Sanne Lin Sætre

Deep Reinforcement Learning Based Parameter Optimisation for Installation Analysis of Marine Cables

Master's thesis in Marine technology
Supervisor: Svein Sævik
Co-supervisor: Dong Trong Nguyen and Dylan van Drunen
June 2023

Sanne Lin Sætre

Deep Reinforcement Learning Based Parameter Optimisation for Installation Analysis of Marine Cables

Master’s thesis in Marine technology
Supervisor: Svein Sævik
Co-supervisor: Dong Trong Nguyen and Dylan van Drunen
June 2023

Norwegian University of Science and Technology

Sanne Lin Sætre

Deep Reinforcement Learning Based Parameter Optimisation for Installation Analysis of Marine Cables

Master's thesis in Marine Technology
Supervisor: Svein Sævik
Co-supervisor: Dylan Van Drunen and Dong Trong Nguyen
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology *Faculty of Engineering*
Department of Marine Technology

Master Thesis Description Spring 2023

for

Stud. Tech. Sanne Lin Sætre

Deep Reinforcement Learning Based Parameter Optimisation for Installation Analysis of Marine Cables
Maskinlæringsbasert parameteroptimering for installasjonsanalyse av marine kabler

Engineering analyses to support installation of marine power cables and umbilicals requires modelling a range of marine operations. The output of these analyses include lay tables or step tables to guide the vessel during the operation. Optimisation of certain parameters such as tension at the touch down point is an important part of the analysis but can be time-consuming. The idea for this master is to train a deep neural network-based agent that can replace the manual work involved in optimising the analysis. The master work to be performed during Spring 2023 represents a continuation of the project work performed in Fall 2022 and is to be carried out as follows:

1. Report the result of the literature review conducted during Fall 2022 into the master thesis document.
2. If deemed necessary, additional literature review into reinforcement learning, cable technology, basis for dynamic response analysis and relevant guidelines and standards related to design and installation analyses of marine cable systems. This is to further support the scope of work identified during the project thesis Fall 2022.
3. Define the selected case scenario including environmental data needed to train the agent algorithm.
4. Extend OpenAI Gym to use OrcaFlex as an environment, to create a reinforcement learning model (agent) including the definition of parameters, the cost function and rewards/penalties related to the parameters the agent should optimize.
5. Train the agent to optimize the following case scenarios:
    a. Cart Pole problem (baseline test)
    b. J-tube pull-in
6. In agreement with the supervisors define eventual additional case scenarios to the J-tube pull-in scenario identified during Fall 2022 and extend the agent to those applications.
7. Conclusions and recommendations for further work.

The work scope may prove to be larger than initially anticipated. Subject to approval from the supervisors, topics may be deleted from the list above or reduced in extent. This is to be notified to the reader in the introduction.

In the master report, the candidate shall present his/her personal contribution to the resolution of problems within the scope of the master work

Theories and conclusions should be based on mathematical derivations and/or logic reasoning identifying the various steps in the deduction.

The candidate should utilise the existing possibilities for obtaining relevant literature.

Master report format
The master report should be organised in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The report shall contain the following elements: A text defining the scope (this document to be included), preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables and equations shall be numerated.

The supervisors may require that the candidate, in an early stage of the work, presents a written plan for the completion of the work.

The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced using an acknowledged referencing system.

The report shall be submitted in electronic format (.pdf):
- Signed by the candidate
- The text defining the scope shall be included (this document)
- Drawings and/or computer models that are not suited to be part of the report in terms of appendices shall be provided on separate (.zip) files.

Ownership
NTNU has according to the present rules the ownership of the master reports. Any use of the report has to be approved by NTNU (or external partner when this applies). The department has the right to use the report as if the work was carried out by a NTNU employee, if nothing else has been agreed in advance.

Thesis supervisors:

Prof. Svein Sævik, NTNU, svein.savik@ntnu.no
Prof. Dong Trong Nguyen, dong.t.nguyen@ntnu.no
Dylan Van Drunen, dylan.van_drunen@nexans.com

Deadline: June, 2023, date according to further information.

Trondheim, January, 2023

[illegible]

Svein Sævik	Dong Trong Nguyen	Dylan Van Drunen
11.01.2023	11.01.2023	11.01.2023

Candidate – date and signature:

11.01.2023

Signature: [illegible]
Sanne Lin Sætre (Jan 11, 2023 13:28 GMT+1)
Email: sannels@stud.ntnu.no

Signature: [illegible]
Dylan Van Drunen (Jan 11, 2023 13:31 GMT+1)
Email: dylan.van_drunen@nexans.com

Signature: [illegible]
Dong Trong Nguyen (Jan 11, 2023 13:35 GMT+1)
Email: dong.t.nguyen@ntnu.no

Preface

This thesis documents the research and development conducted as part of the TMR4930 Marine Technology - Master's thesis at the Norwegian University of Science and Technology (NTNU), carrying a weight of 30 ECTS. It represents the final delivery of a Master of Science degree in the field of Marine Cybernetics. The project builds upon preliminary work completed during a project thesis in the Fall of 2022. The entire thesis has been authored by Sanne Lin Sætre.

The inspiration for this thesis came from an idea proposed by Dylan Van Drunen, a project engineer at Nexans. The initial vision was to explore the application of Reinforcement Learning (RL) to automate and optimize cable laying models in OrcaFlex. The overarching objective is to employ an RL-based agent for parameter optimization in OrcaFlex, specifically targeting the installation analysis of marine cables. This thesis has been developed in collaboration with supervisors at NTNU, Professor Svein Sævik and Professor Dong Trong Nguyen, along with the guidance and insights provided by Dylan Van Drunen from Nexans. As a result, the thesis is titled "Deep Reinforcement Learning-Based Parameter Optimization for Installation Analysis of Marine Cables."

The content of this thesis encompasses a comprehensive literature review that covers submarine power cable technology, cable installation design, Machine Learning (ML), Reinforcement Learning (RL), as well as pertinent software such as OrcaFlex, Python, and OpenAI Gym. Two distinct problem scenarios are addressed: the baseline CartPole problem within OrcaFlex and the J-tube pull-in problem. Specifications for each case are outlined, including the relevant parameters to be controlled by the RL agent. The implementation details of the environment and agent are presented through high-to-low-level flowcharts. Subsequently, the results achieved by the RL agent in different scenarios are presented, followed by a thorough discussion of their performance. Finally, the thesis concludes with key findings and recommendations for future work.

Throughout this thesis endeavor, supervisors Svein Sævik, Dong Trong Nguyen, and Dylan Van Drunen have provided critical insights, posed thought-provoking questions, and offered invaluable guidance and suggestions.

This thesis assumes that the reader possesses a technical background and understanding, along with basic knowledge of submarine cable installation, machine learning, and reinforcement learning concepts.

Sanne Lin Sætre

Sanne Lin Sætre

Summary

The thesis, titled "Deep Reinforcement Learning-based Parameter Optimization for Marine Cable Installation," explores the application of deep reinforcement learning (RL) techniques in the field of marine cybernetics. The thesis aims to revolutionize the modeling of cable installation processes using intelligent agents and advance the field of marine cybernetics through RL techniques.

The thesis begins with an introduction that provides background information on marine cable technology and presents a summary of the literature review. The research questions, objectives, and scope of the thesis are outlined to establish a clear direction for the thesis. The thesis describes its contributions, focusing on the advancement of marine cybernetics through RL techniques and the potential revolutionization of cable installation modeling using intelligent agents.

Subsequently, the thesis examines the technology and design elements of submarine power cables. It highlights their applications, reliability, and various design aspects crucial for successful installations.

The chapter on cable installation design explores critical scenarios in the installation process, including loads and load effects, design curves, and various steps involved in cable installation, such as cable routing, scheduling and timing, obstacle removal, transportation, reel handling, laying campaigns, cable protection, and the use of vessels and cable laying equipment. The chapter also discusses the analysis of submarine cable installation, encompassing cable laying analysis, cable pulling-in analysis, and considerations related to weather conditions.

The methodology provides an overview of machine learning techniques, including supervised and unsupervised learning, with an emphasis on deep neural networks. Reinforcement learning is introduced as the primary focus, along with its problem formulation, value function, and various modern RL techniques, such as exploration vs. exploitation, model-free vs. model-based RL, temporal difference learning, policy optimization, and deep deterministic policy gradient.

The software used for implementation and case scenarios is discussed, with a particular focus on OrcaFlex and OpenAI Gym. The thesis presents the baseline CartPole problem as a case scenario implemented in OrcaFlex, and introduces the J-tube pull-in problem as another case scenario.

In the implementation chapter, the details of the baseline CartPole problem in OrcaFlex are explained, including the OrcaFlex model, vessel motion, and the creation of a custom environment using OpenAI Gym. The execution of the model and the entire simulation is described for both random action and deep Q-network (DQN) approaches.

Similarly, the implementation of the J-tube pull-in problem is presented, covering the pay-out rate of the quadrant winch wire, environment setup, and the execution of the model and simulation.

The experiment and results chapter analyze the outcomes of the baseline CartPole problem and the J-tube pull-in problem. The results are discussed, highlighting the performance and effectiveness of the deep RL techniques in these scenarios.

Finally, the thesis concludes with a summary of the key findings and contributions. It suggests further areas of research and development to expand upon the work presented in the thesis.

Sammendrag

Masteroppgaven "Deep Reinforcement Learning basert parameter optimalisering for installasjonsanalyse av sjøkabler" utforsker anvendelsen av deep reinforcement learning (RL) teknikker innen marin kybernetikk. Målet med denne oppgaven er å revolusjonere modelleringen av kabelinstallasjonsprosesser ved bruk av intelligente agenter og fremme feltet for marin kybernetikk gjennom RL-teknikker.

Oppgaven starter med en introduksjon som gir bakgrunnsinformasjon om marin kabelteknologi og presenterer et sammendrag av litteraturstudiet fra prosjektoppgaven. Forskningsspørsmålene, formålet og omfanget av oppgaven er skissert for å etablere et klart mål med oppgaven. Fremtiden til oppgaven er beskrevet, med fokus på utviklingen av marin kybernetikk gjennom RL-teknikker og den potensielle revolusjonen av kabelinstallasjonsmodellering ved bruk av intelligente agenter.

Deretter ser oppgaven på teknologien og designelementene til undersjøiske strømkabler. Den fremhever deres applikasjoner, pålitelighet og ulike designaspekter som er avgjørende for vellykkede sjøkabel installasjoner.

Kapittelet for design av kabelinstallasjoner utforsker kritiske scenarier i installasjonsprosessen, inkludert belastninger og lasteffekter, designkurver og ulike trinn involvert i kabelinstallasjon som kabelføring, tidsplan og timing, fjerning av hindringer, transport, håndtering av kabel, leggingskampanjer, kabelbeskyttelse, og bruk av fartøy og kabelleggingsutstyr. Kapittelet diskuterer også analyse av sjøkabelinstallasjon, omfattende kabelleggingsanalyse, kabelinntrekkingsanalyse og hensyn knyttet til værforhold.

Metodikken gir en oversikt over maskinlæringsteknikker, inkludert overvåket og uovervåket læring, og legger vekt på bruk av dype nevrale nettverk. Reinforcement learning introduseres som hovedfokus, sammen med problemformuleringen, verdifunksjonen og ulike moderne RL-teknikker som utforskning vs. utnyttelse, modellfri vs. modellbasert RL, tidsforskjellslæring, policyoptimalisering og dyp deterministisk politisk gradient.

Programvaren som brukes til implementering og case-scenarier diskuteres, med spesielt fokus på OrcaFlex og OpenAI Gym. Baseline CartPole-problemet presenteres som et case-scenario implementert i OrcaFlex, og J-tube pull-in-problemet introduseres som et annet case-scenario.

I kapittelet som omhandler implementasjon er detaljene for baseline CartPole-problemet i OrcaFlex forklart, inkludert OrcaFlex-modellen, fartøyets bevegelse og opprettelsen av et tilpasset miljø ved hjelp av OpenAI Gym. Utførelsen av modellen og hele simuleringen er beskrevet for både tilfeldig handling og deep Q-nettverk (DQN) tilnærminger.

På samme måte presenteres implementeringen av J-tube pull-in-problemet, som dekker utbetalingshastigheten for kvadrant-vinsjwire, miljøoppsettet og utførelse av modellen og simuleringen.

Eksperiment- og resultatkapittelet gir en analyse av resultatene av baseline CartPole-problemet og J-tube pull-in-problemet. Resultatene blir diskutert, og fremhever ytelsen og effektiviteten til de dype RL-teknikkene i disse scenariene.

Til slutt avsluttes oppgaven med en oppsummering av de viktigste funnene og bidragene. Ytterligere forsknings- og utviklingsområder foreslås for å utvide arbeidet som presenteres i oppgaven.

Table of Contents

List of Figures

List of Tables

Abbreviation

AGB	Adaptive gradient boost
AI	Artificial intelligence
ANN	Artificial neural network
APF	Artificial potential field
API	In section 3: American Petroleum Institute
API	In other sections: Application programming interface
CDTM	Controlled depth tow method
CLV	Cable-laying vessel
COLREG	Convention on the International Regulations for Preventing Collisions at Sea
DBSCAN	Density-based spatial clustering of applications with noise
DDGP	Deep deterministic policy gradient
DNN	Deep neural network
DNV	Det norske veritas
DQN	Deep Q-Network
DRL	Deep reinforcement learning
DT	Decision tree
GA	Genetic Algorithm
GB	Gradient boosting
HER	Hindsight Experience Replay
HPS-RL	Hyperparameter Search for Reinforcement Learning
HVAC	Heating, Ventilation and Air Conditioning
IEA	International energy agency
KNN	K-nearest neighbors
LR	Logistic regression
MDP	Markov decision process
ML	Machine learning
MLR	Multi-linear regression
PCA	Prinicipal component analysis
RF	Random forest
RL	Reinforcement learning
ROV	Remotely operated vehicle
SVM	Support vector machines
TCP	Transmission Control Protocol
TD	Temporal difference
TDP	Touchdown point
USV	Unmanned Surface Vessel
ZMQ	ZeroMQ

1 Introduction

The combination of reinforcement learning and the use of OrcaFlex, a specialized software for marine cable analysis, is the central focus of this thesis.

Marine cable installations are vital for maintaining global connectivity and powering modern society. Achieving successful and efficient cable deployment requires meticulous analysis and optimization of various parameters. Deep Reinforcement Learning (DRL) has emerged as a powerful approach in recent years, offering solutions to complex optimization problems across diverse domains. This thesis aims to harness the capabilities of DRL specifically for optimizing the parameters involved in the installation analysis of marine cables.

The primary objective of this thesis is to develop a novel framework that leverages DRL techniques to automate and optimize the installation analysis process of marine cables. By employing a reinforcement learning-based agent, the thesis endeavors to maximize the efficiency and accuracy of cable installation procedures by optimizing specific parameters within the OrcaFlex model.

The implementation of DRL techniques for parameter optimization in cable installation analysis presents an exciting opportunity to revolutionize the field of Marine Cybernetics. The potential benefits are multifaceted, including improved cable deployment efficiency, reduced operational costs, and enhanced overall system performance. Moreover, by automating the optimization process, this approach enables engineers to dedicate their efforts to higher-level decision-making tasks, leading to more informed and effective cable installation strategies.

This thesis builds upon preliminary work conducted in the Fall of 2022 and addresses the pressing need for advanced optimization techniques in marine cable installation analysis. By integrating DRL algorithms with existing analysis software, the goal is to develop a robust and adaptable system capable of autonomously optimizing various parameters, such as cable tension, positioning, and route selection, to achieve optimal installation models.

Collaboration between the Norwegian University of Science and Technology (NTNU), Nexans, and industry experts has been instrumental in exploring this cutting-edge research topic. The guidance and support provided by Professor Svein Sævik and Professor Dong Trong Nguyen from NTNU, along with the insights and expertise of Dylan Van Drunen, a project engineer at Nexans, have shaped the direction and scope of this thesis.

Through this thesis, the aspiration is to contribute to the advancement of Marine Cybernetics by demonstrating the potential of Deep Reinforcement Learning in optimizing the installation analysis of marine cables. By harnessing the power of intelligent agents, seeking to unlock new possibilities for enhancing the efficiency, reliability, and sustainability of cable installation operations in the interconnected world.

1.1 Background

The increasing demand for electricity and renewable energy has led to a surge in cable installation projects worldwide. According to the International Energy Agency (IEA), renewable capacity growth is predicted to accelerate significantly in the next four years, with renewable sources accounting for nearly 95% of the global power capacity increase by 2026 (IEA 2021). In Norway, several companies, including Nexans, Global Maritime, Kongsberg Maritime, Subsea 7, and Equinor, have embraced renewable energy as part of their future plans and core values (Nexans 2022, GM 2022, Maritime 2022, Subsea7 2022, Equinor 2022).

Submarine power cables play a critical role in offshore power supply and interconnecting regions separated by water. The installation of these cables requires meticulous design and analysis processes, often utilizing specialized software like OrcaFlex. However, manual creation of suitable models for cable installations can be time-consuming and challenging. To address this issue, this thesis aims to explore the utilization of Deep Reinforcement Learning (DRL) techniques to optimize OrcaFlex models for marine cable analysis.

The installation of submarine power cables involves a complex engineering process that encompasses various stages. It begins with designing the power cable to meet specific scenarios outlined by organizations such as the American Petroleum Institute (API), which include considerations for loads, load effects, and design curves. The installation process itself involves multiple steps, such as route survey, obstacle removal, cable laying, protection, jointing, and testing. Prior to cable laying, the operation undergoes analysis using OrcaFlex. Machine Learning (ML) approaches can be employed to optimize and automate the process of creating OrcaFlex models. This thesis specifically focuses on the potential of Deep Reinforcement Learning (DRL) to develop an agent capable of creating the most effective model for a given installation operation. To accomplish this, the Python programming language and the OpenAI Gym toolkit will be utilized.

By investigating the application of DRL in marine cable analysis, this thesis seeks to address the challenges associated with manual model creation and improve the efficiency and accuracy of cable installation procedures. The potential benefits of leveraging DRL techniques in this context are significant, including enhanced cable deployment efficiency, reduced operational costs, and improved overall system performance. Moreover, automating the optimization process through intelligent agents enables engineers to concentrate on higher-level decision-making tasks, leading to more informed and effective cable installation strategies.

This thesis builds upon previous work in the field and aims to contribute advanced optimization techniques to the domain of marine cable installation analysis. By integrating DRL algorithms with existing analysis software like OrcaFlex, the objective is to develop a robust and adaptable system capable of autonomously optimizing various parameters, including cable tension, positioning, and route selection, to achieve optimal installation outcomes. The collaborative efforts between the Norwegian University of Science and Technology (NTNU), Nexans, and industry experts have facilitated the exploration of this cutting-edge research topic. The guidance and support provided by Professor Svein Sævik and Professor Dong Trong Nguyen from NTNU, along with the insights and expertise of Dylan Van Drunen, a project engineer at Nexans, have been invaluable in shaping the direction and scope of this thesis.

1.2 Literature review

Deep reinforcement learning (DRL) has emerged as a powerful tool for solving complex optimization and control problems in various domains. This literature review explores the applications of DRL in unmanned ships, fluid mechanics, robotic manipulation tasks, HVAC systems, and traffic signal control. By summarizing existing research, it aims to provide an overview of the current state of DRL applications and identify potential research gaps. Specifically, this review aims to address the research gap related to parameter optimization in the installation analysis of marine cables using DRL techniques. By filling this gap, the thesis contributes to advancing DRL-based parameter optimization in the marine industry.

1.2.1 Brief review of Machine Learning (ML) and Reinforcement Learning (RL)

Machine learning enables computers to learn from data, make predictions, and automate decision-making processes (Mitchell 1997). It encompasses various techniques, including supervised and unsupervised learning, as well as the use of deep neural networks for complex modeling tasks.

RL is a branch of machine learning focused on learning to control a system in order to maximize a long-term objective. An illustration of a basic RL scenarion is given below (Figure 1). Unlike other ML paradigms, RL does not have a supervisor but relies on a reward signal and delayed, partial feedback (Nielsen 2015).

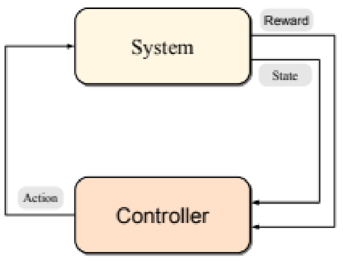


Figure 1: Example of a basic RL scenario, from (Szepesvari 2010)

Key components of RL include the environment (the world in which the agent operates), the agent (the entity that takes actions), the reward function (defines the agent's reward for actions), and the policy (the strategy the agent uses to choose actions). Some of the main challenges in RL are delayed and partial feedback (Nielsen 2015).

The problem formulation in RL is often based on a Markov decision process (MDP), which models the interaction between the agent and the environment. The goal is to find an optimal policy that maximizes the expected long-term return. Dynamic programming methods can be used when the system model is known, while RL methods can be used without a model.

The value function estimates the long-term reward for being in a state or taking an action in a state. The value function helps evaluate and improve policies.

1.2.2 Applications of DRL

"Deep Reinforcement Learning-Based Path Control and Optimization for Unmanned Ships" discusses the use of deep reinforcement learning in solving the optimization problem in path planning and management for unmanned ships. The research proposed a new reward function which considers the environment and control delay of unmanned ships to minimize the total travel time by reducing coordination time between ships. Simulation experiments were conducted to validate the effectiveness of the solution (Wu et al. 2022).

The article presents a deep reinforcement learning-based approach for optimizing path planning and management of unmanned ships. It highlights the challenges and solutions related to path optimization, unmanned ship control, and cluster control. The proposed method aims to improve planning efficiency and find optimal control rules for unmanned ships (Wu et al. 2022).

"Hyperparameter Tuning for Deep Reinforcement Learning Applications" discusses the importance of hyperparameter tuning in deep reinforcement learning (RL) applications. While RL has gained success in various domains, setting the right hyperparameters is crucial for optimal performance and reliability of the deployed models. However, deep RL has seen limited progress in hyperparameter tuning due to its algorithm complexity and the need for simulation platforms. To address this, the article proposes a distributed variable-length genetic algorithm framework called Hyperparameter Search for Reinforcement Learning (HPS-RL) to systematically tune hyperparameters for deep RL. The framework aims to improve training time and robustness by evolving optimal solutions. It demonstrates scalability and compares its performance with Bayesian approaches, showing that it can produce computationally efficient and robust models requiring fewer training episodes (Kiran and Ozyildirim 2022).

The article highlights the challenges in deep RL hyperparameter tuning and the need for practical solutions. It introduces HPS-RL, a genetic algorithm-based approach, to automate the search for optimal hyperparameters. The framework utilizes population-based evolution, crossover, and mutation to find multi-objective optimal hyperparameters for deep RL applications. It emphasizes the importance of selecting the right algorithms and gym environments for deep RL problems. The results show that HPS-RL can significantly impact RL research and applications by improving training time and producing robust models. Overall, the article contributes to advancing deep RL controllers for real-world problems by addressing the critical aspect of hyperparameter tuning (Kiran and Ozyildirim 2022).

"Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization" explores the use of Deep Reinforcement Learning (DRL) in the field of fluid mechanics, particularly in active flow control and shape optimization. DRL has shown promise in handling complex problems characterized by non-linearity, non-convexity, and high dimensionality. In active flow control, DRL algorithms have been used to discover effective control strategies for systems like the 2-D cylinder. Incorporating locality and invariance in the architecture of Artificial Neural Networks (ANNs) enables the discovery of control strategies even for large systems. However, while DRL has been successful in simulations, its application in physical experiments in fluid mechanics is yet to be fully explored due to challenges such as building experiments, acquiring real-time high-accuracy data, and implementing physical actuators. The article highlights the complementary nature of simulations and experiments, with simulations serving as benchmarks and experiments providing real-life validation (Rabault et al. 2020).

In shape optimization, DRL algorithms have been used to optimize the shape of objects interacting with complex flows. Though less developed compared to flow control, DRL shows promise in this area. The article presents two applications: aerodynamic optimization of a missile-like object and lift maximization with a cylinder-like shape. These applications use parametric models for one-shot optimization, but iterative optimization methods could be explored in the future. The article concludes by emphasizing the need for further research in DRL for fluid mechanics, including addressing challenges such as data parallelism, reward function selection, and the curse of dimensionality in large control spaces. The authors envision the extension of DRL to higher Reynolds numbers and 3-D flows, as well as the transition from simulations to real-world experiments. They anticipate DRL becoming a practical tool for industrial applications and a means to empirically explore general properties of flows, highlighting the multidisciplinary nature of this research field (Rabault et al. 2020).

"A review on deep reinforcement learning for fluid mechanics" provides a review of deep reinforcement learning (DRL) applications in the field of fluid mechanics. DRL has been adopted in physics and engineering domains for solving decision-making problems that were previously challenging due to non-linearity and high dimensionality. The review covers various applications of DRL in fluid mechanics, including flow control, shape optimization, and laminar flows past a square cylinder. It discusses the choice of DRL algorithms, problem complexity, and reward shaping. The article also highlights the potential of DRL in fluid dynamics and suggests future possibilities for coupling DRL with fluid dynamics for optimization and control tasks (Garnier et al. 2021).

While there have been several applications of DRL in fluid mechanics, the literature on the topic remains limited. The article presents a comprehensive overview of the current state of DRL in fluid mechanics, describing the numerical context, problem complexity, and algorithm choices. The coupling of DRL algorithms with existing numerical computational fluid dynamics (CFD) solvers is relatively straightforward, offering a wide range of possibilities for optimization and control tasks. The robustness of DRL algorithms in the face of numerical noise is demonstrated, and the use of parallel computing capabilities and transfer learning is highlighted. However, there are still challenges to be addressed, such as applying DRL to highly turbulent and non-linear flows and exploring its behavior in high-dimensional action spaces. The article anticipates further advancements in DRL driven by ongoing progress in the field and the industrial challenges that can benefit from it (Garnier et al. 2021).

"Automatic Parameter Optimization Using Genetic Algorithm in Deep Reinforcement Learning for Robotic Manipulation Tasks" proposes a method for automatically optimizing hyperparameters in deep reinforcement learning (RL) for robotic manipulation tasks. The approach combines Deep Deterministic Policy Gradient (DDPG) and Hindsight Experience Replay (HER) with a Genetic Algorithm (GA) to fine-tune the hyperparameter values. The algorithm is tested on six robotic manipulation tasks, showing a significant reduction in learning time and improved performance compared to existing methods. The thesis highlights the importance of automating the hyperparameter tuning process in RL and provides evidence that the GA-based approach enhances the efficiency of RL algorithms for robotic tasks (Sehgal et al. 2022).

The paper presents a novel algorithm, GA+DDPG+HER, for automatic hyperparameter tuning in deep RL. The algorithm is applied to various robotic manipulation tasks and demonstrates faster learning and improved performance. The thesis emphasizes the significance of automating hyperparameter optimization in RL and provides evidence of the effectiveness of the GA-based approach in enhancing the efficiency of learning agents. The findings suggest that this method can accelerate the learning process and improve performance in robotic manipulation tasks (Sehgal et al. 2022).

1.2.3 Applications of DQN

"Application of deep Q-networks for model-free optimal control balancing between different HVAC systems" explores the application of a deep Q-network (DQN) for model-free optimal control in HVAC systems. The DQN is integrated with an EnergyPlus simulation model of an office building to minimize energy consumption while maintaining indoor CO2 concentration below a certain threshold. The results show that the DQN can improve its control policy based on prior actions, states, and rewards, resulting in a 15.7% reduction in energy usage compared to baseline operation. The DQN also demonstrates the ability to balance control actions among different energy consumers in the building, such as chillers, pumps, and air-handling units (Ahn and C. S. Park 2020).

Traditionally, HVAC systems have been controlled using rule-based or model-based approaches, but these methods have limitations in terms of accuracy and scalability. The model-free approach employed in this thesis, using the DQN, achieves global optimization without relying on a simulation model. The findings suggest that the DQN has the potential to optimize building energy consumption and overcome the challenges associated with simulation models. Future research aims to explore the integration of transfer learning with the DQN to leverage knowledge from previous cases and to investigate the performance of the DQN in different seasons (Ahn and C. S. Park 2020).

"Deep Q-network-based traffic signal control models" focuses on the development and evaluation of traffic signal control models using deep Q-network (DQN), a reinforcement learning algorithm. Two models were developed for an isolated intersection and two coordinated intersections, and their performance was compared with a fixed-time signal control model. The study found that the developed DQN-based models showed superior performance in terms of traffic signal control compared to the fixed-time signal control method. The research highlights the potential of using artificial intelligence, specifically reinforcement learning, to solve complex problems like traffic congestion (S. Park et al. 2021).

Traffic congestion is a prevalent issue in urban areas, and traditional methods of road expansion and construction are not always feasible. Artificial intelligence has gained attention as a promising approach to address traffic congestion through intelligent traffic signal control. This study aims to contribute to this field by developing traffic signal control models using DQN, a reinforcement learning algorithm. The models were tested for an isolated intersection and coordinated intersections in an urban area. The DQN algorithm was selected due to its effectiveness in handling large state and action spaces. The developed models were compared with a fixed-time signal control model and showed better performance in both isolated and coordinated intersection scenarios. The study emphasizes the potential of using AI-based approaches to improve traffic signal control and alleviate traffic congestion in urban areas (S. Park et al. 2021).

"Path Planning of Coastal Ships Based on Optimized DQN Reward Function" proposes a coastal ship path planning model based on the optimized deep Q network (DQN) algorithm. The model combines environment status information and the DQN algorithm to achieve efficient and safe ship path planning. The traditional reward function of DQN is optimized by incorporating the potential energy reward of the target point, adding reward areas near the target point, and including danger areas near obstacles. Experimental comparisons with other algorithms demonstrate that the optimized DQN algorithm improves stability, convergence speed, and calculation time. It enables the ship to navigate autonomously while adhering to navigation rules, enhancing safety, economy, and decision-making capabilities (Guo et al. 2021).

The coastal ship path planning model consists of three components: environmental status information processing, path search, and path smoothing. The environmental processing involves grid-based modeling of the marine environment and adherence to international rules for collision avoidance at sea. The path search utilizes an optimized DQN algorithm to predict a collision-free path from the starting point to the target point. The reward function is optimized to enhance learning efficiency and convergence speed. Finally, path smoothing is applied to ensure the planned path adheres to safe navigation rules. The proposed model improves the effectiveness and safety of ship navigation, contributing to the development of intelligent ships and reducing the risk of maritime accidents (Guo et al. 2021).

"A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field" presents a path planning method for unmanned surface vessels (USVs) that incorporates collision avoidance based on deep reinforcement learning (DRL) and artificial potential field (APF). The proposed method uses a DRL algorithm, specifically Deep Q-learning network (DQN), to learn optimal action strategies in a visually simulated environment, with real-time sensor information as input. To address collision avoidance during USV navigation, the location of obstacle ships is divided into four zones according to the International Regulations for Preventing Collisions at Sea (COLREGS). The APF algorithm is employed to enhance the DQN's action space and reward function, addressing the sparse reward issue. Simulation experiments demonstrate the effectiveness of the method in achieving autonomous collision avoidance path planning (L. Li et al. 2021).

The paper presents a path planning strategy that combines DRL and APF for USVs, ensuring optimal actions for collision avoidance. The DQN algorithm learns from simulated sensor data, while the APF algorithm improves the action space and reward function. The method is validated through simulation experiments and shown to effectively address collision avoidance in various scenarios. Future work involves considering uncertain environmental factors and ship kinematics models for improved reliability and precision in real-world applications. Combining model-based DRL algorithms is also a potential direction to enhance the method's applicability and stability (L. Li et al. 2021).

1.2.4 Research gap

The research gap in the literature review related to the thesis on "Deep reinforcement learning-based parameter optimization for installation analysis of marine cables" can be identified as follows:

- Limited exploration of deep reinforcement learning-based parameter optimization in marine cable installation: Existing literature covers various applications of deep reinforcement learning in fields such as unmanned ships, fluid mechanics, robotic manipulation tasks, HVAC systems, and traffic signal control. However, there is a lack of research specifically addressing parameter optimization for marine cable installation using deep reinforcement learning techniques. The literature review does not provide direct evidence or studies that focus on optimizing parameters for marine cable installation using deep reinforcement learning.
- Lack of studies specifically addressing installation analysis of marine cables: The literature review does not specifically discuss the installation analysis of marine cables using deep reinforcement learning. While it covers applications of deep reinforcement learning in related fields such as unmanned ships and fluid mechanics, there is a gap in research specifically focusing on the installation analysis of marine cables. This research gap highlights the need for studies that specifically address the challenges and opportunities of using deep reinforcement learning for optimizing the installation process of marine cables.

To address these gaps, further research is needed to specifically explore the application of deep reinforcement learning-based parameter optimization techniques for the installation analysis of marine cables. This thesis would contribute to filling the gap in the literature and provide valuable insights into how deep reinforcement learning can be applied to optimize the parameters involved in the modeling of installation processes, leading to improved efficiency and effectiveness in the marine cable industry.

1.3 Objective and scope

1.3.1 Objective

The objective of this thesis is to develop a deep neural network-based agent using deep reinforcement learning (DRL) techniques to automate and optimize models for the analysis of marine cable installation. The aim is to replace manual work involved in parameter optimization when modeling in OrcaFlex. The focus is on improving the efficiency and accuracy modelling for the analysis process, ultimately contributing to the advancement of marine engineering practices.

This thesis aims to bridge the existing gap in the field of marine cable installation analysis by incorporating DRL techniques. While previous studies have investigated various aspects of submarine power cable technology, cable installation design, and optimization algorithms, the utilization of DRL for parameter optimization in the context of marine cable installation analysis remains relatively unexplored. By applying DRL algorithms to automate and optimize the models used in OrcaFlex, this thesis introduces a novel approach that has the potential to enhance the efficiency, accuracy, and cost-effectiveness of marine cable installation processes. Through a comprehensive exploration of DRL techniques and their application to marine cable installation, this thesis contributes to the advancement of both the marine engineering and machine learning domains.

1.3.2 Research questions

This thesis aims to address the following research questions:

- Can deep reinforcement learning (DRL) be effectively utilized for parameter optimization in the installation analysis of marine cables?
- How accurate and efficient is the trained DRL agent in optimizing critical parameters related to cable installation, such as tension at the touch-down point?

1.3.3 Scope

The scope of this thesis revolves around deep reinforcement learning-based parameter optimization for the installation analysis of marine cables. The analysis involves various marine operations and requires modeling and optimization of critical parameters. The work builds upon the preliminary project thesis conducted in Fall 2022 and extends it by incorporating DRL techniques for automated parameter optimization in OrcaFlex models.

For a detailed description of the scope, please refer to the ”Master Thesis Description Spring 2023 for Stud. Tech. Sanne Lin Sætre.”

1.4 Contribution

This thesis seeks to make significant contributions to the field of Marine Cybernetics by combining reinforcement learning (RL) techniques with marine power cable installation modeling.

1.4.1 Advancement of Marine Cybernetics through RL techniques

By applying RL principles and algorithms to automate the creation of installation models, this thesis contributes to the evolution of Marine Cybernetics, an interdisciplinary field that integrates marine engineering, computer science, and automation.

1.4.2 Revolutionizing cable installation modelling process using intelligent agents

The automation of model creation using RL techniques has the potential to revolutionize the cable installation process, reducing human effort, minimizing errors, and improving overall efficiency. The intelligent agents developed in this thesis enable the generation of accurate and reliable OrcaFlex models, facilitating safer and more effective cable installations.

Through the combination of RL and marine power cable installation modeling, this thesis aims to pave the way for innovative approaches in the field, offering new possibilities for optimizing installation operations and supporting the growth of offshore energy systems.

The subsequent chapters of this thesis will study the theoretical foundations of RL, provide a comprehensive review of marine cable installation procedures, explore the capabilities of the OrcaFlex software, present the development and integration of RL algorithms, evaluate the performance of automated models, and discuss potential future developments. By the end of this thesis, the aim is to demonstrate the effectiveness and practicality of RL techniques in automating the creation of OrcaFlex models for marine power cable installation, contributing to the advancement of marine engineering and fostering sustainable energy transmission infrastructure.

1.5 Outline

The thesis is structured as follows:

1. Introduction: Provides background information, research objectives, and the contribution of the thesis in advancing Marine Cybernetics through RL techniques and revolutionizing cable installation.

2. Submarine Power Cable Technology: Covers applications, reliability, design elements, and challenges associated with cable technology.

3. Cable Installation Design: Discusses critical scenarios, cable installation process, route planning, and selection of vessels and equipment.

4. Analysis of Submarine Cable Installation: Explores cable laying analysis, cable pulling-in analysis, weather conditions, and challenges involved in cable laying and pulling-in operations.

5. Methodology: Explains the machine learning techniques used, emphasizing reinforcement learning (RL) and its integration in cable installation optimization.

6. Implementation: Details the software tools used, setup, and configuration, along with the integration of RL algorithms with cable installation simulation models.

7. Experiments and Results: Presents findings, analysis, and comparison of RL agents' performance with traditional methods, highlighting the impact of different RL configurations and training parameters.

8. Conclusion: Summarizes the thesis' conclusions, contributions, and potential for RL techniques in revolutionizing marine cable installation. Identifies future research areas and the integration of real-time data for adaptive decision-making.

2 Submarine power cable technology

The use of submarine power cables has become increasingly common in recent years, particularly as the demand for renewable energy sources and new interconnections between countries and regions grows (IEA 2021). Figure 2 from the *Renewables 2021* report provided by the International Energy Agency (IEA) shows the growth of renewable electricity capacity by region/country from 2015-2021 as well as predicted increase until 2026.

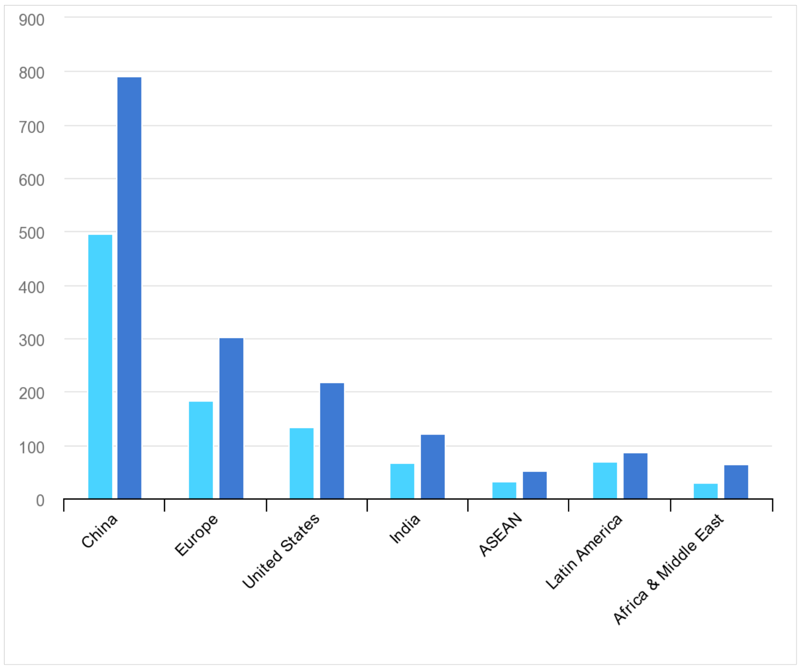


Figure 2: Renewable electricity capacity growth (in GW) by region/country, main case 2015-2020 (light blue) and 2021-2026 (dark blue), from (IEA 2021)

Submarine power cables are an important means of transmitting electrical power and data over long distances, and are often used to connect offshore wind farms, tidal energy installations, and other renewable energy sources to the electrical grid. They are also used to connect different countries and regions, allowing for the exchange of electrical power and data between them (called interconnectors). The growing number of submarine power cable installation projects are a result of this need for greater electrical energy supply diversity. In order to perform such projects, an understanding of what a submarine cable is, applications and design elements are necessary. In this section, the term *installation* will include all handling of finished cable from the factory, through loading, transfer to a laying or transportation vessel, transportation to the installation site, cable laying and jointing, and protection on or under the seabed by various means (Cigre 2022).

2.1 Applications

Through decades submarine power cables has had different major uses. Some examples are listed below:

- Power supply to islands
- Connection of autonomous grids
- Offshore wind farms
- Supply of marine platforms

- Short-haul crossings (installation of cables over relatively short distances)

In recent years, submarine power cables have been widely used in the oil and gas industry for connecting offshore facilities. However, with the growing demand for green energy, the use of submarine power cables in conjunction with offshore wind parks is expected to increase in the coming years (Worzyk 2009).

2.2 Reliability

Generally submarine cables are very reliable, however typical causes of failure are:

- Physical damage due to human activities
- Nature forces, e.g. seismic activity
- Breakdown of electrical insulation
- Hydraulic failure (if fluid-filled tubes in umbilical cables)

Repairing submarine power cables presents unique challenges compared to land power cables due to the complex underwater environment. To enhance the reliability of submarine power cables, several approaches can be taken. One strategy is to install multiple cables with sufficient spacing between them. This arrangement allows for easier repair without affecting the operation of other cables. By having redundant cables, even if one cable experiences a failure, there will still be operational cables to ensure continuous power transmission (IEEE 2005).

Alternatively, protective measures can be employed to safeguard submarine power cables. Methods such as rock dumping, burial, concrete mattresses, and horizontal directional drilling (HDD) can be used to shield the cables from potential damage. These protection methods are often more cost-effective compared to installing multiple cables.

When evaluating the reliability of submarine power cables, factors beyond installation must be considered. Ensuring repairability and preparing for maintenance activities are crucial aspects to assess. Adequate provisions should be made to facilitate efficient repair and minimize downtime in the event of a cable failure.

2.3 Design elements

There are several key design elements that are used in the construction of submarine power cables, which is illustrated in Figure 3. These elements can vary depending on factors such as the invention, development, manufacturing, testing, and installation of the cables. Some of the different types of submarine cables and construction elements that are used in the design include (Worzyk 2009, IEEE 2005):

- The conductor
- The insulation system
- The water-blocking sheath
- Armoring
- Outer serving
- Number of core cables
- Coaxial cables

- Optical fibres inside submarine power cables

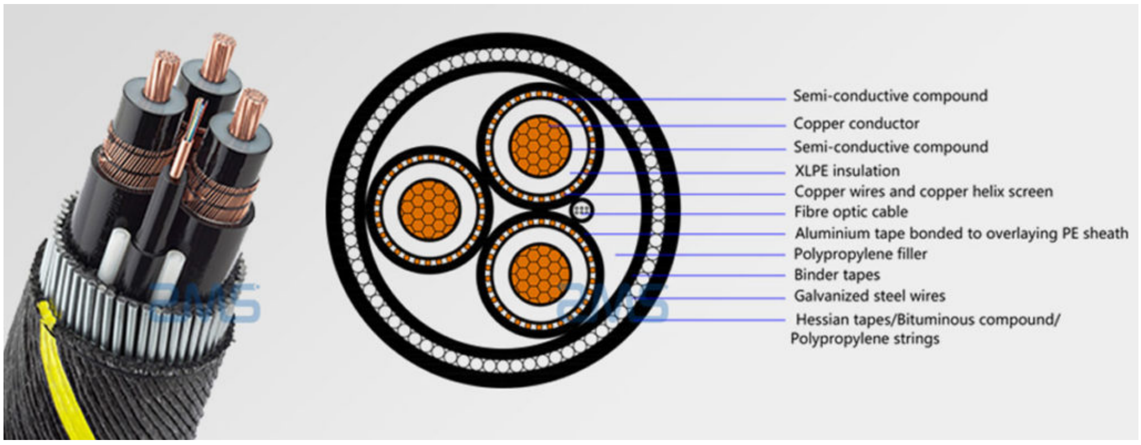


Figure 3: Illustration of design elements used in submarine power cables, from (KVCable.com 2022)

Five generic cable types, which represent the majority of submarine power cables, and their rated voltage, insulation, typical application, maximum length and typical rating are listed in Figure 4 (Worzyk 2009).

Cable Type No.	1	2	3	4	5
Rated voltage U_0	33 kV a.c.	150 kV a.c	420 kV a.c.	320 kV d.c.	450 kV d.c.
Insulation	XLPE, EPR	XLPE	Oil/paper or XLPE	Extruded	Mass-impregnated
Typical application	Supply of small islands, connection of offshore WTG	Connection of islands with large population, OWP export cables	Crossing of rivers/straights with large transmission capacity	Long-distance connections of offshore platforms or wind parks	Long-distance connection of autonomous power grids
Max. length	20–30 km	70–150 km	< 50 km	>500 km	>500 km
Typical rating	30 MW	180 MW	700 MW/ three cables	1000 MW/ cable pair	600 MW/ cable

Figure 4: Five generic submarine power cable types, from (Worzyk 2009)

3 Cable installation design

In this section, the cable installation design will be described. The term *pipeline*, *submarine power cable* and *cable* will be used. *Pipeline(s)* describe a series of welded steel pipe joints, which are stiffer in bending than cables. *Submarine power cable(s)* and *cable(s)* will be used interchangeably to refer to a cross-section without bending stiffness that is used to transmit electrical power subsea. This should be kept in mind for the remainder of this thesis.

3.1 Critical scenarios

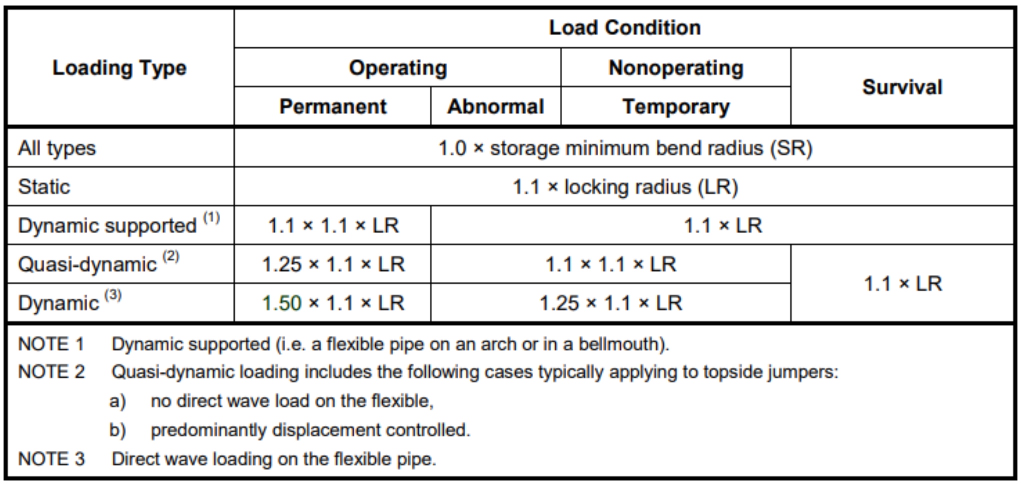
When designing a submarine power cable, it is important to follow certain requirements and standards in order to produce a certified product. Such requirements can be found in different standards provided by organizations such as Det norske veritas (DNV) or the American Petroleum Institute (API). Some of the requirements for submarine power cables can be found in the API *Specification for Unbonded Flexible Pipe* API specification 17J (API 2014b) and the API *Recommended Practice for Flexible Pipe* API recommended practice 17B (API 2014a). These standards provide guidelines for design, manufacturing, testing, and installation of flexible pipes which is a product that is similar to power cables specially with respect to installation requirements . By following these requirements, manufacturers can produce high-quality, reliable, and safe submarine power cables.

3.1.1 Loads and load effects

In general, the design of a submarine power cable is based on the information provided by the purchaser (API 2014b). The specification 17J classifies loads into three types: functional (permanent and variable), environmental (external), or accidental. The specific load combinations and classes that should be considered in the design of a submarine power cables can be found in Appendix A.1 and A.2 of specification 17J. The design requirements for the cable should be shown to be met under the load conditions specified. In addition to the specified loads, variations of the loads, load effects, and environmental and soil conditions should also be analyzed. The design load effects that should be considered include tension/compression, bending, and torsion.

3.1.2 Design curves

The design of the layers of a submarine power cables should be in accordance with the criteria specified in Appendix A.3 and Section 5 of the API 17J specification. The manufacturer should evaluate the potential for buckling failure in the pressure armors, carcass, and tensile armors, and confirm through analysis and testing that the design requirements are met. The minimum bend radius for storage should be calculated based on the criteria in Appendix A.3, and the bend radius required to cause locking in the interlocked layers should also be calculated. The minimum bend radius design criteria for different load conditions and load type combinations are shown in Figure 5.



Loading Type	Load Condition			
	Operating		Nonoperating	Survival
	Permanent	Abnormal	Temporary	
All types	1.0 × storage minimum bend radius (SR)			
Static	1.1 × locking radius (LR)			
Dynamic supported (1)	1.1 × 1.1 × LR	1.1 × LR		
Quasi-dynamic (2)	1.25 × 1.1 × LR	1.1 × 1.1 × LR		1.1 × LR
Dynamic (3)	1.50 × 1.1 × LR	1.25 × 1.1 × LR		

NOTE 1 Dynamic supported (i.e. a flexible pipe on an arch or in a bellmouth).
NOTE 2 Quasi-dynamic loading includes the following cases typically applying to topside jumpers:
a) no direct wave load on the flexible,
b) predominantly displacement controlled.
NOTE 3 Direct wave loading on the flexible pipe.

Figure 5: Minimum bend radius design criteria, from (API 2014b)

The cable manufacturer should provide a curvature-tension graph that shows the allowable curvature at different tensions, or vice versa. Curvature is defined as the reciprocal of the bend radius, $\kappa = \frac{1}{\text{bend radius}}$.

Alternatively, the design of a submarine power cable can be based on reliability. In this case, all relevant design criteria must be considered and the level of safety must be approved by the purchaser.

3.2 Cable installation steps

Submarine power cables are used to transmit electrical power under the ocean, typically from offshore wind farms or underwater power plants to the mainland (Worzyk 2009). The installation of submarine power cables involves several specialized techniques and technologies to ensure that the cables are properly placed and protected from damage. Some of the key steps involved in the installation of submarine power cables include (Cigre 2022, IEEE 2005, Worzyk 2009):

- Route survey: The first step in installing a submarine power cable is to survey the route where the cable will be installed. This involves using specialized equipment, such as sonar and underwater drones, to map the seabed and identify any potential obstacles or hazards that could damage the cable.
- Removal of obstacles: When the route survey is complete, there might be discovered that there is a need to removed old cables or other obstacles along the planned path. This can be done using a grapple, dredgers or excavators.
- Cable laying: Once the route has been surveyed and obstacles are removed, the cable is typically spooled onto the laying drum/turntable of a specialized cable-laying vessel (CLV) and laid on the seabed. The vessel is equipped with a cable-laying mechanism that unspools the cable, first through a tensioner, then over a laywheel where it is guided into desired position on the seabed. During the installation process, continuous testing of the fiber optic cable is performed.
- Cable protection: Submarine power cables are typically buried under the seabed or rock dumped if the seabed is too hard, to protect them from damage by fishing gear, anchors, and other hazards. Trenching is done using a plow or trencher with high pressure water jets which liquidizes the seabed and creates a trench for the cable to fall into.
- Cable jointing: Submarine power cables are often installed in sections, with each section typically being several kilometers long. The sections are joined together using specialized

connectors, known as cable joints, which are designed to provide a waterproof and electrically-sealed connection between the cables.

- Cable testing: Once the cable has been installed and jointed, it is typically tested once more to ensure that it is functioning properly. This usually involves injecting a test current into the cable and measuring the voltage and other electrical parameters, to ensure that the cable is able to transmit power efficiently and without losses.

3.2.1 Cable routing

The process of installing a submarine cable begins with a survey of the route where the cable will be laid. Cable routing refers to the process of determining the route that a submarine cable will follow. This involves surveying the underwater terrain and existing infrastructure, as well as taking into account factors such as sea depths, currents, and potential obstacles or hazards. Such hazards include (Worzyk 2009):

- Shipping lanes, anchorages, harbour entrances
- Fishing grounds
- Boulder fields, outcrops, submarine canyons and steep slopes
- Areas with strong water currents

This survey typically involves both site visits and desktop investigative work, and is used to gather information about the underwater terrain, existing infrastructure, and other factors that may affect the cable's installation (Cigre 2022). Based on this information, a recommended corridor and preliminary route are chosen which ensures the cable will be laid in a way that follows applicable industry practices and minimizes the risk of damage to the cable or other underwater infrastructure. Additionally, the chosen route should be designed with future maintenance and potential expansion in mind, to ensure that the cable can be easily accessed and serviced if necessary (Worzyk 2009). Once the survey and route planning are complete, the next step is to design and manufacture the cable itself, if not done concurrently. Overall, cable routing is an important part of the submarine cable installation process, as it helps to ensure that the cable is laid in a safe and efficient manner as well as influencing cost, constructability, reliability, and reparability and electrical benefits (IEEE 2005).

3.2.2 Schedule and timing

After the cable design, route, and termination facilities have been determined, there are a number of factors that can influence the installation methods and timing. These can include factors such as the type of terrain the cable will be passing through, the availability of equipment and personnel, and any potential environmental or regulatory constraints. It is important to carefully consider these factors during the planning phase to ensure that the installation can be completed efficiently and effectively. Such factors include: tidal velocity, tide height, wind velocity, wave action, fog, precipitation, snow, ice, icebergs, marine traffic, fishing seasons and environmental constraints (IEEE 2005).

The laying of submarine power cables is often a critical part of a project. Before beginning a laying operation, it is essential to complete the seabed preparations and create a "Lay Plan" that provides the crew of the CLV with the necessary parameters for vessel and cable position, touchdown point (TDP), speed, and tension, as well as a predicted residual tension after laying. It is important to note that the residual tension should always be positive but relatively small. Residual tension is the tension that remains in the cable after it has been laid on the seabed. A positive residual tension indicates that there is still some tension in the cable, which helps to prevent the cable from violating the minimum bend radius in the cable catenary or kinking where the cable twists on itself (IEEE 2005).

3.2.3 Removal of obstacles

The chosen route for a cable may be obstructed by debris that needs to be removed before the cable can be installed. This is particularly common in underwater environments, where the seabed may be covered in sediment, rocks, or other debris. If these obstacles are not identified and removed before the installation process begins, they can cause significant delays and added expenses (IEEE 2005).

To remove obstacles from the seabed, specialized equipment such as dredgers or excavators may be used. In some cases, explosives may also be needed to clear large, hard objects that cannot be removed by other means. It is important to carefully survey the route and identify potential obstacles before starting the installation process to ensure that the installation can be completed efficiently and effectively (IEEE 2005).

3.2.4 Transportation

The size and weight of a cable can influence its delivery method, with smaller cables being delivered by land and larger cables being delivered by rail or sea. Special accommodations may need to be made to receive the cable, such as having a staging area in a nearby port or preparing the job site for direct delivery. It is important to carefully plan the delivery of the cable to ensure that it arrives at the job site in a timely and cost-effective manner. This may involve coordinating with the cable manufacturer and third-party logistics providers to choose the most appropriate delivery method (IEEE 2005).

3.2.5 Reel handling

If a cable is delivered on a reel, the reel may be large and heavy, and the user's normal reel-handling equipment may not be adequate to unload it. In these cases, a crane with a large spreader bar or two cranes may be used to unload the cable safely and effectively. Once the cable is unloaded, a motorized reel-turning stand may be used to unspool the cable from the delivery reel. It is important to have the necessary equipment and personnel on hand to handle the cable reel safely and efficiently to ensure that the cable can be prepared for installation without any delays or complications (IEEE 2005).

3.2.6 Laying campaign

During a laying campaign, it is critical to monitor the catenary, or the shape of the cable in the water column between the point where it exits the CLV and the point where it touches down on the seabed (Cigre 2022). This allows for the calculation of the forces applied to the cable. The catenary can be monitored using a remotely operated vehicle (ROV) adjacent to the cable, an ROV riding on the cable, or a cable catenary scanner attached to the vessel near the cable chute (Worzyk 2009). It is important to note that the catenary does not account for the bending stiffness of the cable, but this has a relatively small impact due to the size of the cable itself. Finite element analysis can be used to calculate the behavior of the cable in the water without assuming a catenary shape (Cigre 2022).

The main parameters involved in the laying of a submarine power cable are (An example is given in Figure 6):

- w [N/m]: The unit weight of the cable per meter in the water, equal to the unit weight in air minus the weight of the volume of water occupied by a meter of cable, accounting for water infiltration into the spaces between the armor wires, servings, and bedding layers.
- H [N]: The bottom tension, or the force applied at the catenary foot point (the point where the cable first touches the seabed).

- xp [m]: The distance of the catenary foot from the installation wheel on the ship.
- R [m]: The minimum radius of curvature between the installation sheave and the catenary foot.
- d [m]: The water depth at the point where the cable leaves the installation wheel.
- T0 [N]: The catenary tangent tension at the point where the cable exits the CLV (neglecting the cable's bending stiffness and friction on the installation sheave).

By knowing these parameters, the residual tension and touchdown point of the cable can be calculated, or vice versa. Because H is given, then T is given at any point by:

$$T = H + w_s y \tag{1}$$

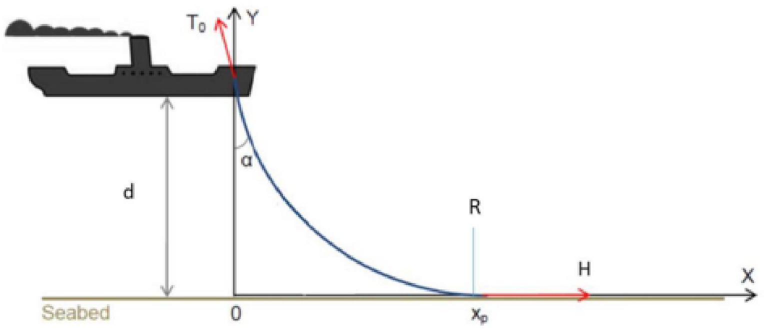


Figure 6: Example of catenary mechanical forces and parameters, from (Cigre 2022)

In addition, it is important to monitor and log various data related to the vessel, the cable, and the pay-out speed of the cable (IEEE 2005). Different contractors may have their own software (e.g. OrcaFlex) to continuously run these calculations. In the planning phase, it is important to define the target catenary angle and residual tension, taking into account the cable's parameters and a safety margin (Cigre 2022).

If the residual tension is too low, the pay-out speed can be decreased to prevent kinking or over-bending of the cable. If the residual tension is too high, the pay-out speed can be increased to prevent cable damage or free span. When laying cables in a trench, it is important to reduce the speed and residual tension when approaching bends or corners to prevent the cable from crawling out of the trench (Cigre 2022).

The pay-out speed of the cable must be controlled and synchronized with all equipment handling the cable, such as turntables, pick-up arms, caterpillars, and cable ways. Coordinating the start and stop of the vessel movement with the pay-out of the cable is important. In some cases, multiple cables may be bundled together for installation. It is important to use strong, flexible material for the bundling to allow the cables to move during installation without breaking (Cigre 2022).

3.2.7 Cable protection

Submarine power cables are valuable assets that must be protected from external hazards. Protection methods involve four steps: selecting a suitable cable route, designing suitable cable armoring, protecting the seafloor (e.g. by burying), and active after-installation protection. Active after-installation protections refers to the measures that are taken to protect the cables after they have been installed on the seabed, including use of burial techniques to cover the cables with protective layers of sand or other materials, use of protective devices, such as cable armor or protective sleeves, to prevent the cables from being damaged by external factors, such as fishing gear or marine traffic. A well-designed cable protection can increase the reliability and availability of the cable system, as well as reduce operational costs for repair and maintenance (Worzyk 2009).

Protective measures should be considered during submarine cable installation in areas with a risk of cable damage from external threats, such as intertidal zones, shipping lanes, areas with trawling fishing activity, and other areas with frequent maritime activity that could disturb the seabed. These measures can reduce the risk of damage and minimize operational and maintenance costs. These costs include financial costs, environmental disturbance, and disturbance to other sea users (Cigre 2022).

There are several ways to protect a submarine cable from physical damage. These include cable spacing, cable embedment, burial depth, and various embedment techniques. Some of these techniques include water jet plowing, vibratory plowing, high-force plowing, trench excavation, using conduit, cable protectors, cable chases, concrete tiles, and more. Horizontal directional drilling (most common for landfalls) and microtunneling can be used to install a conduit under a waterway, while rock dumping, mattresses, blankets, and scour mats can be used to provide mechanical protection (IEEE 2005).

Cable burial is a protective measure used during submarine cable installation to reduce the risk of external damage to the cable. Cable burial involves digging a trench in the seabed and placing the cable in the trench to protect it from external threats. This method can provide physical protection to the cable and can also reduce the risk of the cable being disturbed by human activity or natural events. This can be done simultaneously with laying, post laying or remedial (Cigre 2022).

3.2.8 Vessel and cable laying equipment

The configuration of a Cable Laying Vessel (CLV) can vary depending on its specific purpose and operational area. Nevertheless, there are commonalities among these vessels, as they typically share similar laying equipment and components. Presented below (Figure 7) is an illustrative CVL layout with numbered annotations describing the various parts of the vessel and its equipment (Poulsen 2022).

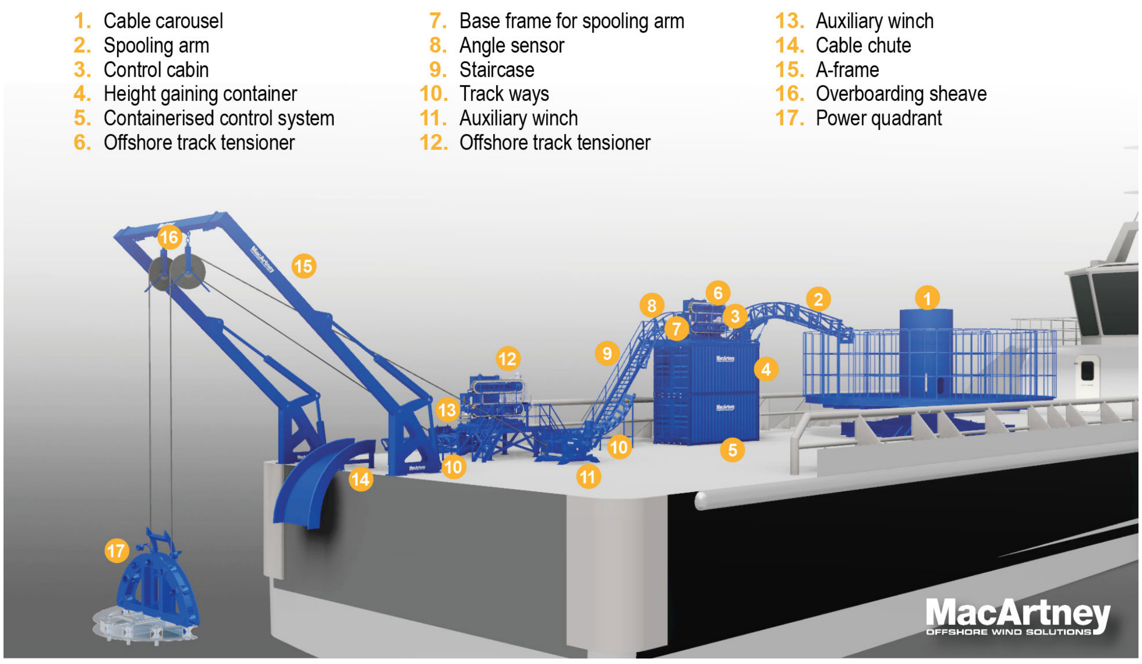


Figure 7: Outlay and explanation of cable laying vessel and equipment (Poulsen 2022).

It is important to note that the specific layout and equipment arrangement can differ between different CLVs, as they are tailored to the vessel's intended use and the requirements of cable laying projects. Variations may include vessel size, cable capacity, specialized equipment, and additional features to support specific tasks like deep-sea operations or offshore wind farm installations.

When evaluating a CLV, there are several factors that should be considered. These include the vessel's anchoring equipment, availability for installation and repair, cable-coiling facilities, cable-tensioning machines, dynamic positioning with interface with GPS, lay control equipment, laying sheave or sheaves, propulsion system, response to wave and wind action, and weight limitations. The vessel should also have adequate storage for the cable, manoeuvrability to provide accurate positioning of the cable, cable tension control equipment and deployment system, deck facilities for cable installation and recovery, workshop facilities for equipment repair, control rooms for equipment and data logging, and positioning systems (IEEE 2005). It should also have seaworthiness certificates. The specific length, freeboard deck, beam, moulded draught, moulded depth, and bollard pull of the vessel will depend on the requirements of the cable installation project. It is important to carefully evaluate these and other factors when choosing a CLV to ensure that it is suitable for the specific installation project (Cigre 2022).

In addition, it is important to consider the navigation and communication capabilities of the vessel. This includes the ability to follow a route within a given tolerance, available communication frequencies, communication system, required bottom position accuracy, and survey control system. These factors can help ensure that the vessel can operate safely and effectively during the installation process (IEEE 2005).

Another important factor to consider when choosing a CLV is the minimum bending radius of the cable. Exceeding the minimum bending radius can damage the cable and compromise its performance, it is therefore important to carefully design and place equipment such as reels, coils, sheaves, rollers, and fantail, as well as tensioning equipment and turntables, to avoid exceeding this radius. Ensuring that the minimum bending radius is not exceeded is crucial for protecting the cable and ensuring performance (Worzyk 2009).

3.2.9 Installing cable on land

After installing the cables on the seabed the cable is then connected to land, a platform, a windfarm or similar. The cables can be installed using a variety of methods, including winches, rollers, and sheaves. These tools can be used to reduce pulling tension and protect the cable from abrasion. Motorized rollers or linear machines may be used on long runs to further reduce tension. The cable may also be pulled through a pre-installed conduit system or laid in a trench (IEEE 2005).

However, there are several constraints to consider when installing cable on land. These include sidewall pressure, pulling tension, and tidal currents, wind, and vessel maneuvering ability. These factors can limit the ability to uncoil, cut, and float slack cable off the laying vessel at the receiving end of the cable installation (IEEE 2005).

3.3 Submarine cable installation analysis

It is important to properly analyze the cable's mechanical limitations prior to and during the cable laying process. This analysis should consider factors such as dynamic and static top tension, touch down tension, residual tension, friction, bending radius, curvature, and buoyancy of the cable. The analysis can be done using various standard calculation tools, such as OrcaFlex. The installation contractor should propose or confirm a suitable cable laying and protection methodology, including burial, along with the necessary vessels and equipment. The client and contractor should agree on bad weather conditions and reasonable endeavor criteria for the cable laying and burial before the contract is awarded. The installation windows should also be considered, taking into account weather conditions and permitting constraints (Cigre 2022).

3.3.1 Cable laying analysis

The cable laying analysis is a critical step in the installation of submarine power cables because it allows the engineers and technicians involved in the project to understand the forces that the cable will be subjected to during the laying process and to design the installation accordingly. This analysis is typically carried out using specialized software tools that take into account a range of factors such as the cable geometry, the vessel used for laying, the wave and current conditions, and the response of the cable to dynamic loading (Cigre 2022).

One key aspect of the cable laying analysis is the determination of the maximum allowable significant wave height for different wave directions. This is important because the cable must be able to withstand the forces imposed on it by the waves and currents during the laying process. The analysis also includes a fatigue analysis to ensure that the cable will not suffer local damage due to the repeated loading it will experience during the laying process (Cigre 2022).

Finally, the cable laying analysis includes a two-part installation analysis to determine the feasibility and safety of the operation. This includes a detailed assessment of the cable route and the vessel used for laying, as well as an evaluation of the risks and hazards associated with the installation process (Cigre 2022).

3.3.2 Cable pulling-in analysis at landfall and at offshore asset

The purpose of the cable pulling-in analysis is to determine the maximum winch force required for the pulling-in of the cable, and to determine the post-pulling-in configuration of the cable. This analysis is important because it allows for the safe and effective execution of the cable pulling-in operation, which is a critical part of the overall cable laying process. The analysis takes into account a variety of factors, including the weight of the cable, the friction of the cable along its route, the radii of the route, the sea state, and the movement of the vessel during the pull-in. The cable pulling-in analysis is often carried out using a finite element method, which allows for the modeling of complex forces acting on the cable. This method involves dividing the cable into a number of smaller elements, and solving for the forces and deformations in each element using equations that describe the behavior of the cable. The results of the analysis can provide valuable information about the maximum winch force required to safely pull in the cable, and the post-pulling-in configuration of the cable (Cigre 2022).

It is important to carefully consider the input parameters and uncertainties in the cable pulling-in analysis, and to investigate the sensitivity of the results to changes in these parameters. This can help ensure that the analysis is accurate and reliable, and that the cable pulling-in operation can be carried out safely and effectively (Cigre 2022).

3.3.3 Weather conditions

Weather conditions are an important consideration in the planning of cable laying and pull-in operations. These operations are sensitive to weather conditions, and a sufficient weather window must be available to complete the work. Key metocean data, including wind speed and direction, wave height and direction, current velocity and direction, sea ice, visibility, and lightning, must be considered in the planning process. The persistence value, or P-value, is a probability of non-exceedance of weather downtime criteria, and can be used to assess the likelihood of adverse weather conditions. In the event of severe weather, cable laying should be stopped, and the vessel turned to minimize dynamic movements of the cable. A prepared strategy should be in place for cutting and sealing the cables in case of offshore jointing or cable laying. Maintenance downtime, where essential maintenance is performed on tools, should be accounted for in the project schedule (Cigre 2022, Worzyk 2009).

Weather, including wind and waves, can have a major impact on the safety and success of cable laying operations. Wind generates waves, and the strength and direction of the wind, as well as the duration and water depth, can all affect the size and characteristics of the waves. The significant wave height, or Hs, is defined as the average height of the highest third of all waves, and is a commonly used measure of sea state. The cable laying process is sensitive to the movement of the laying wheel, which is caused by waves, and the resulting sea state. In general, cable laying should be stopped in the event of severe weather, and the vessel turned to minimize dynamic movements of the cable (Cigre 2022).

4 Methodology

4.1 Machine learning

Machine learning (ML) is a topic related to Artificial Intelligence (AI), which includes learning from data and making predictions and/or decisions (Y. Li 2018). The ML algorithm is able to learn from data (Goodfellow et al. 2016). A quote from Tom M. Mitchell (Mitchell 1997) describing ML is "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 task in T, as measured by P, improves with experience E". ML can be categorized into supervised learning, unsupervised learning and Reinforcement Learning (RL).

- Supervised learning, using labeled datasets.
- Unsupervised learning, where the agent analyze and cluster unlabeled datasets.
- RL, where sequential decision making and evaluative feedback is utilised. (Will be discussed in separate section).

(Delua 2021, Y. Li 2018).

4.1.1 Supervised learning

In supervised learning labeled datasets is given to the agent. The purpose of the datasets is to train or "supervise" the agent (Delua 2021). The agent tries to use the given data, examples of output/input pairs in order to learn the function that has produced these pairs. The agent is therefore attempting to find a general function in order to predict an outcome from the given examples. In many cases the outputs must be provided by a human or "supervisor", while in other cases this is collected automatically, but is still called supervised learning. Supervised learning can further be divided into two types; regression and classification (Goodfellow et al. 2016).

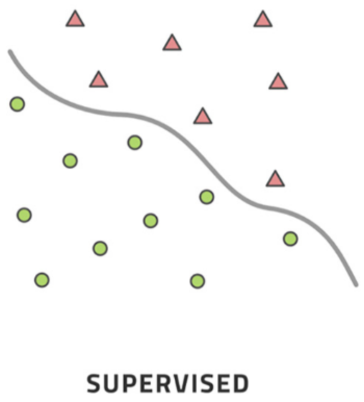


Figure 8: Illustration of supervised learning, from (Patrick 2021)

Regression is to predict a numerical value y given a data point x. Then the learning algorithm produces a function f, which predicts $y = f(x)$ (Grimstad 2022, Goodfellow et al. 2016). In other words regression uses the algorithm to understand the relationship between dependent and independent variables. These types of models can be helpful when prediction numerical values based on different data points. Examples of regression algorithms are linear regression, logistic regression (LR) and polynomial regression (Delua 2021).

Classification is when the learning algorithm produces a function f, which assigns x to a class $y = f(x)$. The algorithm specifies which of k categories a data point x belongs to (Grimstad 2022).

In classification the algorithm accurately assign test data into specific categories, e.g. sorting spam mail from other mail. Some commonly used classifiers are linear classifiers, support vector machines (SVM), decision trees (DTs) and random forest (RF) (Delua 2021).

Other examples of supervised ML algorithms is artificial neural network (ANN), gradient boosting (GB), adaptive gradient boost (AGB), multi-linear regression (MLR), LR and K-nearest neighbors (KNN) (Hoss and Alireza 2021).

4.1.2 Unsupervised learning

Unsupervised learning differs from supervised learning because the datasets given is not labeled. Therefore, existing pattern is looked for in the data input (Hoss and Alireza 2021). E.g. as basis for clustering and density estimation (Y. Li 2018). The algorithm is unsupervised in the form of discovering hidden patterns in data without human intervention. Unsupervised learning is commonly used for three main tasks; clustering, association and dimensionality reduction (Delua 2021).

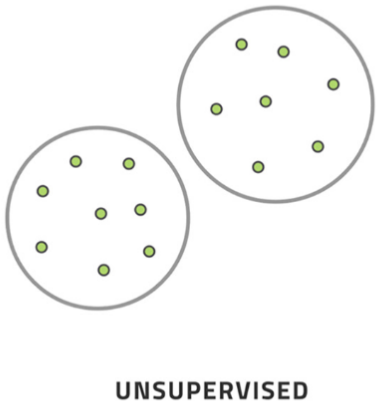


Figure 9: Illustration of unsupervised learning, from (Patrick 2021)

Clustering is a technique for grouping unlabeled data based on differences or similarities, and is a data mining technique. This technique could be helpful when dealing with segmentation, image representation and similar issues (Delua 2021).

A different type of unsupervised learning is association, which is applied in order to find relationships between variables in a given dataset by applying different rules. This type of unsupervised learning is commonly used for market basket analysis such as "A customer who bought this item, also bought this item" (Y. Li 2018).

The third type of unsupervised learning is dimensionality reduction. This is a technique where the number of features (dimensions) in a given dataset is too high. In order to obtain a manageable size of data, the algorithm reduces the number of data inputs, while preserving the data integrity. Dimensionality reduction is frequently used for preprocessing data (Delua 2021).

Other examples of unsupervised learning is k-means clustering, hierarchical clustering, density-based spatial clustering of applications with noise (DBSCAN), principal component analysis (PCA), and apriori algorithm (Hoss and Alireza 2021).

4.1.3 Deep neural networks

Artificial Intelligence can be described as a collection of analytical and numerical tools which tries to learn and imitate a process (Hoss and Alireza 2021). When this is accomplished, the AI can handle and respond to new situations. The essential building blocks of AI are neural networks, genetic algorithms, and fuzzy logic (Mohaghegh 2000). The inspiration for neural networks originates from the neurons in the human brain. Artificial neural network is a mathematical device trying to replicate the behaviour of the human brain neural network (Patrick 2021).

Logistic ANN is a commonly used multi-layer neural network containing something called sigmoid neurons. A sigmoid neuron is a type of artificial neuron that uses a sigmoid activation function. This function is defined mathematically as (Lin 1993, Nielsen 2015):

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Where x is the input value and $f(x)$ is the output value. This function has the shape of an "S" curve.

The sigmoid function has several useful properties. It is continuous, differentiable, and monotonically increasing, which means that its output value always increases as the input value increases. It also has a range of [0, 1], which means that the output value is always between 0 and 1. (Lin 1993, Nielsen 2015).

Figure 10, illustrates a neural network. The leftmost layer is called the input layer, which includes what is called the input neurons (Nielsen 2015). The input layer has several neurons equal to the number of elements in the input vector. The output layer, which is the rightmost layer includes the output neurons (here illustrated as one single output neuron). The output layer contains the same number of elements in the output vector (Patrick 2021). The layer placed in the middle is called the hidden layer (here two hidden layers), also referred to as the topology of the neural network. The reason for the name is simply because these neurons are neither input nor output neurons, hence their values are not observed. A neural network might contain one or more hidden layers depending on the architecture(Grimstad 2022).

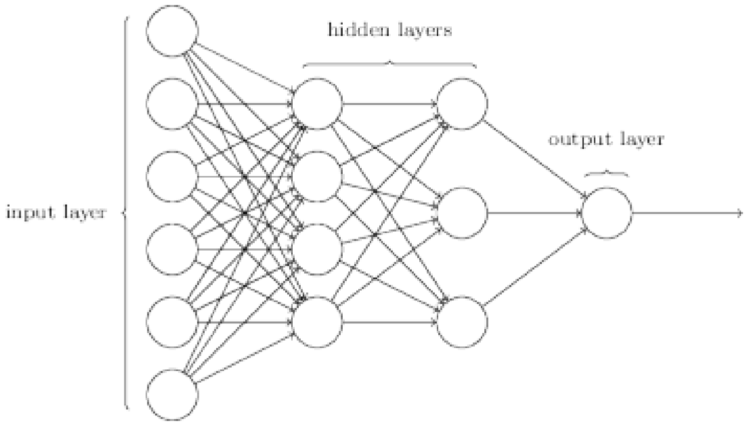


Figure 10: Illustration of neural network with input layer, two hidden layers and output layer, from (Nielsen 2015)

The design of input and output layer (or layers) is relatively simple compared to designing the hidden layer (or layers). The input layer can be both linear and nonlinear (Lin 1993). Design of hidden layers might be an intricate process because there exists few simple rules of thumb. The hidden layers are mainly responsible for feature extraction and provide increased dimensionality, as well as accommodate classification and pattern recognition. However some neural networks researchers have developed several design heuristics for hidden layers, which can manipulate the networks to give their desired behaviour (Mohaghegh 2000).

Again, looking at Figure 10, it illustrates what is called a feed-forward neural network. Feed-forward neural networks are a type of ANN in which information only flows in one direction, from the input nodes to the output nodes, without any feedback loops (Nielsen 2015). This type of neural network is illustrated in Figure 11. In contrast, recurrent neural networks are a type of ANN that includes feedback loops, allowing information to flow in multiple directions. This type of network is a closer representation of the human brain's neural network and is useful for tasks involving the recognition or prediction of temporal sequences of patterns. However, recurrent neural networks are currently less powerful than feed-forward networks and are less widely used for this reason (Lin 1993).

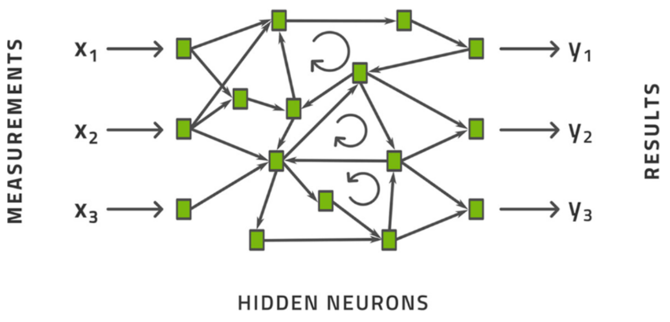


Figure 11: Illustration of recurrent neural network with input layer, hidden layers with feed-back loops and output layer, from (Patrick 2021)

4.2 Reinforcement learning

Reinforcement learning is a branch of machine learning that focuses on learning to control a system with the goal of maximizing some numerical value representing a long-term objective (Sutton and Barto 2018). It is different from other ML paradigms in that there is no supervisor, only a reward signal, and the feedback is delayed and only partial. Time is also accounted for in RL, as the predictions made by the agent can have longer-term effects and may influence future states of the controlled system. RL has many practical applications in fields such as AI, operations research, and control engineering (Silver 2015, Szepesvari 2010).

In Figure 12 the basic idea of RL is represented where the controller receives the state of the system as well as a reward and executes an action based on the state and reward. Then a new state and reward is given to the controller. From this the controller or agent can learn how to maximize the reward and execute actions based on this. RL is of special interest due to the large number of practical applications where RL can be useful, ranging from problems in artificial intelligence (AI) to operations research or control engineering (Szepesvari 2010).

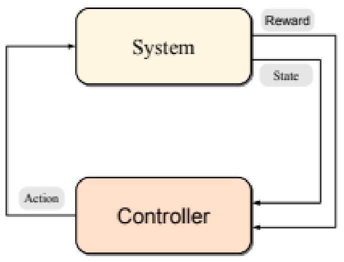


Figure 12: Example of a basic RL scenario, from (Szepesvari 2010)

In order to solve a RL problem, the following components must be defined:

- The environment: This is the world in which the agent operates. It includes the state of the environment, the possible actions that the agent can take, and the transitions between states that result from those actions.

- The agent: This is the entity that takes actions within the environment. It receives information about the state of the environment, and chooses actions based on that information in order to maximize the reward signal.
- The reward function: This is a function that defines the reward that the agent receives for taking a particular action in a particular state. The goal of the agent is to maximize the total reward it receives over time.
- The policy: This is the strategy that the agent uses to choose actions in different states. The policy is learned by the agent through interaction with the environment, and determines how the agent behaves in different situations.

Once these components are defined, the RL problem can be solved by training the agent to interact with the environment and learn the optimal policy. This typically involves using algorithms such as Q-learning or policy gradient methods to update the policy based on the rewards received by the agent (Lin 1993).

One of the key challenges in RL is the fact that the feedback is often delayed and partial. This means that the agent may not receive immediate feedback for its actions, and it may not receive complete information about the state of the environment. As a result, the agent must learn to act based on incomplete and noisy information, and it must be able to adapt its behavior over time as it receives more information (Silver 2015, Szepesvari 2010).

Despite these challenges, RL has many practical applications. For example, it can be used to control robotic systems, such as autonomous vehicles or robots that operate in unstructured environments. It can also be used to optimize decision making in complex systems, such as supply chain management or energy management. In these and other applications, RL can help agents learn to make better decisions and maximize long-term rewards (Silver 2015, Szepesvari 2010).

4.2.1 Problem formulation

In reinforcement learning, a problem is typically formulated as a Markov decision process (MDP) (Sutton and Barto 2018, Lin 1993). An MDP is a mathematical framework for modeling decision-making problems in which an agent interacts with an environment (Mnih et al. 2013, Mitchell 1997). It is defined as a 5-tuple consisting of a state space, action space, state transition probability, reward function, and discount factor. The discount factor determines the importance of future rewards relative to immediate rewards, and it is typically set to a value between 0 and 1. The transition probability describes how the environment changes in response to the actions taken by the agent (Y. Li 2018).

In an MDP, the goal of the agent is to find a policy that maximizes the expected long-term return, which is the discounted, accumulated reward received by the agent over time. To do this, the agent can use dynamic programming methods such as policy evaluation, value iteration, and policy iteration to find an optimal policy. These methods require knowledge of the model of the system, which specifies the state transition probabilities and reward function (Anthony et al. 2017).

In a mathematical sense, at each time step t, the agent receives a state s_t, selects an action at based on that state, and receives a scalar reward r_t based on the action taken. The agent then transitions to a new state s_{t+1} according to the environment dynamics. The goal of the agent is to learn a policy, which is a mapping from states to actions, that will maximize the long-term return, or the discounted, accumulated reward. The return at time t is defined as (Y. Li 2018, Mnih et al. 2013):

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3}$$

where γ is the discount factor, which is a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards.

To find an optimal policy, dynamic programming methods can be used when the system model is known. These methods include policy evaluation, which calculates the value or action value function for a given policy, and value iteration and policy iteration, which are used to find an optimal policy. In the absence of a model, RL methods can be used to directly learn a policy from interactions with the environment (Y. Li 2018).

In the case of cable laying, the state of the environment could be the position of the cable and the vessel, the weather conditions, and other relevant factors. The actions could be the movements of the vessel and the tension on the cable. The reward could be the progress made in laying the cable, with higher rewards for laying the cable more quickly and accurately. The RL algorithm would learn to take actions that maximize the reward by making predictions about the effects of different actions on the state of the environment.

4.2.2 Value function

The value function is a key concept in reinforcement learning. It is a function that estimates the long-term reward that an agent can expect to receive for being in a particular state, or for taking a particular action in a particular state (Y. Li 2018, Anthony et al. 2017, Mnih et al. 2013, Nagabandi et al. 2017, Feinberg et al. 2018).

In reinforcement learning, the value function represents the expected long-term return of an agent following a certain policy in a given state. The value of a state is the expected return starting from that state, and the value of a state-action pair is the expected return starting from that state, taking the specified action, and following the policy thereafter (Y. Li 2018, Mnih et al. 2013, Nagabandi et al. 2017, Feinberg et al. 2018).

The value function can be used to evaluate a given policy and determine whether it is optimal. A policy is considered optimal if it maximizes the expected return for the agent in all states. The value function can also be used to improve a given policy by finding a better action for each state (Y. Li 2018, Mnih et al. 2013, Nagabandi et al. 2017).

The value function for a state s under a policy π is defined as (Y. Li 2018, Anthony et al. 2017, Mnih et al. 2013, Nagabandi et al. 2017):

$$V_\pi(s) = \mathbb{E}_\pi[R_t | s_t = s], \tag{4}$$

where R_t is defined as in Equation 3 and is the return at time t, $\mathbb{E}_\pi$ is the expectation under the policy π, and s_t is the state at time t.

The value function for a state-action pair (s, a) under a policy π is defined as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_t | s_t = s, a_t = a], \tag{5}$$

where $\mathbb{E}_\pi$ is the expectation under the policy π, R_t is defined as in Equation 3 and is the return at time t, s_t is the state at time t, and a_t is the action at time t.

To calculate the value function, dynamic programming methods can be used when the system model is known. These methods include policy evaluation, which calculates the value or action value function for a given policy, and value iteration and policy iteration, which are used to find an optimal policy. In the absence of a model, RL methods can be used to directly learn a policy from interactions with the environment (Y. Li 2018, Mnih et al. 2013, Nagabandi et al. 2017).

In the cable installation problem, the state could represent the current location of the cable installation team, the actions could represent the different possible routes to the next installation location, and the transition model would indicate which locations can be reached from the current location. The reward could be the negative of the distance traveled, and the goal would be to find the shortest path by minimizing the total distance traveled. To solve this problem using RL, the agent could learn a policy that specifies the best action to take at each state (i.e., the route that will minimize the total distance traveled). The optimal value for each state would be the shortest distance from that state to the destination.

4.2.3 Modern RL non-exhaustive taxonomy

In modern Reinforcement Learning (RL), there are various algorithms that have been developed to solve different types of problems. Figure 13 shows a non-exhaustive taxonomy that covers some of the commonly used RL algorithms (OpenAI 2022c).

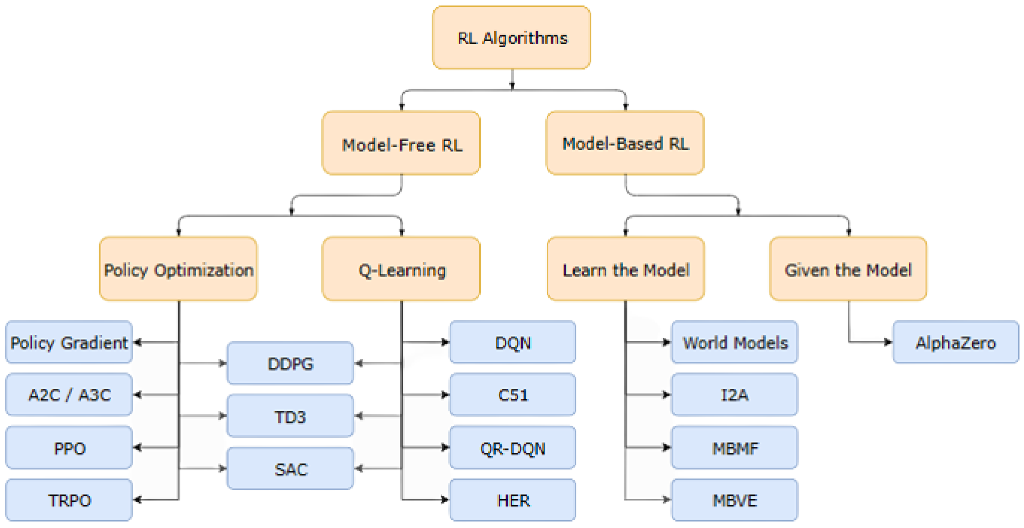


Figure 13: A non-exhaustive taxonomy of algorithms in modern RL, from (OpenAI 2022c)

Model-Free Algorithms:

Value-based methods in reinforcement learning (RL) involve estimating the value of different states or state-action pairs. One such algorithm is Q-Learning, which iteratively updates the action-value function using the Bellman equation. Another approach is Deep Q-Networks (DQN), where a deep neural network is used to approximate the action-value function. Additionally, Double Q-Learning is an algorithm that aims to mitigate overestimation bias present in traditional Q-learning (Y. Li 2018).

Policy-based methods directly optimize the policy to find the best action in each state. REINFORCE is an example of such an algorithm, which uses policy gradient methods to maximize the expected return. Proximal Policy Optimization (PPO) is another policy-based algorithm that updates the policy by optimizing a surrogate objective. Trust Region Policy Optimization (TRPO) is designed to ensure that the policy update does not deviate too far from the old policy.

Actor-critic methods combine elements of both value-based and policy-based methods. One such algorithm is Advantage Actor-Critic (A2C), which maintains both a value function (critic) and a policy (actor). Asynchronous Advantage Actor-Critic (A3C) is an extension of A2C that parallelizes multiple agents to speed up training.

Model-Based Algorithms:

Model-based algorithms in RL rely on having a learned model of the environment. Monte Carlo Tree Search (MCTS) is an example of a model-based algorithm that builds a search tree by sampling action sequences and estimating their values. Model-Predictive Control (MPC) uses the learned environment model to plan a sequence of actions over a finite horizon. Guided Policy Search (GPS) combines model-free and model-based approaches to learn a policy (Y. Li 2018).

Exploration algorithms play a crucial role in RL by balancing exploration and exploitation. Epsilon-

Greedy is a common exploration strategy that randomly selects actions with a certain probability, striking a balance between exploration and exploitation. Another approach is Upper Confidence Bound (UCB), which selects actions based on an upper confidence bound on their value estimates. Thompson Sampling employs a Bayesian approach to select actions based on posterior probability distributions.

Multi-agent RL involves multiple agents interacting and learning simultaneously. Independent Q-Learning is a simple approach where each agent learns independently without considering other agents. Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is an algorithm that uses a centralized critic and decentralized actors for multiple agents. Counterfactual Multi-Agent Policy Gradient (COMA) estimates the counterfactual value to improve multi-agent policies.

Please note that while this taxonomy provides an overview of various RL algorithms, it is important to acknowledge that there are many other specialized and hybrid algorithms that have been developed to address specific challenges in RL.

4.2.4 Exploration vs. exploitation

In reinforcement learning, exploration vs exploitation is the fundamental dilemma of whether to explore uncertain policies or exploit the current best policy. The exploration-exploitation trade-off is important because an agent needs to balance the need to learn about new actions and states in order to improve its policy with the need to exploit its current knowledge to maximize the expected reward (Y. Li 2018, Anthony et al. 2017, Lin 1993).

Exploration refers to the process of trying out new actions and states in order to gather more information about the environment. This can help the agent to learn more about the rewards and transitions that are possible in the environment, and to improve its ability to make informed decisions (Y. Li 2018, Anthony et al. 2017, Lin 1993).

On the other hand, exploitation refers to the process of taking the actions that are known to be the most rewarding, based on the information that the agent has already learned. This can help the agent to maximize its rewards in the short term, but it may also prevent the agent from discovering new, potentially more rewarding actions and states (Y. Li 2018, Anthony et al. 2017, Lin 1993).

One simple approach to dealing with this dilemma is the ε-greedy algorithm, where ε is a small value close to 0. In ε-greedy, the agent selects the greedy action (i.e., the action with the highest estimated value) with probability $1 - \varepsilon$, and selects a random action with probability ε (Y. Li 2018, Silver 2015, Sutton and Barto 2018). This means that the agent will exploit its current value function estimation with probability $1 - \varepsilon$ and explore with probability ε (Y. Li 2018, Anthony et al. 2017, Lin 1993).

4.2.5 Model-free vs. model-based RL

When the model of the system is unknown or not fully known, the agent can use model-free RL methods to learn an optimal policy. These methods do not require knowledge of the model, and instead learn directly from experience by trial and error. Model-free RL methods can be applied to both fully observable and partially observable environments. In a partially observable environment, the agent only has access to partial information about the state of the environment, and must use this information to make decisions (Nagabandi et al. 2017, Y. Li 2018, Nagabandi et al. 2017, Feinberg et al. 2018).

Model-free and model-based reinforcement learning are two different approaches to solving RL problems. Model-free RL refers to a class of algorithms that do not explicitly model the environment, and instead learn directly from the rewards and transitions that are observed during the learning process. Model-free RL is often faster and simpler to implement, as it does not require building a model of the environment. However, it can be less efficient, as the agent must learn from experience, and may need to explore the environment extensively in order to learn good policies (Y. Li 2018, Mnih et al. 2013, Nagabandi et al. 2017, Liang et al. 2017, Feinberg et al. 2018).

In contrast, model-based RL involves building a model of the environment, which is used to simulate different actions and predict their outcomes. This allows the agent to plan its actions in advance, and to take into account the long-term consequences of its actions. Model-based RL can be more efficient, as the agent can use its model to plan and evaluate different actions before actually taking them. However, it can be more complex to implement, and may require more computational resources (Y. Li 2018, Mnih et al. 2013, Liang et al. 2017, Feinberg et al. 2018).

Recent papers on model-based RL have proposed several methods that combine the benefits of model-free and model-based approaches. These methods use techniques such as temporal difference models, value prediction networks, and recursive tree structure neural networks to integrate model-free and model-based learning (Y. Li 2018). The methods have been applied to a variety of tasks, including continuous control tasks, 2D navigation tasks, and Atari games. Model-based RL offers the potential to improve the sample efficiency and performance of RL algorithms by leveraging knowledge of the environment dynamics to guide decision-making and learning (Y. Li 2018).

Overall, the choice between model-free and model-based RL depends on the specific problem that needs to be solved, and the trade-offs between efficiency, simplicity, and computational resources.

4.2.6 Temporal difference learning

Temporal difference (TD) learning is a fundamental method in RL. It involves using experience from the environment to learn and update estimates of the value function in a model-free, online, and fully incremental way. TD learning is a prediction problem that uses the TD error (i.e., the difference between the expected and observed values) to update the value function. It learns the value function directly from experience with the TD error. The update rule for TD learning is given by (Y. Li 2018, Mnih et al. 2013, Lin 1993):

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)], \tag{6}$$

where $V(s)$ is the current estimate of the value function for state s, r is the reward observed in state s, γ is the discount factor, and $V(s')$ is the estimated value of the next state s'. The parameter α is the learning rate, which determines how much the value function should be updated based on the TD error.

Q-learning is an off-policy control method that learns the action-value function $Q(s, a)$. It uses the following update rule to refine the policy greedily with respect to the action values:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \text{max}^{a'} Q(s', a') - Q(s, a)], \tag{7}$$

where $Q(s, a)$ is the current estimate of the action-value function for state s and action a, r is the reward observed in state s after taking action a, γ is the discount factor, $Q(s', a')$ is the estimated action-value for the next state s' and action a', and $\text{max}^{a'} Q(s', a')$ is the maximum value of $Q(s', a')$ over all possible actions a' in the next state s'. The parameter α is the learning rate, which determines how much the action-value function should be updated based on the TD error (Y. Li 2018, Mnih et al. 2013, Lin 1993).

TD learning is related to other RL methods such as Q-learning and SARSA. Q-learning is an off-policy control method that learns the action-value function and refines the policy greedily with respect to the action values. SARSA is an on-policy control method that learns the action-value function and updates the policy based on the current state and action. Both Q-learning and SARSA use bootstrapping, where the value function is estimated based on subsequent estimates, to enable online and continual learning (Y. Li 2018).

TD learning, Q-learning, and SARSA can converge to the optimal value function under certain conditions. From the optimal value function, an optimal policy can be derived. These methods have been widely used in various RL tasks and have proven to be effective in learning value functions and finding optimal policies (Y. Li 2018).

4.2.7 Policy optimization

Policy-based methods are a type of reinforcement learning that directly optimizes the policy, which is the function that determines the actions that the agent takes in different states. Policy-based methods optimize the policy by gradient ascent, updating the policy parameters by moving in the direction of the gradient of the policy. This is in contrast to value-based methods like TD learning and Q-learning, which estimate the value function and then use it to determine the optimal policy (Y. Li 2018, Anthony et al. 2017, Mnih et al. 2013, Nagabandi et al. 2017, Feinberg et al. 2018).

Policy-based methods have several advantages over value-based methods. They can learn stochastic policies, which are important in situations where the optimal policy is not deterministic. They are also effective in high-dimensional or continuous action spaces, and they have better convergence properties. However, they are also more likely to converge to local optima, they are less efficient to evaluate, and they have higher variance (Y. Li 2018, Mnih et al. 2013, Nagabandi et al. 2017, Feinberg et al. 2018).

In mathematical terms, policy-based RL methods optimize the policy function, $\pi(a|s;\theta)$, with respect to some set of parameters θ. This is typically done by using gradient ascent, which involves computing the gradient of the policy function with respect to θ and then updating θ in the direction of the gradient. The gradient of the policy function is given by the following expression:

$$\nabla_\theta \pi(a \mid s;\theta) = \pi(a \mid s;\theta) \nabla_\theta \log \pi(a \mid s;\theta) \tag{8}$$

Here, the gradient of the log probability of the policy is known as the score function or likelihood ratio. The policy gradient theorem states that the policy gradient can be computed as the expected value of the gradient of the log probability of the policy with respect to the expected return:

$$\nabla_\theta \pi(a \mid s;\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi(a \mid s;\theta) Q_{\pi_\theta}(s,a)] \tag{9}$$

REINFORCE is a popular policy-based RL algorithm. It updates the policy parameters in the direction of the policy gradient, which is given by the gradient of the log probability of the actions taken by the agent. This gradient is multiplied by the return, which is the sum of the rewards that the agent receives. In some cases, a baseline function is subtracted from the return to reduce the variance of the gradient estimate (Williams 1992, Y. Li 2018).

Overall, policy-based methods are an important type of RL algorithm that have unique advantages and disadvantages compared to value-based methods. They are effective in certain situations, but they require careful tuning and regular evaluation to ensure that they are performing well.

4.2.8 Deep deterministic policy gradient

Deep deterministic policy gradient (DDPG) is an actor-critic reinforcement learning algorithm that can be used to learn policies for continuous action spaces. It is an extension of the deterministic policy gradient algorithm, which is a model-free, off-policy algorithm for learning policies in such environments. This allows for faster learning and better convergence properties compared to REINFORCE (Y. Li 2018, Feinberg et al. 2018).

DDPG uses deep neural networks (DNNs) as function approximators for both the actor and the critic. The actor network is used to represent the policy, which maps states to actions. The critic network is used to estimate the action-value function, which represents the expected return for a given state and action (Y. Li 2018, Feinberg et al. 2018).

Mathematically, the actor network is parameterized by a set of weights θ^μ, and the critic network is parameterized by a set of weights θ^Q. At each iteration, a mini-batch of transitions (s_i, a_i, r_i, s_{i+1}) is sampled from the replay buffer and used to update the networks according to the following equations:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i^N \nabla_a Q(s,a|\theta^Q)|s = s_i, a = \mu(s_i)\nabla\theta^\mu \mu(s|\theta^\mu)|_{s_i} \tag{10}$$

$$\theta^\mu \leftarrow \theta^\mu + \alpha\nabla_{\theta^\mu} J \tag{11}$$

Here, $Q(s,a|\theta^Q)$ is the action-value function estimated by the critic network, and $\mu(s|\theta^\mu)$ is the policy represented by the actor network. The gradient of the critic network with respect to the action is computed using the Bellman equation, and the gradient of the actor network is computed using the chain rule (Y. Li 2018).

The critic network is also updated using a similar procedure, based on the Bellman equation:

$$\nabla_{\theta^Q} J \approx \frac{1}{N}\sum_i^N \left(r_i + \gamma Q(s_{i+1}, \mu(s_{i+1}|\theta^\mu)|\theta^Q) - Q(s_i, a_i|\theta^Q)\right) \nabla_{\theta^Q} Q(s,a|\theta^Q)|_{s_i,a_i} \tag{12}$$

$$\theta^Q \leftarrow \theta^Q + \beta\nabla_{\theta^Q} J \tag{13}$$

In both equations, α and β are the learning rates for the actor and critic networks, respectively. The discount factor γ is used to balance the importance of short-term and long-term rewards.

DDPG is an off-policy algorithm, meaning that it can learn from actions that were not selected by the current policy. This is useful because it allows the algorithm to learn from a large amount of experience without the need to explore the entire state space. However, this also means that the algorithm can potentially learn from suboptimal actions, so it is important to carefully tune the hyperparameters to ensure that the learning process is efficient and stable (Y. Li 2018, Feinberg et al. 2018).

During training, the agent collects experience by interacting with the environment, storing the observed transitions in a replay buffer. At each iteration, a mini-batch of transitions is sampled from the replay buffer and used to compute the gradient of the critic network with respect to the actor network's parameters. This gradient is then used to update the actor network, in order to improve the policy (Y. Li 2018, Feinberg et al. 2018).

Overall, DDPG is an effective algorithm for learning policies in continuous action spaces, and has been used to solve a wide range of RL tasks. It is particularly useful in cases where the action space is high-dimensional, as it can handle such spaces efficiently.

4.2.9 Deep Q-learning (DQN)

Deep Q-Network, or DQN, is a reinforcement learning algorithm that combines Q-Learning with deep neural networks to let RL work for complex, high-dimensional environments, like video games or robotics (OpenAI 2017). It approximates a state-value function in a Q-Learning framework with a neural network (Mnih et al. 2013).

This type of Q-learning works by approximating a state-value function in a Q-Learning framework with a neural network (Paszke and Towers 2023). In the case of Atari games, for example, several frames of the game are taken as input and the neural network outputs state values for each action1. The DQN architecture has two neural networks, the Q network and the Target network, and a component called Experience Replay. The Q network is trained to produce the optimal state-action value, while Experience Replay interacts with the environment to generate data to train the Q Network2.

DQN interacts with the environment through a continuous cycle of observation, action, and reward. At each time step, the agent observes the current state of the environment and chooses an action based on its policy. The environment then transitions to a new state and returns a reward that indicates the consequences of the action (Paszke and Towers 2023). The agent uses this information to update its policy and improve its performance over time.

The deep Q-Network updates its policy by using a variant of Q-Learning, which is a model-free reinforcement learning algorithm. In Q-Learning, the agent learns an action-value function that estimates the expected reward of taking a given action in a given state and following a fixed policy thereafter. It is also known as the Q-function, and it is denoted by $Q(s, a)$, where s is the state and a is the action (Foy 2021). The action-value function is updated at each time step using the Bellman equation (see Equation 14), which expresses the relationship between the value of a state and the values of its successor states (Paszke and Towers 2023).

$$Q(s, a) = r + \gamma \mathrm{max} Q(s', a') \tag{14}$$

where r is the immediate reward, γ is the discount factor, s' is the next state, and a' is the next action (Foy 2021).

In DQN, the action-value function is approximated using a neural network, which is trained to predict the expected value for each action, given the input state. The network is updated using experience replay, which stores the transitions that the agent observes and allows us to reuse this data later. By sampling from it randomly, the transitions that build up a batch are decorrelated, which has been shown to greatly stabilize and improve the DQN training procedure (Paszke and Towers 2023).

5 Software used for implementation and case scenario

5.1 Software used for implementation

5.1.1 OrcaFlex

OrcaFlex is a software program that is used for modeling and analyzing dynamic systems, such as offshore oil and gas facilities, pipelines, and ships. It allows users to simulate the behavior of these systems under various conditions, and analyze their performance using mathematical modeling and simulation techniques. This helps engineers and designers understand the behavior of their systems and make informed decisions about their design and operation (Orcina 2022a).

5.1.1.1 Documentation

In order to use and understand the OrcaFlex software, the documentation of the software is important. The documentation for OrcaFlex includes instructions on how to build a model, run the program, and extract results, as well as the underlying theory and technical notes. The documentation is available in the form of a dedicated OrcaFlex help browser and can also be downloaded or viewed online (Orcina 2022c).

5.1.1.2 Modelling and analysis

OrcaFlex is a software program for modeling and analyzing dynamic systems, such as offshore oil and gas facilities, pipelines, and ships. It uses mathematical modeling and simulation techniques to represent the behavior of the system under different conditions, such as different sea states, wave heights, and loads. This allows users to understand the behavior of their system and make predictions about its performance. It also makes it possible for the user to extract data, graphs, etc. for analysis (e.g bending radii, touchdown tension, stress/strain for the cable during the operation). OrcaFlex is commonly used in the offshore oil and gas industry to analyze the performance of structures and inform design and operational decisions (Orcina 2022c).

5.1.1.3 PythonAPI

Python is an object-oriented, dynamically typed scripting language that offers many advantages when used with the OrcaFlex API. It is interpreted, and therefore does not require a compile or link step before running, and the interpreter handles many programming tasks automatically, such as memory management and type casting. The Python interface to OrcaFlex is a wrapper to the OrcFxAPI DLL and simplifies the use of the C API by wrapping multiple function calls into a single function call and handling differences in data types. This makes the Python interface a good choice for developing pre and post-processing applications with OrcaFlex. (Orcina 2022b).

Flowcharts further explaining how OrcaFlex and Python are related can be seen in Figures 18, 19 and 20 in Chapter 6.

5.1.2 OpenAI gym

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a wide range of environments, from simple games to complex physical simulations, that can be used to train and evaluate RL algorithms. OpenAI Gym also includes a collection of benchmark tasks, as well as tools and utilities for evaluating and comparing the performance of different algorithms on those tasks. OpenAI Gym is designed to be flexible and modular, so that it can be easily extended and customized to support different types of RL environments and scenarios. It is developed and maintained by the research team at OpenAI, and is widely used by researchers and developers in the RL community (OpenAI 2022a).

5.1.2.1 Cartpole example

For this thesis the CartPole problem will serve as a baseline problem, where OrcaFlex will be used as the environment. The classic CartPole problem involves an agent learning to balance a pole on a moving cart by controlling the cart's movement left or right. The state of the system is represented by the position and velocity of the cart, as well as the angle and angular velocity of the pole. The agent's actions are to move the cart left or right, and the goal is to keep the pole balanced for as long as possible to maximize the reward. To solve the problem, the agent can use an RL algorithm such as DDPG to learn a policy that maps states to actions. The agent can then use this policy to make decisions and keep the pole balanced. To implement the CartPole problem in Python, a library or framework that supports dynamic systems and RL algorithms must be chosen, such as PyBullet, Gym, or TensorFlow (OpenAI 2022b, Barto et al. 1983, Surma 2018). A visual representation of the problem is given below:

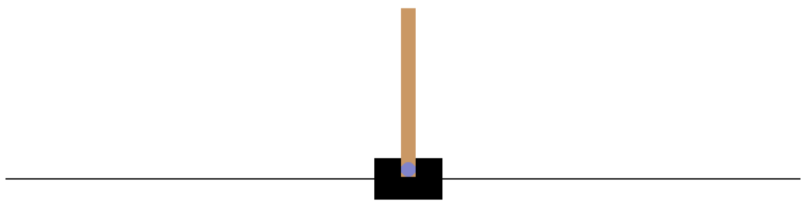


Figure 14: Classic CartPole problem, from OpenAI 2022b

5.2 Baseline CartPole problem in OrcaFlex

The baseline CartPole problem in OrcaFlex is an adapted version of the classical CartPole problem. In this problem, the cart is represented by a vessel, and the pole is represented by a 6D buoy that acts as a constrained pole (see Figure 15). The objective is to keep the pole upright for as long as possible, with a discrete action space with two actions:

- 0: push the vessel left
- 1: push the vessel right

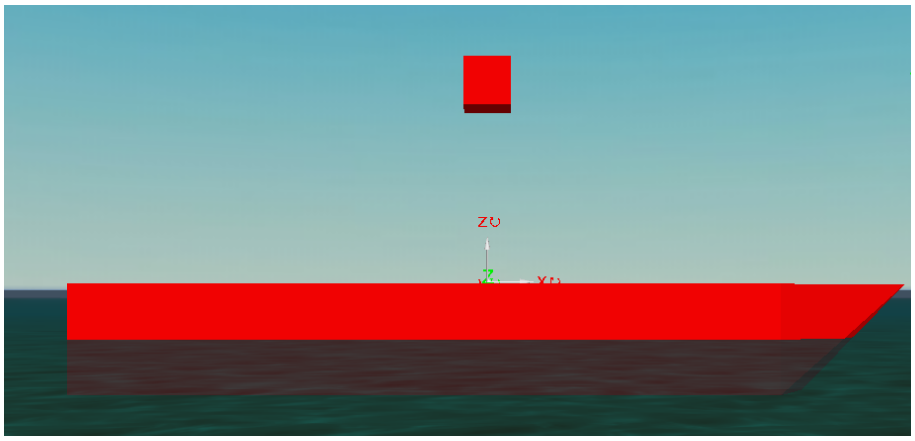


Figure 15: CartPole problem model in OrcaFlex

The observation space comprises four variables: vessel position, vessel velocity, pole angle, and pole angular velocity. Each variable has specific minimum and maximum ranges, as detailed in Table 1.

Num	Observation	Min	Max
0	Vessel Position	-20m	20m
1	Vessel Velocity	$-\infty$	∞
2	Pole Angle	$-40°$	$40°$
3	Pole Angular Velocity	$-\infty$	∞

Table 1: Observation Space of Baseline CartPole problem in OrcaFlex

The termination conditions for an episode are as follows: the vessel position exceeds the range of $(-10m, 10m)$, the pole angle is outside the range of $(-20°, 20°)$, or the maximum reward of 500 is reached. The system terminates when any of these conditions are met.

The goal of the agent is to maximize the duration for which the pole remains upright. The agent receives a reward of 1 for each step taken, and the episode concludes when one of the termination conditions is met.

By training an agent using reinforcement learning techniques, the objective is to develop an intelligent controller that can effectively balance the pole on the vessel for an extended period.

5.3 J-tube pull-in problem

A case scenario was provided by Nexans describing the steps of a J-tube pull-in into a jacket at North sea wind farm. The scenario can be divided into the following steps:

Preparation

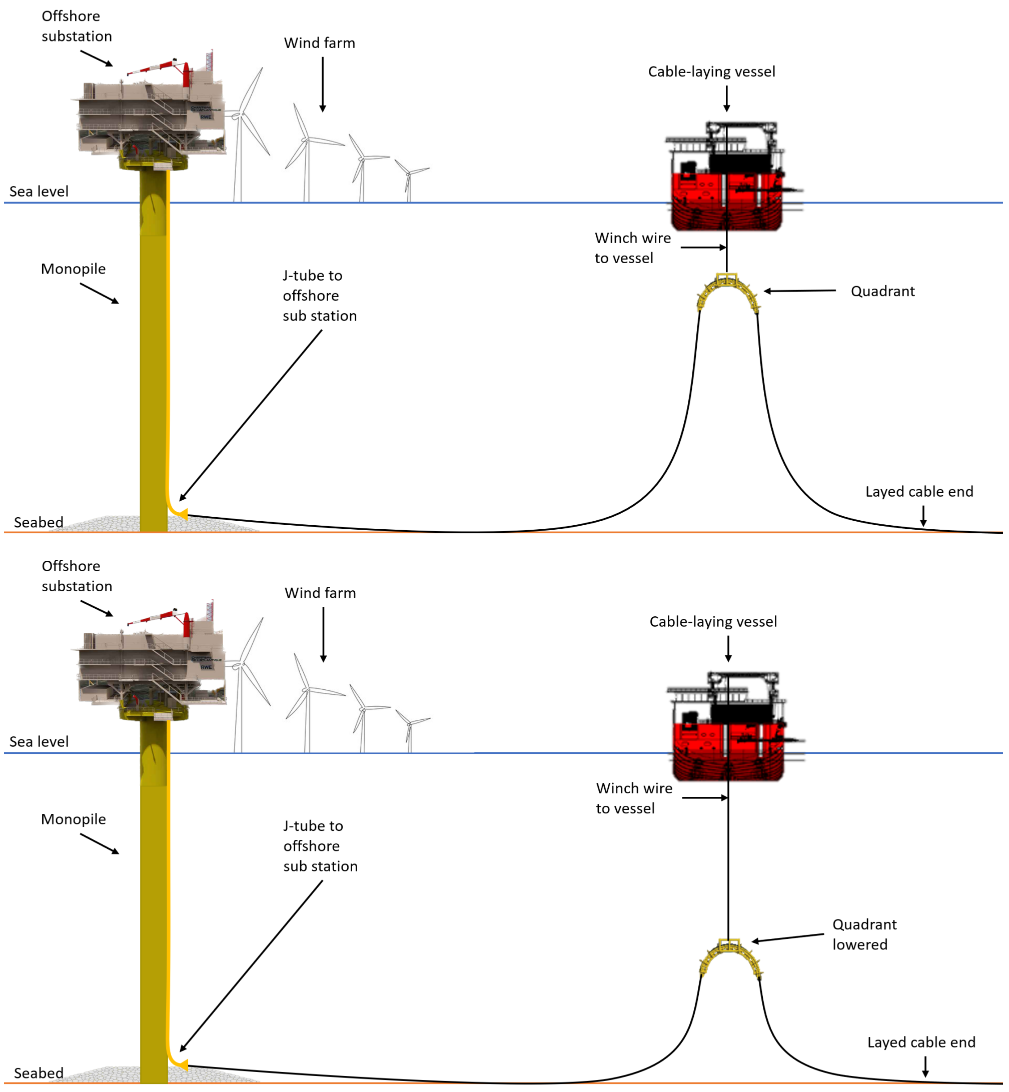
1. Pull the platform winch wire out of the J-tube and to a predetermined distance from the platform where the vessel will be during the procedure.

Procedure

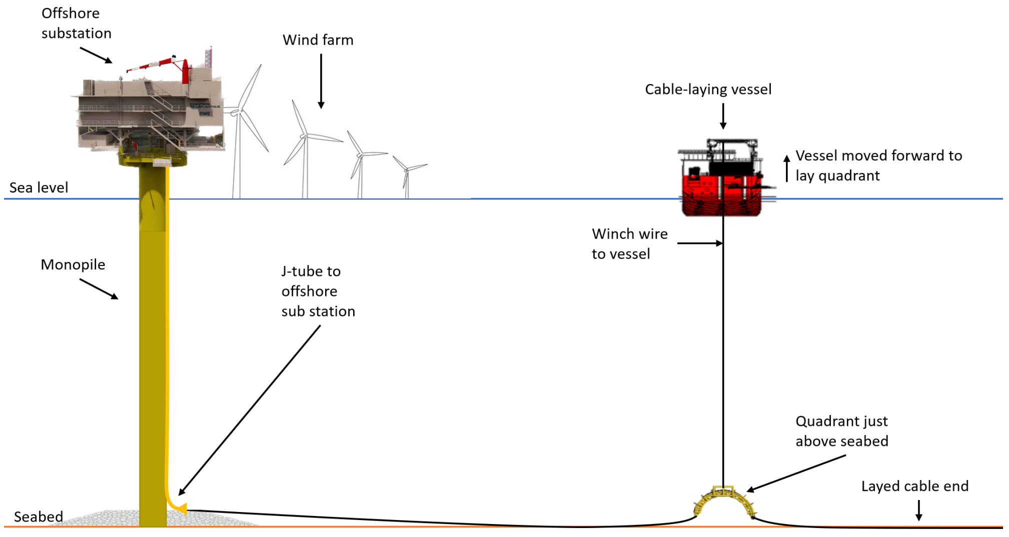
1. Lay the cable to a known reference point which is ”x” m away from the platform. There is a mark on the cable which should land on this reference point at the touchdown point. If the mark is closer to the platform than required the cable will be pulled up for a small distance and the overlength layed in an ”S” shape to take up the extra overlength.
2. Move the vessel forward and lay the rest of the cable on the seabed until the end is out of the cable carousel/turn table.
3. Route the cable end with the pull-in head around the quadrant and over the port laywheel or chute.
4. Move the vessel backwards and lower the pull-in head end to the seabed while recovering cable over the starboard laywheel.
5. When the pull-in head reaches the seabed, stop the vessel and connect the pull-in head to the winch wire using the ROV.
6. Start hauling in on the platform winch
7. Slide the quadrant aft along the deck. Both port and starboard cable will be paid out over their respective laywheels but the port side will be paid out faster because it is being hauled in by the platform winch.
8. Overboard the quadrant using an a frame or crane (stop the platform winch for this step)
9. Lower the quadrant to the seabed.
10. Just before the quadrant reaches the seabed. Move the vessel forward to lay the quadrant on the seabed.

The vessel is static for steps 5 to 9 and moves forward at the end of step 10 to lay the quadrant flat on the seabed. The main task is to have an agent to control the pay out rate of the quadrant in step 9 and 10, which is illustrated in Figure 16. Step 9 of the scenario involves lowering the quadrant to the seabed, while step 10 involves moving the vessel forward to lay the quadrant flat on the seabed. Proper installation and protection of the cable can increase its reliability, and reduce the need for costly repairs and maintenance. It is therefore important to carefully plan and execute these steps in order to determine the feasibility and safety of this operation.

In step 10, the vessel moves forward to lay the quadrant on the seabed. This movement will cause the cable to move and experience additional forces, which must be carefully considered to avoid exceeding the cable’s bending radius and damaging the cable. The vessel’s speed and acceleration must also be carefully controlled to avoid exceeding the cable’s maximum allowable tension and to ensure that the quadrant is properly laid on the seabed. A detailed analysis of the cable’s response to the vessel’s movement is necessary to ensure the safety and success of this operation.



(a) Step 9



(b) Step 10

Figure 16: Illustration of what happens in steps 9-10

Similarly to the baseline problem, the action space is discrete with two actions:

- 0: decrease the pay out rate
- 1: increase the pay out rate

The observation space however is different. The observation space is only determined by the touchdown tension of the cable and has to be within (tdt_min - tdt_max, tdt_max · 2), where:

- tdt_min = 0kN
- tdt_max = 2kN

The agent aims to maintain the touchdown tension within the specified bounds for as long as possible, receiving a reward of 1 for each step taken. An episode concludes when the threshold is reached or the maximum reward of 6000 is reached.

The overall system configuration, as depicted in Figure 17, includes the vessel, quadrant, cable, and winch wire. The agent's control objective is to manage the payout of the winch wire while maintaining the specified touchdown tension range. The pull-in rate of the cable is pre-determined at 0.1 m/s and remains constant throughout the procedure.

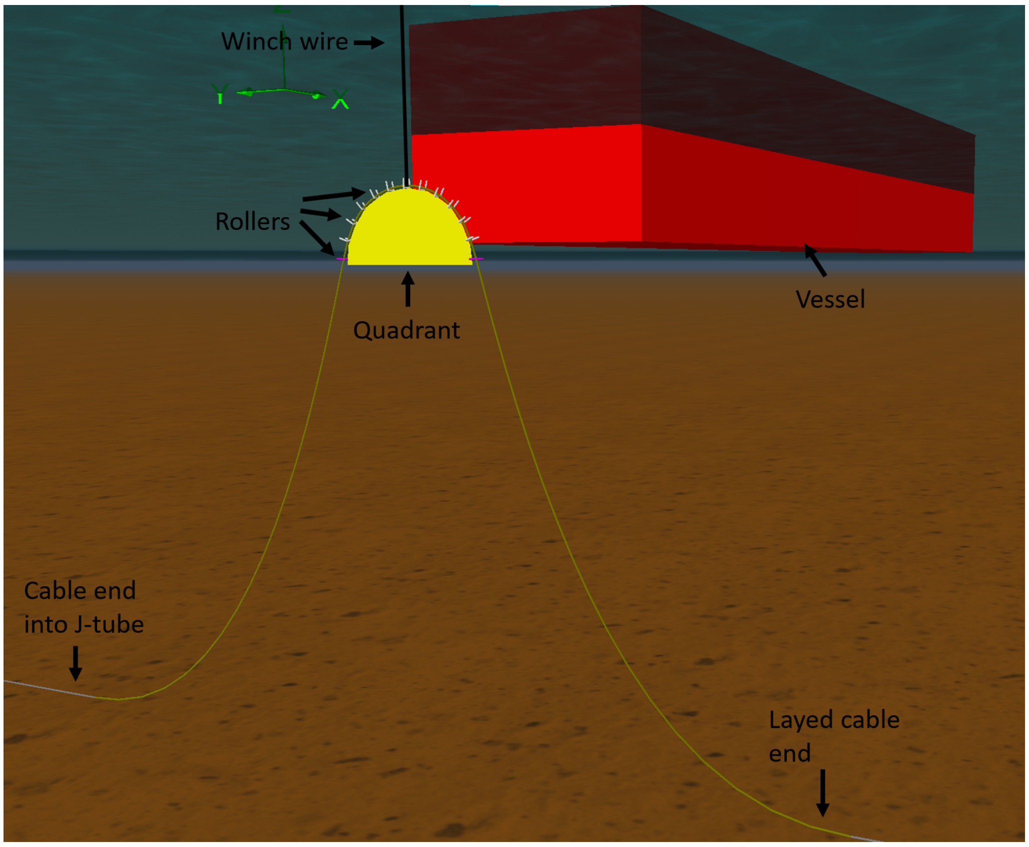


Figure 17: OrcaFlex model of J-tube pull-in

6 Implementation

6.1 Baseline CartPole problem in OrcaFlex

The high-level architecture of the workflow consists of two main components: the OrcaFlex model and the Neural Network, as shown in Figure 18. The OrcaFlex model represents the simulation model used for cable installations in OrcaFlex, capturing the dynamics and behavior of the cable installation process. The Neural Network refers to the neural network employed in Deep Reinforcement Learning (DRL) to optimize the OrcaFlex model. The workflow involves the exchange of information between these components. The OrcaFlex model provides the current state and reward information to the Neural Network, enabling it to learn and generate optimized actions. These actions are then fed back into the OrcaFlex model, influencing its decision-making and behavior during the cable installation process. By utilizing this architecture, the Neural Network learns to improve the parameter optimization for the installation analysis of marine cables, ultimately enhancing the overall performance and reliability of the process.

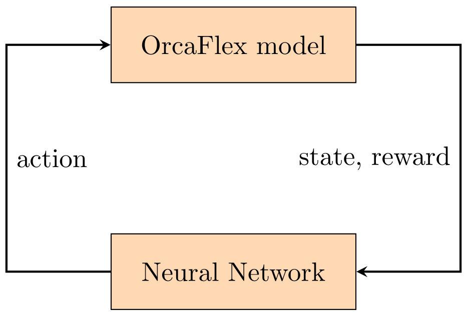


Figure 18: High level architecture of workflow

The lower-level architecture of the OrcaFlex model is depicted in Figure 19. The OrcaFlex model serves as the simulation framework for cable installations, capturing the dynamics and behavior of the process. It interacts with an external script named "ExternallyCalculatedVesselMotion.py" to calculate vessel motion, an important aspect of the installation analysis.

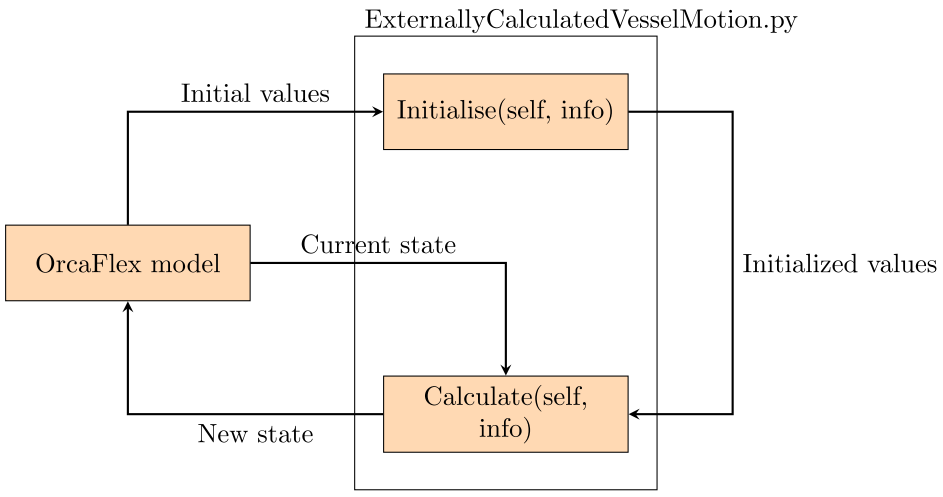


Figure 19: Lower level architecture for OrcaFlex model

The architecture involves two main functions within the OrcaFlex model. The "Initialise(self, info)" function initializes the model with initial values, receiving relevant information from an external source. These initial values are then passed to the "Calculate(self, info)" function, which performs calculations using the initialized values and additional information. This calculation process generates a new state, representing the updated state of the cable installation process for each time step.

The external script ”ExternallyCalculatedVesselMotion.py” plays a specific role in calculating vessel motion, which is an essential component of the cable installation process. By utilizing this lower-level architecture, the OrcaFlex model can effectively incorporate external calculations and generate accurate and up-to-date states for the installation analysis of marine cables.

The lower-level architecture for the neural network is illustrated in Figure 20. It consists of several key components that work together to facilitate the execution of the neural network and its interaction with the OrcaFlex model.

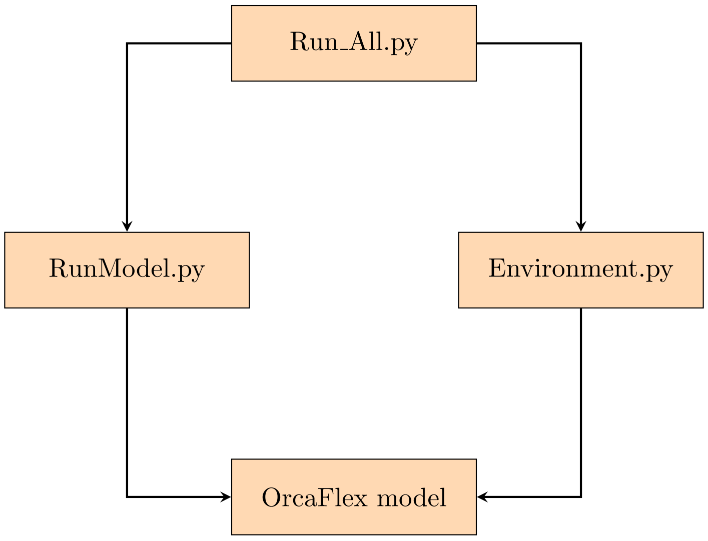


Figure 20: Lower-level architecture for neural network

The main script, ”Run_All.py,” serves as the control center for the overall workflow. It coordinates the execution of the different components involved in the process. It communicates simultaneously with ”Environment.py” and ”RunModel.py” to build both the OrcaFlex environment and the neural network model.

The ”Environment.py” script handles the OrcaFlex environment and serves as an interface between the neural network and the OrcaFlex model. It facilitates the exchange of information between the two components, allowing the neural network to send relevant actions to the OrcaFlex model and receive updated information or states from the model. This communication enables the neural network to interact with the simulated cable installation environment and learn from the feedback provided by the OrcaFlex model.

The ”RunModel.py” script is responsible for executing the OrcaFlex model concurrently with the rest of the code. It manages the execution of the OrcaFlex model and handles the saving of the simulations. It receives actions from the neural network through the ”Environment.py” interface and applies them to the OrcaFlex model, influencing its behavior during the cable installation process.

The OrcaFlex model itself represents the simulation model used for cable installations. It captures the dynamics and behavior of the cable installation process in OrcaFlex. It provides a rendering mode that allows for the visualization of episodes during the simulation, enabling monitoring and analysis of the cable installation process.

Within this architecture, the ”Run_All.py” script communicates with ”Environment.py” and ”RunModel.py” to build the necessary components and establish the interaction between the neural network and the OrcaFlex model. This interaction allows for the seamless integration of the neural network with the simulated cable installation environment, enabling learning and decision-making based on the behavior and dynamics of the OrcaFlex model.

By utilizing this lower-level architecture, the neural network can effectively interface with the OrcaFlex model, allowing for the optimization of the cable installation process through the application of Deep Reinforcement Learning techniques.

6.1.1 OrcaFlex model

The OrcaFlex model created to represent the CartPole problem successfully simulates the dynamics of the system using the components available in OrcaFlex. The default vessel serves as the cart, and a 6D buoy represents the pole that needs to be balanced. The 6D buoy is connected to the vessel via a constraint that allows rotation around the y-axis only, replicating the beam-like structure of the CartPole system.

By combining these components and applying the necessary constraints, the OrcaFlex model accurately represents the behavior of the CartPole system. This model can be used for various purposes, such as analyzing the system dynamics, developing control algorithms, or conducting simulation experiments within the OrcaFlex environment.

The visual representation of the CartPole problem model in OrcaFlex, as shown in Figure 21, provides a clear illustration of how the components are connected and how the dynamics of the system are simulated.

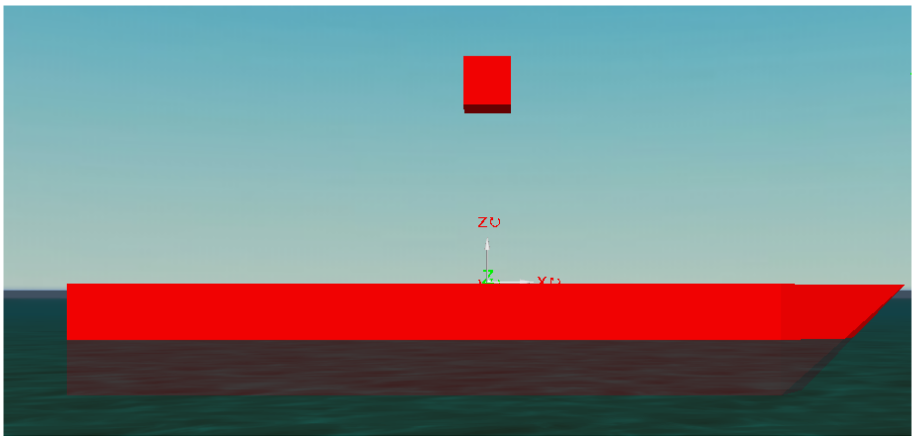


Figure 21: CartPole problem model in OrcaFlex

6.1.1.1 Vessel motion

The "ExternallyCalculatedVesselMotion.py" script is responsible for replicating the desired behavior of the vessel motion in the OrcaFlex model, similar to how the cart moves in the classic CartPole problem. It follows a specific flow of execution, as depicted in Figure 22.

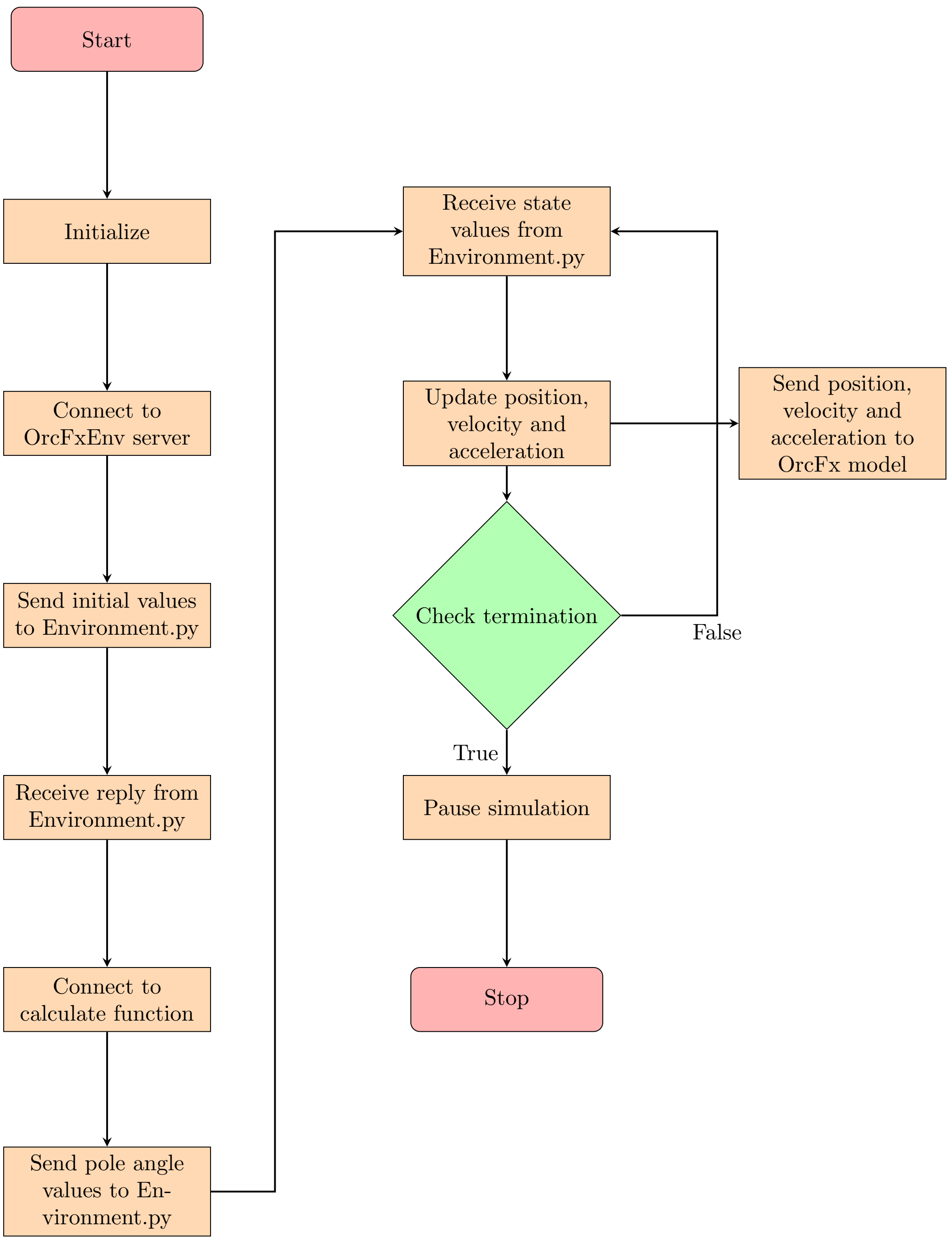

Start
Initialize
Connect to OrcFxEnv server
Send initial values to Environment.py
Receive reply from Environment.py
Connect to calculate function
Send pole angle values to Environment.py
Receive state values from Environment.py
Update position, velocity and acceleration
Send position, velocity and acceleration to OrcFx model
Check termination
False
True
Pause simulation
Stop

Figure 22: Flowchart for applying externally calculated motion to a vessel in OrcaFlex

The script uses Python and imports necessary libraries to perform the simulation. It contains two main methods: "Initialize" and "Calculate".

"ExternallyCalculatedVesselMotion.py" begins by initializing the necessary variables and establishing a connection with the OrcFxEnv server. It sends the initial values, such as velocity and acceleration, to the server and waits for confirmation to ensure successful initialization.

During each calculation step of the simulation, the script executes the "Calculate" method. It retrieves the current simulation time and checks if it is valid. If the time is valid, the script retrieves the constraint variable ”Ry,” which represents the pole angle, from the OrcaFlex model. This value, along with the simulation time, is sent to the server.

The server performs calculations based on the received values and computes updated position, velocity, acceleration, and a termination flag. These updated values are then received by the script.

Upon receiving the updated values, the script updates the corresponding parameters in the OrcaFlex model, allowing it to incorporate the externally-calculated vessel motion. It also checks if the termination condition is met based on the received values. If the condition is satisfied, indicating that the vessel motion has reached a threshold, the script pauses the simulation and prints relevant information.

In summary, the ”ExternallyCalculatedVesselMotion.py” script plays a crucial role in simulating vessel motion within the OrcaFlex model. It initializes the variables, establishes communication with the OrcFxEnv server, sends and receives motion-related values, and updates the motion parameters during the simulation

6.1.2 OpenAI gym custom environment

To accurately calculate vessel motion and enable the implementation of a neural network in the future, a custom environment called ”OrcFxCartPoleEnv” was created using the OpenAI Gym framework. This environment is designed to solve the CartPole problem, following the structure and conventions of the classic CartPole environment.

The OrcFxCartPoleEnv class consists of three functions: ”init()”, ”step()”, and ”reset()”.

During each step, the environment calculates the new state of the vessel and returns it to the OrcaFlex model. The OrcFxCartPoleEnv workflow is depicted in Figure 23.

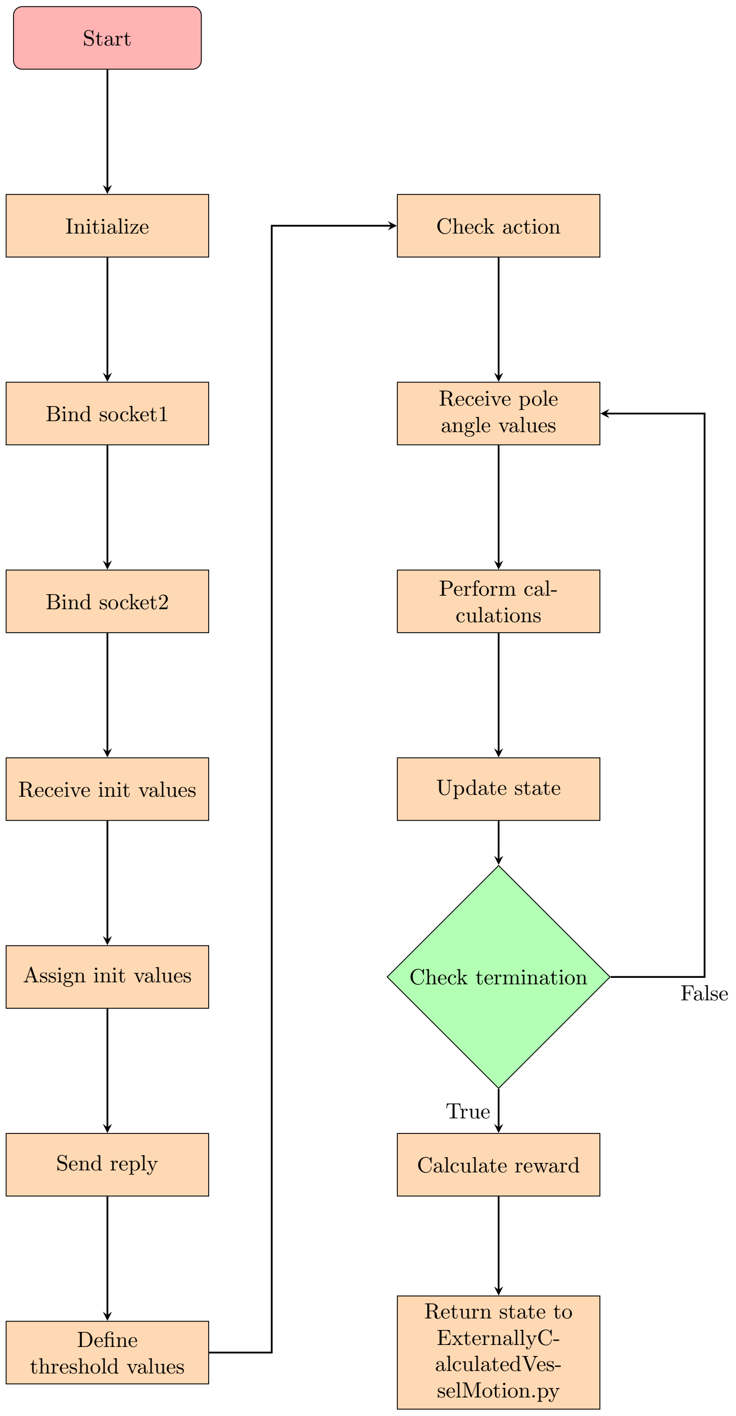


Figure 23: Flowchart for the OrcFxCartPoleEnv class

The "init()" function initializes the environment by setting up the necessary variables, establishing communication using ZeroMQ sockets, retrieving the episode number, and initializing state variables and parameters specific to the CartPole problem.

The "step()" function executes an action within the environment. It takes an action as input, performs calculations based on the current state and the received action, updates the state accordingly, and receives the pole angle (Ry) and simulation time (t) through a ZeroMQ socket. The calculations consider the dynamics equations of the pole, update the state using a chosen integration method, and check for termination conditions. A reward value is assigned based on the termination status, and the function returns the updated state, reward, termination flag, and any additional information.

The "reset()" function resets the environment to its initial state. It resets the internal state variables and can accept optional parameters if needed. The function returns the initial state of the environment.

Additionally, there is a "render()" method that handles visualization or displaying the current state of the environment. In this case, the visualization is done within the OrcaFlex simulation environment.

The OrcFxCartPoleEnv class enables the integration of the CartPole problem with the OpenAI Gym framework, allowing for the use of various reinforcement learning algorithms to solve the problem. It facilitates communication with the OrcaFlex simulation environment through ZeroMQ sockets, enabling dynamic simulations and advanced calculations.

6.1.3 Execution of model

To simulate each episode of the CartPole problem, the OrcaFlex model needs to be executed using the Python OrcaFlex API. This task is accomplished by a script called "RunModel.py", which is responsible for running the simulation and saving the desired results. The flow of the code is shown in Figure 24.

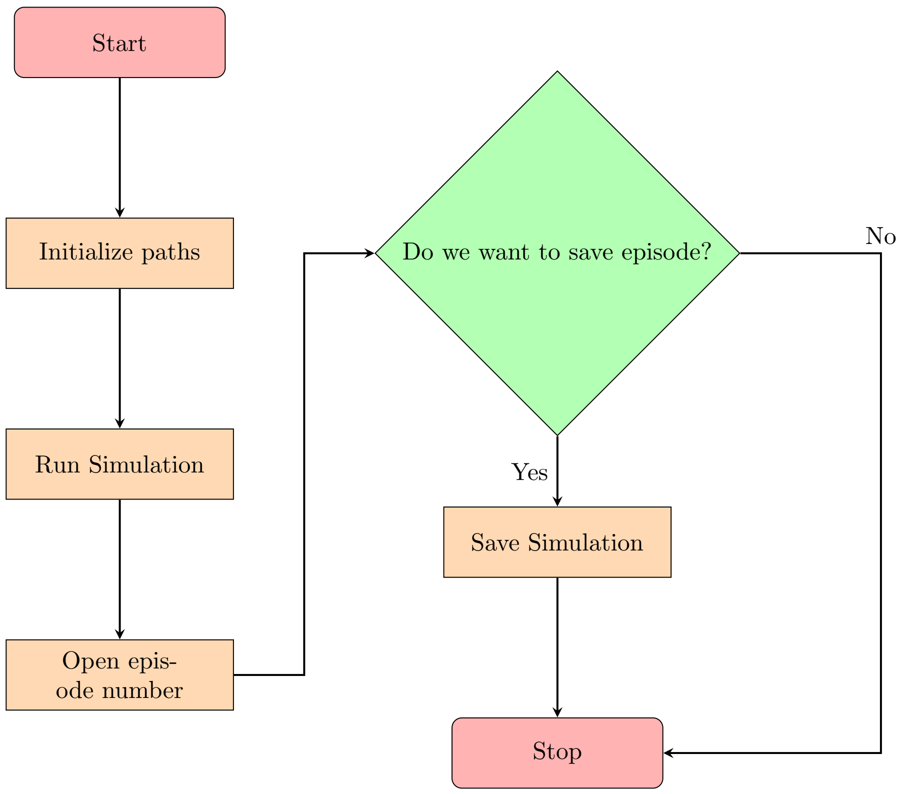


Figure 24: Flowchart for running and saving the OrcaFlex model

The code sets up the necessary paths, executes a simulation in OrcaFlex using a specified input file, reads the episode number from a text file, constructs a path for saving the simulation results based on the episode number, and stores the simulation data in the designated file. This allows for further analysis, evaluation, or visualization of the CartPole problem within the OrcaFlex environment.

6.1.4 Execution of whole simulation

The "Run_All.py" script integrates the OrcaFlex model, vessel motion calculation, environment setup, and execution into a comprehensive simulation. It offers two variations: "Run_All_Random.py" and "Run_All_DQN.py", representing different approaches to running the simulation.

1. "Run_All_Random.py": This variation of the script executes the simulation using random actions, either left or right, without any specific decision-making algorithm or agent. It serves as a baseline or comparison for evaluating the performance of other approaches. The vessel motion is calculated randomly, and the simulation proceeds based on these random actions.
2. "Run_All_DQN.py": This variation incorporates a neural network-based agent using the Deep Q-Network (DQN) algorithm. The DQN agent learns from the environment and makes decisions based on its learned knowledge to achieve better performance. It utilizes a neural network model and reinforcement learning techniques to optimize its decision-making process. The agent takes into account the current state of the environment, such as the vessel's position and velocity, and uses the neural network to determine the optimal action to take. This approach enables the agent to learn and improve its performance over time through interactions with the environment.

Both variations of the script enable the execution of the complete simulation, including vessel motion calculation, interaction with the OrcaFlex model, and decision-making based on either random actions or the DQN algorithm. These scripts provide a framework to evaluate and compare different approaches to solving the CartPole problem and assess the effectiveness of the DQN-based agent.

6.1.4.1 Random action

The random actions refer to the approach of selecting actions (either left or right) randomly without any specific decision-making algorithm or strategy. This approach directly replicates the concept of random games in the classic CartPole problem (Pylessons 2019).

In the classic CartPole Random Games, the agent takes random actions without considering the current state or any learned knowledge. These actions are chosen purely based on chance, simulating a random decision-making process. The purpose of this approach is to establish a baseline performance by comparing it with other more sophisticated strategies or algorithms.

By building the "Run_All_Random.py" script in the same way as the classic CartPole Random Games, it ensures that the random actions are implemented using a similar methodology, enabling fair and accurate comparisons of performance across different approaches. See Figure 25 for code flow.

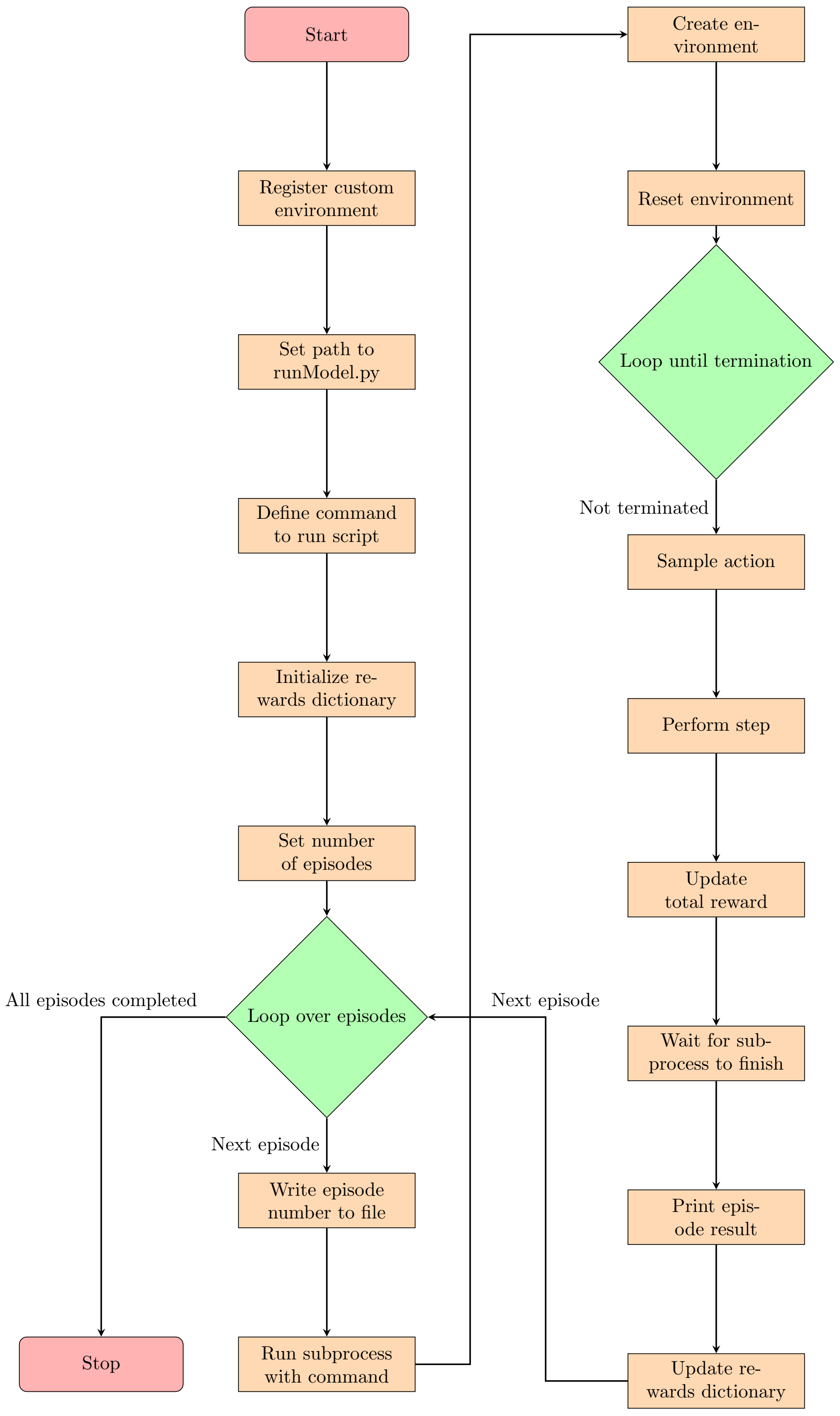


Figure 25: Flowchart for running the OrcFxCartPole-v0 environment

In summary, the code registers a custom Gym environment, runs the environment for a specified number of episodes, accumulates rewards, and stores the rewards in a dictionary. It utilizes subprocesses to run the external environment simulation and communicates with the environment using the reset() and step() methods.

6.1.4.2 DQN

The Deep Q-Network (DQN) agent, implemented by Pylessons to solve the CartPole problem in OrcaFlex, serves as a reference and source of inspiration (Pylessons 2019). Pylessons successfully applied the DQN algorithm, a reinforcement learning technique, to solve the classic CartPole problem.

Adapting and implementing the DQN agent for the CartPole problem in the OrcaFlex environment involved modifying the agent to account for vessel motion, integrating it with the OrcaFlex model, and appropriately defining the state, action, and reward spaces.

By leveraging insights from Pylessons' implementation and adapting the DQN agent to the specific requirements of the OrcaFlex simulation, the goal was to develop an effective agent capable of solving the CartPole problem in this context.

The flowchart in Figure 26 illustrates the steps involved in the CartPole problem with the DQN agent.

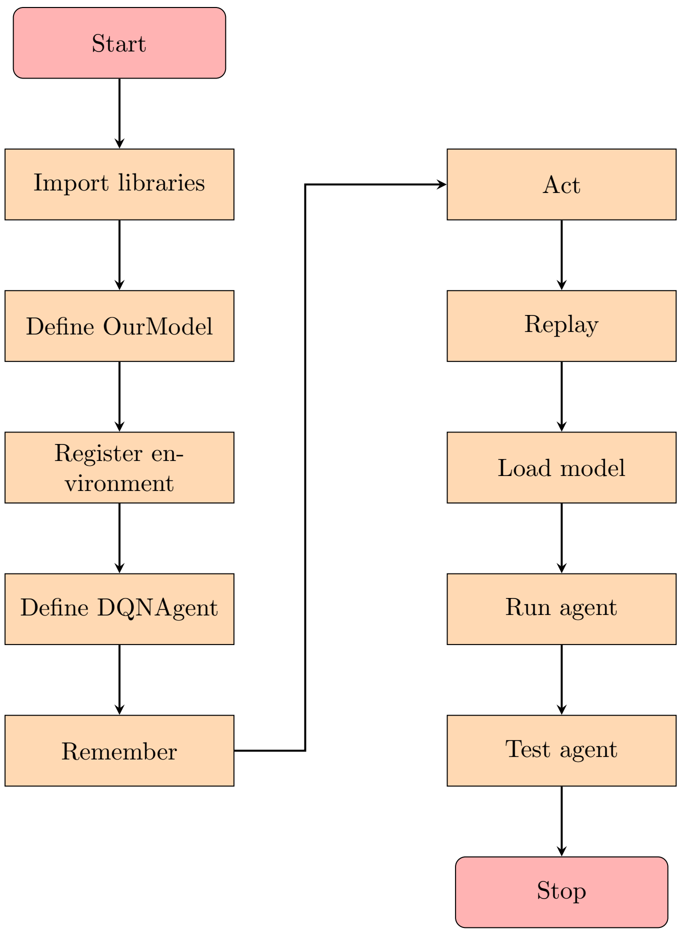


Figure 26: Flowchart CartPole problem with DQN agent

Creating a DQN agent involves defining a loss function that represents the gap between our prediction and the target. The loss function incorporates the reward obtained from an action, the maximum predicted future reward discounted by a gamma value, and the current prediction (Pylessons 2019). By subtracting the prediction from the target and squaring the result, we can punish large losses and treat negative values as positive ones. Keras simplifies this process by automatically handling the calculation of the loss during training. The learning rate, which determines how much the neural network learns in each iteration, is also taken care of by Keras.

The DQN algorithm includes the "remember" and "replay" methods, which are important for the agent's learning process. These methods are used to store and retrieve experiences in memory, enabling the agent to learn from past interactions. Although the original DQN design has various tweaks, a simplified version is often preferred for better understanding.

In reinforcement learning, several crucial parameters need to be configured for the DQN agent to optimize its performance. These parameters encompass the number of games or episodes the agent will partake in, the discount rate (gamma) employed to calculate future discounted rewards, the exploration rate (epsilon) governing the frequency of random action selection versus predicted actions, the rate at which epsilon decays to gradually reduce exploration (epsilon_decay), the minimum value for epsilon (epsilon_min), the learning rate of the neural network (if applicable), and the batch size utilized for training the DQN.

These parameters play a vital role in shaping the agent's learning process and can be fine-tuned based on the specific requirements of the problem at hand. It is worth noting that as the agent becomes more proficient in playing games, the objective is to gradually decrease the number of exploratory actions by adjusting the exploration rate (epsilon) accordingly.

Within the provided code, the DQN agent initializes these parameters and sets the maximum memory size to 2000, along with default values for gamma, epsilon, epsilon_min, epsilon_decay, batch_size, and train_start. By carefully adjusting these parameters, researchers and developers can optimize the DQN agent's learning and decision-making capabilities to achieve superior performance in the given task. The exact values are taken from Pylessons DQN agent and are set to (Pylessons 2019):

memory	=	2000
gamma	=	0.95
epsilon	=	1.0
epsilon_min	=	0.001
epsilon_decay	=	0.999
batch_size	=	64
train_start	=	1000

The implementation used the Gymnasium library to connect the DQN agent with the OrcFxCartPole environment. The DQN agent is a neural network-based algorithm for reinforcement learning. The implementation included defining a neural network model, registering the environment, initializing the DQN agent, storing experiences in memory, selecting actions based on an epsilon-greedy strategy, training the agent through replaying experiences, and saving and loading the trained model. The agent's performance was evaluated by running episodes and printing the scores.

6.2 J-tube pull in problem

In the J-tube pull-in scenario, the objective is to control the payout rate of the a-frame winch wire during the operation. The architecture of the solution follows a similar pattern as the CartPole problem in OrcaFlex, explained in the previous section, utilizing the principles of reinforcement learning and the DQN agent. However, there are notable distinctions in terms of the OrcaFlex model and the specific control parameters.

The OrcaFlex model was built to simulate the dynamic characteristics of a cable being laid through a J-tube during the pull-in scenario. It incorporates multiple variables, including cable tension, vessel motion, and the interaction between the cable and the J-tube. By considering these factors, the model accurately simulates the behavior of the cable as it is pulled into the J-tube using a quadrant lowered by a winch with a payout rate controlled by an agent. The model offers valuable insights into the cable tension and other pertinent states, which can be effectively utilized by the DQN agent to make informed decisions.

In the J-tube pull-in scenario, unlike the CartPole problem, the focus shifts to controlling the payout rate of the winch wire connected to the quadrant instead of the vessel's position. The payout rate determines the speed at which the cable is hauled-in or paid-out during the pull-in operation. By adjusting the payout rate, the agent aims to achieve an optimal residual tension that ensures the minimum bend radius or minimum tension limits are not violated. The threshold for this scenario is determined by the touchdown tension of the cable being laid, representing the acceptable range of cable tension during the operation. It is typically set between 0 kN and 2 kN to avoid excessive tension or slack. The agent's objective is to control the payout rate to maintain the tension within this threshold while achieving the desired cable laying objectives.

Throughout the training process, the DQN agent actively engages with the OrcaFlex model, constantly monitoring the system's present state, which includes factors like cable tension and other pertinent variables. Based on the observed state, the agent determines the payout rate, with the objective of maximizing the reward signal while ensuring adherence to the tension threshold constraint. For each timestep that the touchdown tension remains within the limits, the agent receives a reward of 1. To facilitate replay and training, the agent stores its experiences, encompassing the observed states, chosen actions, obtained rewards, and subsequent states, in memory.

The DQN agent's training loop involves iterating over episodes, each comprising multiple steps. At each step, the agent selects an action (the payout rate) based on the observed state, following an exploration-exploitation trade-off guided by the epsilon-greedy strategy. The action is then applied to the OrcaFlex model, and the resulting state, reward, and termination information are obtained. The agent uses these experiences to update its Q-value estimates and improve its policy over time.

In summary, the J-tube pull-in scenario in OrcaFlex employs a similar architecture to the CartPole problem, but with a different OrcaFlex model and control objective. The agent's task is to control the payout rate of the winch wire, aiming to maintain the cable tension within the acceptable range of 0 kN to 2 kN. Through the reinforcement learning process, the agent learns to make optimal decisions based on observed states and rewards, ensuring successful cable laying while respecting the tension threshold constraint.

6.2.1 Pay-out rate of quadrant winch wire

The J-tube Pull-in Script is designed specifically for controlling the tension of a winch wire during a J-tube pull-in operation in OrcaFlex. It showcases the application of externally-calculated payout rate to maintain the tension within a specified threshold range. The script utilizes variables like payout rate, force magnitude, state, and thresholds for tension.

In terms of the OrcaFlex model, the script focuses on simulating the dynamics of the winch wire, J-tube, and cable tension. It takes into account factors such as the interaction between the cable and the J-tube, as well as the forces involved in the pull-in operation. The model provides feedback on the cable tension and other relevant states, which the script's functionality relies on for decision-making. Note that the J-tube was not modeled here for simplicity.

Communication with the OrcaFlex environment is established using the ZeroMQ library, enabling data exchange between the Python script and the simulation. The script's external function class, "ExternallyCalculatedWinchPayOutRate," initializes the necessary variables and facilitates communication with OrcaFlex. It receives tension data and controls the payout rate of the winch wire accordingly.

The termination condition for the J-tube Pull-in script is based on the received touchdown tension reaching a specified threshold. Once this threshold is exceeded, the script terminates the pull-in operation.

Overall, the J-tube Pull-in script is tailored to the specific task of controlling the tension of a winch wire during a J-tube pull-in operation. It employs an OrcaFlex model that simulates the dynamics of the relevant components and utilizes externally-calculated payout rate to maintain tension within a specified range.

6.2.2 Environment

The Environment.py script is similar to the Environment.py for the CartPole proble. While they have similarities in terms of their overall structure and functions, their specific calculations and parameters are tailored to their respective environments.

The step() function implemented is taking an action as input and returning the next state, reward, termination status, and additional information. However, the calculations within the step function differ based on the specific environment being modeled. For the J-tube pull-in the step calculates the pay-out rate of the winch wire, instead of vessel motion.

In summary, OrcFxCartPoleEnv and OrcFxEnv are distinct implementations of different environments with specific dynamics and variables. While they share similarities in structure and functionality, their calculations and parameters are customized to suit their respective environments.

6.2.3 Execution of model

The runModel.py script for both cases, CartPole problem and J-tube pull-in problem, shares the same code structure and functionality. The only difference lies in the specific paths used within the script. The main purpose of the runModel.py script is to execute the and save the simulation.

6.2.4 Execution of whole simulation

Similarly to the CartPole problem, the Run_All.py script for the J-tube pull-in scenario follows the same setup with two variations: Run_All_Random.py and Run_All_DQN.py. These variations represent different approaches to running the simulation.

In Run_All_Random.py, the simulation is conducted using random actions and does not incorporate any specific decision-making algorithm or agent. This approach serves as a baseline for comparison, allowing the evaluation of other strategies or algorithms.

The ”Run_All_DQN.py” script implements a Deep Q-Network (DQN) agent, a powerful algorithm based on neural networks. The DQN agent leverages the capabilities of neural networks and reinforcement learning to learn from the environment and make informed decisions in the J-tube pull-in scenario. The hyperparameters used in this implementation are set to the same values as those employed in the CartPole problem:

$$\begin{array}{ccc}
\text{memory} & = & 2000 \\
\text{gamma} & = & 0.95 \\
\text{epsilon} & = & 1.0 \\
\text{epsilon\_min} & = & 0.001 \\
\text{epsilon\_decay} & = & 0.999 \\
\text{batch\_size} & = & 64 \\
\text{train\_start} & = & 1000
\end{array}$$

By having these two variations, the Run_All script enables the exploration and evaluation of different approaches to the J-tube pull-in simulation. It allows for a comprehensive analysis of the performance and effectiveness of both random actions and the more advanced DQN-based agent in this specific context.

7 Experiment and results

7.1 Baseline CartPole problem

The objective of this experiment was to solve the CartPole problem in OrcaFlex by training a Deep Q-Network (DQN) agent. The goal for the agent was to obtain a reward of 500, with a reward of 1 given for each time step (0.1s) in which the agent successfully balanced the pole and kept it within the frame. If the pole angle exceeded the ±20 degrees angle limits or if the vessel moved outside the ±10m frame limits, the simulation for that episode was terminated, and the next episode was executed. This approach ensured that the agent focused on learning successful strategies and avoided training on episodes where failure occurred.

By imposing these constraints, the experiment aimed to assess the agent's ability to learn and maintain stability throughout the simulation duration. The training process involved multiple episodes, during which the agent learned from its experiences and adjusted its actions to maximize the cumulative reward. The agent's performance was evaluated based on the achieved rewards, specifically aiming for rewards of 500 to indicate successful solution of the CartPole problem.

7.1.1 Results and discussion

Analyzing the results obtained from the training process, a consistent improvement in the agent's performance as indicated by the graph of the learning rate can be observed. The total reward achieved by the agent during every 20th episode of training is depicted in Figure 27, with the x-axis representing the number of episodes and the y-axis representing the total reward. The graph illustrates how the agent's total reward gradually increased over time, reaching a maximum reward of 500.

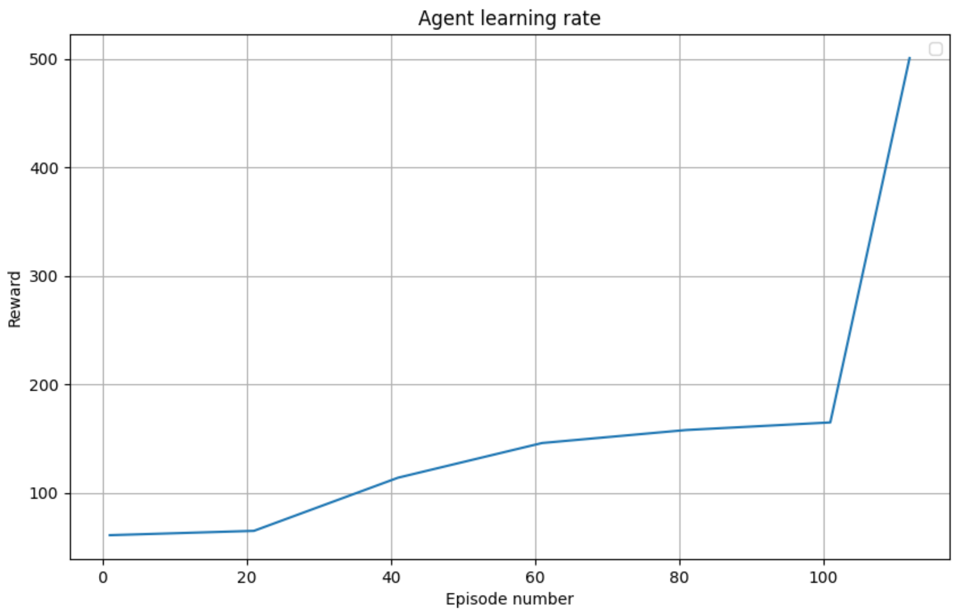


Figure 27: Agent learning rate for CartPole problem

Upon closer examination of the data points, it becomes apparent that the agent's reward consistently increased throughout the training process. In the initial episodes, the agent achieved a total reward of 61, indicating a relatively low level of proficiency. However, as the training progressed, the agent's performance improved. By episode 112, the agent achieved its highest reward of 501, marking a significant peak in its learning curve.

This learning rate pattern could be attributed to the reinforcement learning algorithm employed during the training process. Reinforcement learning algorithms typically involve a trial-and-error process, where the agent explores different actions and learns from the resulting rewards. As the agent receives feedback in the form of rewards, it adjusts its behavior to maximize future rewards. This iterative learning process gradually refines the agent's decision-making and leads to improved performance over time.

The parameters used in the training process, such as the number of episodes, discount rate (gamma), exploration rate (epsilon), learning rate, and batch size, all play crucial roles in the agent's learning progress. The specific values chosen for these parameters greatly influence the rate of learning and the final performance of the agent. These values were set as the same as for the DQN agent solving the classic CartPole problem presented by Pylessons (Pylessons 2019).

Overall, the new results suggest that the applied reinforcement learning algorithm successfully enhanced the agent's proficiency in solving the CartPole problem. The increasing reward trend followed by a significant peak at episode 112 demonstrates the agent's ability to learn from its experiences and adapt its actions to achieve better outcomes.

In order to gain a comprehensive understanding of the training process and the agent's performance, Figure 28 presents a graph illustrating the vessel movement during every 20th training episode. The graph provides a visual representation of the agent's ability to control the vessel's motion and maintain stability throughout the training process. By analyzing the vessel's movement over time,the agent's learning progression and its capacity to optimize critical parameters related to cable installation can be examined.

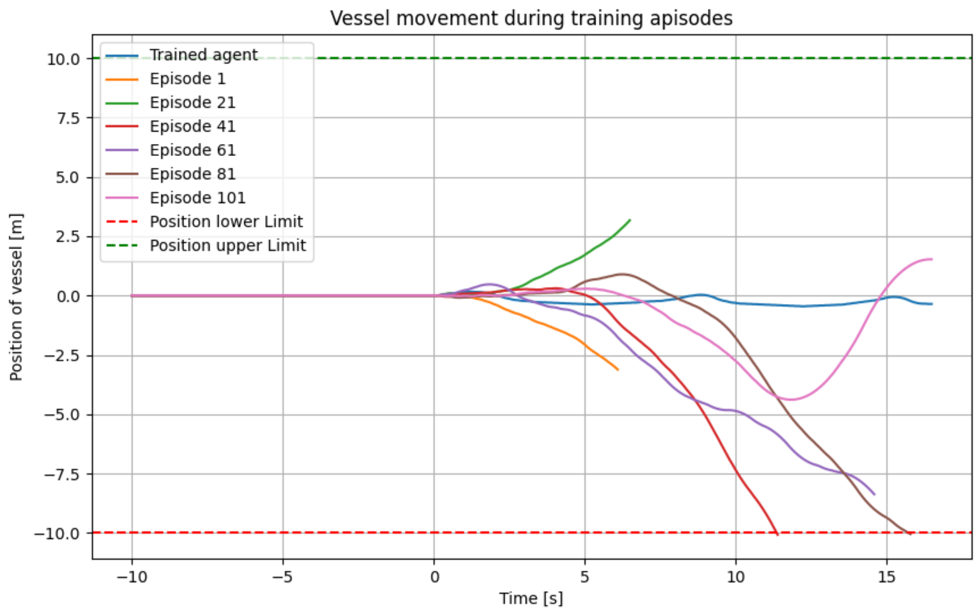


Figure 28: Vessel movement of every 20th episode of during training process

The analysis of the vessel movement graph reveals significant variations among the training episodes. Notably, episodes 41 and 81 were terminated due to the pole angle reaching the lower limit. This suggests that the agent encountered challenges in maintaining balance during these specific episodes. It is possible that the agent made aggressive or incorrect movements, leading to the pole angle exceeding the acceptable threshold. On the other hand, the completely trained agent demonstrates much greater stability, with minimal deviation from the vertical position. This indicates that the agent has learned to perform more controlled and measured movements, resulting in a steady vessel movement.

Examining the remaining training episodes, it becomes apparent that the vessel movement generally follows a consistent pattern. The episodes tend to start moving in a particular direction and maintain that direction throughout the duration. This suggests that the agent initially adopts a specific strategy or movement pattern to maintain balance. However, it is important to note that there can still be variations and fluctuations in the vessel movement during these episodes, indicating ongoing adjustments and adaptations by the agent to optimize its performance.

Turning to the pole angle results (Figure 29), the remainder of the episodes (1, 21, 61, 101) were terminated by the pole angle threshold. This implies that the agent faced difficulties in keeping the pole within the acceptable range during these instances. The agent's movements during these episodes may not have been effective in countering the pole's dynamics, resulting in excessive deviation from the vertical position.

It is also insightful to examine the reasons behind the failure of each training episode. To gain a better understanding, an analysis of the pole angle for every 20 episodes of the training process were also conducted. Figure 29 represents the pole angle during the training episodes.

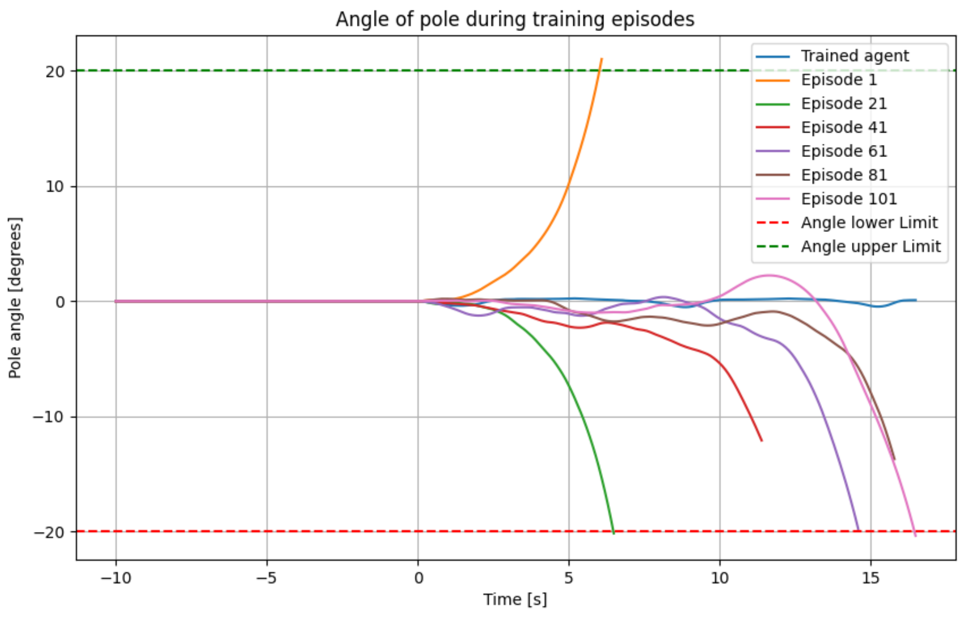


Figure 29: Angle of pole for every 20th episode of during training process

In the early episodes, where the agent reached the pole angle limits, it struggled to maintain the pole within the acceptable range (see episode 1 and 21). However, as the training progressed, the agent exhibited increased steadiness and stability, keeping the pole angle closer 0 for a longer period. This indicates that the agent learned to balance the pole more effectively over time. The ability to keep the pole angle near 0 for longer durations suggests that the agent has acquired better control and understanding of the dynamics involved in maintaining the pole's upright position.

Overall, these results illustrate the learning progress of the agent throughout the training process. The termination of episodes due to pole angle limits in the early stages highlights the initial challenges faced by the agent. However, as the training advanced, the agent demonstrated enhanced stability and improved control over the pole angle. This improvement is evident in the more consistent vessel movement and the agent's ability to maintain the pole angle around 0 for extended periods.

To gain a deeper understanding of the agent's behavior, Figure 30 illustrates the pole angle over time when applying the agent to the environment, with the ±20 degrees limits marked. This graph provides valuable insights into how the agent controls the vessel's motion to prevent the pole from exceeding the set angle limits, showcasing its ability to maintain stability and balance the pole within the desired range.

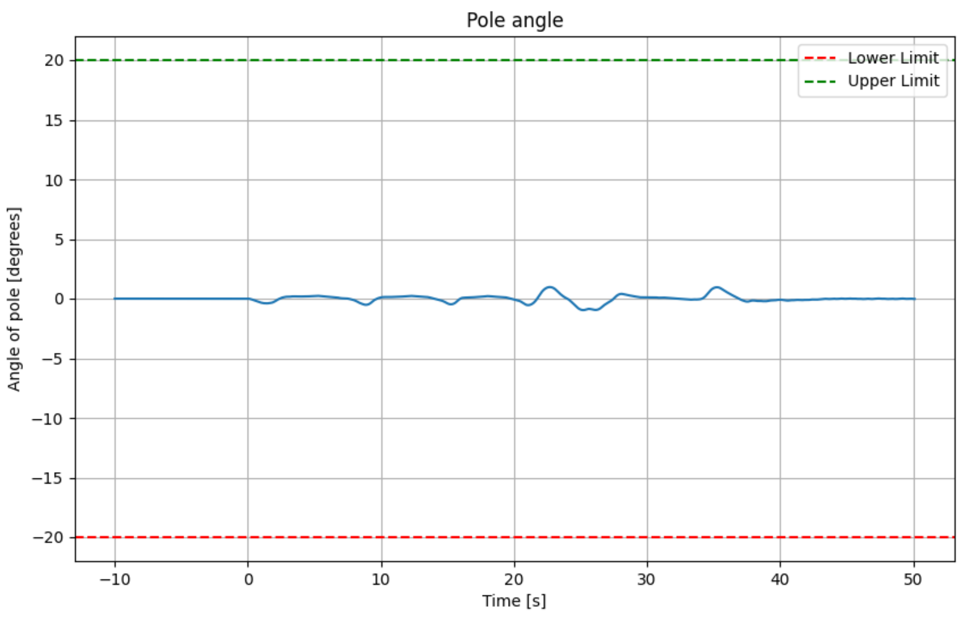


Figure 30: Angle of pole over time when agent is applied

The graph reveals that the pole angle experiences only minor fluctuations around 0 degrees throughout the episode. This indicates that the agent effectively controls the pole angle, maintaining it close to the vertical position. The ability to keep the pole angle near 0 degrees demonstrates the agent's capability to maintain balance and stability during the CartPole task.

The agent's proficiency in controlling the pole angle is crucial for solving the CartPole problem. By continuously adjusting its actions based on feedback from the environment, the agent ensures that the pole remains balanced and does not deviate significantly from the vertical position. This precise control of the pole angle contributes to higher rewards and successful episodes.

Upon observing the graph, it becomes evident that the pole angle consistently remains well within the predefined limits throughout the entire episode. This observation suggests that the termination of the episode is primarily attributed to the agent reaching the maximum reward rather than the pole angle exceeding the limits. The agent's ability to effectively control the vessel's motion and maintain stability is demonstrated by the pole angle staying within the desired range. Hence, the termination of the episode can be attributed to the agent successfully achieving its goal of maximizing the reward.

The agent's ability to maintain the pole angle around 0 degrees throughout the episode highlights its stability and consistency. It showcases the agent's learning capabilities and adaptability, as it continuously adjusts its actions to prevent the pole from falling and to achieve the desired outcome. By effectively controlling the pole angle, the agent demonstrates its proficiency in solving the CartPole problem and accomplishing the task at hand.

To gain further insights into the agent's behavior, Figure 31 displays the vessel's x-position over time, with the ±10m limits indicated. This graph provides evidence of the agent consistently operating within this predefined range, which further reinforces its effective control over the vessel's motion.

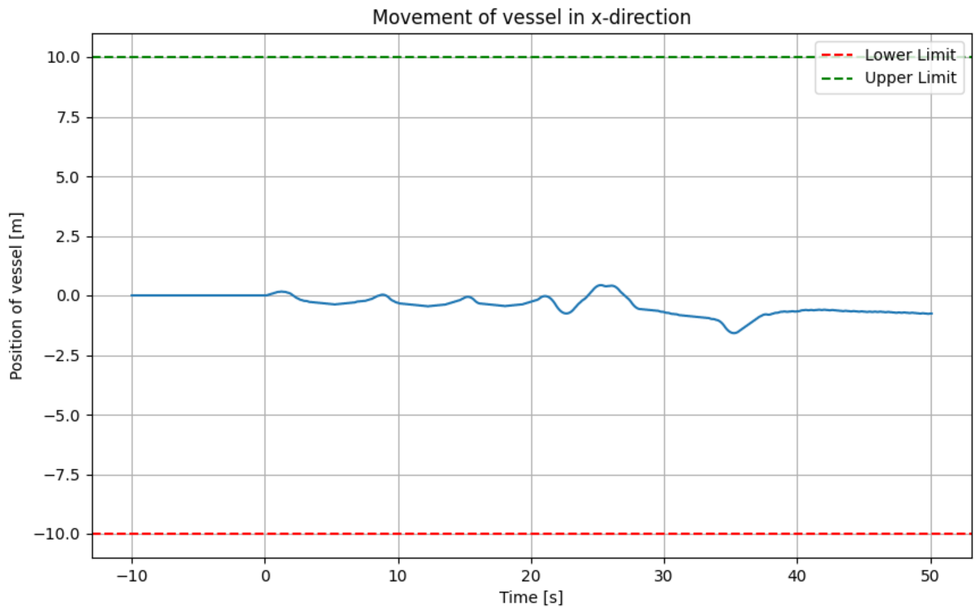


Figure 31: CartPole vessel position when agent is applied

The above graph displays the vessel's x-position on the y-axis and the elapsed time on the x-axis. As the training progresses, the agent learns to adjust the vessel's movements to maintain the pole within the acceptable angle limits. This is reflected in the change in the vessel's position over time, which correlates with the changes in the pole angle.

In addition it showcases the agent's ability to respond to the pole's movements. When the pole angle leans towards one side, the agent promptly shifts the vessel in the opposite direction to counterbalance the pole. This action can be observed in the graph as the vessel's x-position adjusts accordingly, ensuring the pole angle remains within the specified limits. The agent's effective coordination of the vessel's movements with the pole angle demonstrates its capability to maintain stability and prevent excessive tilting.

It is important to note that the small fluctuations observed in the vessel's movement do not contribute to episode termination. Instead, as indicated by the results, episodes are terminated upon reaching the maximum reward. This suggests that the agent successfully controls the vessel's movement to fluctuate around 0m, minimizing deviations and ensuring effective task completion.

The agent's ability to maintain both the vessel's position and the pole angle within their respective limits is crucial for solving the CartPole problem. By operating within the defined range, the agent maximizes its chances of maintaining balance and preventing the pole from falling. This ultimately leads to higher rewards and successful completion of episodes, showcasing the agent's proficiency in mastering the task.

To further evaluate the effectiveness of the trained agent, a comparison was made between random actions, the trained agent, and the performance of the agent in the classic CartPole problem as presented by Pylessons (Pylessons 2019). The total rewards achieved by these three approaches were analyzed over 21 simulations. Figure 32 illustrates the graph depicting the total rewards obtained from random actions, the trained agent, and the performance of the agent in the classic CartPole problem.

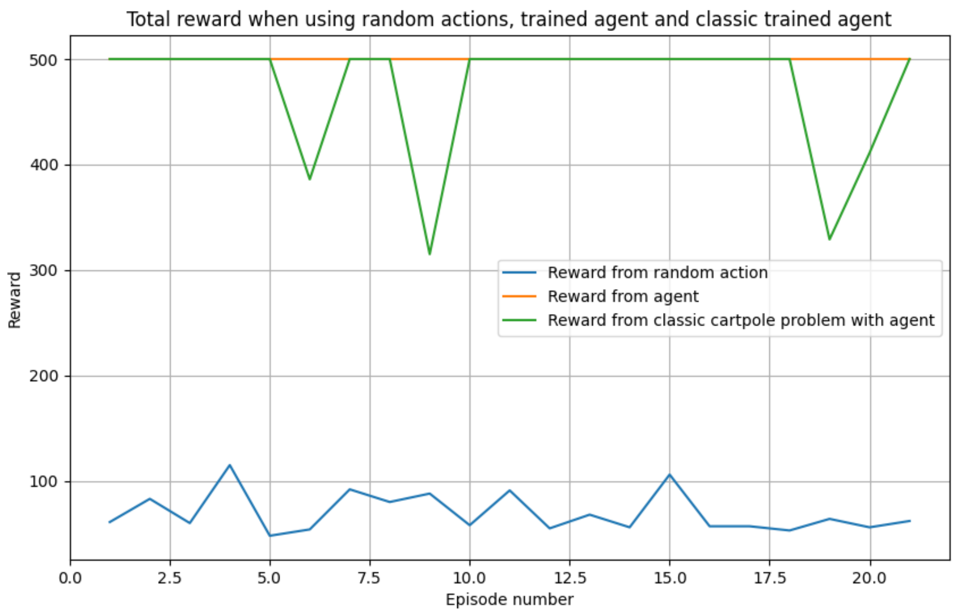


Figure 32: Rewards from random actions compared to use of agent

In the case of random actions, the total rewards varied considerably across the 21 simulations, with values ranging from 48 to 115. This wide range of rewards indicates the inconsistency and unreliability of random actions in achieving desirable outcomes.

On the other hand, the trained agent consistently achieved a total reward of 500 across all 21 simulations. This demonstrates the agent's ability to make informed decisions and take actions that lead to a desirable outcome in the CartPole problem. The agent's performance outperforms random actions, indicating its effectiveness in solving the problem.

Additionally, the performance of the agent in the classic CartPole problem, as presented by Pylessons, was also considered. The agent consistently achieved a maximum reward of 500 in approximately 17 out of the 21 simulations. This demonstrates the agent's ability to perform well in a well-known benchmark problem.

The comparison between the random actions, the trained agent, and the agent in the classic CartPole problem provides valuable insights into the effectiveness of the trained agent. It shows that the trained agent outperforms random actions and performs at a similar level to the agent in the classic CartPole problem. This highlights the success of the training process and the agent's ability to adapt to the specific problem domain.

In conclusion, the trained agent demonstrates its superiority over random actions and performs comparably to the agent in the classic CartPole problem. The results validate the effectiveness of the training process and highlight the agent's ability to solve the CartPole problem. Further analysis and optimization may be required to enhance the agent's performance and achieve even higher rewards in future iterations.

7.2 J-tube pull-in problem

In the J-tube pull-in installation scenario, the DQN agent was trained to control the pay-out rate of the winch wire, ensuring that the touchdown tension of the cable remains within the desired range of 0 to 2 kN. By providing a reward of 1 for each time step the agent maintains the tension within the specified limits, the agent is encouraged to learn the optimal pay-out rate strategy.

Wave loads were excluded from the simulation because the winch wire is generally heave compensated for a real-world pull-in to counteract the effects of waves, ensuring that the pay-out rate remains stable and unaffected by wave-induced movements.

The primary objective of the experiment is to assess the performance of the DQN agent in achieving consistent rewards within the desired range. By successfully maintaining the touchdown tension, the agent demonstrates its ability to control the pay-out rate effectively and ensure the safe and accurate installation of the cable.

By evaluating the performance of the agent, the experiment provides insights into the learning progress of the agent and its ability to generalize its knowledge to new scenarios. The analysis of the experiment's results and their subsequent discussion shed light on the effectiveness of the agent in achieving the desired outcomes and contribute to the understanding of its potential applications in real-world cable installation operations.

7.2.1 Results and discussion

The results obtained from the experiment provide insights into the performance of the DQN agent in the J-tube pull-in installation scenario. Several graphs were generated to analyze different aspects of the behavior of the agent and the achieved outcomes.

Figure 33 depicts the velocity of quadrant pay-out for the J-tube pull-in installation with the trained agent applied. The behavior of the agent in controlling the pay-out rate of the winch wire can be observed. The graph provides insights into how the agent adjusts the velocity over time to ensure the successful deployment of the quadrant.

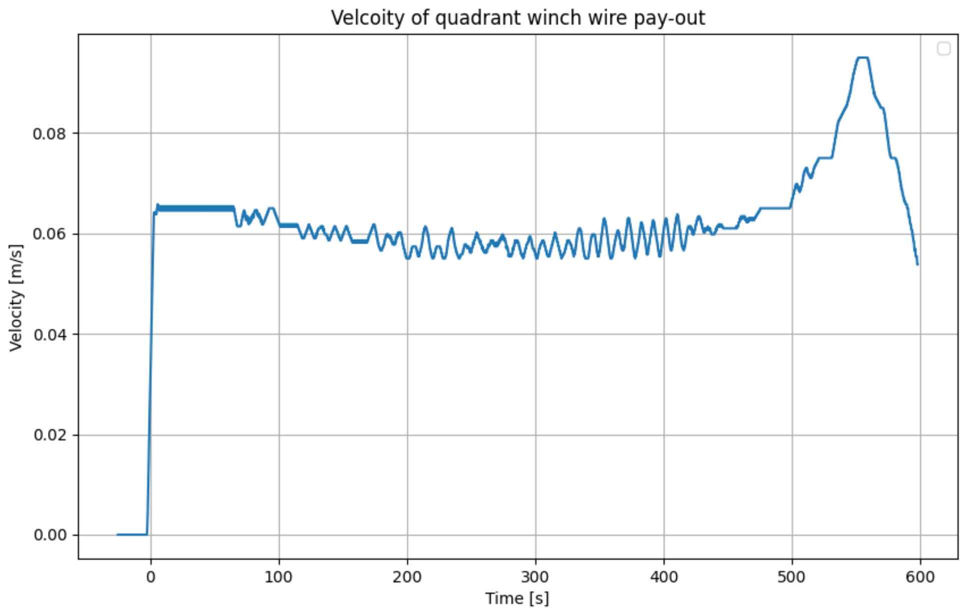


Figure 33: Velocity of quadrant pay-out for J-tube pull-in when agent is applied

At the beginning of the scenario, the velocity of pay-out increases rapidly from 0 to approximately 0.06-0.065 m/s. This initial increase indicates the agent's proactive response to the installation process as the platform winch is hauling in at a constant velocity of 0.1 m/s, initiating the deployment of the quadrant with an appropriate speed. Subsequently, the pay-out velocity exhibits fluctuations around the value of 0.06, which demonstrates the ability of the agent to maintain a consistent rate during the intermediate stages of the installation. This velocity aligns with the logical progression of the operation, where the quadrant needs to be lowered at roughly half the velocity of the pull-in winch.

Towards the end, there is a notable change in the pay-out velocity. The graph shows that the velocity is increased to around 0.09, indicating the adjustment to accommodate the specific requirements of the installation process at that point. The relationship between the haul-in velocity and pay-out velocity shifts from roughly half to approximately the same velocity. This increase in velocity suggests that the agent recognizes the need for a higher pay-out rate during this phase, potentially to ensure proper positioning or alignment of the quadrant.

After reaching the peak velocity, the pay-out rate starts to decrease again. This reduction indicates the response to completing the installation of the quadrant or transitioning to the next phase of the operation. The control demonstrated by the agent in adjusting the pay-out velocity throughout the scenario reflects its capability to adapt and optimize the deployment process, ensuring the successful installation of the J-tube pull-in.

To evaluate the learning progress of the agent, the total reward obtained in every 20th simulation during the training process was plotted. Figure 34 provides valuable insights into how the agent's performance evolves as it gains experience through training iterations.

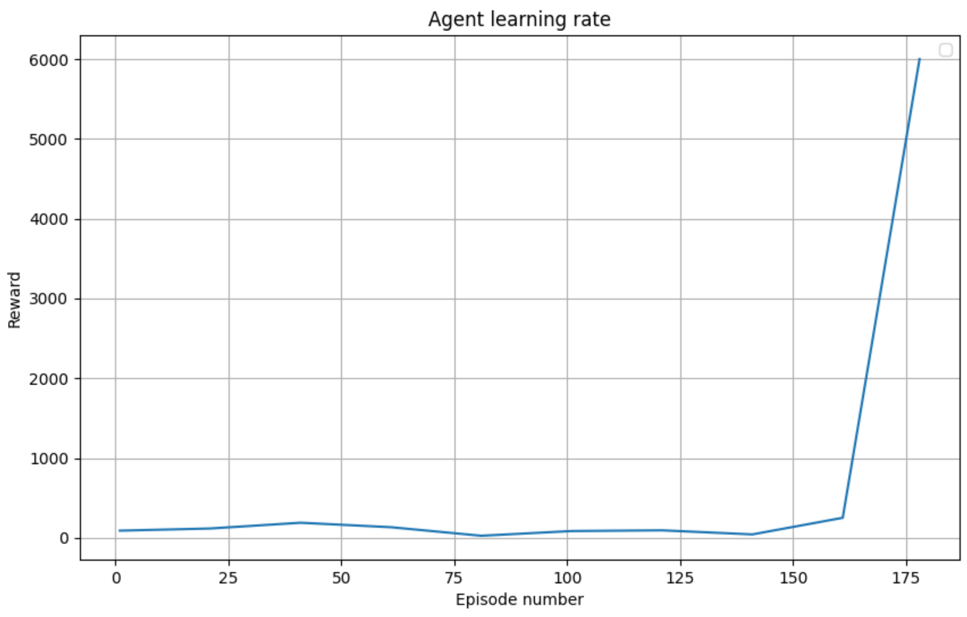


Figure 34: Agent learning rate for J-tube pull-in problem

The graph shows the total reward obtained at different episodes of training. It can be observed that initially, in the first few episodes, the reward is relatively low, at around 91.

A significant milestone in the training process is evident when the total reward of 6000. This point corresponds to a critical stage in the scenario where the vessel moves forward to lay the quadrant on the seabed. The successful achievement of this step is crucial for the overall success of the installation process. The fact that the agent reaches a total reward of 6000 signifies its ability to effectively control the pay-out rate strategy, leading to the desired outcome of quadrant deployment.

The learning rate displayed in the total reward graph does not exhibit a consistent increase but rather remains flat, with a noticeable inflection point where the reward abruptly jumps to 6000. This sudden change in performance raises questions about the factors contributing to this shift in the agent's learning process.

One possible explanation for the flat reward curve during the initial phase is that the agent focused on exploration. During exploration, the agent tries different actions to gain a better understanding of the environment. This process may have resulted in a relatively stagnant reward curve. However, at the inflection point, the agent could have transitioned to exploitation, where it began utilizing the acquired knowledge to maximize its reward. This change in strategy could account for the sudden increase in performance.

Another reason for the inflection point could be the agent's ability to overcome initial challenges. The initial episodes of training might have presented significant difficulties as the agent struggled to comprehend the environment's dynamics and determine successful strategies. However, after a certain number of episodes, the agent may have overcome these challenges and discovered effective approaches to maximize its reward. This breakthrough could explain the abrupt improvement in performance.

Deep reinforcement learning algorithms often involve training neural networks that approximate the Q-values or policy of the agent. It is plausible that the network took time to converge and learn meaningful representations of the environment and optimal policies. Once the network achieved a certain level of convergence, its performance could have improved dramatically, resulting in the observed increase in reward.

The parameters used in the training process, such as the number of episodes, discount rate (gamma), exploration rate (epsilon), learning rate, and batch size, play important roles in the learning progress of the agent. In this experiment, the agent was trained for a total of 137 episodes. The discount rate (gamma) was set to 0.95, indicating the weight given to future rewards in the agent's decision-making process. The exploration rate (epsilon) started at 1.0 and gradually decayed over time to encourage the agent to exploit its learned knowledge. The learning rate determines how much the neural network learns in each iteration, and the batch size determines the number of samples used for training the DQN. As for the baseline CartPole problem, the hyperparameters remained the same as for the DQN soving the classic CartPole problem (Pylessons 2019).

The variation in the pay-out rate for each training simulation was examined by plotting the results for every 20th episode, as shown in Figure 35. The graphs illustrate significant fluctuations in the pay-out rate across different episodes.

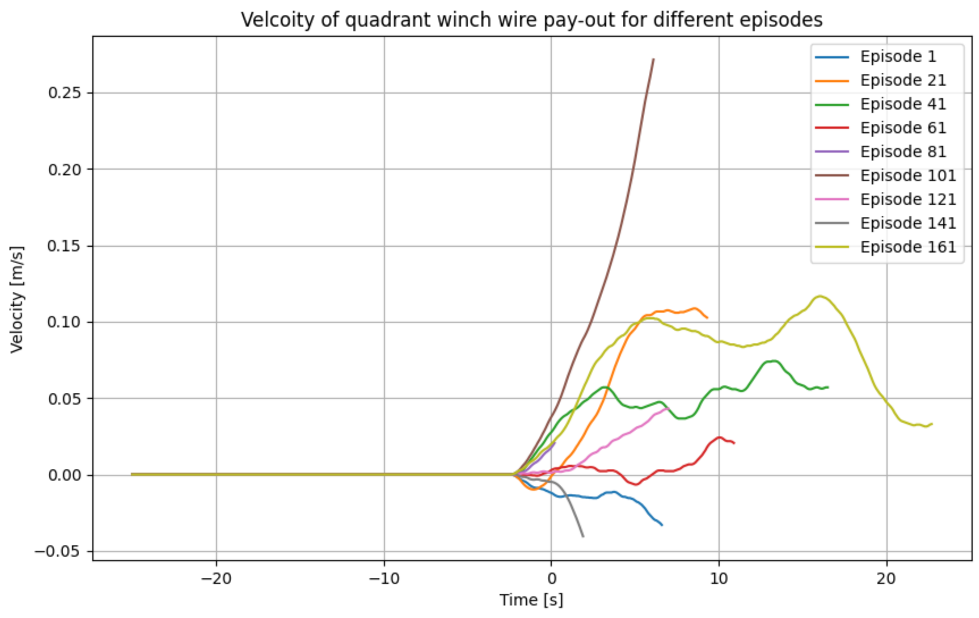


Figure 35: Pay-out rate for different training episodes for J-tube pull-in

Upon analysis of the graphs, it is observed that episode 101 achieved the highest pay-out velocity, slightly exceeding 0.25 m/s. This indicates that the agent successfully controlled the pay-out rate to ensure efficient deployment of the quadrant. Conversely, episode 141 displayed the lowest pay-out velocity, measuring less than -0.05 m/s. In this case, the negative value implies that the cable was being hauled in instead of being paid out, indicating an unsuccessful deployment attempt.

Episode 161, which recorded the highest reward during training, exhibited an initial increase in pay-out velocity, fluctuating around 0.1 m/s. This indicates that the agent effectively controlled the pay-out rate in the early stages of the installation process. However, towards the later stages of the episode, the pay-out velocity decreased, leading to a failure in the deployment.

The results highlight the challenges faced by the agent in consistently controlling the pay-out rate throughout the training process. The fluctuations in pay-out velocity indicate the complexity of the J-tube pull-in problem and the need for adaptive strategies to address varying scenarios. It is essential for the agent to dynamically adjust the pay-out rate based on the specific requirements of each situation to ensure successful quadrant deployment.

During the simulation using the trained agent, an important aspect that was analyzed is the maximum and minimum tension in the cable over time. The tension in the cable is a critical factor to monitor, as it directly impacts the safety and effectiveness of the J-tube pull-in installation. Figure 36 provides a visual representation of the tension in the cable throughout the simulation. It clearly shows that the touchdown tension, which refers to the tension at the point of cable contact with the seabed, remained consistently within the defined limits of 0 kN to 2 kN.

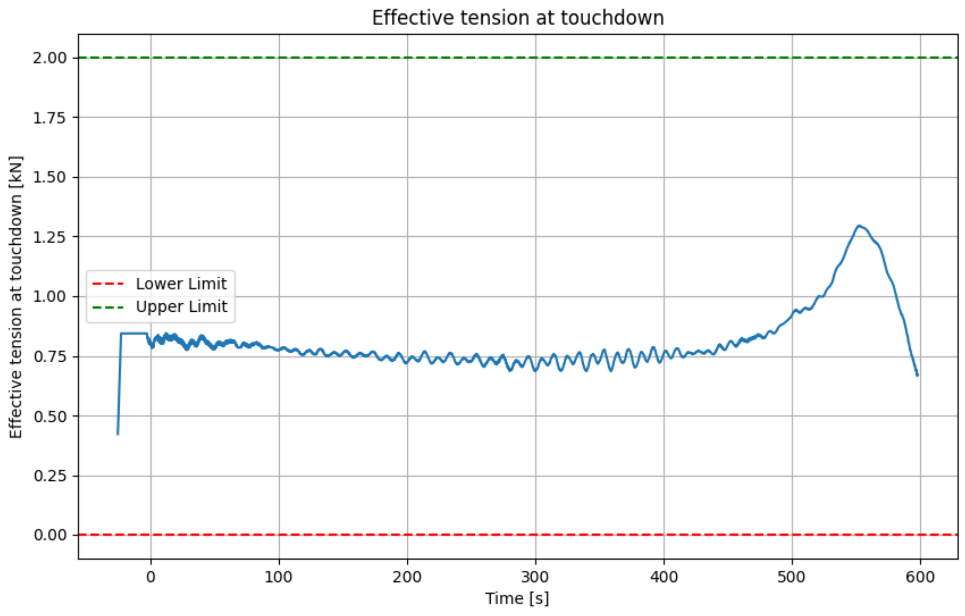


Figure 36: Touchdown tension for J-tube pull-in

In addition to analyzing the tension within the defined limits, a comparison was made between the tension experienced during episode 161 of the training simulation and the tension achieved by the trained agent. The plot, depicted in Figure 37, illustrates the variation in effective tension at touchdown for both scenarios. It can be observed that during episode 161, the tension exhibits a significantly higher level of variability and instability compared to the trained agent, specifically within the time frame of the episode. The episode concludes at approximately 25 seconds, when the lower touchdown limit of 0 kN is reached.

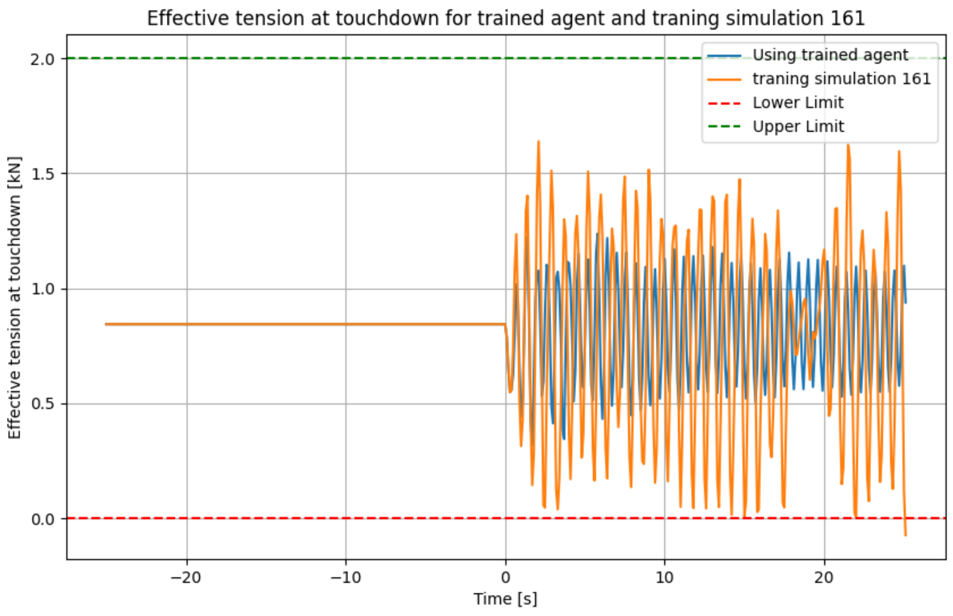


Figure 37: Touchdown tension for trained agent and training episode 161 J-tube pull-in

The ability of the agent to maintain the tension within the specified range indicates its successful control of the pay-out rate. By adjusting the rate at which the winch wire pays out, the agent effectively manages the tension in the cable, ensuring that it does not exceed the maximum allowable limit of 2 kN or fall below the minimum limit of 0 kN. This precise control of tension is crucial to prevent damage to the cable, such as buckling or overbending, and ensure safe deployment.

The above graphs provides a clear visual confirmation of the agent's ability to regulate the tension in real-time during the J-tube pull-in. By consistently keeping the tension within the defined limits, the agent demonstrates its capability to make accurate decisions regarding the pay-out rate, effectively responding to changes in the installation scenario. The successful control of the cable residual tension is a significant achievement, as it contributes to the overall safety and efficiency of the installation process. Maintaining appropriate tension ensures that the cable is properly laid and minimizes the risk of any operational issues or damages.

In addition, the comparison between episode 161 and the trained agent's performance offers compelling evidence of the agent's ability to control the pay-out rate and maintain the desired tension range throughout the simulation. This successful outcome further emphasizes the effectiveness of the trained agent in accurately managing the J-tube pull-in installation and highlights its practical value in real-world marine operations.

In order to obtain a comprehensive view of the curvature behavior throughout the length of the cable in the J-tube pull-in installation scenario, a range graph was generated. This graph, depicted in Figure 38, presents the minimum, maximum, and mean values of the curvature observed.

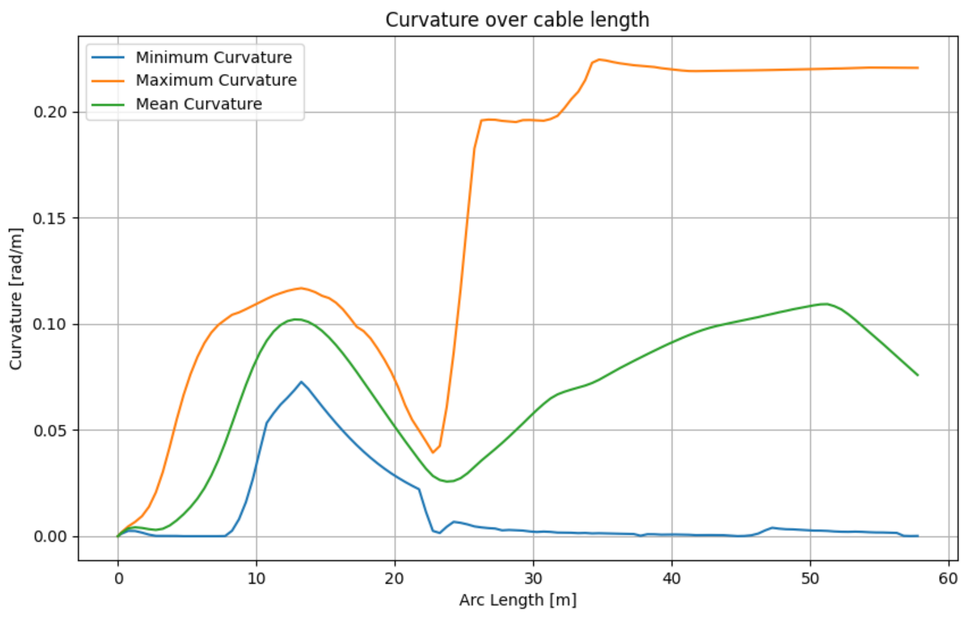


Figure 38: Min, max and mean curvature over length of cable for J-tube pull-in

By examining the graph, it is evident that the curvature values spanned a range from 0 to 0.25 rad/m. The minimum curvature value represents the point at which the cable exhibits the least bending along its length. In this scenario, the minimum curvature observed was approximately 0 rad/m, indicating sections of the cable that were nearly straight. On the other hand, the maximum curvature value signifies the regions where the cable experienced the highest degree of bending. In this case, the maximum curvature reached around 0.25 rad/m, highlighting areas of significant curvature along the cable's length. It is likely that the curvature of the quadrant between the cable lengths of 26 m to 32 m contributed to the high curvature in the cable. Upon closer examination of the maximum curvature, it is evident that the highest curvature value occurs at 34.75 m along the cable's length. This specific location is further analyzed in Figure 39.

The results presented in Figure 38 further support the agent's proficiency in regulating the pay-out rate strategy. The range of curvature values reflects the agent's adaptive behavior in response to varying installation conditions, maintaining the cable's shape within acceptable bounds throughout the deployment process.

Figure 39 highlights important aspects of the curvature of the cable during the J-tube pull-in installation scenario, as observed in the simulation where the agent was utilized. Monitoring the curvature is crucial as it directly influences the behavior and structural integrity of the cable.

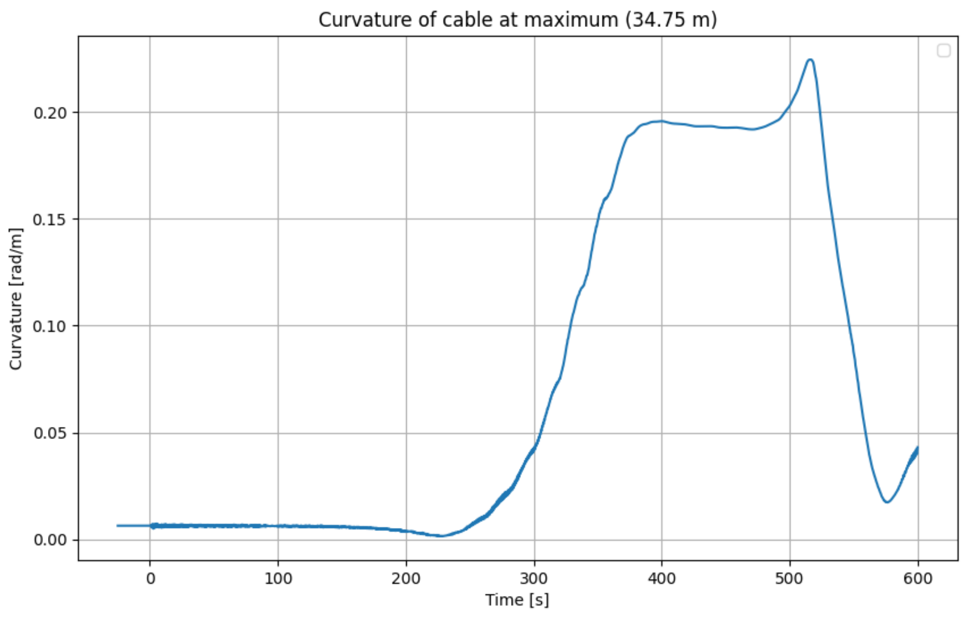


Figure 39: Curvature of cable at maximum for J-tube pull-in

While the agent did not have a predefined limit or range for curvature as an input, this aspect is still important to consider. Typically, cables have defined limits provided by the manufacturer. If the minimum bend radius was given as 2.5 m,, which is a typical limit, the curvature could be checked like this:

$$\kappa = \frac{1}{r} = \frac{1}{2.5} = 0.4 \tag{15}$$

where κ is curvature and r is minimum bend radius. This implies that the curvature would be within limits. Therefore, it was sufficient for the agent to solely control the pay-out rate based on the tension at touchdown limits for this specific case.

The successful control and maintenance of the desired curvature by the agent hold significant practical implications for real-world marine operations. It ensures the accurate alignment and positioning of the cable, contributing to the overall effectiveness and efficiency of the j-tube pull-in installation process. By effectively managing the cable's curvature, the agent enhances the project's success and minimizes the risk of potential issues or damage during deployment.

In order to evaluate the performance of the trained agent in comparison to random actions as depicted in Figure 40. This graph presents the total rewards obtained from both the trained agent and random actions during 21 simulations of the J-tube pull-in installation scenario.

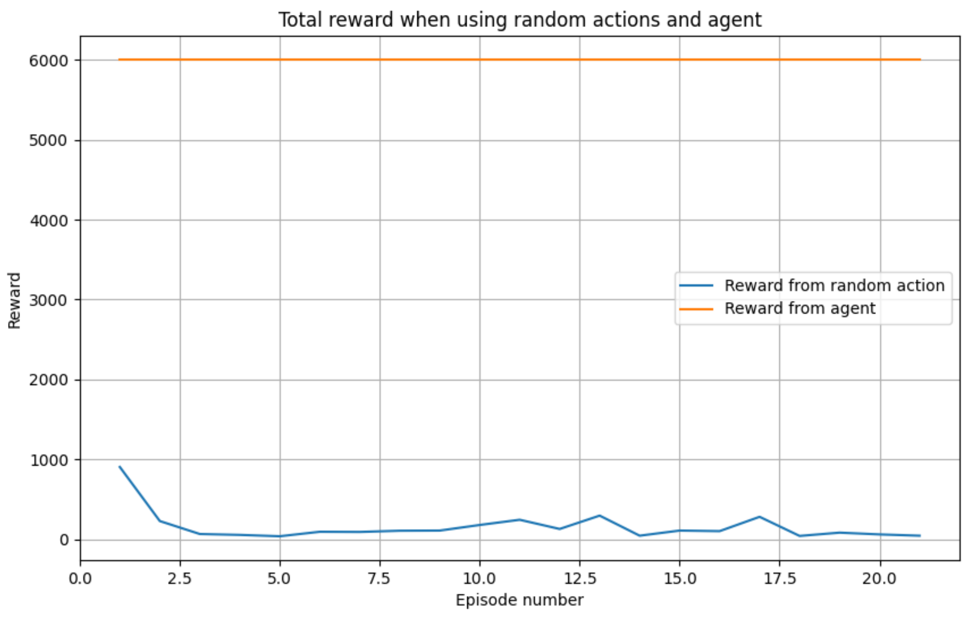


Figure 40: Rewards from random actions compared to use of agent

When analyzing the rewards obtained from random actions, it becomes evident that there is a significant variation in the rewards achieved. The graph illustrates a range of rewards, with some simulations resulting in relatively low rewards while others yield higher rewards, although significantly lower than with the trained agent. The random actions, lacking any systematic strategy, lead to inconsistent outcomes in terms of the total reward obtained.

On the other hand, when utilizing the trained agent, a consistent and superior performance is observed. In each of the 21 simulations conducted with the trained agent, the total reward consistently of 6000. This consistent high performance indicates that the agent effectively controlled the pay-out rate, leading to the maintenance of the desired touchdown tension within the specified range.

The comparison between the trained agent and random actions demonstrates the significant advantage of utilizing the trained agent in achieving successful outcomes in the J-tube pull-in installation scenario. The agent's ability to consistently receive a total reward of 6000 highlights its proficiency in controlling the pay-out rate and ensuring the safe and efficient deployment of the cable.

These results further validate the effectiveness of the trained agent and emphasize the practical implications of its utilization in real-world marine operations. By employing the trained agent when modelling, operators can benefit from its consistent and reliable performance in the end, enabling precise control over the cable installation process and ensuring the success of the project.

Overall, the results of this experiment confirm the effectiveness of the DQN agent in controlling the pay-out rate of the winch wire in the J-tube pull-in installation. The agent successfully maintains the desired touchdown tension within the specified limits, ensuring the safe and efficient deployment of the cable. These findings have practical implications in real-world marine operations where accurate control of cable installation is crucial for the success of the project.

8 Conclusion and further work

This thesis aimed to develop a deep neural network-based agent using Deep Reinforcement Learning (DRL) techniques to automate and optimize modelling for the analysis of marine cable installation. The objective was to replace manual work involved in parameter optimization, particularly in determining critical parameters like tension, with an efficient and accurate automated process.

Chapter 1 provided the background and motivation for the study, along with a summary of the literature review highlighting the current state of marine cable installation analysis. Research questions were defined, and the objective and scope of the thesis were outlined. The contributions of this work included advancements in Marine Cybernetics through RL techniques and revolutionizing the modeling of cable installation processes using intelligent agents.

Chapter 2 presented an overview of submarine power cable technology, including its applications, reliability, and design elements. This chapter provided the necessary foundation for understanding the complexities and challenges associated with cable installation.

Chapter 3 focused on cable installation design, discussing critical scenarios, loads, and load effects, as well as the various steps involved in the cable installation process. It covered topics such as cable routing, schedule and timing, removal of obstacles, transportation, reel handling, laying campaign, cable protection, and the required vessel and cable laying equipment. The chapter also discussed the analysis of submarine cable installation, including cable laying analysis, cable pulling-in analysis, and considerations for weather conditions.

Chapter 4 introduced the methodology used in this thesis, highlighting the concepts of machine learning, supervised and unsupervised learning, and deep neural networks. It then looked into the principles of Reinforcement Learning (RL), including problem formulation, value function, exploration vs. exploitation, model-free vs. model-based RL, temporal difference learning, policy optimization, and deep deterministic policy gradient.

Chapter 5 provided an overview of the software used for implementation and case scenarios. It discussed OrcaFlex, a software package used for modeling and analysis, and OpenAI Gym, a popular framework for RL experiments. The baseline CartPole problem in OrcaFlex and the J-tube pull-in problem were introduced as the case scenarios for implementation.

Chapter 6 described the implementation details of the baseline CartPole problem in OrcaFlex, including the OrcaFlex model, vessel motion, and the custom environment created using OpenAI Gym. It also presented the execution of the model using random actions and the DQN algorithm. It further explored the implementation of the J-tube pull-in problem, discussing the payout rate of quadrant winch wire and the environment setup. It provided insights into the execution of the model and the simulation process.

Chapter 7 presented the experimental results and discussions for both the baseline CartPole problem and the J-tube pull-in problem. It analyzed the performance of the developed agent and provided insights into its effectiveness and efficiency in optimizing the cable installation process.

8.1 Conclusion

In conclusion, this thesis/project successfully addressed the research questions raised in the introductory chapter:

"Can deep reinforcement learning (DRL) be effectively utilized for parameter optimization in the installation analysis of marine cables?"

The findings demonstrate that deep reinforcement learning techniques, specifically employing a Deep Q-Network (DQN) agent, can be effectively utilized for parameter optimization in the installation analysis of marine cables. The trained DQN agents showcased remarkable learning capabilities and adaptability in both the CartPole problem and the J-tube pull-in installation scenario. By leveraging intelligent agents and reinforcement learning algorithms, critical parameters related to cable installation, such as tension at the touchdown point, were optimized with high accuracy and efficiency.

Similarly, in the CartPole problem, the trained DRL agent exhibits accuracy and efficiency in maintaining stability and balance, as evidenced by the consistent reward achieved across multiple simulations. The agent successfully learns to control the vessel's movements and pole angle to prevent the pole from falling, indicating its proficiency in optimizing critical parameters related to cable installation.

"How accurate and efficient is the trained DRL agent in optimizing critical parameters related to cable installation, such as tension at the touch-down point?"

The trained DQN agents exhibited impressive accuracy and efficiency in optimizing critical parameters related to cable installation. In the CartPole problem, the agent consistently achieved high rewards, effectively controlling the vessel's motion, maintaining stability, and balancing the pole within the desired range. Similarly, in the J-tube pull-in installation scenario, the agent successfully adjusted the pay-out velocity of the winch wire, ensuring the safe and effective deployment of the quadrant. The agent's ability to maintain tension in the cable within specified limits further confirmed its accuracy in optimizing critical parameters.

In summary, the significant findings of this thesis highlight the potential of deep reinforcement learning and DQN agents in addressing complex control problems in marine applications. The trained agents exhibited high levels of performance, adaptability, and learning capabilities. By successfully optimizing critical parameters, such as tension at the touchdown point, the utilization of intelligent agents and reinforcement learning techniques improves the accuracy and efficiency of cable installation processes.

These findings have broader implications for the development of autonomous control systems in marine operations. The application of deep reinforcement learning and intelligent agents can enhance safety, efficiency, and overall operational outcomes. By advancing marine cybernetics, this research contributes to the future of marine cable installations and could pave the way for further advancements in autonomous control systems in the marine industry.

8.2 Further work

In order to enhance the effectiveness and applicability of the proposed control algorithm, several avenues for further research and development can be explored.

An important aspect when looking at the J-tube problem and other marine operations is to test the algorithm's performance in varying environmental conditions. Incorporating factors such as wind, waves, and current into the testing scenarios would allow for a comprehensive evaluation of the algorithm's robustness and adaptability in real-world conditions. This exploration can provide valuable insights into how the algorithm handles disturbances caused by environmental factors and help refine its control strategies accordingly.

Moreover, it would be beneficial to assess the algorithm's performance under different sea states, ranging from mild to severe. By examining the algorithm's behavior in various sea states, researchers can gain a deeper understanding of its robustness and adaptability in adverse conditions. This exploration could lead to improvements in the algorithm's performance and the development of strategies to ensure safe and reliable operation across a wide range of sea states.

Expanding the control variables beyond the current limited set would also be a valuable avenue for further investigation. By incorporating more control inputs and reducing the number of constant variables, the algorithm can be equipped to handle a broader range of operating conditions. This expansion can enhance the algorithm's adaptability to dynamic environments and improve its overall control performance.

To provide a comprehensive assessment of the proposed control algorithm, it is essential to test it in different real-world scenarios. This could involve evaluating its performance in various water bodies such as rivers, lakes, and coastal regions. Additionally, exploring the algorithm's effectiveness in specific tasks like path planning, obstacle avoidance, or target tracking would provide valuable insights into its capabilities and limitations. By subjecting the algorithm to diverse scenarios, researchers can validate its practicality and identify areas for further improvement.

To establish the competitiveness of the proposed control algorithm, it would be valuable to compare its performance against other existing control algorithms. This comparison can be based on metrics such as control accuracy, stability, energy efficiency, and response time. By benchmarking against state-of-the-art control strategies, the strengths and weaknesses of the proposed algorithm can be identified. This analysis will facilitate further improvements and advancements in the algorithm, positioning it as an effective and competitive solution.

Lastly, comparing the proposed control algorithm with the widely used proportional-integral-derivative (PID) controller would be an interesting avenue to explore. The PID controller is a well-established control technique, and comparing its performance with the proposed algorithm can provide valuable insights. This comparison would shed light on the advantages, if any, of the proposed algorithm over the conventional PID approach and help identify potential areas for improvement or alternative control strategies.

By addressing these aspects in further research and development, the proposed control algorithm can be refined, validated, and positioned as an effective and reliable solution.

Bibliography

Ahn, Ki Uhn and Cheol Soo Park (2020). 'Application of deep Q-networks for model-free optimal control balancing between different HVAC systems'. In: *Science and Technology for the Built Environment* 26.1, pp. 61–74. DOI: 10.1080/23744731.2019.1680234. URL: https://doi.org/10.1080/23744731.2019.1680234.

Anthony, Thomas, Zheng Tian and David Barber (2017). 'Thinking Fast and Slow with Deep Learning and Tree Search'. In: *CoRR* abs/1705.08439. arXiv: 1705.08439. URL: http://arxiv.org/abs/1705.08439.

API, American Petroleum Institute (2014a). *Recommended Practice for Flexible Pipe.* Pages: 138-161. Accessed: 2022–09-19.

— (2014b). *Specification for Unbonded Flexible Pipe.* Pages: 22-28. Accessed: 2022–09-19.

Barto, Andrew G., Richard S. Sutton and Charles W. Anderson (1983). 'Neuronlike adaptive elements that can solve difficult learning control problems'. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5, pp. 834–846. DOI: 10.1109/TSMC.1983.6313077.

Cigre (2022). *Installation of Submarine Power Cables.* URL: https://e-cigre.org/publication/883-installation-of-submarine-power-cables.

Delua, Julianna (2021). 'Supervised vs. Unsupervised Learning: What's the Difference?' In: *IBM.* URL: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

Equinor (2022). *Kort om Equinor.* URL: https://www.equinor.com/no/om-oss/kort-om-equinor.

Feinberg, Vladimir et al. (2018). 'Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning'. In: *CoRR* abs/1803.00101. arXiv: 1803.00101. URL: http://arxiv.org/abs/1803.00101.

Foy, Peter (2021). *Fundamentals of Reinforcement Learning: Policies, Value Functions & the Bellman Equation.* URL: https://www.mlq.ai/reinforcement-learning-policies-value-functions-bellman-equation/.

Garnier, Paul et al. (2021). 'A review on deep reinforcement learning for fluid mechanics'. In: *Computers & Fluids* 225, p. 104973. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2021.104973. URL: https://www.sciencedirect.com/science/article/pii/S0045793021001407.

GM (2022). *Global Maritime.* URL: https://www.globalmaritime.com/.

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep Learning.* http://www.deeplearningbook.org. MIT Press.

Grimstad, Bjarne (2022). *TTK28 - Modelling with neural networks.* Accessed: 2022–10-18.

Guo, Siyu et al. (2021). 'Path Planning of Coastal Ships Based on Optimized DQN Reward Function'. In: *Journal of Marine Science and Engineering* 9.2. ISSN: 2077-1312. DOI: 10.3390/jmse9020210. URL: https://www.mdpi.com/2077-1312/9/2/210.

Hoss, Belyadi and Haghighat Alireza (2021). *Machine Learning Guide for Oil and Gas Using Python : A Step-by-Step Breakdown with Data, Algorithms, Codes, and Applications.* Gulf Professional Publishing. ISBN: 9780128219294. URL: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2643527&site=ehost-live&scope=site.

IEA, International Energy Agency (2021). *Renewables 2021.* Paris: International Energy Agency. URL: https://www.iea.org/reports/renewables-2021.

IEEE (2005). 'IEEE Guide for the Planning, Design, Installation, and Repair of Submarine Power Cable Systems'. In: *IEEE Std 1120-2004*, pp. 1–45. DOI: 10.1109/IEEESTD.2005.95937.

Kiran, Mariam and Melis Ozyildirim (2022). *Hyperparameter Tuning for Deep Reinforcement Learning Applications.* arXiv: 2201.11182 [cs.LG].

KVCable.com (2022). *Submarine Cable.* URL: https://kvcable.com/products/submarine-cable/.

Li, Lingyu et al. (2021). 'A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field'. In: *Applied Ocean Research* 113, p. 102759. ISSN: 0141-1187. DOI: https://doi.org/10.1016/j.apor.2021.102759. URL: https://www.sciencedirect.com/science/article/pii/S0141118721002352.

Li, Yuxi (2018). 'Deep Reinforcement Learning'. In: *CoRR* abs/1810.06339. arXiv: 1810.06339. URL: http://arxiv.org/abs/1810.06339.

Liang, Eric et al. (2017). 'Ray RLLib: A Composable and Scalable Reinforcement Learning Library'. In: *CoRR* abs/1712.09381. arXiv: 1712.09381. URL: http://arxiv.org/abs/1712.09381.

Lin, Long-Ji (1993). 'Reinforcement Learning for Robots Using Neural Networks'. UMI Order No. GAX93-22750. PhD thesis. USA.

Maritime, Kongsberg (2022). *Kongsberg Maritime.* URL: https://www.kongsberg.com/no/maritime/.

Mitchell, Tom M. (1997). *Machine learning.* McGraw-Hill Science/Engineering/Math. URL: http://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.

Mnih, Volodymyr et al. (2013). 'Playing Atari with Deep Reinforcement Learning'. In: *CoRR* abs/1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

Mohaghegh, Shahab (Sept. 2000). 'Virtual-Intelligence Applications in Petroleum Engineering: Part 1 - Artificial Neural Networks'. In: *Journal of Petroleum Technology - J PETROL TECHNOL* 52, pp. 64–73. DOI: 10.2118/58046-MS.

Nagabandi, Anusha et al. (2017). 'Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning'. In: *CoRR* abs/1708.02596. arXiv: 1708.02596. URL: http://arxiv.org/abs/1708.02596.

Nexans (2022). *Nexans Norge.* URL: https://www.nexans.no/no/company.html.

Nielsen, M.A. (2015). *Neural Networks and Deep Learning.* Determination Press. URL: https://books.google.no/books?id=STDBswEACAAJ.

OpenAI (2017). *OpenAI Baselines: DQN.* URL: https://openai.com/research/openai-baselines-dqn.

— (2022a). *Gym Documentation.* URL: https://www.gymlibrary.dev/.

— (2022b). *Gym Documentation - CartPole.* URL: https://www.gymlibrary.dev/environments/classic_control/cart_pole/.

— (2022c). *Part 2: Kinds of RL Algorithms.* URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.

Orcina (2022a). *OrcaFlex – World-leading software that goes beyond expectation.* URL: https://www.orcina.com/orcaflex/.

— (2022b). *OrcaFlex API Documentation.* URL: https://www.orcina.com/webhelp/OrcFxAPI/Default.htm.

— (2022c). *OrcaFlex Documentation.* URL: https://www.orcina.com/webhelp/OrcaFlex/Default.htm.

Park, Sangmin et al. (Sept. 2021). 'Deep Q-network-based traffic signal control models'. In: *PLOS ONE* 16.9, pp. 1–14. DOI: 10.1371/journal.pone.0256405. URL: https://doi.org/10.1371/journal.pone.0256405.

Paszke, Adam and Mark Towers (2023). *Reinforcement learning (DQN) tutorial.* URL: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html.

Patrick, Bangert (2021). *Machine Learning and Data Science in the Oil and Gas Industry : Best Practices, Tools, and Case Studies.* Gulf Professional Publishing. ISBN: 9780128207147. URL: https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2518914&site=ehost-live&scope=site.

Poulsen, Michael Fly (2022). *Cable & Pipe handling.* https://www.macartney.com/offshore-wind-solutions/cable-pipe-handling/. Accessed: 2022–09-10.

Pylessons (2019). *Introduction to Reinforcement Learning.* URL: https://pylessons.com/CartPole-reinforcement-learning.

Rabault, Jean et al. (2020). 'Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization'. In: *Journal of Hydrodynamics.* URL: https://doi.org/10.1007/s42241-020-0028-y.

Sehgal, Adarsh et al. (2022). *Automatic Parameter Optimization Using Genetic Algorithm in Deep Reinforcement Learning for Robotic Manipulation Tasks.* arXiv: 2204.03656 [cs.RO].

Silver, David (2015). *Lectures on Reinforcement Learning.* URL: https://www.davidsilver.uk/teaching/.

Subsea7 (2022). *Subsea 7 - about us.* URL: https://www.subsea7.com/en/about-us.html.

Surma, Greg (2018). *Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning).* URL: https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction.* Cambridge, MA, USA: A Bradford Book. ISBN: 0262039249.

Szepesvari, Csaba (2010). *Algorithms for Reinforcement Learning.* Morgan and Claypool Publishers. ISBN: 1608454924.

Williams, Ronald J. (1992). 'Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning'. In: *Machine Learning* 8, pp. 229–256.

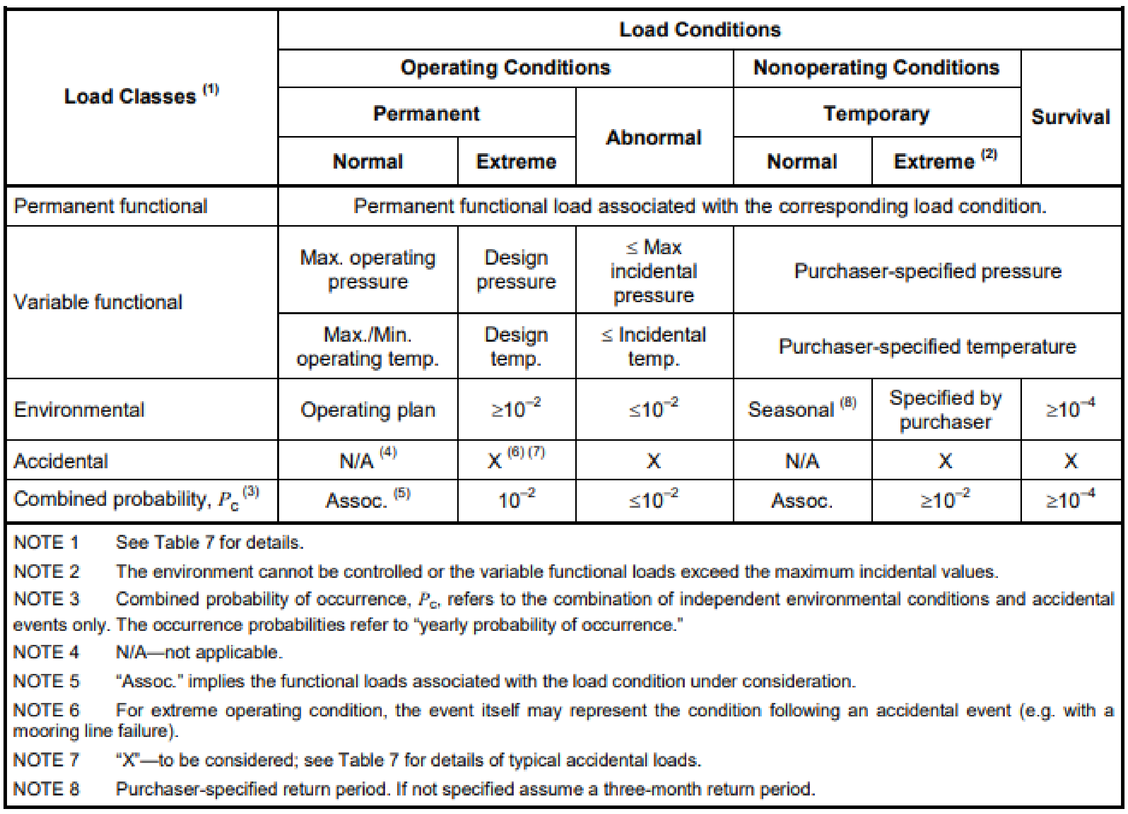
Worzyk, T. (2009). *Submarine Power Cables: Design, Installation, Repair, Environmental Aspects.* Power Systems. Springer Berlin Heidelberg. ISBN: 9783642012709. URL: https://books.google.no/books?id=X8QfRT%5C_SYDgC.

Wu, Di et al. (2022). 'Deep Reinforcement Learning-Based Path Control and Optimization for Unmanned Ships'. In: *Wireless Communications and Mobile Computing.* URL: https://doi.org/10.1155/2022/7135043.

Appendix

A Specification

A.1 Load Combinations of load classes, load conditions

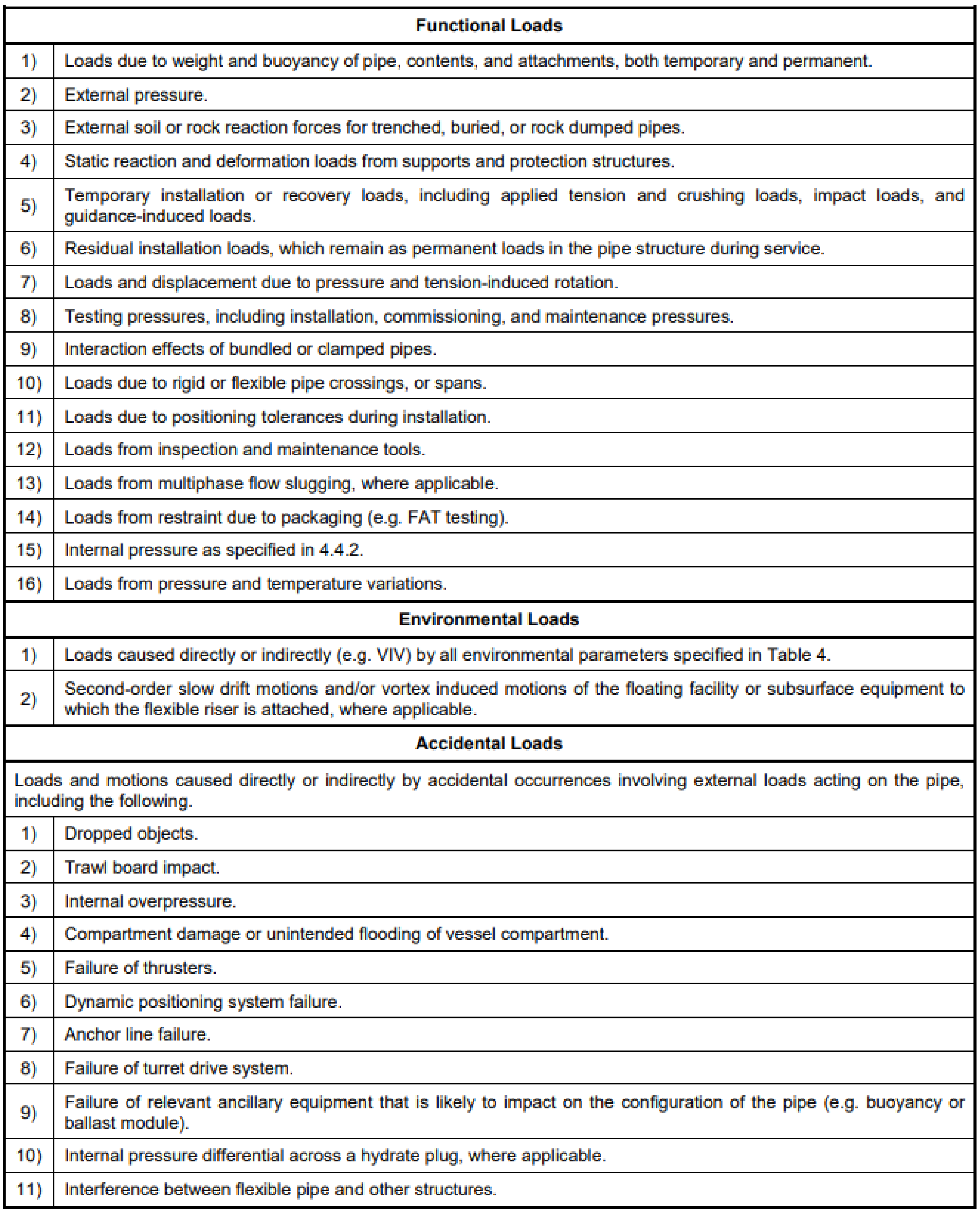


Load Classes [1]	**Load Conditions**					
	Operating Conditions			**Nonoperating Conditions**		**Survival**
	Permanent		**Abnormal**	**Temporary**		
	Normal	**Extreme**		**Normal**	**Extreme** [2]	
Permanent functional	Permanent functional load associated with the corresponding load condition.					
Variable functional	Max. operating pressure	Design pressure	$\leq$ Max incidental pressure	Purchaser-specified pressure		
	Max./Min. operating temp.	Design temp.	$\leq$ Incidental temp.	Purchaser-specified temperature		
Environmental	Operating plan	$\geq 10^{-2}$	$\leq 10^{-2}$	Seasonal [8]	Specified by purchaser	$\geq 10^{-4}$
Accidental	N/A [4]	X [6] [7]	X	N/A	X	X
Combined probability, P_c [3]	Assoc. [5]	10^{-2}	$\leq 10^{-2}$	Assoc.	$\geq 10^{-2}$	$\geq 10^{-4}$

NOTE 1 See Table 7 for details.
NOTE 2 The environment cannot be controlled or the variable functional loads exceed the maximum incidental values.
NOTE 3 Combined probability of occurrence, P_c, refers to the combination of independent environmental conditions and accidental events only. The occurrence probabilities refer to "yearly probability of occurrence."
NOTE 4 N/A—not applicable.
NOTE 5 "Assoc." implies the functional loads associated with the load condition under consideration.
NOTE 6 For extreme operating condition, the event itself may represent the condition following an accidental event (e.g. with a mooring line failure).
NOTE 7 "X"—to be considered; see Table 7 for details of typical accidental loads.
NOTE 8 Purchaser-specified return period. If not specified assume a three-month return period.

Load Combinations of load classes, load conditions, from (API 2014b)

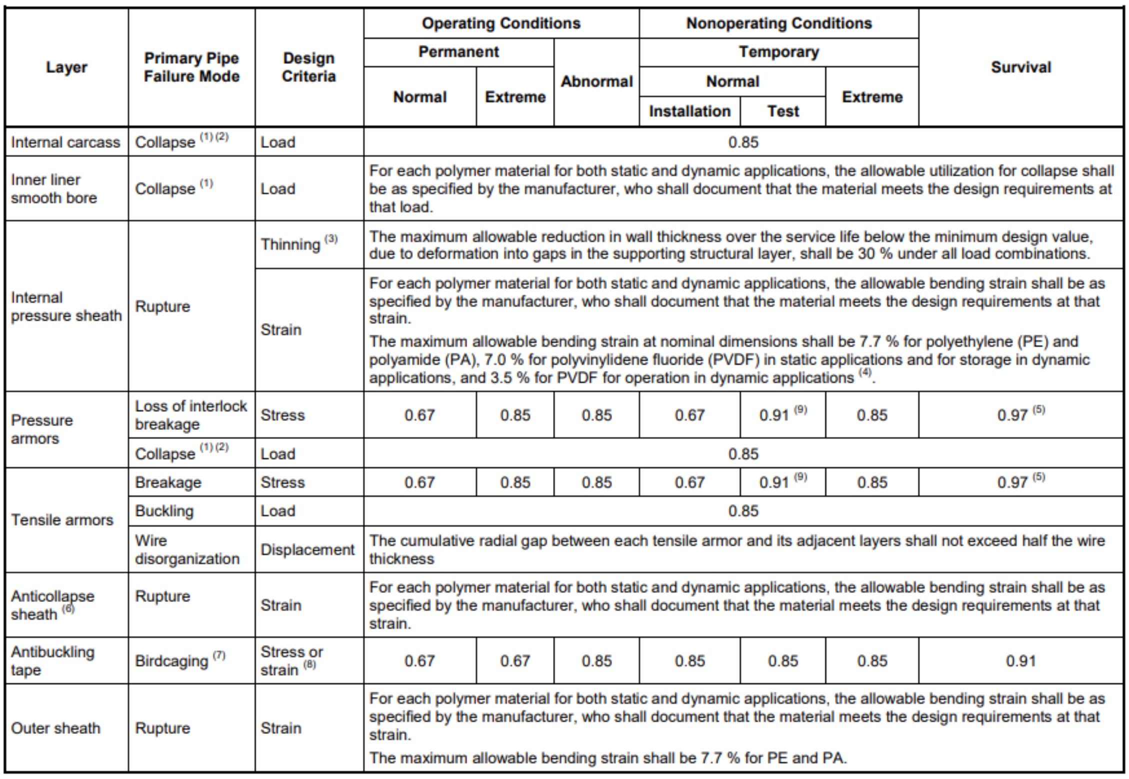
A.2 Typical load classes



Functional Loads	
1)	Loads due to weight and buoyancy of pipe, contents, and attachments, both temporary and permanent.
2)	External pressure.
3)	External soil or rock reaction forces for trenched, buried, or rock dumped pipes.
4)	Static reaction and deformation loads from supports and protection structures.
5)	Temporary installation or recovery loads, including applied tension and crushing loads, impact loads, and guidance-induced loads.
6)	Residual installation loads, which remain as permanent loads in the pipe structure during service.
7)	Loads and displacement due to pressure and tension-induced rotation.
8)	Testing pressures, including installation, commissioning, and maintenance pressures.
9)	Interaction effects of bundled or clamped pipes.
10)	Loads due to rigid or flexible pipe crossings, or spans.
11)	Loads due to positioning tolerances during installation.
12)	Loads from inspection and maintenance tools.
13)	Loads from multiphase flow slugging, where applicable.
14)	Loads from restraint due to packaging (e.g. FAT testing).
15)	Internal pressure as specified in 4.4.2.
16)	Loads from pressure and temperature variations.
Environmental Loads	
1)	Loads caused directly or indirectly (e.g. VIV) by all environmental parameters specified in Table 4.
2)	Second-order slow drift motions and/or vortex induced motions of the floating facility or subsurface equipment to which the flexible riser is attached, where applicable.
Accidental Loads	
Loads and motions caused directly or indirectly by accidental occurrences involving external loads acting on the pipe, including the following.	
1)	Dropped objects.
2)	Trawl board impact.
3)	Internal overpressure.
4)	Compartment damage or unintended flooding of vessel compartment.
5)	Failure of thrusters.
6)	Dynamic positioning system failure.
7)	Anchor line failure.
8)	Failure of turret drive system.
9)	Failure of relevant ancillary equipment that is likely to impact on the configuration of the pipe (e.g. buoyancy or ballast module).
10)	Internal pressure differential across a hydrate plug, where applicable.
11)	Interference between flexible pipe and other structures.

Typical load classes, from (API 2014b)

A.3 Flexible Pipe Layer Design Criteria



Layer	Primary Pipe Failure Mode	Design Criteria	Operating Conditions			Nonoperating Conditions			Survival
			Permanent		Abnormal	Temporary			
						Normal		Extreme	
			Normal	Extreme		Installation	Test		
Internal carcass	Collapse (1)(2)	Load	0.85						
Inner liner smooth bore	Collapse (1)	Load	For each polymer material for both static and dynamic applications, the allowable utilization for collapse shall be as specified by the manufacturer, who shall document that the material meets the design requirements at that load.						
Internal pressure sheath	Rupture	Thinning (3)	The maximum allowable reduction in wall thickness over the service life below the minimum design value, due to deformation into gaps in the supporting structural layer, shall be 30 % under all load combinations.						
		Strain	For each polymer material for both static and dynamic applications, the allowable bending strain shall be as specified by the manufacturer, who shall document that the material meets the design requirements at that strain. The maximum allowable bending strain at nominal dimensions shall be 7.7 % for polyethylene (PE) and polyamide (PA), 7.0 % for polyvinylidene fluoride (PVDF) in static applications and for storage in dynamic applications, and 3.5 % for PVDF for operation in dynamic applications (4).						
Pressure armors	Loss of interlock breakage	Stress	0.67	0.85	0.85	0.67	0.91 (9)	0.85	0.97 (5)
	Collapse (1)(2)	Load	0.85						
Tensile armors	Breakage	Stress	0.67	0.85	0.85	0.67	0.91 (9)	0.85	0.97 (5)
	Buckling	Load	0.85						
	Wire disorganization	Displacement	The cumulative radial gap between each tensile armor and its adjacent layers shall not exceed half the wire thickness						
Anticollapse sheath (6)	Rupture	Strain	For each polymer material for both static and dynamic applications, the allowable bending strain shall be as specified by the manufacturer, who shall document that the material meets the design requirements at that strain.						
Antibuckling tape	Birdcaging (7)	Stress or strain (8)	0.67	0.67	0.85	0.85	0.85	0.85	0.91
Outer sheath	Rupture	Strain	For each polymer material for both static and dynamic applications, the allowable bending strain shall be as specified by the manufacturer, who shall document that the material meets the design requirements at that strain. The maximum allowable bending strain shall be 7.7 % for PE and PA.						

Flexible Pipe Layer Design Criteria, from (API 2014b)