain more realistic failure scenarios. Use of the heuristic shows sub-optimal results.

Chapter 2

Background

2.1 Machine learning

The approaches used to perform the safety analysis in this thesis is based on the concept of machine learning. Tom M. Mitchell defines the process of machine learning as

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. (Mitchell (1997))

That is, the computer program is said to learn if its experiences make it become better at the task it is supposed to perform. As an illustrative example, consider a program that is supposed to decide if there is a cat or a dog in a picture. Such a program is called a *classifier*. According to the definition of Mitchell, the classifier is learning if it became better at predicting if there is a cat or a dog in the picture, after some learning experience.

Mohri et al. discusses different common learning scenarios. Among these scenarios are *supervised learning* and *reinforcement learning*. Reinforcement learning is elaborated on in Section 2.3.

Supervised learning describes the scenario where the system is handed labeled examples as training data (Mohri et al., 2012). In the cat- or dog-classifier case, the training data could consist of images of cats and dogs with associated labels. After training, the system is then supposed to do predictions on unseen data. In the example cat- or dog-classifier system, the unseen data could be new images of cats and dogs that were kept separate from the training data and if the training was successful, it would be able to label the images correctly. Supervised learning is the most common form learning scenario within machine learning (LeCun et al., 2015).

2.1.1 Artificial Neural Networks (ANN)

ANN is a common framework used to perform supervised machine learning. Inspired by how the human brain works, an ANN is composed of a network of artificial neurons that together is able to form functions of desired complexity, including nonlinear functions. The artificial neuron

is also known as a *unit* or a *node* and will be referred to as a *node* in this thesis. Let $\boldsymbol{x}$ be the input vector with entries $[x_0, x_1, ..., x_{n-1}, x_n]$ and $\boldsymbol{w}$ be the weight vector belonging to the node, with entries $[w_0, w_1, ..., w_{n-1}, w_n]$. Let $\mathbf{o}(x)$ be the dot product of the vectors $\boldsymbol{x}$ and $\boldsymbol{w}$:

$$\boldsymbol{o}(\boldsymbol{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} \tag{2.1}$$

and $y = f(\boldsymbol{o}(\boldsymbol{x}))$ be the activation function of the node output. The node is illustrated in Figure 2.1.

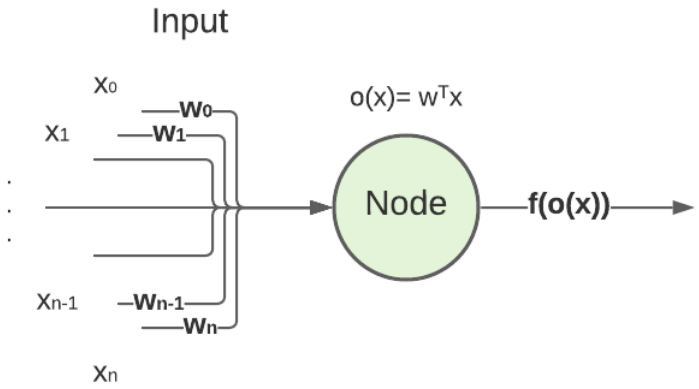


Figure 2.1: The artificial neuron, also known as a node.

Each node performs a simple computation of its input (Russell et al., 1995),

$$y = f(\boldsymbol{o}(\boldsymbol{x})) = f(\boldsymbol{w}^T \boldsymbol{x})$$
$$w_0 = -1$$

where f is known as the activation function, which is a function that makes the neuron give output on the desired format. By default, w_0 equals -1 and together with x_0 it forms the node *bias*. A typical activation function is the sigmoid:

$$f(\boldsymbol{o}(\boldsymbol{x})) = \frac{1}{1 + e^{\boldsymbol{o}(\boldsymbol{x})}}$$

which squashes the input to be between 0 and 1. The rectified linear unit, also known as the ReLU function, is another commonly used activation function.

$$f(\boldsymbol{o}(\boldsymbol{x})) = max(0, \boldsymbol{o}(\boldsymbol{x}))$$

A single node, or a network with only one layer of nodes, can only represent linearly separable functions (Russell et al., 1995). To learn and represent problems that are not linearly separable, more layers of nodes must be added. These layers are called *hidden layers*, and are elaborated on in the section on deep learning and multilayer neural networks, see Section 2.2.

2.2 Deep learning

2.2.1 Multilayer neural networks

To solve problems that are not linearly separable, nodes must be combined in a network. An example network topology is shown in Figure 2.2.

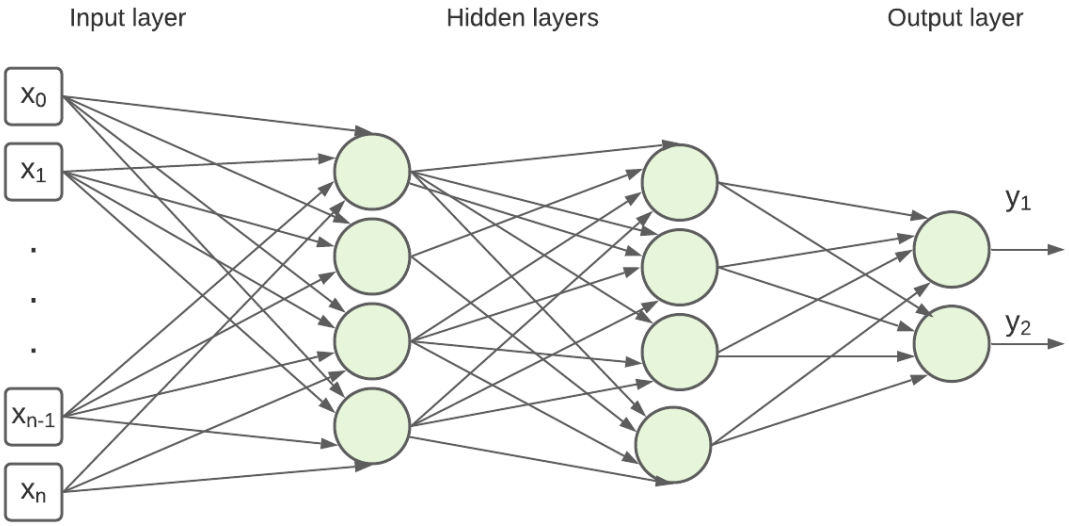


Figure 2.2: Example neural network topology.

In multilayer neural networks, there are one or more *hidden layers*, which are nodes that are not directly connected to the output.

Neural networks can be expanded with more nodes and layers to meet the complexity of the problem it is supposed to solve, but adding complexity can lead to computational inefficiency in training.

2.2.2 Training

In a training process, the node weights $\boldsymbol{w}$ are updated in several rounds called *epochs*. A multilayer neural network can be trained using the *backpropagation* algorithm. This algorithm indicates how the weights of the nodes in the network should change, according to the error between the output and the labels (LeCun et al., 2015). The error is defined by a predefined objective function. In the example cat- or dog-classifier introduced in Section 2.1, the output could be a vector describing the probability of the picture being a cat or a dog, and the error could be the difference between these probabilities and the actual binary classification, e. g.

$$\text{error} = \begin{bmatrix} \text{P(cat)} \\ \text{P(dog)} \end{bmatrix} - \begin{bmatrix} \text{Cat in picture (0 or 1)} \\ \text{Dog in picture (0 or 1)} \end{bmatrix} \tag{2.2}$$

The error then first affects the output nodes by indicating the direction and magnitude by which the output nodes missed the target. Subsequently, the error is propagated backwards through

the layers to adjust the weights of the network in order to output a probability that fit this picture a little bit better. This process is then repeated for many labeled images of cats and dogs, until the network weights are adjusted to identify a more generalized pattern indicating *cat* or *dog* and the performance metric, e. g. the accuracy when tested on unseen data, reaches a sufficient level.

2.3 Reinforcement learning

Reinforcement learning (RL) differs fundamentally from supervised learning. Instead of training on labeled training data, the RL agent learns by navigating in its environment and collect positive or negative rewards based on the action it takes.

To illustrate, lets consider two different learning processes of a child. An example of supervised learning of a child could be a parent telling the child that the name of an apple is "apple". The child then rehearses pronouncing "apple" by adjusting the pronounciation according to the feedback of the parent.

But a lot of the learning that happens during childhood is not supervised. It is based on the experiences the child makes when it navigates the world. For example, a child learns not to touch a hot plate, because the last time it did, it got burnt. Not necessarily because someone explicitly told the child to avoid it, but because the experience itself made the child aware of what this action led to - pain. RL is analogous to this kind of learning.

RL is a computational approach to learning from interacting with an environment (Sutton et al., 2018). The general structure of RL can be illustrated as in Figure 2.3.

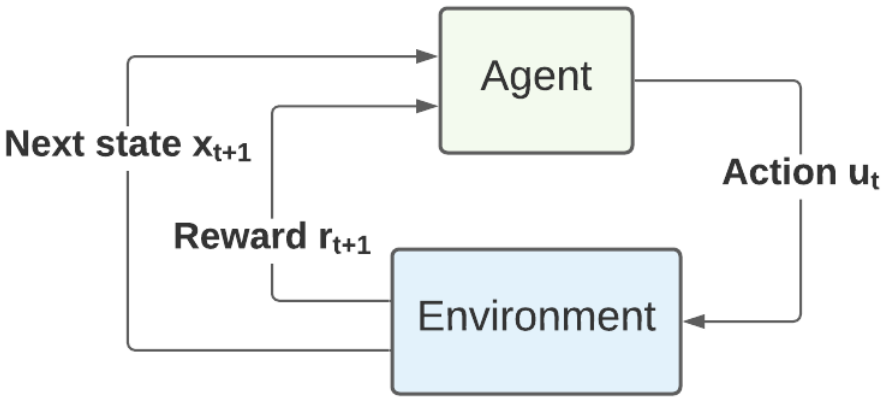


Figure 2.3: The RL structure, inspired by (Sutton et al., 2018).

In a RL problem, an agent navigates within an environment, just like the child navigates the world. The agent takes actions that potentially can impact the environment, the reward the agent gets, and the next state the agent will find itself in. In the child with the hot plate case, the action would be "put hand on plate". The next state the child will be in is with its hand on the hot plate, and the reward of this state is pain. The next time the child observes this combination of state, environment and possible action, it will probably not make the same action, as it is now conscious of the negative reward this will produce. In other words, the

child has learned from its experience. In the same manner, the RL agent gradually learns to avoid actions and states that lead to negative rewards, and instead tends toward behaviour that maximize the possible reward. The RL problem can be mathematically formulated as a *Markov Decision Process*.

2.3.1 Markov Decision Process (MDP)

A MDP is an idealized mathematical formulation of the RL problem (Sutton et al., 2018). It is formalized as a tuple (X, U, P_u, R_u) where

- X is the finite set of states the agent can be in, i. e. the *state space*.
- U is the finite set of actions the agent can take, i. e. the *action space*.
- $P_u(x, x')$=$P(X_{t+1} = x'|X_t = x, U_t = u)$ is the *transition function*, which specify the probability of a possible next state x' given a state-action pair (x, u).
- $R_u(x, x') = R(X_t = x, X_{t+1} = x', U_t = u)$ is the *reward function*, which specify the reward the agent obtains as a consequence of transitioning from state x to state x' due to the action u.

The state X_t of the agent is observed at every timestep t, and an action U_t is chosen. A *state-action history* $\sigma_t = \{X_0, U_0, X_1, U_1, ..., X_t\}$ is obtained. The mapping from the state-action history to an action at timestep t, is called a *policy* $\pi_t(\sigma_t) = U_t$. A MDP adheres to two assumptions (Russell et al., 1995):

- *Markov assumption*: the future is conditionally independent of the past, given the present. In particular, U_t is conditionally independent of the state-action history given present state X_t, i. e. $\{X_0, U_0, X_1, U_1, ..., X_{t-1}, U_{t-1}\} \perp\!\!\!\perp U_t|X_t$,
- *Stationarity assumption*: The transition function P_u and reward function R_u are the same for all t.

Under these assumptions, the policy is simplified to $\pi_t(\sigma_t) = \pi_t(X_t) = \pi(X_t)$. In cases where the policy is stochastic, it becomes a probability measure of taking action U_t in state X_t, given as $\pi(U_t \mid X_t)$ (Sutton et al., 2018).

The task of the agent is to find the *optimal policy* π^* and act thereafter. In decision theory, the optimal policy is defined as the policy that maximize the expected utility (Russell et al., 1995). The utility of a policy is a numerical measure of how useful the policy will be to the agent. In a Markov Decision Process the expected utility of a policy is expressed as a sum of the expected rewards called the *state-value function* $V^\pi(x)$:

$$V^\pi(x) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x \right]$$

where $\mathbb{E}_\pi$ denotes the expected value of the evaluated expression when following policy π. The expression consists of r_{t+k}, the reward received at timestep $t+k$ and the constant γ, which is a *discount factor* between $[0, 1]$, which discounts rewards further into the future.

The expression can be reformulated as the output of the reward function at the current step together with a term describing the probability of moving to the next states with a recursive term that states the value of said states:

$$V^\pi(x) = R_{\pi(x)}(x) + \gamma \sum_{x'} P_u(x'|x, \pi(x)) V^\pi(x')$$

The state-value function $V^\pi(x)$ is thus the sum of expected cumulative reward when starting in state x and executing π thereafter. A similar function that also accounts for the action chosen is the *action-value function*, which also can be decomposed into a recursive structure:

$$\begin{aligned} Q^\pi(x, u) &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x, U_t = u \right] \\ Q^\pi(x, u) &= R_u(x) + \gamma \sum_{x'} P_u(x'|x, u) Q^\pi(x', \pi(x')) \end{aligned}$$

The action-value function $Q(x, u)$ is the expected cumulative reward when starting in state x, choosing action u and executing π thereafter.

If the agent acts according to the optimal policy π^*, the value functions become the optimal value functions, i.e.

$$\begin{aligned} V^*(x) &= \max_\pi \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x \right] \\ Q^*(x, u) &= \max_\pi \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | X_t = x, U_t = u \right] \end{aligned}$$

Another value metric is called the *advantage function*.

$$A^\pi(x, u) = Q^\pi(x, u) - V^\pi(x) \tag{2.3}$$

The advantage function compares the state-value function and the action-value function in order to describe how much better it is to choose action u than to follow the policy π at the current timestep (Francois-Lavet et al., 2018).

2.3.2 Reinforcement learning algorithms

RL algorithms can be used to solve MDPs where

- **The rewards are unknown to the agent**. The agent must thus learn the characteristics of the MDP model by exploring it.
- **The time horizon is infinite**, with discounts on future rewards with discount factor γ. The discount factor makes the agent take decisions that have emphasis on more rewards in the near future.

RL algorithms are categorized into *model-based* and *model-free* algorithms.

Model-based algorithms include a model of the environment, with e.g. estimates of the transition function and/or the reward function (Sutton et al., 2018). These methods perform *planning* according to their model of the environment. The main drawback of these methods is that an environment model often is hard to obtain, and in these cases the agent has to learn the characteristics of the environment by exploration, which can be inefficient.

Model-free methods rely on learning either a value function, or a direct representation of the policy π and adjust the policy thereafter (Francois-Lavet et al., 2018). Thus, the agent does not attempt to describe the entire environment, but rather tries to get an idea of what the consequences of its actions within the environment will be.

Model-free methods can again be separated into *action-value* methods and *policy gradient methods*. Action-value methods are methods where the agent learn the value of actions, and selects new ones based on the learned value functions. A classic example of an action-value method is Q-learning, in which the agent learns an approximation of the action-value function $Q(x, u)$. Policy gradient methods instead learn a parameterized policy, and the agent chooses its actions directly from this without the need to approximate any value functions, although some policy gradient methods incorporate value functions in order to learn the policy. This method is applied in this thesis, and it is elaborated on in Section 2.3.3.

It is worth noting that these categorizations of methods are mostly used to get a general overview of the existing methods. It can be hard to distinguish the methods from each other as there are several RL algorithms that combine the different approaches.

2.3.3 Policy optimization

Policy gradient methods

Policy gradient methods maximizes the performance of the agent by updating a parameterized policy via gradient ascent steps. The parameter vector of the policy is denoted $\boldsymbol{\theta}$ and the policy is denoted as

$$\pi(u|x, \boldsymbol{\theta}) = P\{U_t = u|X_t = x, \boldsymbol{\theta}_t = \boldsymbol{\theta}\} = \pi_\theta(u|x)$$

which represent the probability of action u given state x and the parameter vector $\boldsymbol{\theta}$ at timestep t (Sutton et al., 2018). The parameter vector is then updated using gradient ascent, with an approximated gradient of some cost function $J(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\widehat{\nabla J(\boldsymbol{\theta})}$$

where α is the gradient ascent step size and $\widehat{\nabla J(\boldsymbol{\theta})}$ is the estimate of the stochastic cost function. The most common cost function is (Schulman, Wolski, et al., 2017):

$$J(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t\big[\log\pi_\theta(U_t|X_t)\hat{A}_t\big]$$

which consists of the approximated expectation of the log of the policy, multiplied by an approximation of the advantage function A_t, see Equation (2.3). As the problem is of stochastic nature, the expectation $\mathbb{E}$ is approximated by an average $\hat{\mathbb{E}}_t$ over a finite batch of samples.

The cost function gradient used in the gradient ascent step then becomes:

$$\widehat{\nabla J(\boldsymbol{\theta})} = \hat{\mathbb{E}}_t\big[\nabla_\theta\log\pi_\theta(U_t|X_t)\hat{A}_t\big] \tag{2.4}$$

2.3.4 Generalized Advantage Estimation Generalized Advantage Estimation (GAE)

Generalized Advantage Estimation is a method for estimating the advantage function Equation (2.3), which in turn can be used to compute the cost function gradient in Equation (2.4). The method defines an augmented advantage function called the discounted advantage function:

$$A^{\pi,\gamma}(x,u) := Q^{\pi,\gamma}(x,u) - V^{\pi,\gamma}(x) \tag{2.5}$$

where γ is a discount factor for future rewards (Schulman, Moritz, et al., 2018).

2.3.5 Trust Region Policy Optimization Trust Region Policy Optimization (TRPO)

Trust region methods, such as TRPO, apply a constraint to the maximization problem in order to contain the gradient ascent step inside some desired trust region. These methods can be used in order to step the parameterized policy.

TRPO is, as the name implies, a policy optimization method that implements a trust region constraint (Schulman, Levine, et al., 2017). It is deemed fit for large nonlinear policies such as

policies represented by neural networks. This is due to the fact thatthe policy updates become smoother than with action-value methods, as policies computed through a value function can change dramatically due to small changes in the value function.

The optimization objective in TRPO becomes:

$$\max_{\theta} \hat{\mathbb{E}}_t\big[\frac{\pi_\theta(U_t|X_t)}{\pi_{\theta_{old}}(U_t|X_t)}\hat{A}_t\big] \tag{2.6}$$

$$\text{s.t.}\hat{\mathbb{E}}_t\big[\text{KL}\big[\pi_{\theta_{old}}(\circ|X_t), \pi_\theta(\circ|X_t)\big] \leq \delta \tag{2.7}$$

where $\pi_{\theta_{old}}$ is the policy before the gradient ascent update, δ is the trust region constraint threshold and KL is the *Kullback-Leibler divergence*, which is a measure of the difference of the between the distibutions $\pi_{\theta_{old}}$ and π_θ (Schulman, Wolski, et al., 2017). The trust region constraint thus prohibits the gradient update to perform steps that will change the shape of the policy too drastically.

2.3.6 DRL

Deep Reinforcement Learning techniques has led to new applications and improvement of algorithms in problems that involve sequential decision making (Francois-Lavet et al., 2018).

DRL combines multilayer neural networks (see Section 2.2) and reinforcement learning (see Section 2.3) by using a multilayer neural network to represent the learning objective, being e.g. a model of the environment, a value function, or the policy π.

This thesis implements a DRL method by introducing a multilayer neural network to represent the policy in a policy gradient method. GAE is applied in order to estimate the cost function gradient and TRPO is used to step the policy, see Section 2.3.4 and Section 2.3.5.

2.4 Adaptive Stress Testing

AST is a reinforcement learning approach to perform safety validation. The method is implemented using a simulator of the system under test. The goal of the agent is to lead the system into the most likely failure modes.

The AST method can be implemented in many ways. Consider the case of stress testing a marine vehicle simulator. If the objective is to make the system fail by adding disturbances, then the RL agent can be in control of the disturbance value. If the objective is to discover situations where the vehicle collides with other vessels, the agent can be set up to control the motion of the other vessels. Both these cases will be presented in this thesis.

In the AST method, the problem of bringing the system to failure is modeled as a discrete time, continuous state MDP and then optimized by RL. The system consists of

- The simulator $\mathcal{S}$

- The environment $\mathcal{E}$
- The System Under Test (SUT) $\mathcal{M}$
- The RL agent $\mathcal{A}$

and is structured in a somewhat unintuitive way, see Figure 2.4.

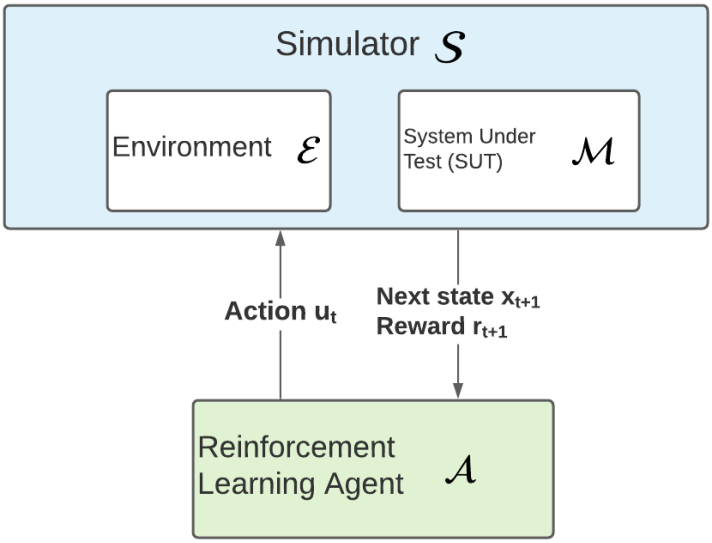


Figure 2.4: The AST structure, inspired by (Lee, Mengshoel, et al., 2020).

In AST, the environment, the system under test and the simulator is merged into one proxy environment, which the RL agent interacts with. This is due to the fact that failure modes often includes parts of the environment, and that controlling parts of the environment, such as other vessels or obstacles, can be beneficial in order to provoke the system into failure. This differs from traditional RL setups where the agent usually represents the system under test.

As mentioned, AST attempts to find the *most likely failure path*, which is the history of actions with the highest likelihood subject to the constraint that the final state is a failure event, i. e. $x_{t_{end}} \in E$, where E is the *event space* $E \subset X$ and t_{end} is the final timestep of the simulation (Lee, Mengshoel, et al., 2020:p7). Formulated mathematically, AST optimizes Equation (2.9).

$$\max_{u_0,\ldots,u_{end}} \prod_{t=0}^{t_{end}-1} p(u_t|x_t) \tag{2.8}$$

$$\text{subject to } x_{t_{end}} \in E \tag{2.9}$$

2.4.1 Formulation

The AST problem is formulated as an MDP as follows (Lee, Mengshoel, et al., 2020).

- The state x of the MDP is the state of the simulator.
- The agent observes the state x and chooses action u.
- The transition to the next state is given by the transition behaviour of the simulator which is the combined behaviour of $\mathcal{M}$ and $\mathcal{E}$.
- The reward functions is set to optimize Equation (2.9).

2.4.2 The reward function

The reward function used in AST is designed to find the most likely failure states of the system. It is formulated in different manners in different AST papers. Koren, Corso, et al. (2020) presented a general formulation as:

$$R = \begin{cases} 0 & \text{if } x \in E \\ -\alpha - \beta f(x), & \text{if } x \notin E, t \geq t_{end} \\ -g(u) - \eta h(x), & \text{if } x \notin E, t < t_{end} \end{cases} \tag{2.10}$$

where α is a large number in order to penalize trajectories that do not lead to failure, $\beta f(x)$ is an optional, possibly domain specific, heuristic. $g(u)$ is the action reward, proposed to be proportional to $logP(u)$, i. e. the log likelihood of the action which will make the agent tend towards the most likely failures and $\eta h(x)$ is an optional training heuristic given at each timestep.

2.5 Marine vessel dynamics and control

2.5.1 Dynamics of marine vessel

The simulators used in this thesis portrays simplified 3-Degrees of Freedom (DOF) horizontal plane models of marine vessels, which will be elaborated on in this chapter.

Let an inertial frame be approximated by the earth-fixed reference frame {e} called North East Down (NED). A marine vessel is fully described by a 6-DOF model with the state-space representation

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

where (x, y, z) represents *surge, sway* and *heave*, describing the body position in 3D space (Fossen, 2011). (ϕ, θ, ψ) are the orientation angles *roll, pitch* and *yaw*, see Figure 2.5 for an illustration.

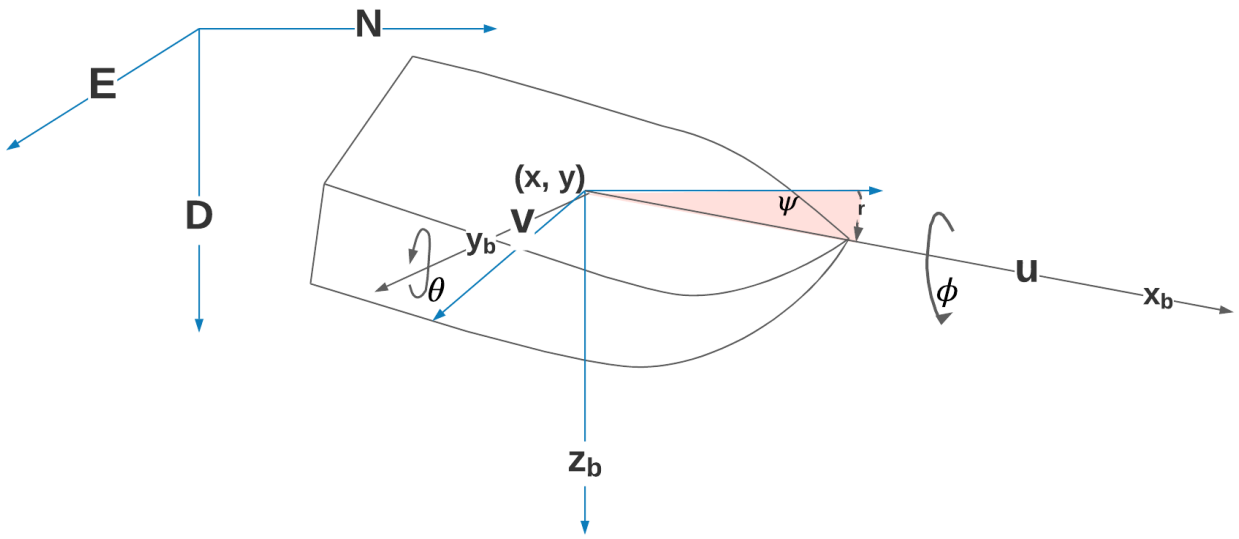


Figure 2.5: Vessel with state parameters, inspired by (Fossen, 2011).

in Figure 2.5, x_b, y_b and z_b makes up the body reference frame of the vessel. As a simplification, one often assumes that the movement in the z-direction is arbitrarily small. The *heave* parameter z is then removed from η together with the vertical angles (ϕ, θ), yielding the 3-DOF state vector

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$$

The velocity vector for the 3-DOF model is given as

$$\boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

where (u, v, r) is called *surge speed*, *sway speed* and *yaw rate*, see Figure 2.5.

As can be deducted by inspection of Figure 2.5, the following relation between η and ν can be obtained:

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u\cos(\psi) - v\sin(\psi) \\ u\sin(\psi) + v\cos(\psi) \\ r \end{bmatrix} = \boldsymbol{R}_{z,\psi}\nu$$

where

$$\boldsymbol{R}_{z,\psi} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When the assumption of no heave motion is combined with the assumption that there is no external disturbances to the vessel such as wind, ocean currents and waves, the 3-DOF vessel dynamics can be formulated as:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\nu \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} \ &+ \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \end{aligned} \tag{2.11}$$

Here, $\boldsymbol{\tau}$ is the vector of forces and torque acting on the vessel, which can include e. g. disturbances and control actuation. $\boldsymbol{D}(\boldsymbol{\nu})$ is the hydrodynamic damping matrix, $\boldsymbol{M}$ is the mass matrix and $\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis and centripetal matrix. The matrices $\boldsymbol{M}$ and $\boldsymbol{C}(\boldsymbol{\nu})$ represent the combination of the dynamics of the rigid body and the dynamics due to the hydrodynamic effect of added mass, denoted RB and A respectively:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{M}_{RB} + \boldsymbol{M}_{A} \\ \boldsymbol{C}(\boldsymbol{\nu}) &= \boldsymbol{C}(\boldsymbol{\nu})_{RB} + \boldsymbol{C}(\boldsymbol{\nu})_{A} \end{aligned}$$

2.5.2 Path following

Guidance is a method to obtain state references in order to make a vessel follow a certain trajectory. One form of guidance system is *path following* for straight line paths. This can be implemented as a simple LOS look-ahead path controller. This controller gives actuation to minimize the shortest distance to the path and the look ahead angle between the LOS point on the path and the vehicle, as seen in Figure 2.6. In addition, it can be set to minimize the error in surge velocity according to some desired velocity u_d.

A new desired value for the heading is found for every time step, given by

$$\begin{aligned} \psi_d &= \alpha_k + \chi_r \\ &= \arctan2(y_{\boldsymbol{p}_{k+1}} - y_{\boldsymbol{p}_k}, x_{\boldsymbol{p}_{k+1}} - x_{\boldsymbol{p}_k}) + \arctan2(-e, \Delta) \end{aligned}$$

Where α_k is the path relative angle to the NED frame, χ_r is the velocity-path relative angle given by the smallest distance between the vessel and the path e, and Δ, the lookahead distance (the LOS vector projection onto the path). LOS systems are implemented in both of the simulators used in this thesis.

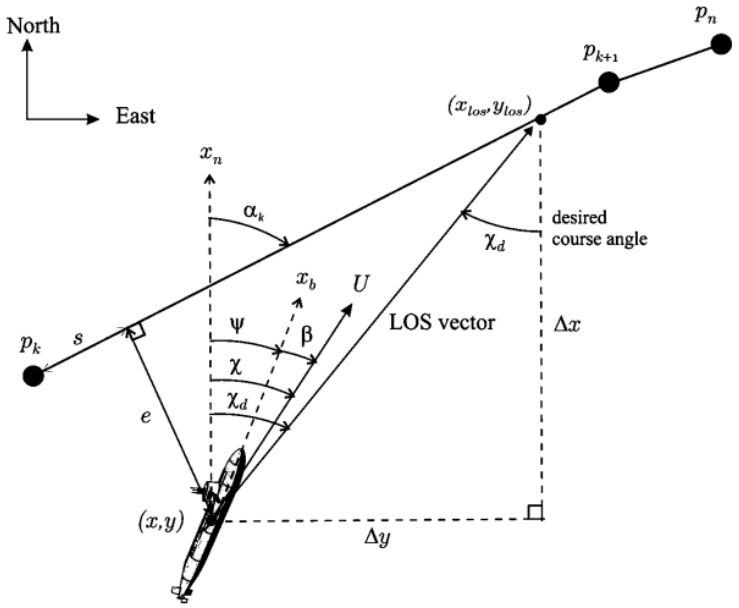


Figure 2.6: LOS following of a simple line. From (Fossen, 2011)

2.5.3 Collision avoidance and the Single Path Velocity Planner (SP-VP)

Autonomous marine vehicles need collision avoidance systems when navigating amidst obstacles such as other vehicles or land. In this thesis, a simulator of the milliAmpere ferry collision is used, in which the the vessel is simulated in traffic with other vehicles and the collision avoidance system is tested.

The collision avoidance system implemented on the milliAmpere ferry is based on the SP-VP approach, which is a path-velocity decomposition method proposed in (Thyri et al., 2020).

The method projects all obstacles into a path-time space and builds a graph of different velocity profiles along a path to a predefined waypoint. An optimal velocity profile along the path is then selected based on the cost attached to every node in the graph. An example is shown in Figure 2.7.

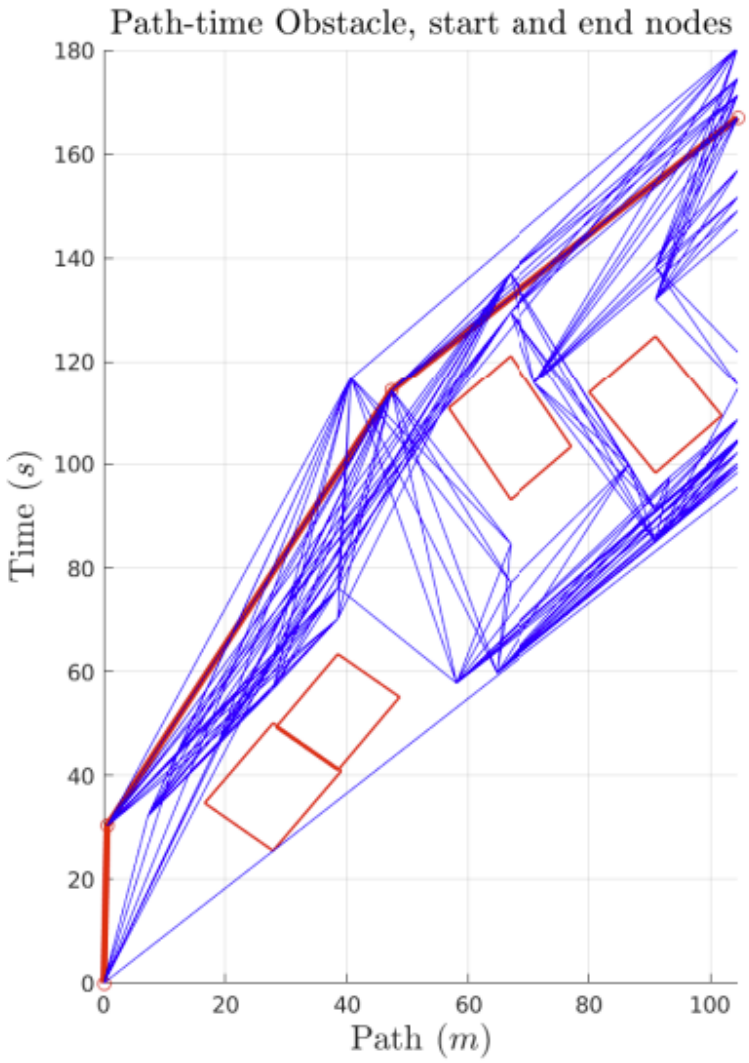


Figure 2.7: Possible velocity profiles along a predefined path in the path-time space. The squares are obstacles that the system is supposed to avoid. The chosen, and thus optimal velocity profile is highlighted in red. Figure obtained from (Thyri et al., 2020).

The SP-VP method is a robust and simple approach to collision avoidance as it produces predictable and intuitive trajectories. As a trade off of the simplicity of the SP-VP approach, the method is not able to handle situations where e. g. an obstacle is driving head on towards the vessel as it is unable to find a path that does not lead to collision. In cases like these, the algorithm breaks down due to the infeasible nature of the problem.

Chapter 3

Methodology

Two different simulators are used in this thesis. First, a simple simulator hereby referred to as the SimpleSim, which simulates a simplified model of the milliAmpere ferry. Then a more complex simulator obtained from Zeabuz which simulates the milliAmpere ferry COLAV system is used, which is hereby referred to as the milliAmpere COLAV simulator.

To apply AST to the objectives in this thesis, an open source Python package called the AST toolbox was used (Koren, Ma, et al., n.d.). The SimpleSim was constructed in order to do a proof of concept of the method. The AST Toolbox is quite comprehensive, which called for a simplified approach first, in order to get familiarized with the concept and the toolbox itself.

The SimpleSim was tested by injecting disturbances into the system dynamics to cause the vessel to drift off the path it originally was set to follow. The milliAmpere COLAV simulator was tested by adding actuation in the control input of an obstacle vehicle in order to search for scenarios where the obstacle and the milliAmpere vessels crashed.

In this chapter, the two simulators will be presented. Then, the AST implementation will be elaborated on, as well as the experiment setup.

3.1 Constructing the SimpleSim

In order to gain knowledge of the AST toolbox and how it could be applied to a simulator, a simple milliAmpere simulator was constructed.

The simulated vessel was controlled by a control input $\boldsymbol{\tau_c}$ and disturbed by a vector $\boldsymbol{w}$. The vessel dynamics represented in the simulator can then be described by an augmented version of Equation (2.11):

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_{z,\psi}\nu \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} \ &+ \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau_c} + \boldsymbol{w} \end{aligned} \tag{3.1}$$

The kinetic parameters of the vessel, including $\boldsymbol{M}$, $\boldsymbol{C}$ and $\boldsymbol{D}$, was obtained from Zeabuz and

corresponds to the kinetics of the milliAmpere ferry.

The simple simulator vessel was instructed to follow a straight line $y = 0$ by applying a LOS guidance controller starting with the initial position and velocity

$$\boldsymbol{\eta}_0 = \begin{bmatrix} 0 \\ 1 \\ \frac{\pi}{4} \end{bmatrix}, \boldsymbol{\nu}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

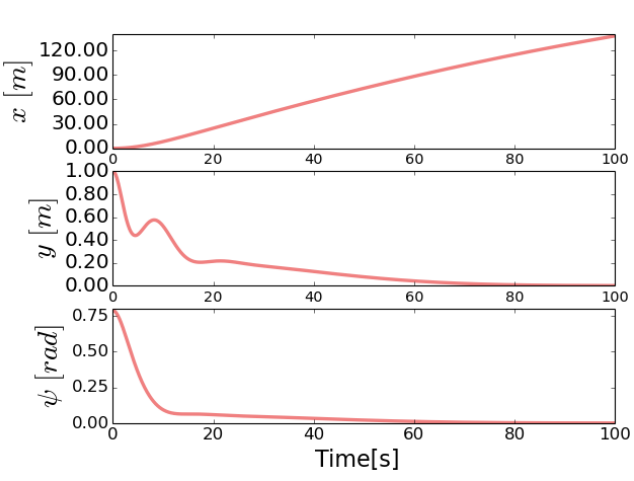
The system was also set up with two different controller schemes, namely PD and PID controllers, in order to investigate if controller type would have an effect on the AST results.

3.1.1 PD regulation and LOS guidance

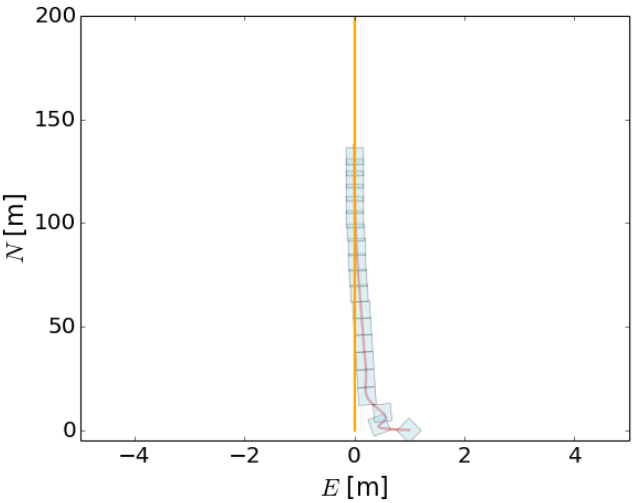
PD regulation and LOS guidance was implemented, which resulted in the following control input:

$$\begin{aligned} \boldsymbol{\tau_c} &= \boldsymbol{R}_{z,\psi}^{\top} \boldsymbol{\tau_{reg}} \\ \boldsymbol{\tau_{reg}} &= -\mathbf{K}_p(\boldsymbol{\eta} - \boldsymbol{\eta}_d) \; - \mathbf{K}_d \boldsymbol{R}_{z,\psi}(\boldsymbol{\nu} - \boldsymbol{\nu}_d) \\ \boldsymbol{\nu}_d &= \begin{bmatrix} u - u_d \\ 0 \\ \mathrm{K}_\psi(\psi - \psi_d) \end{bmatrix} \end{aligned} \tag{3.2}$$

where $\mathbf{K}_p$ and $\mathbf{K}_d$ are tunable vector gains and K_ψ is the yaw gain. $\boldsymbol{\eta}_d$ was set to $[100, 0, 0]$ and increased to $[200, 0, 0]$ if the vessel reached $x = 80$. The guidance and regulator parameters were tuned until a satisfactory result was obtained. The vessel converges to the path and follows it throughout the simulation. The result is shown in Figure 3.1.



(a) Position, heading and reference values with PD regulation and LOS guidance.



(b) Trajectory of vessel with PD regulation and LOS guidance.

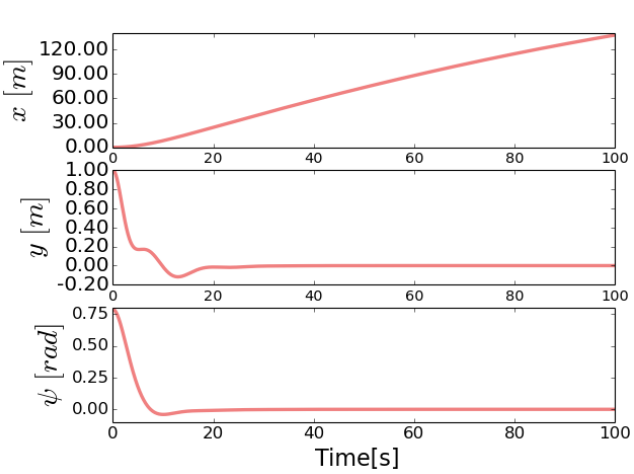
Figure 3.1: PD regulated vessel.

3.1.2 PID regulation and LOS guidance

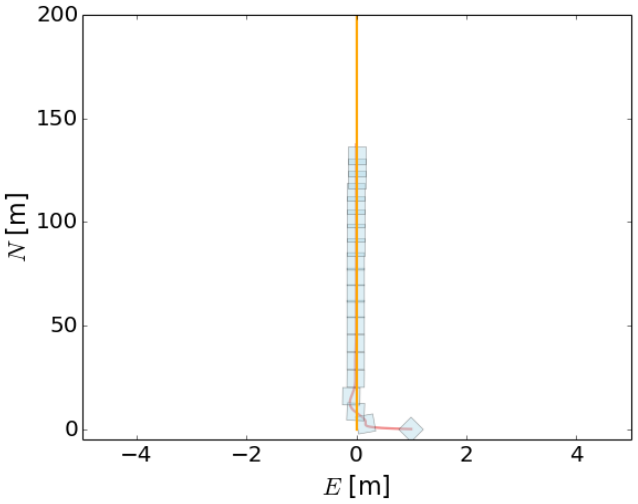
Integral action was introduced in addition to the previous LOS guidance control in order to obtain more robust control. This resulted in the following augmentation of the control input Equation (3.2):

$$\boldsymbol{\tau_{reg}} = -\mathbf{K}_p(\boldsymbol{\eta} - \boldsymbol{\eta}_d) \; - \mathbf{K}_d \boldsymbol{R}_{z,\psi}(\boldsymbol{\nu} - \boldsymbol{\nu}_d) \; - \mathbf{K}_i \int (\boldsymbol{\eta} - \boldsymbol{\eta}_d)$$

Where $\mathbf{K}_i$ is an additional tunable regulator constant. The guidance and regulator parameters were tuned until a satisfactory result was obtained. The result is shown in Figure 3.2.



(a) Position, heading and reference values with PID regulation and LOS guidance.



(b) Trajectory of vessel with PID regulation and LOS guidance.

Figure 3.2: PID regulated vessel.

3.1.3 Adding disturbances

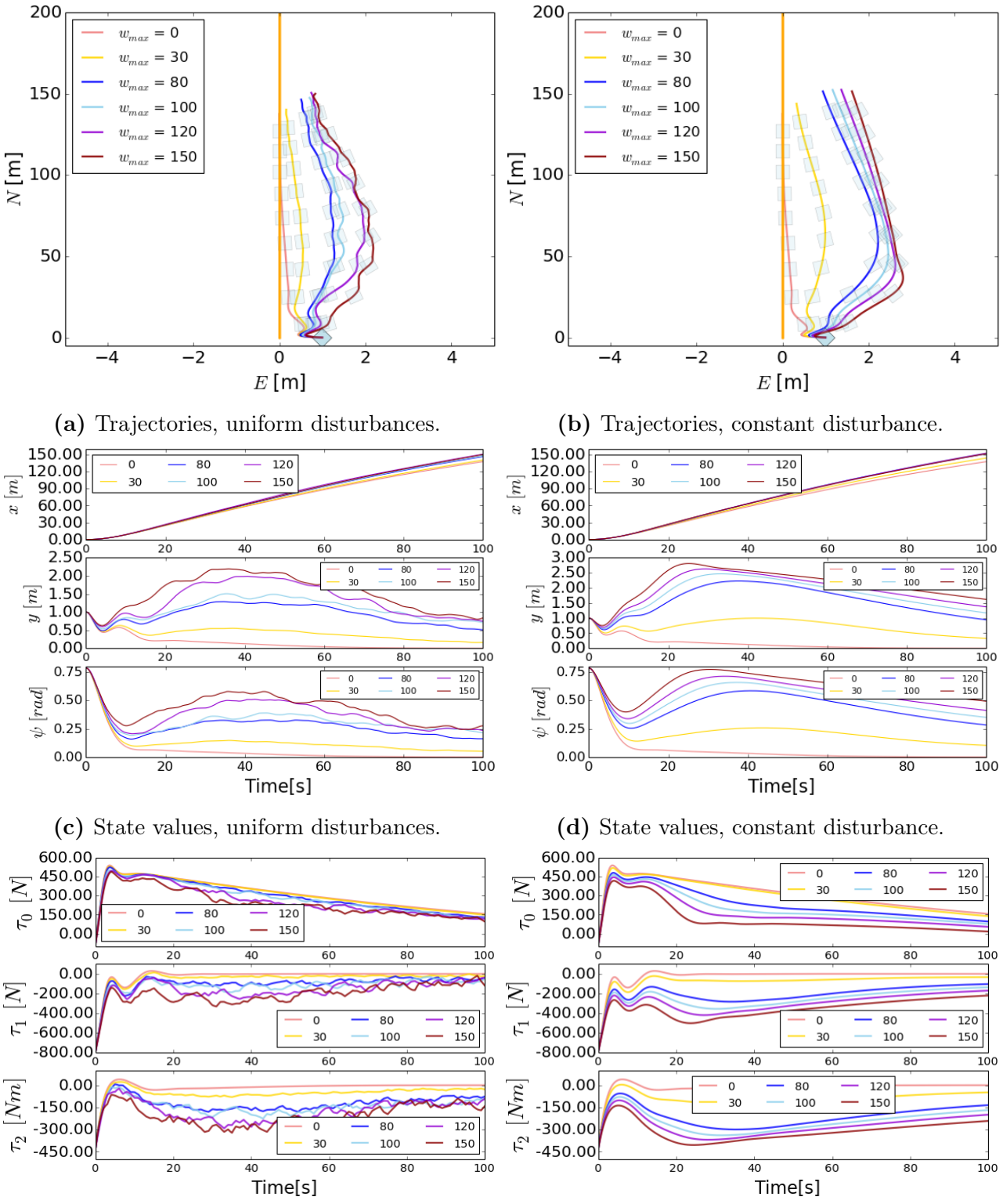
The AST method was supposed to disturb the vessel so that it was unable to follow the path. In order to investigate how much disturbance the SimpleSim control system was able to counteract, thus creating a baseline for the following experiments, both uniformly distributed disturbance values and a constant disturbance was introduced into the system.

The uniformly distributed disturbances was drawn from a uniform distribution over $[0, w_{max}]$. The value of w_{max} was in increased to test the systems ability to counteract disturbances.

From the resulting trajectories and reference plots, it can be observed that the system is able to counteract disturbances quite effectively with regards to staying on or near the path for quite high values of w_{max}. The offset from the path increases for higher values of w_{max}, but in all cases the offset is reduced towards the end of the simulation.

The PID regulated system responded better to disturbance than the PD regulated system. This is expected, as the integral effect of PID control is known to reduce offsets from the reference

values. The offset from the path, introduced by the disturbances, is significantly higher for the PD regulated system in both the uniform disturbance case and the constant disturbance case. The resulting trajectories, corresponding state values and the regulator control input for both the PD and the PID regulated vessel are presented in Figure 3.3 and Figure 3.4.



(a) Trajectories, uniform disturbances.

(b) Trajectories, constant disturbance.

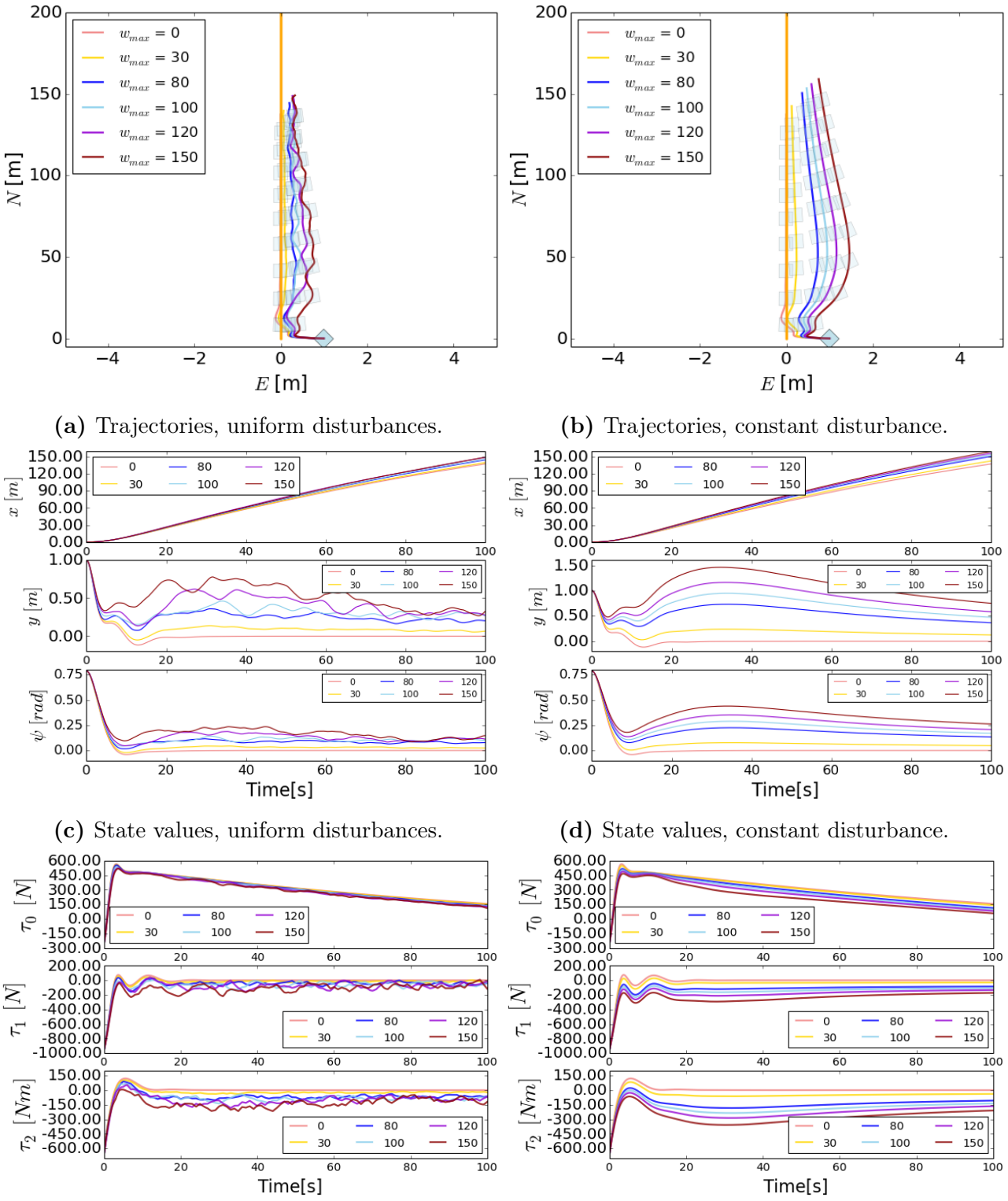
(c) State values, uniform disturbances.

(d) State values, constant disturbance.

(e) Regulator control inputs, uniform disturbances.

(f) Control input, constant disturbance.

Figure 3.3: Resulting trajectories for PD regulated vessel with increasing values of w_{max}.



(a) Trajectories, uniform disturbances.

(b) Trajectories, constant disturbance.

(c) State values, uniform disturbances.

(d) State values, constant disturbance.

(e) Regulator control inputs, uniform disturbances.

(f) Control input, constant disturbance.

Figure 3.4: Resulting trajectories for PID regulated vessel with increasing values of w_{max}.

3.1.4 SimpleSim baseline

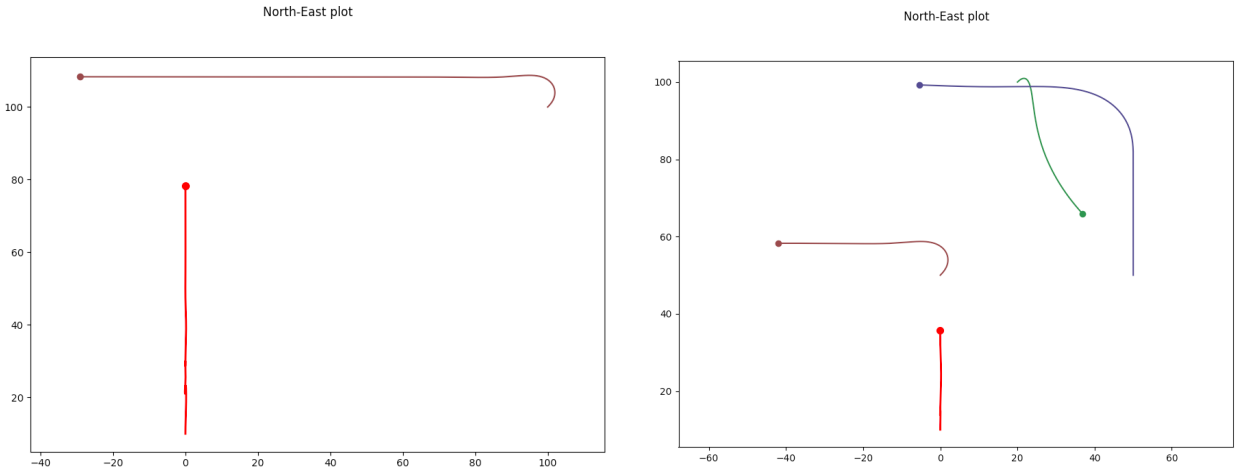
The results from the disturbance tests was used as a baseline in order to compare the performance of the AST method. From the tests, some remarks can be made:

- The maximum path offset, which can be seen as the y-value difference from 0 in the reference plots, of the PD regulated is ∼2.8m, obtained in the constant disturbance scenario.

- For the PID regulated vessel, the maximum path offset ∼1.4, also obtained in the constant disturbance scenario.

3.2 The milliAmpere COLAV simulator

The milliAmpere simulator is a simulator obtained from Zeabuz, which simulates a milliAmpere ferry together with one or more obstacles, see Figure 3.5. The obstacle can be implemented using different vehicle models, controller and guidance schemes. The simulator is centered around the collision avoidance (COLAV) algorithm, to prove the concept and to expose its vulnerabilities. The COLAV algorithm of the milliAmpere vehicle is based on the SP-VP algotihm, see section 2.5.3.



(a) One obstacle scenario.

(b) Three obstacle scenario.

Figure 3.5: Examples from visualization of the Zeabuz simulator.

It is worth noting that the simulator does not simulate the entire milliAmpere ferry system, only the COLAV algorithm. The SP-VP algorithm is designed to fail fast, and in the full system, the state machine of the ferry will take over when it fails and handle the situation by e.g. halting the speed and eventually stop. This happens when the SP-VP algorithm deems the situation infeasible and throws an error. In the milliAmpere COLAV simulator, this error is supposed to terminate the program. In order to avoid termination of the simulation and approximate the behaviour of the full system, functionality was added to the simulator in order to make the system stop when the problem became infeasible.

3.3 Adaptive stress testing implementation

3.3.1 AST specification

In this thesis, the version of the AST method specified in (Koren, Alsaif, et al., 2018) is implemented. As mentioned in Section 1.2, this paper proposed the use of DRL as the reinforcement learning solver and reported better solutions using DRL than with the use of MCTS for use in the autonomous automobile scenario.

The DRL version of the AST method uses a neural network to represent the parameterized policy π_θ. The method runs batches of simulation episodes which is then used to train the policy network in epochs. The length of the episodes, the batch size and the number of epochs are predetermined. The GAE method is used to estimate the policy-gradient from the batches, then TRPO is used to step the policy.

In addition to the replacement of the RL solver, this paper suggested an augmentation of the reward function to include a domain specific heuristic, namely the distance from the vehicle to the pedestrians, see Equation (3.3),

$$R = \begin{cases} 0 & \text{if } x \in E \\ -100000 - 10000\text{dist}(p_v, p_p), & \text{if } x \notin E, t \geq t_{end} \\ -\log(1 + M(u, \mu_u|x)), & \text{if } x \notin E, t < t_{end} \end{cases} \tag{3.3}$$

where p_v is the position of the vehicle and p_p is the position of the pedestrian, and $M(u, \mu_u|x)$ is the Mahalanobis distance between the action u and the expected action μ_u given the state x (Mahalanobis, 1936). In both the SimpleSim and the milliAmpere COLAV simulator case, a domain specific heuristic was implemented.

A tutorial for the AST Toolbox was followed when constructing the AST wrapper for the simulators (*Contents — AdaptiveStressTestingToolbox 2020.09.01.0 Documentation* 2021). The AST method works as a wrapper around the simulator that it tests. In this way, the AST simulator can do multiple batches of simulation to train the RL agent in order to optimize its actions towards failure mode. In this thesis, the AST method was implemented in *closed loop* mode, meaning that the method would be able to step the simulator, evaluate the observation returned and then provide a new action. This required access for the AST simulator to the simulator functions

- `reset(`x_0`)`: reset the simulator to the fixed initial position x_0
- `step_simulation(u)`: step the simulation with action u
- `is_terminal()`: return true if the simulation has finished

3.3.2 Wrapping the SimpleSim

The AST method was implemented to control the disturbance $\boldsymbol{w}$ in the SimpleSim dynamics, see Equation (3.1). The AST method was run with increasing values of w_{max}, meaning that the

agent was able to draw actions from the continuous action space $U = [0, w_{max}]$.

The reward function

The reward function was adjusted to the SimpleSim domain, resulting in

$$R = \begin{cases} 0 & \text{if } x \in E \\ -100000 - 10000\frac{1}{\text{dist}(path,p)}, & \text{if } x \notin E, t \geq t_{end} \\ -\log(1 + M(u, \mu_u|x)), & \text{if } x \notin E, t < t_{end} \end{cases} \tag{3.4}$$

where $\text{dist}(path, p)$ is the shortest distance between the path and the vessel. This domain specific heuristic was implemented in order to penalize the RL agent additionally if the distance to the path was small at the end of trajectories that did not lead to failure.

3.3.3 Wrapping the milliAmpere COLAV Simulator

As mentioned in Section 3.2, the milliAmpere COLAV Simulator provided the possibility of simulating several obstacles of different models and controllers. In this thesis, the scenario with only one obstacle is chosen, as this was considered as more illustrative of the AST concept. The obstacle was implemented as a simple first order model with a LOS guidance control system that instructed it to follow a straight line. The AST method was set to give control input to the obstacle in order to drive it towards failure.

The reward function

In resemblance to Equation (3.3), a domain heuristic based on the distance to the obstacle was implemented in the reward function:

$$R = \begin{cases} 0 & \text{if } x \in E \\ -100000 - 10000\text{dist}(p_v, p_{obs}), & \text{if } x \notin E, t \geq t_{end} \\ -\log(1 + M(u, \mu_u|x)), & \text{if} x \notin E, t < t_{end} \end{cases}$$

where $\text{dist}(p_v, p_{obs})$ is the distance between the ferry and the obstacle. This domain specific heuristic was implemented in order to penalize the RL agent additionally if the distance from the ferry to the obstacle was small. The simulator was also tested using an additional training heuristic $h(x)$:

$$R = \begin{cases} 0 & \text{if } x \in E \\ -100000 - 10000\text{dist}(p_v, p_{obs}), & \text{if } x \notin E, t \geq t_{end} \\ -\log(1 + M(u, \mu_u|x)) - h(x), & \text{if} x \notin E, t < t_{end} \end{cases}$$

The heuristic was constructed in order to obtain a better model of the obstacle driver probability, i. e. it was meant to penalize actions that was less likely to be executed by a human driver. The heuristic was constructed based on the following assumptions:

- It is likely that the obstacle will take actions that reduce the risk of collision with the milliAmpere ferry.
- If the distance between the vessels is less than a tolerable limit, and the milliAmpere ferry is within the field of view of the obstacle, it is likely that the obstacle will take actions that takes it further away, and/or heads in another direction.
- A simplified measure can be used to assess the risk of collision based on the distance between the vessels and the difference in angle between the obstacle and the ferry, and the heading of the obstacle.

A heuristic was developed, based on the field of view of the obstacle and the distance between the vessels. Let

- L be the distance threshold
- FOV be the maximum field of view angle
- $D = \text{dist}(p_v, p_{obs})$ be the distance between the vessels
- ϕ be the angle following the distance vector between the vessels
- β be the absolute value of difference between the obstacle heading ψ_{obs} and ϕ, illustrated in Figure 3.6.

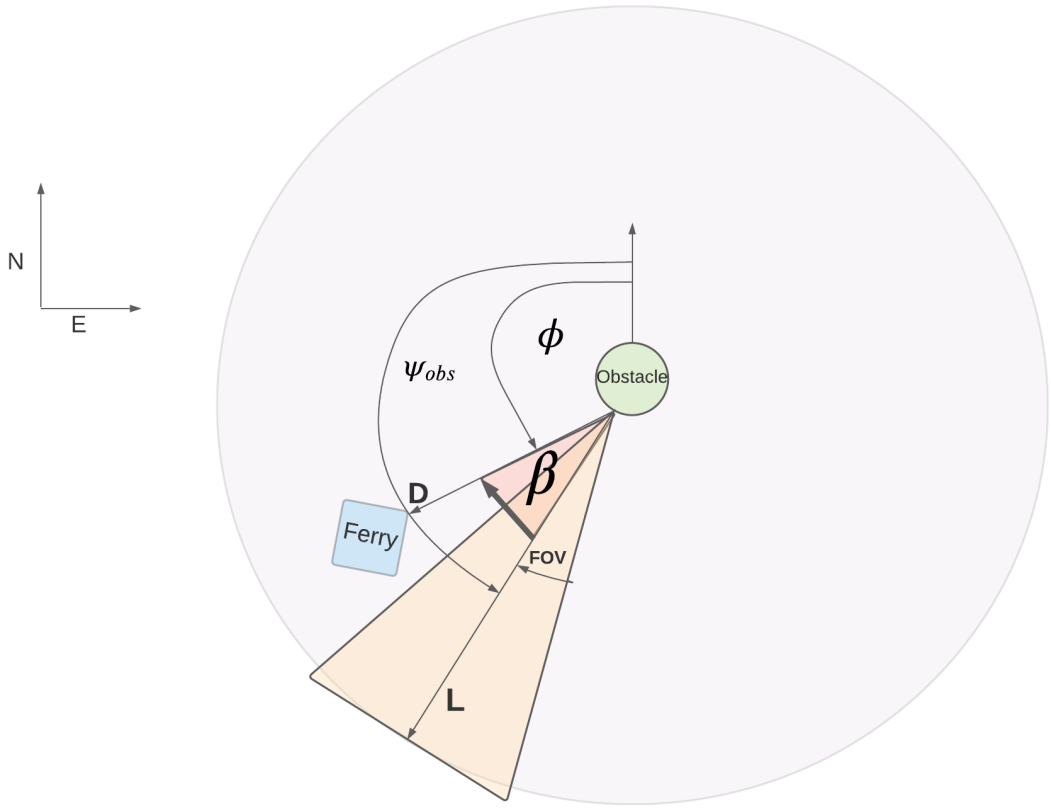


Figure 3.6: Illustration of parameters used in driver model heuristic.

Let r denote the risk measure. The risk measure was set to be

$$r = \begin{cases} 0 & \text{if D} > \text{L} \\ 0.5\frac{(L-D)}{L}, & \text{if D} < \text{L}, \beta > FOV \\ 0.5\frac{(L-D)}{L} + 0.5\frac{(FOV-\beta)}{FOV}, & \text{if D} < \text{L}, \beta < FOV \end{cases} \tag{3.5}$$

Thus, the risk increases if the obstacle is close to the ferry, and it increases further if the obstacle is headed towards the ferry. If the obstacle is very close to the ferry and it is also headed towards it, the risk measure approaches 1.

To obtain the heuristic value, the risk had to be evaluated before and after the action. Let r_t be the risk measure before the action, r_{t+1} the risk measure after the action, and $\delta_r = r_{t+1} - r_t$. The resulting heuristic became

$$h = \begin{cases} 0, & \text{if } \delta_r < 0 \\ log(1 + r_{t+1}), & \text{if } \delta_r > 0 \end{cases} \tag{3.6}$$

Thus, the system receives negative reward if the action does not reduce the risk, and the penalty increases with greater risk. This is a simplified approximation of a probability model of the obstacle driver.

3.4 AST Experiment Setup

The AST method contain several hyperparameters that need to be specified in order to run the experiments.

When testing both the SimpleSim and the milliAmpere COLAV simulator, the mean action used in the Mahalanobis distance in the reward function (see Section 3.3.1) was set to zero. The zero value makes the reward function penalize actions that are far from zero, which gives a probability estimate that favours smaller actions. This makes sense as an approximation in both simulator cases, as big disturbances or driver actions are then considered less likely.

3.4.1 Experiment with the SimpleSim

The SimpleSim was tested with increasing values for w_{max}, in order to compare the results to the disturbance test scenarios from Section 3.1. The goal of the AST system was set to bringing the vessel to a desired offset from the path in East direction. While testing, the desired offset was altered in order to investigate the AST performance for different values. The experiments was set up with the same initial values as was used in the testing of the system in Section 3.1, namely:

Parameter	Value
$\boldsymbol{\eta}_0$	$\begin{bmatrix} 0 \\ 1 \\ \frac{\pi}{4} \end{bmatrix}$
$\boldsymbol{\nu}_0$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

The simulation hyperparameters were set as follows:

Parameter	Value
Batch size	10 000
Epochs	30
t_0	0 s
t_{end}	100 s
dt	0.1 s
AST step size	1 s
Max number of AST steps	100
Desired offset from path	1.3m, 1.5m, 2m, 2.5m, 3m

3.4.2 Experiment with the milliAmpere COLAV Simulator

The experiment was set up with the following initial values:

Parameter	Value
$\boldsymbol{\eta}_0$	$\begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}$
$\boldsymbol{\nu}_0$	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
$\boldsymbol{\eta}_{obs_0}$	$\begin{bmatrix} 100 \\ 100 \\ 0 \end{bmatrix}$
$\boldsymbol{\nu}_{obs_0}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

When running the COLAV simulator with these parameters, the ferry is able to stop in order to let the obstacle pass and later continue, see Figure 3.7.

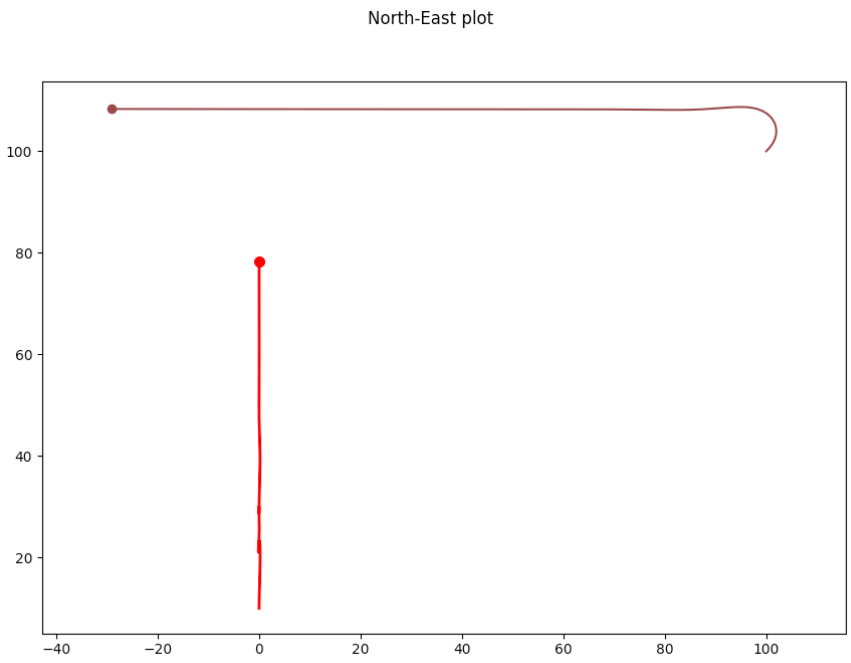


Figure 3.7: One obstacle scenario used in the experiment. The milliAmpere ferry stops at $x \sim 25$ and awaits for the obstacle to pass, before it continues.

The simulation hyperparameters were set as follows:

Parameter	Value
Batch size	20 000
Epochs	50
t_0	0 s
t_{end}	400 s
dt	0.1 s
AST step size	4
Max number of AST steps	100
Crash distance	5 m
w_{min}	-0.1
w_{max}	0.1
Training heuristic parameters	Value
L	50 m
FOV	20 °

The goal of the AST system was to bring the obstacle into a collision with the ferry. The crash distance, namely what distance between the vessels that would evaluate to a collision, was set to 5m. The magnitude of the obstacle actuation, w_{min} and w_{max}, was limited to the action space [-0.1, 0.1] as higher values gave unrealistic movements of the obstacle. The obstacle in the simulation is, as mentioned in Section 3.3, a simple model based on first order processes, and was highly affected by actuation.

3.4.3 AST steps, simulator timesteps and batch size

In the SimpleSim case, the step size of the AST simulator is set to 1s, meaning that one AST step corresponds to 1s in the vessel simulators. The step size of the SimpleSim, namely dt, is set to 0.1. This indicates that for every step of the AST simulator, the vessel simulators does 10 timesteps of size 0.1. The AST simulator thus evaluates the state for every second and then provides a new action based on the evaluation. In the milliAmpere COLAV simulator case, the step size of the AST simulator is set to 4s, meaning that for every step of the AST simulator, the vessel simulators does 40 timesteps of size 0.1.

The evaluation was chosen not to be performed at every timestep due to the fact that this would cause the DRL network to become unnecessary large, which possibly could introduce unwanted noise in the network. In the milliAmpere COLAV simulator case, evaluating the state and providing a new action at an interval of 4s was also deemed similar to human behaviour.

The maximum number of AST timesteps is set to 100, which corresponds to a vessel simulation of 100s for the SimpleSim and and 400s for the milliAmpere COLAV simulator. The timespan is reduced when the goal is reached before the maximum number of steps is performed. The batch size specify how many AST steps the AST simulator runs in one epoch. To illustrate: in the milliAmpere COLAV simulator, the batch size is set to 20 000, which means one epoch results in 200 trajectories if every trajectory in an epoch uses the full 100 AST timesteps. If the goal is reached before the maximum number of AST steps are performed, there is room for more trajectories in an epoch, and the number of trajectories per epoch increases.

Chapter 4

Results and discussion

The results from the tests are presented and discussed. In both simulator cases, the AST method was able to identify failure modes. Some of the failure scenarios are demonstrated and discussed.

Two resulting RL statistics are presented in every test case, namely the:

- Average discounted return: the mean of the discounted return through the epoch. This measure indicates to which degree the agent is able to improve the discounted return throughout the AST simulation epochs.

- Number of episodes: The number of episodes ran for every epoch. For some of the epochs this number is higher than the baseline of batch size divided by the maximum number of AST steps, which indicates that the agent found failures using a reduced number of AST steps which gave room for more trajectories in that epoch. Thus, a higher number of epochs indicate a higher number of trajectories that led to failure.

As described in Section 3.3.1, the actions of the AST agent in the SimpleSim case amount to disturbances in the vessel dynamics and the goal was hit when the vessel was driven off the path, to some desired offset value. In the milliAmpere COLAV simulator case, the actions of the agent amount to actuation in the obstacle vehicle and the goal was hit when the obstacle collided with the ferry.

4.1 Results using the SimpleSim

In the cases where the AST method was able to identify failures, the action sequences that led to failure was stored in order to replay these in the SimpleSim and gather the results for analysis and discussion. Due to the high number of failure cases, only every 100th action sequence was obtained. The RL statistics are presented together with the state values, the resulting trajectories, the action sequences and the regulator control input. Early trajectories are plotted in a lighter color in order to emphasize the latter trajectories, which are the trajectories the AST agent converges to.

4.1.1 Results with the PD regulator

With the PD regulated vessel, the AST method was able to identify failures with a path offset of 2m and 2.5m. The test was also performed with an offset of 3m, but no results were obtained in this case.

In the 2m case, failure modes were identified for w_{max} values of 80, 100, 120 and 150. In the 2.5m case, only $w_{max} = 120$ and 150 gave results. This is coherent with the baseline tests with constant disturbance, where $w_{max} = 80$, 100, 120 and 150 resulted in an offset past 2m, and $w_{max} = 120$ and 150 resulted in an offset past 2.5m.

Path offset	w_{max} values	Number of trajectories with failure	Total number of trajectories	Goal percentage
2m	30	0	3 000	0%
	80	8 035	8 158	98.5%
	100	8 796	9 388	93.7%
	120	13 924	14 325	97.2%
	150	18 825	19 032	98.9%
2.5m	30	0	3 000	0%
	80	0	3 000	0%
	100	0	3 000	0%
	120	10 749	11 297	95.1%
	150	11 337	12 238	92.6%
3m	-	-	-	0%

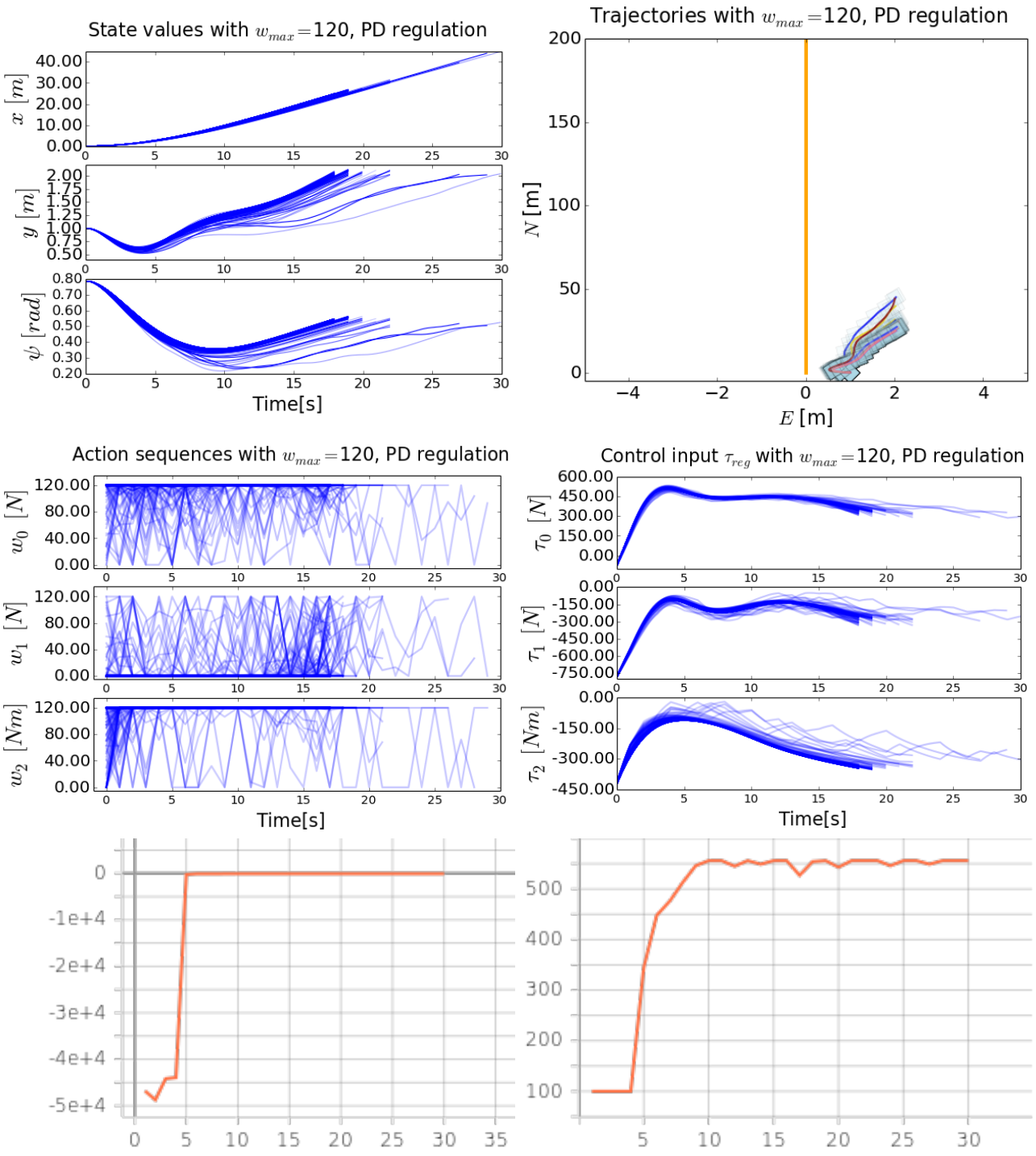
In order to illustrate the failure modes identified by the AST method, results for the two test cases 2m and 2.5m with $w_{max} = 120$ are shown in Figure 4.1 and Figure 4.2.

All the failure trajectories that were identified are in both cases highly similar, which is evident in the trajectory plots.

The results show similar patterns for both the 2m offset and the 2.5m offset case. The agent tends towards giving full disturbance in surge speed and yaw rate, while the sway speed disturbance is kept low. In the 2m offset case, the agent tends towards 0-values for the sway speed disturbance throughout the simulation, while in the 2.5m offset case it tends towards giving a high magnitude nudge in the sway speed towards the end of the episodes.

Although the AST agent is penalized during training for high action values, it is evident that the reward of reaching the goal triumphs the desire to keep action values low. The time of the sequences is reduced throughout the AST simulation, while the magnitude of the sway speed disturbance is increased, illustrating how the AST agent is willing to apply higher action values in order to reach the goal faster.

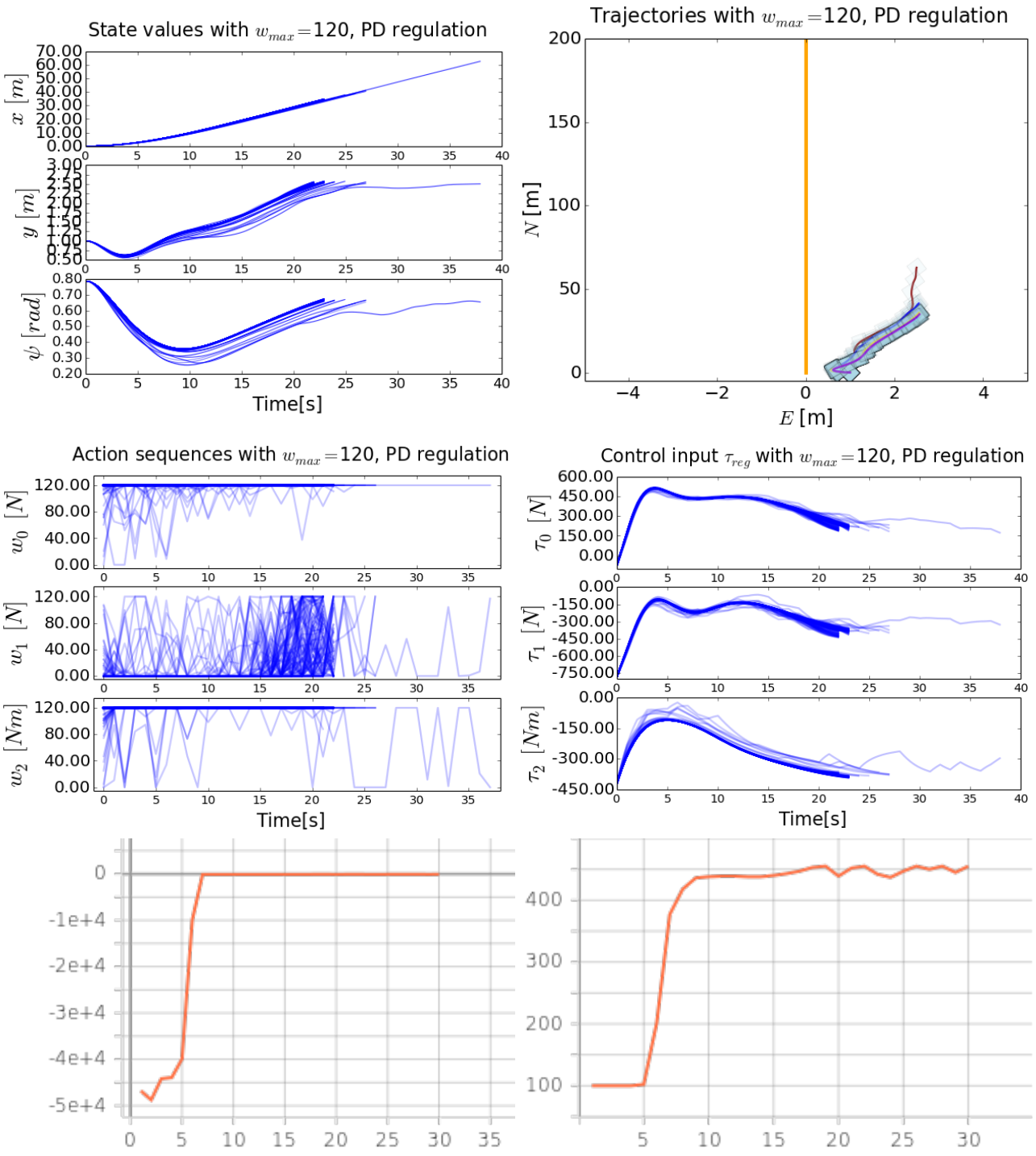
The resulting RL statistics show that the agent quickly converges to smaller values for the average discounted return, as it identifies ways to make the system fail and thus omits the big penalty for not failing. The number of episodes approaches 550 and 500 for the 2m and 2.5m case, respectively. This is coherent with the reduced time span of the failure trajectories, from 100s to ∼20s or less, which correspond to a reduction from 100 to ∼20 AST timesteps.



(a) Average discounted return

(b) Number of episodes

Figure 4.1: Simulation results from AST identified failure scenarios with desired path offset of 2m with w_{max} = 120 for PD regulated vessel.

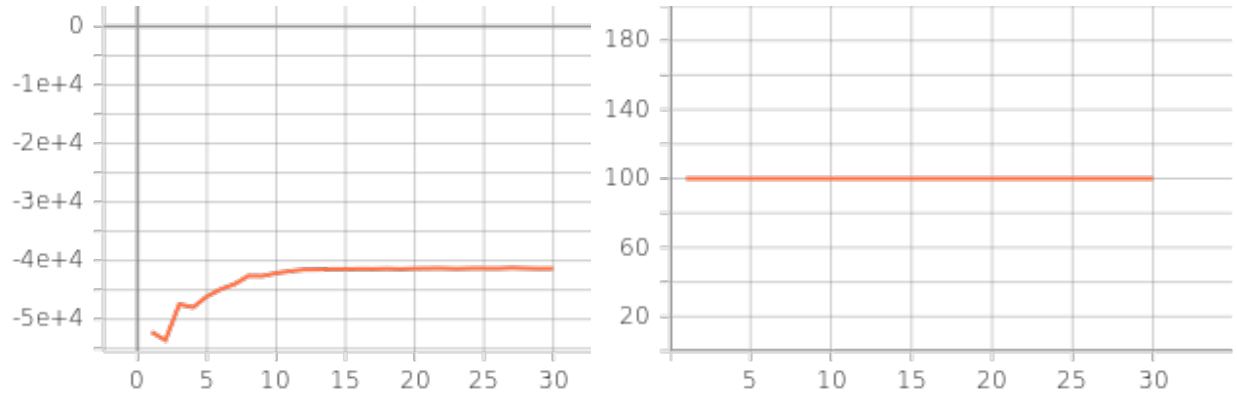


(a) Average discounted return

(b) Number of episodes

Figure 4.2: Simulation results from AST identified failure scenarios with desired path offset of 2.5m with $w_{max} = 120$ for PD regulated vessel.

In the cases where no failure modes were found, the RL statistics were very similar. The average discounted return converged to some large negative value, and the number of episodes stayed constant throughout the epochs as all episodes were performed with the maximum number of AST steps. An example of this is shown for the 2.5m offset case, with $w_{max} = 80$ in Figure 4.3, where no failure was found.



(a) Average discounted return **(b)** Number of episodes

Figure 4.3: Reinforcement learning statistics for all epochs with $w_{max} = 80$ in the 2.5m offset case.

4.1.2 Results with the PID regulator

The tests with desired offset of 2m, 2.5m and 3m was run with the PID regulated vessel as well, but no failure modes were identified. The offset was then lowered to 1.5m and 1.3m, and in the latter case, failure modes were identified for $w_{max} = 150$. As in the PD case, this is again coherent with the results from the baseline tests where only $w_{max} = 150$ resulted in an offset past 1.3m.

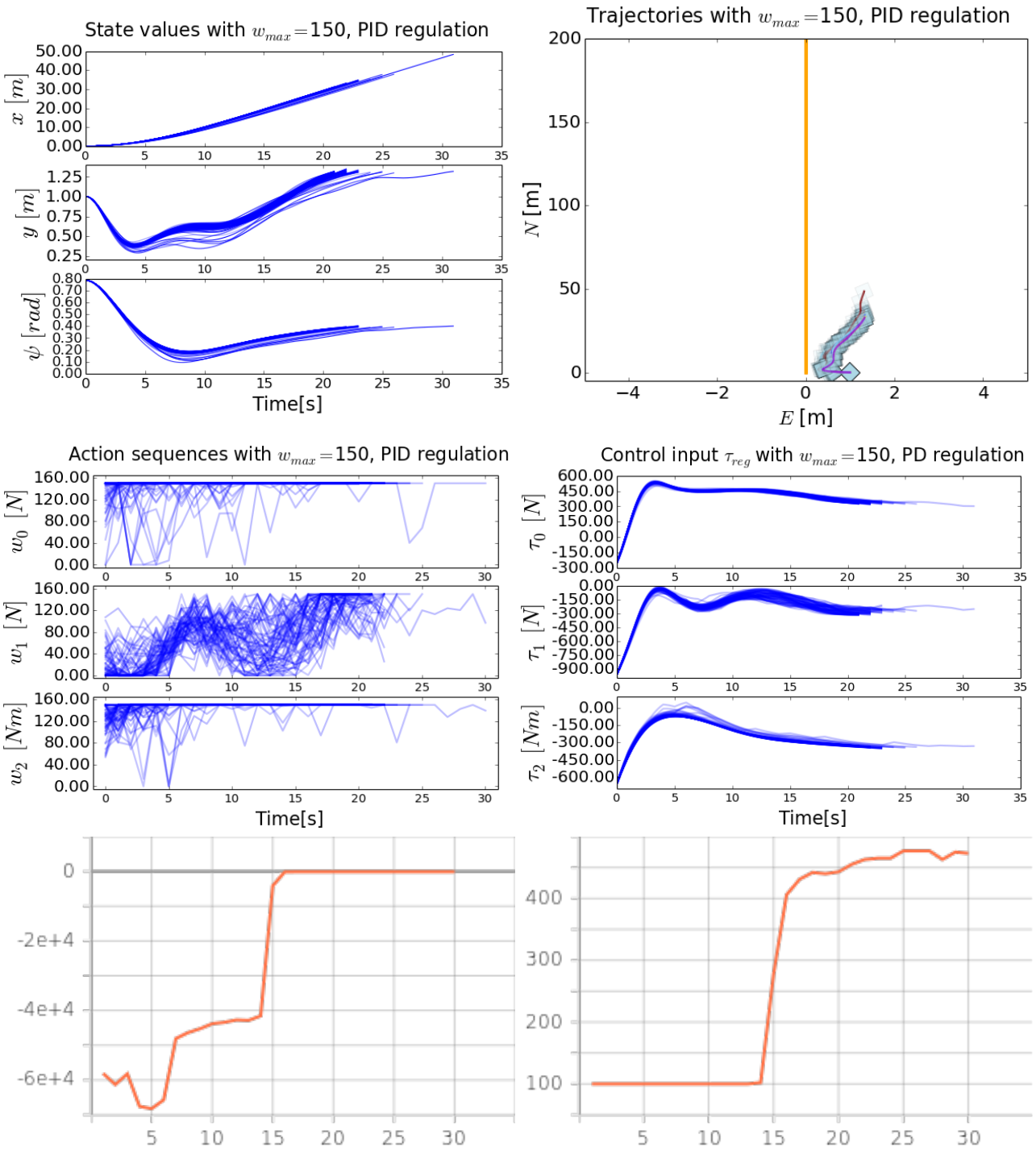
Path offset	w_{max} values	Number of trajectories with failure	Total number of trajectories	Goal percentage
1.3m	30	0	3 000	0%
	80	0	3 000	0%
	100	0	3 000	0%
	120	0	3 000	0%
	150	7 107	8 523	83.4%
1.5m	-	-	-	0%
2m	-	-	-	0%
2.5m	-	-	-	0%
3m	-	-	-	0%

The resulting failure modes show that the AST agent converges to a strategy similar to the ones found in the PD cases. The surge speed and yaw rate disturbances are kept at full value throughout the majority of the episodes. An interesting variation in this case is that the disturbances in sway speed tend to a pattern of two nudges of high value: one between 5s and 10s, and one at the end of the episode. The first nudge takes place around the same time as the control input is at its lowest values as the vessel is at its closest to the path, which is a strategically good point to nudge the vessel off the path.

As in the PD case, the failure trajectories that were identified are highly similar.

The resulting RL statistics show that the agent found failure at approximately epoch 15. Interestingly, the average discounted return seem to converge twice during the simulation. This

illustrates how the AST agent in this case is able to explore new strategies even though it has converged to a local maximum.



(a) Average discounted return

(b) Number of episodes

Figure 4.4: Simulation results from AST identified failure scenarios with $w_{max} = 150$ for PID regulated vessel.

4.1.3 Further discussion of the SimpleSim case

The AST method is able to disturb the system more efficiently than the uniform disturbances and performs quite similar to the constant disturbances with respect to path offset. What is

interesting about the AST results, is that the disturbances are not constant. Thus, the method has identified disturbance patterns of smaller magnitude than the constant disturbances which pose similar effect on the vessel. The agent tends towards full disturbance magnitude for the surge speed and yaw rate disturbances, while the values for the sway speed disturbances are kept lower and only used to nudge the vessel of the path. This indicates that the most important disturbances to provoke the vessel of the path are the ones for surge speed and yaw rate, which translates to a strategy of attempting to turn the vessel away from the path and apply force to increase the surge speed, which is a logical strategy. The results illustrate how the AST method might not be more efficient than testing with constant disturbance values, but shows how it can identify patterns in the disturbances which might not have been identified otherwise.

Once the AST agent is able to identify a failure, it seems to lock itself to this strategy and improve upon only this strategy. After a failure case is identified, the majority of the subsequent trajectories are highly similar and lead to failure. These trajectories spend less AST steps as the goal is found before the maximum number of AST steps is reached, which leads to an increased number of trajectories which all lead to failure. This is evident in the high values of the goal percentages, and indicates that there is little exploration once a good strategy is found. Another indication of this is in the tests that did not result in failure modes, where it seems that the AST agent converges to a local maximum, as illustrated in Figure 4.3. A known issue of the AST method it the problem of sparse rewards, i.e. the fact that the agent does not receive rewards that pushes it towards failure until after the episode has ended, which makes the problem a hard-exploration field. In the case of hard-exploration fields, methods can be implemented in order to urge the system to explore additionally in the case that little new rewards are obtained. Koren and Kochenderfer (2020) implemented a method called Go-Explore in order to counteract this problem that showed promising results, which could have improved the performance in these cases as the agent may have been led to explore more creative solutions. Another way to counteract the phenomenon of similar trajectories, could be to augment the reward function as in Corso, Du, et al. (2019), where an additional negative reward proportional to a measure of the dissimilarity between the obtained trajectories was implemented, which successfully gave less similar trajectories.

Although the AST method obtains results faster and for smaller disturbances for the PD regulated system than the PID regulated system, it is worth repeating that the AST method is not exhaustive, as stated in Section 1.2. Therefore these results are no formal guarantee for PID outperforming PD. The results can, however, function as an indication of this. In order to get a better idea of the comparison of these to regulators, the Differential Adaptive Stress Testing (DAST) method presented in (Lee, Kochenderfer, et al., 2015) and (Lee, Mengshoel, et al., 2020) could have been applied. This method simulates both systems simultaneously and optimizes for scenarios where it finds failures in one system but not the other. This method does not generate a formal proof of out-performance either, but it does directly compare the result of action sequences in both systems which at least could lead to formal proof of one regulator compensating for an action sequence better than the other.

The disturbance profiles vary abruptly in many of the cases. This is not likely as disturbances are physical phenomenon that will effect the system in a more continuous manner. This issue will be discussed further in Section 4.2.4.

4.2 Results using the milliAmpere COLAV simulator

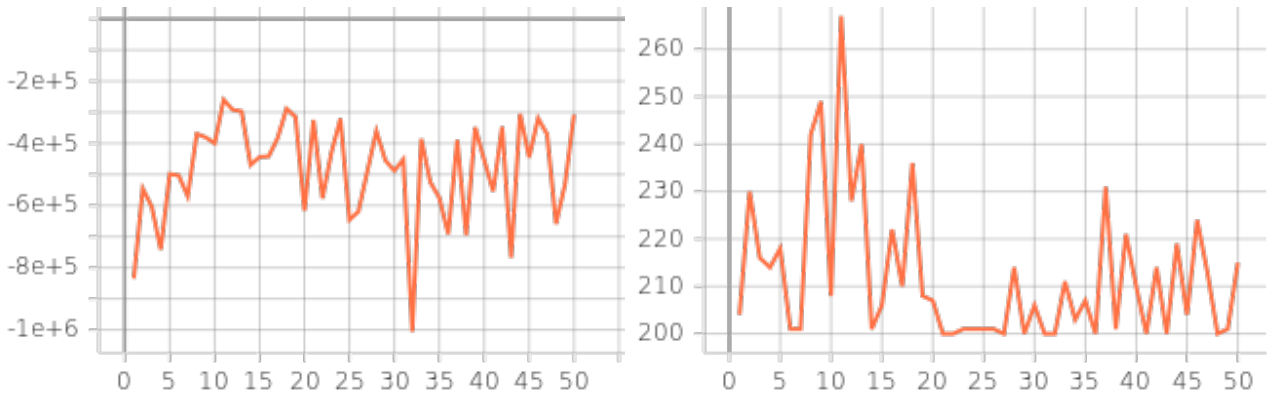
The AST method was run with the milliAmpere COLAV simulator both with and without the driver model heuristic described in Section 3.3.3. The resulting plots show simplified trajectories for both the milliAmpere ferry and the simulated obstacle, where the vessels are drawn at a given interval determined by the length of the sequence. The colour of the center of the vessels fade to darker colours towards the end of the simulation. In addition, a line is drawn between the two vessels to indicate the position of the vessels for simultaneous timesteps.

4.2.1 Without driver model heuristic

	Number of trajectories that led to failure	Total number of trajectories	Goal percentage
Without heuristic	1067	10 606	10.0%

Without the driver model heuristic, the AST agent quickly tends towards actions that makes the obstacle attack the ferry head on. As there is no penalty for this behaviour, this pattern is continued with different approaches. In the start, the obstacle attacks from above, which makes the ferry stop midway. This tactic is continued for a while, before the agent starts to experiment with attacking from below. These policies increases the time it takes until crash is reached and thus the agent receives more negative reward in total, which makes the agent gradually tends towards attacking higher again, and ends up attacking from the side at the end of the AST simulation.

The resulting RL statistics presented in Figure 4.5 show that the training process is a lot more unstable for the millAmpere COLAV simulator case than the SimpleSim case. Both the average discounted reward and the number of episodes is quite unstable and does not converge, which indicates that the system has not converged to a policy. As mentioned, the experiment was set up with 40 simulator timesteps of 0.1s for every AST step. This interval is quite big, and thus a small change in the action sequence can lead to a very different vessel trajectory, which may be causing the AST agent to explore more. The advantage of this is that there is much exploration and the resulting trajectories are quite dissimilar. This does however lead to less clear strategies towards failure, which can make it harder to present this result to system designers.



(a) Average discounted return **(b)** Number of episodes

Figure 4.5: Reinforcement learning statistics for all epochs without heuristic.

Examples of resulting trajectories are shown in Figure 4.6 - Figure 4.8.

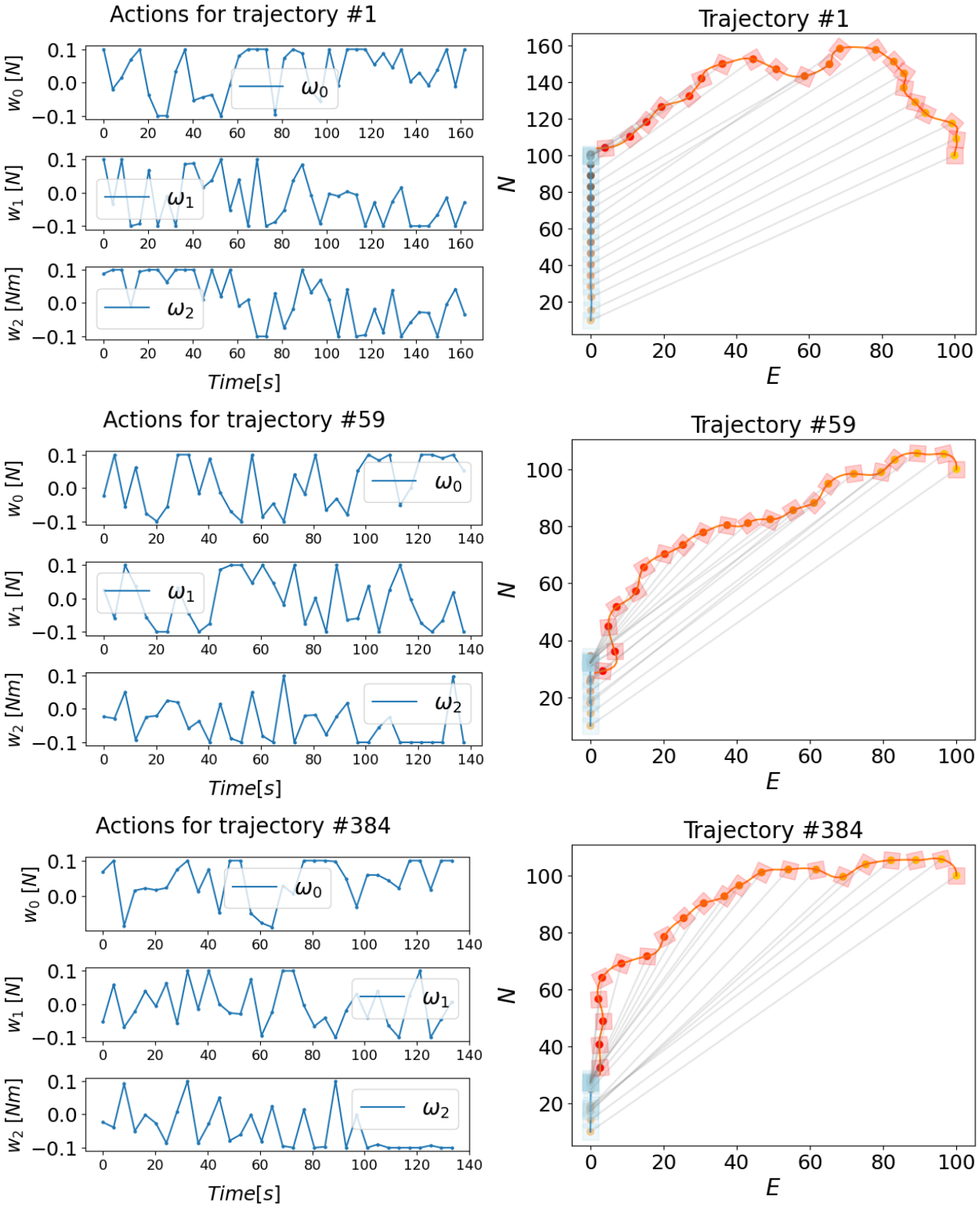


Figure 4.6: The majority of the early discovered trajectories are scenarios where the obstacle attacks the ferry from above. This strategy makes the ferry stop in some cases, as in the two latter plots shown here.

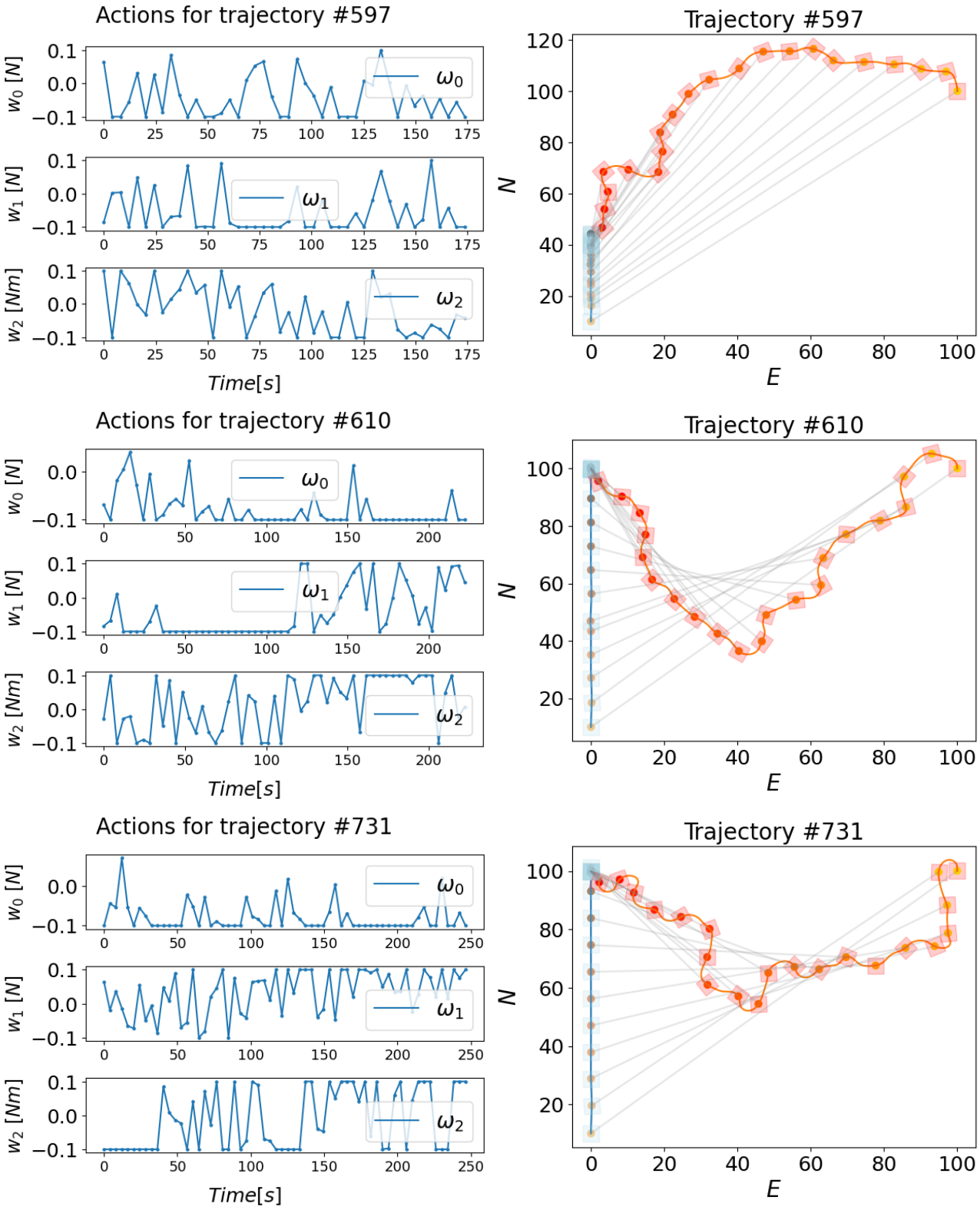


Figure 4.7: The system does some experimenting with the obstacle attacking from below.

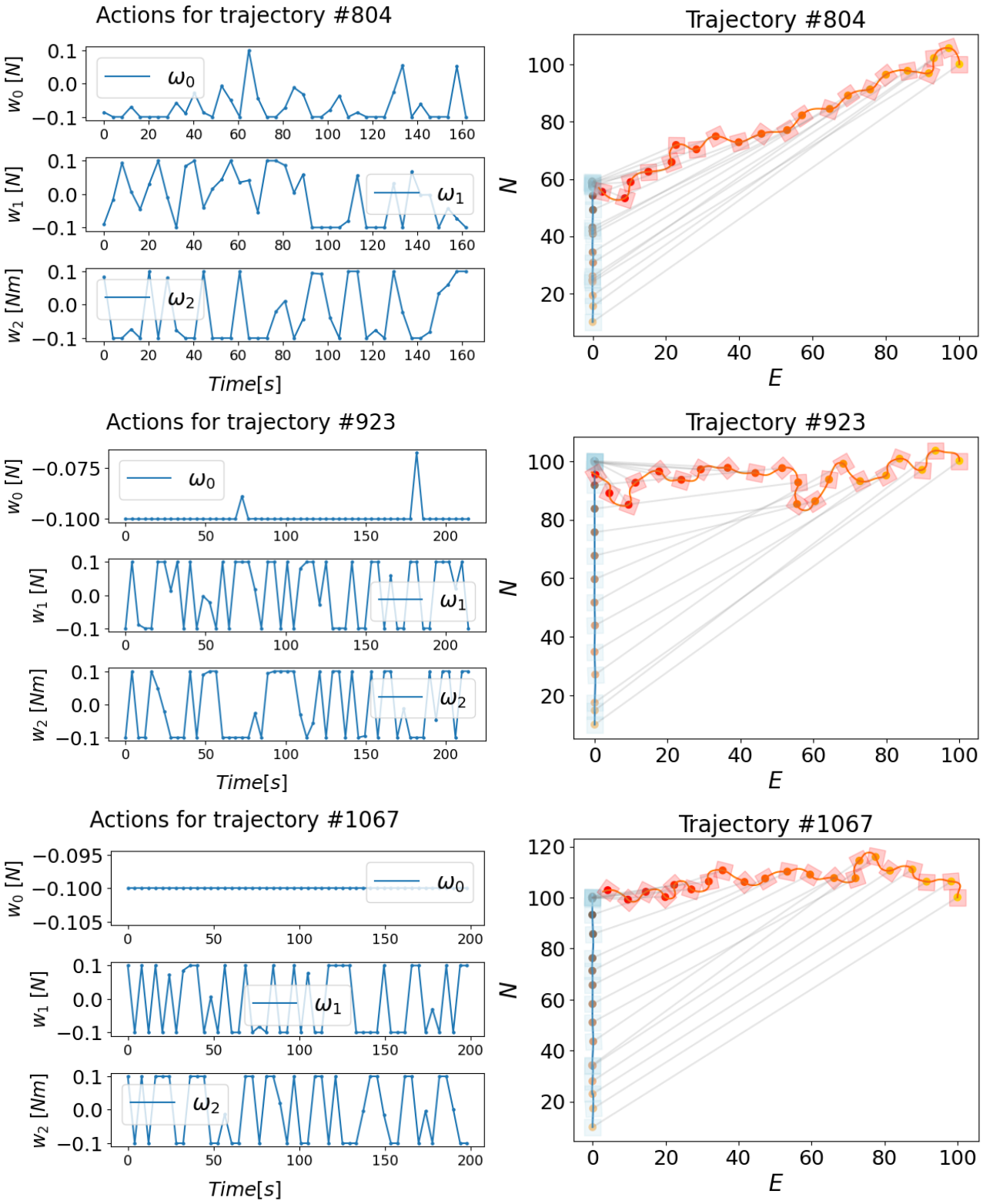


Figure 4.8: The trajectories reported at the end of the AST simulation present scenarios where the obstacle attacks the ferry from the side.

4.2.2 With driver model heuristic

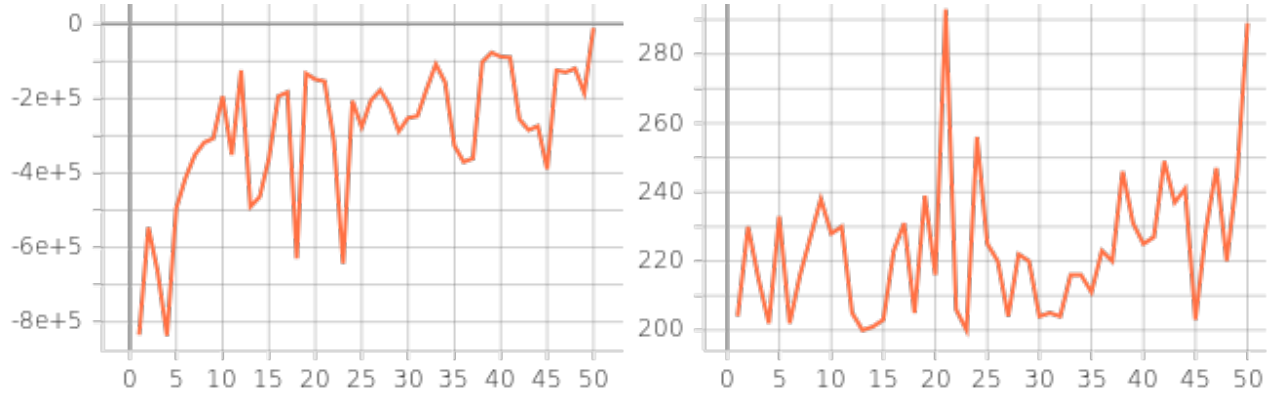
	Number of trajectories that led to failure	Total number of trajectories	Goal percentage
With heuristic	2857	11 182	25.5%

With the driver model training heuristic, the resulting trajectories take slightly different form. The trajectories express more creative ways to reach the goal without the obstacle spending long periods of drive driving towards the ferry.

The agent spends some time trying to attack the ferry from above, as it had success with without the heuristic, but it gradually tends towards doing one or more detours before attacking the ferry at the end. This makes sense as this detour prevents the negative reward of driving towards the ferry. Unfortunately, although the trajectories are more creative, they do not seem more realistic or likely.

The RL agent does not seem to prioritize spending less time as the majority of the trajectories spend over 200 s, which is likely due to the fact that the negative reward for driving the obstacle straight towards the ferry is more significant than the reward it receives for the action at each timestep. Thus, the system has more incentive of not driving towards the ferry than it has to reach the goal quicker.

The resulting RL statistics presented in Figure 4.5 interestingly present more system and signs of convergence in the average discounted return than in the case without heuristic. This may be due to the fact that there is restrictions on the system behaviour which results in less room for exploration which in turn leads to more optimization on the strategies it finds, which again is reflected in the fact that a lot more trajectories were discovered in this case than without the heuristic.



(a) Average discounted return **(b)** Number of episodes

Figure 4.9: Reinforcement learning statistics for all epochs with heuristic.

Examples of resulting trajectories are shown in Figure 4.10 - Figure 4.12.

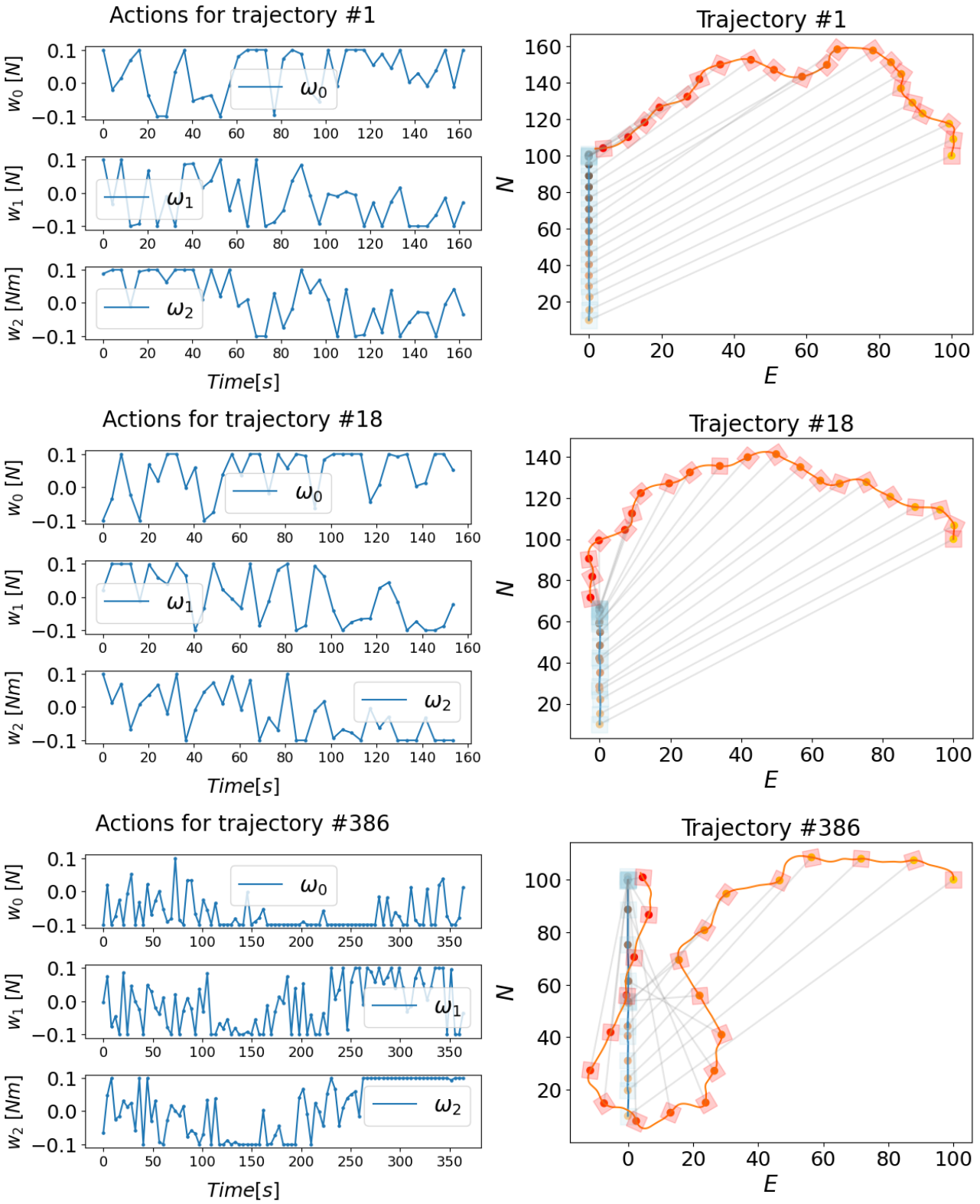


Figure 4.10: The system quickly starts experimenting with creative ways to crash without having the obstacle drive directly towards the ferry.

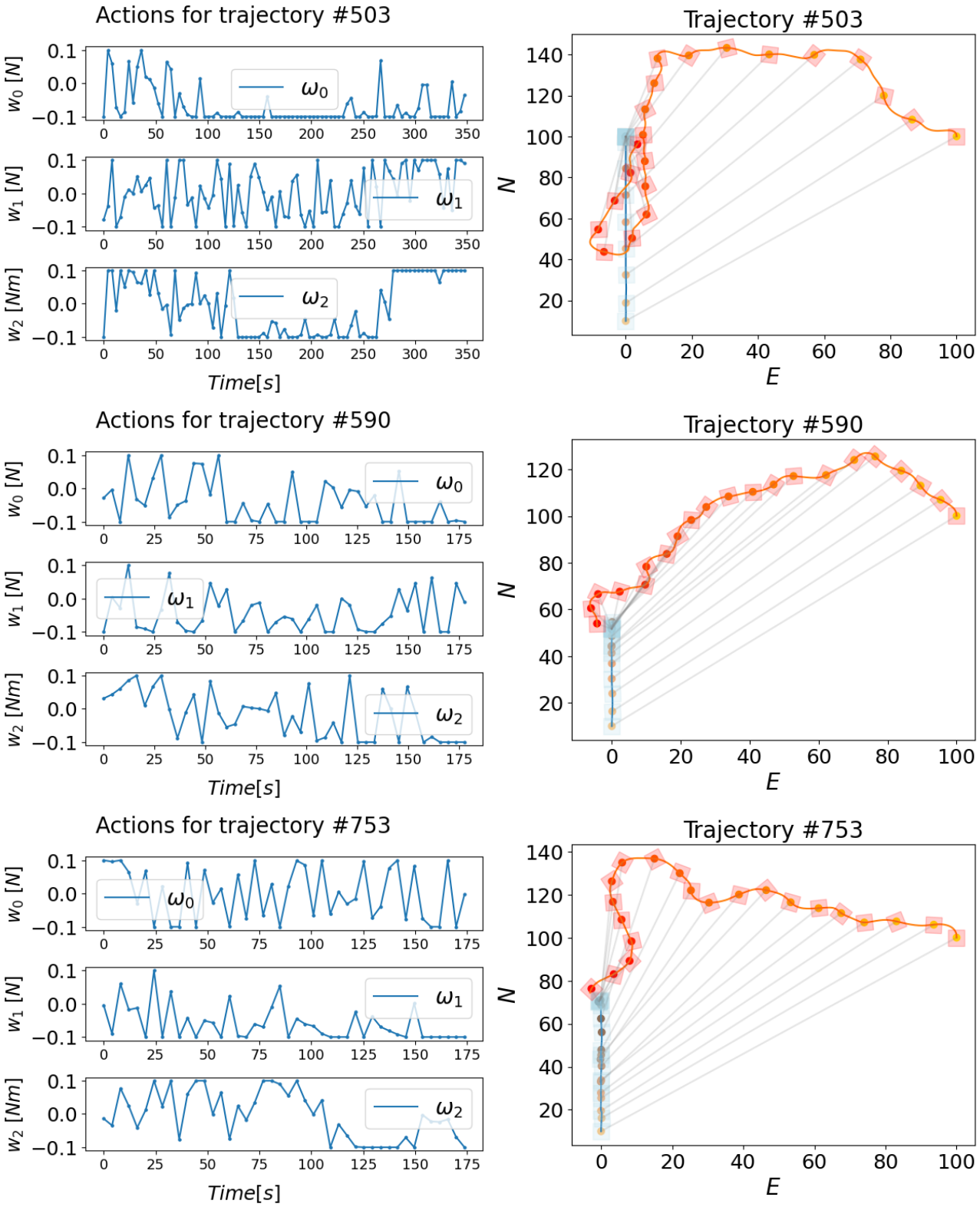


Figure 4.11: Midway in the simulation, the agent experiments with some scenarios where the obstacle attacks the ferry from above.

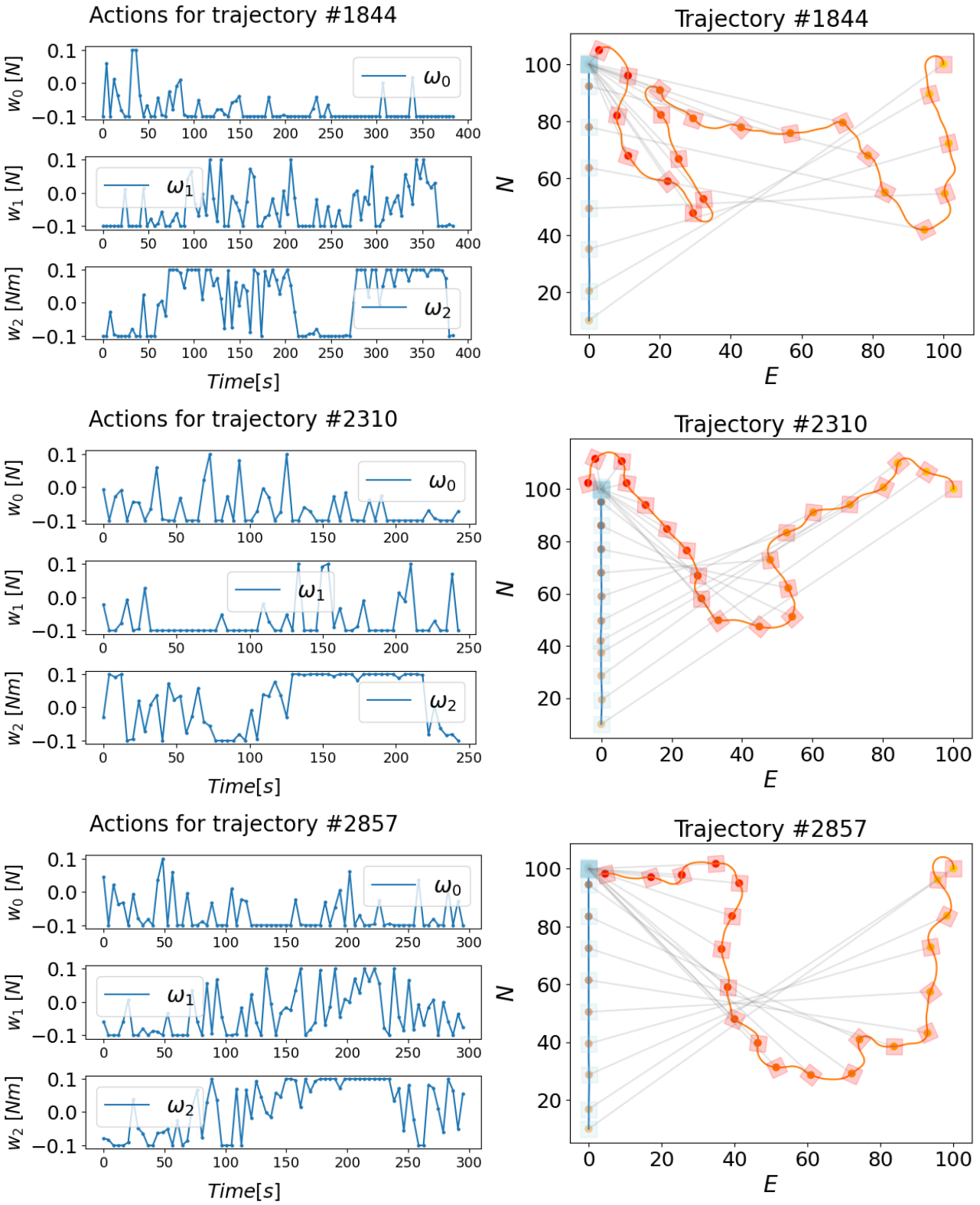


Figure 4.12: The last found trajectories tend to contain scenarios where the obstacle made one or more drastic detours before attacking the ferry.

4.2.3 Discussion for the milliAmpere COLAV simulator results

The training heuristic that was supposed to make the obstacle exhibit less irrational behaviour, did not show satisfying results. Although there were less trajectories where the obstacle drove

straight towards the ferry, the new trajectories still exhibited unlikely obstacle behaviour. The obstacle no longer drove directly towards the ferry for long periods of time, but the new behaviours still show an aggressive approach of the obstacle and the majority of the failures show the obstacle driving towards the ferry for some shorter period of time, typically towards the end of the episode.

As stated in Section 1.2, the AST method is known to identify scenarios where failure is unavoidable. Augmentations in the reward function in resemblance to the ones in (Corso, Du, et al., 2019) could have been made in order to avoid this issue. Corso, Du, et al. (2019) implemented a model that provided a negative reward at the end of episodes that did not lead to failure, if the trajectory was not compliant with the Responsible-Sensitive Safety driving rules. This could have been implemented in a similar manner by making use of the COLREGs, which is a similar rule framework for marine navigation. The heuristic was a measure implemented in order to prevent these unavoidable failure scenarios, but the heuristic reward was distributed at each AST step and not at the end of the trajectory, as in (Corso, Du, et al., 2019). It seems reasonable to evaluate the probability of the full trajectory rather than the probability of every action, as actions can seem irrational over a few timesteps, but make sense in the bigger picture. Also, the simulations could have been conducted using another milliAmpere ferry model as the obstacle, with its own SP-VP COLAV system, to examine if dangerous situation could occur in such a system where both vehicles attempt to avoid each other.

In both cases, the training process is quite unstable and there is little pattern in the action sequences. The number of simulator timesteps per AST step could be reduced in further work in order to be able to identify patterns that lead to crash more clearly. There is, however, an upside to this, which is that the agent does not lock itself to one strategy and only improve upon that one, as in the SimpleSim case.

Similar to the SimpleSim case, the action sequences vary abruptly, which is deemed unlikely and dissimilar to human behaviour. The issue will be discussed further in Section 4.2.4.

4.2.4 General discussion

TRPO was used for optimization in both cases. Proximal Policy Optimization (PPO) is a similar, but simpler algorithm that has shown to outperform TRPO on a collection of benchmark tasks (Schulman, Wolski, et al., 2017). Thus, PPO could potentially have made the system perform better.

In both simulator cases, the action sequences vary abruptly, which leads to unrealistic action sequences. A measure to counteract this could be to evaluate the likelihood of the entire trajectory at the end of an episode, as done in (Corso, Du, et al., 2019), or to add a training heuristic that penalized the magnitude of the action sequence derivative. Other methods for evening out the action sequence could also be applied, such as filtering.

The results in both simulator cases has the potential of being valuable to a design team on the mission to implement an algorithm in their systems. The results in the SimpleSim case demonstrate the robustness of the PID regulator compared to the PD regulator, and illustrate patterns in the disturbances which can have big effects the vessel. Failure modes are uncovered for the the milliAmpere COLAV system, and different failures are found for the cases with and without the heuristic, which can be useful to the Zeabuz design team. However, both simulator cases illustrate drawbacks of the AST method. The SimpleSim case demonstrated how the AST

method finds very similar trajectories, and the milliAmpere COLAV case illustrate how many of the failure scenarios contain unavoidable failure, as the obstacle drove directly towards the ferry in the majority of the obtained failure scenarios. As stated in both simulator cases, measures could have been made in order to avoid this and to obtain more realistic scenarios which could have provided more value in a system design process.

To improve the value for the Zeabuz design team, a system could have been made in order to cluster the output trajectories of the milliAmpere COLAV simulations, in order to provide system designers with more indication of patterns and tendencies in the trajectories the system finds.

Chapter 5

Conclusion and Further Work

5.1 Conclusion

The RL-based AST method provides a framework for efficient identification of system failure modes. The AST method successfully integrates with both a simple milliAmpere simulator called the SimpleSim and a more complex COLAV simulator of the milliAmpere vessel. In both cases, the AST system is able to optimize towards and discover failure modes in the simulated systems. Both test cases demonstrate how the AST method is equally relevant for marine systems as for aircraft and automobile systems, where it previously has shown promising results.

In the SimpleSim case, the vessel implemented with PID control was shown to be harder to provoke into failure by adding disturbances, compared to the vessel implemented with PD control. This illustrates how the method can be used as an indication of method performance in comparison to other methods, although it is not a formal proof of this. One way to perform more direct comparison could be to implement the DAST method proposed in (Lee, Kochenderfer, et al., 2015), which optimizes towards failures that occur in one system but not the other. The AST method is also shown to identify efficient disturbance patterns in order to provoke the vessel off the path. A drawback of the AST method is illustrated in this test case, as the failure modes that are obtained are very similar to each other, and in the cases where the system is unable to find failure it seems to stagnate in its search process by converging to a local maximum. This indicates that the system is not sufficiently incentivized to explore new trajectories as it seems to lock itself to strategies that appear good. This may be due to the problem of sparse rewards, i. e. that the system is not given rewards which lead it toward failure until the end of the episode. Koren and Kochenderfer (2020) discussed this issue of the AST method and proposed a solution using the Go-Explore method, which has shown good effects in other so called hard-exploration fields.

In the second simulator case, failure modes are successfully identified for the milliAmpere COLAV system. The results show that the method can be used to identify collision events and expose potentially dangerous scenarios. However, another drawback of the AST method is illustrated in this test case, namely that the majority of the failures found contain unavoidable failure scenarios as the obstacle deliberately drives into the ferry. This is of course an unlikely scenario, and in discussion with Zeabuz it is also identified as an irrelevant case as the milliAmpere COLAV system is not meant for preventing deliberate attacks. The issue was addressed by implementing

a training heuristic which was supposed to make the obstacle behave more realistically. The results were not satisfactory, as this only made the obstacle do detours before attacking the ferry in the end of the trajectory. The issue of unavoidable failures has been addressed in the AST literature, and a method that has proven effective in the automobile case is to evaluate the full trajectory after the episode is finished and consider if it complies with rules that model human reasoning in traffic (Corso, Du, et al., 2019).

The information obtained from the AST method does provide insight into the tested systems of some value to the Zeabuz ferry design team, but the value could definitely be improved by introducing measures for obtaining more realistic scenarios that are not unavoidable.

5.2 Further Work

In further work, augmentations to the AST reward function with respect to the COLREGs can be made in order to obtain failure scenarios where both vessels strive to comply with marine navigation rules. In addition, a COLAV system could be implemented in the obstacle model as well, in order to emphasize the need to avoid scenarios where the obstacle deliberately drives into the milliAmpere ferry. The maximum number of simulator time steps per AST step could be reduced in order to limit the search space to obtain more clear patterns in the policy. If the problem then exposes signs of being a hard-exploration field, by obtaining similar failure modes, the Go-Explore method could be implemented as in Koren and Kochenderfer (2020).

Furthermore, more work could be conducted in order to structure the information obtained from the AST method by e. g. clustering the resulting trajectories and presenting them in a way that give system designers clear insights as to what scenarios need to be taken into consideration when assessing the safety of the system. In addition, implementing the DAST method could provide developers with a helpful tool to directly compare two different algorithms or systems.

As the AST method is flexible with regards to the simulator and the test objective, the method could also be implemented in other marine technology aspects to supplement the safety validation process.

References

Autoferry - NTNU (2021). URL: https://www.ntnu.edu/autoferry (visited on 11/02/2021).

Autonomous Robotics Systems | Accenture (2021). URL: https://www.accenture.com/gr-en/services/industry-x/autonomous-robotic-systems (visited on 12/08/2021).

Autonomous Ships Market Growth, Companies, Trends by 2030 (2021). Allied Market Research. URL: https://www.alliedmarketresearch.com/autonomous-ships-market (visited on 12/07/2021).

Contents — AdaptiveStressTestingToolbox 2020.09.01.0 Documentation (2021). URL: https://ast-toolbox.readthedocs.io/en/latest/ (visited on 11/02/2021).

Corso, Anthony, Peter Du, et al. (2019). "Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validatio". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 163–168. DOI: 10.1109/ITSC.2019.8917242.

Corso, Anthony, Robert J. Moss, et al. (2021). "A Survey of Algorithms for Black-Box Safety Validation of Cyber-Physical Systems". In: *Journal of Artificial Intelligence Research* 72. ISSN: 1076-9757. DOI: 10.1613/jair.1.12716. arXiv: 2005.02979. URL: http://arxiv.org/abs/2005.02979 (visited on 10/31/2021).

Fossen, Thor I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control =: Vademecum de Navium Motu Contra Aquas et de Motu Gubernando*. Wiley: Chichester, West Sussex. 575 pp. ISBN: 978-1-119-99149-6.

Fourth Greenhouse Gas Study 2020 (2021). URL: https://www.imo.org/en/OurWork/Environment/Pages/Fourth-IMO-Greenhouse-Gas-Study-2020.aspx (visited on 12/08/2021).

Francois-Lavet, Vincent et al. (2018). "An Introduction to Deep Reinforcement Learning". In: *Foundations and Trends® in Machine Learning* 11(3-4), pp. 219–354. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000071. arXiv: 1811.12560. URL: http://arxiv.org/abs/1811.12560 (visited on 10/15/2021).

Goal 11 | Department of Economic and Social Affairs (2021). URL: https://sdgs.un.org/goals/goal11 (visited on 12/08/2021).

Interagency Report for Second Global Sustainable Transport Conference | Department of Economic and Social Affairs (2021). URL: https://sdgs.un.org/publications/interagency-report-second-global-sustainable-transport-conference (visited on 12/08/2021).

Koren, Mark, Saud Alsaif, et al. (2018). "Adaptive Stress Testing for Autonomous Vehicles". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1–7. DOI: 10.1109/IVS.2018.8500400.

Koren, Mark, Anthony Corso, and Mykel J. Kochenderfer (2020). *The Adaptive Stress Testing Formulation*. arXiv: 2004.04293 [cs, eess, stat]. URL: http://arxiv.org/abs/2004.04293 (visited on 11/09/2021).

Koren, Mark and Mykel J. Kochenderfer (2019). "Efficient Autonomy Validation in Simulation with Adaptive Stress Testing". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 4178–4183. DOI: 10.1109/ITSC.2019.8917403.

Koren, Mark and Mykel J. Kochenderfer (2020). "Adaptive Stress Testing without Domain Heuristics Using Go-Explore". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294729.

Koren, Mark, Xiaobai Ma, et al. (n.d.). "AST Toolbox: An Adaptive Stress Testing Framework for Validation of Autonomous Systems". In: p. 4.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". en. In: *Nature* 521(7553), pp. 436–444. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature14539. URL: http://www.nature.com/articles/nature14539 (visited on 09/07/2021).

Lee, Ritchie, Mykel J. Kochenderfer, et al. (2015). "Adaptive stress testing of airborne collision avoidance systems". In: *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. ISSN: 2155-7209, pp. 6C2–1–6C2–13. DOI: 10.1109/DASC.2015.7311450.

Lee, Ritchie, Ole J. Mengshoel, et al. (2020). "Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 69, pp. 1165–1201. ISSN: 1076-9757. DOI: 10.1613/jair.1.12190. URL: https://jair.org/index.php/jair/article/view/12190 (visited on 10/15/2021).

Mahalanobis, P.C. (1936). "On the generalized distance in statistics". In: pp. 49–55.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill: New York. 414 pp. ISBN: 978-0-07-042807-2.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of machine learning*. en. Adaptive computation and machine learning series. MIT Press: Cambridge, MA. ISBN: 978-0-262-01825-8.

Moss, Robert J. et al. (2020). "Adaptive Stress Testing of Trajectory Predictions in Flight Management Systems". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), pp. 1–10. DOI: 10.1109/DASC50938.2020.9256730.

One Autonomous Taxi, Please (2021). MIT News | Massachusetts Institute of Technology. URL: https://news.mit.edu/2021/autonomous-taxi-roboats-1027 (visited on 12/08/2021).

Russell, Stuart J. and Peter Norvig (1995). *Artificial intelligence: a modern approach*. en. Prentice Hall series in artificial intelligence. Prentice Hall: Englewood Cliffs, N.J. ISBN: 978-0-13-103805-9.

Schulman, John, Sergey Levine, et al. (2017). *Trust Region Policy Optimization*. arXiv: 1502.05477 [cs]. URL: http://arxiv.org/abs/1502.05477 (visited on 10/15/2021).

Schulman, John, Philipp Moritz, et al. (2018). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. arXiv: 1506.02438 [cs]. URL: http://arxiv.org/abs/1506.02438 (visited on 11/15/2021).

Schulman, John, Filip Wolski, et al. (2017). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs]. URL: http://arxiv.org/abs/1707.06347 (visited on 11/15/2021).

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*. Second edition. Adaptive Computation and Machine Learning Series. The MIT Press: Cambridge, Massachusetts. 526 pp. ISBN: 978-0-262-03924-6.

Tang, Christopher S. and Lucas P. Veelenturf (2019). "The Strategic Role of Logistics in the Industry 4.0 Era". In: *Transportation Research Part E: Logistics and Transportation*

Review 129, pp. 1–11. ISSN: 13665545. DOI: 10.1016/j.tre.2019.06.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1366554519306349 (visited on 12/08/2021).

The Dark Secret at the Heart of AI (2021). MIT Technology Review. URL: https://www.technologyreview.com/2017/04/11/5113/the-dark-secret-at-the-heart-of-ai/ (visited on 11/05/2021).

Thyri, Emil H., Morten Breivik, and Anastasios M. Lekkas (2020). "A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries in Confined Waters". In: *IFAC-PapersOnLine* 53(2), pp. 14628–14635. ISSN: 24058963. DOI: 10.1016/j.ifacol.2020.12.1472. URL: https://linkinghub.elsevier.com/retrieve/pii/S240589632031884X (visited on 12/07/2021).

Yara Birkeland | Yara International (2021). Yara None. URL: https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/ (visited on 12/07/2021).

ZEABUZ (2021). *ZERO EMISSION AUTONOMOUS URBAN AND COASTAL MOBILITY*. ZEABUZ. URL: https://zeabuz.com/ (visited on 12/07/2021).

Zhao, Xingyu et al. (2020). "Assessing Safety-Critical Systems from Operational Testing: A Study on Autonomous Vehicles". In: *Information and Software Technology* 128, p. 106393. ISSN: 09505849. DOI: 10.1016/j.infsof.2020.106393. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950584919302356 (visited on 12/07/2021).