Optimal Energy Management and Emissions Reduction for Hybrid Marine Propulsion Plant using Reinforcement Learning

Oikonomou Kyriakos

Diploma Thesis

School of Naval Architecture and Marine Engineering

National Technical University of Athens

Supervisor: Assistant Prof. George Papalambrou

Committee Member : Assistant Prof. Christos Papadopoulos
Committee Member : Prof. Nikolaos Tsouvalis

October 2022

Acknowledgements

This work signifies the final step of my undergraduate studies and is the result of a consistent and rigorous effort not only from my part but also from individuals who interminably supported me throughout this period. This my appreciation to them.

First of all , I would like to thank my supervisor Assistant Professor G.Papalamprou for giving me the opportunity to work on this really interesting project . His precious guidance and general insight on the field paved the way for the successful completion of this work .

This work would not be possible without the significant assistance of PhD candidate Vasilios Karystinos. I am grateful for the time he devoted on brainstorming and discussing our ideas on the project which turned out to be truly fruitful and impactful to the completion of this work . I would also like to thank Orfea Bourcha who provided useful insight into Reinforcement Learning related concepts .

Many sincere thanks go to my friends Konstantinos,Spyros,Christos,Chronis,Nikitas,Sotiris, Giorgos and Harrys who staunchly supported me not only during this work but also throughout my academic years as an undergraduate student.

Finally,I would truly like to express my gratitude and appreciation to my family for the constant support and trust they showed me , enabling me to achieve my goals

Abstract

In the following thesis , the optimal control problem concerning the optimal power split between the power sources of the Hybrid Diesel-Electric Propulsion Plant (HIPPO-2) established on the Laboratory of Marine Engineering (LME) is tackled with the use of Reinforcement Learning methodologies . This optimization problem relies on the development of a energy management and emissions minimization system and its control via the Policy Gradient methodologies provided by the field of Reinforcement Learning . The implementation of the Reinforcement Learning methodologies were enabled by the utilization of Julia Programming language which is a nascent programming language discerned for its higher level-simple syntax and its noteworthy speed close to that of lower level languages such as C/C++ . The goal is to replicate the EMEMS system , achieve its real time control via a Reinforcement Learning agent and compare the results to more renowned control schemes such as the Model Predictive Control (MPC).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the recent years industries have been under enormous pressure concerning their environmental footprint.Global warming has alerted many governments worldwide and has incentivized them to take drastic measures in order to reduce their negative impact on the environment.Specifically , the shipping industry which contributes to about 15 % of the NOx emissions worldwide has been forced to discover ways to reduce their impact . The shipping industry , like any other industry , is mainly economically driven which means that the possible sanctions imposed in case of noncompliance with the measures enforced by the governments and the maritime organizations such as the IMO & MARPOL has motivated shipowners to explore ways to reduce their impact on the environment. The regulatory framework proposed by IMO & MARPOL according to Annex VI concerning the gradual reduction of emissions is characterized by the implementation of operational and design limitations .For instance ,the NOx production limits have an upper bound related to the rated engine speed and the area in which the vessel operates . The three NOx emission tiers currently applied are presented in figure 6.3 .

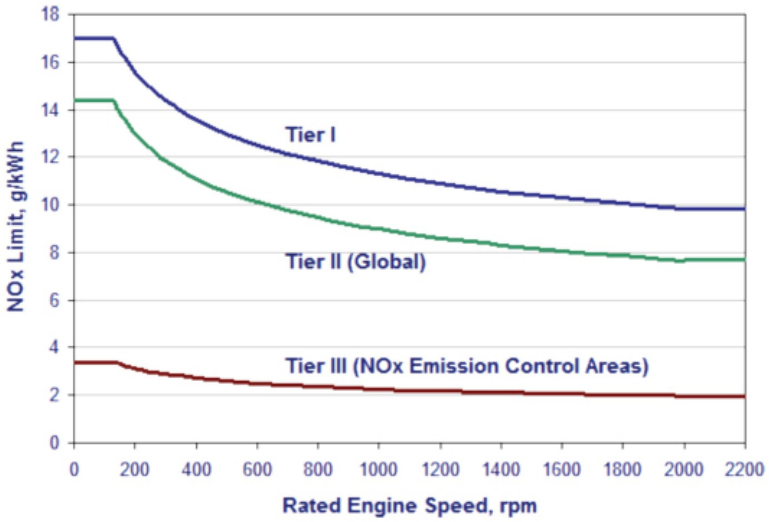

Figure 1.1: NOx Emissions Tiers

Furthermore , in the emission control areas (ECA - TierIII) efficiency metrics such as the EEDI index for new vessels and the EEXI for existing ones , are examined referring to the CO_2 emissions. The regulation mainly aimed to the increase of the hull efficiency however the trend nowadays is the direct regulation of the propulsion plant . Most vessels' propulsion plants are composed of diesel engines which are the main power suppliers . Considering that diesel engines have essentially reached their peak concerning performance and thermal efficiency , novel methods and technologies are vital in order to meet the emission reduction requirements .

According to [5] several promising technologies have been proposed. Some of them aim to decrease the power demand, (i.e. by optimizing the efficiency of the hull and propeller) and others to increase the efficiency of power plant itself. Regarding the second manner, a lot of new recent technologies promise to reduce both emissions and fuel consumption.A number of these efforts, are related with optimizing the existing diesel engines performace mance regarding the emissions, directly (e.g. the EGR), or indirectly (e.g SCR). Moreover,alternative fuels (e.g. LNG and bio-fuels) and renewable sources of power have been also proposed. Furthermore, advances in battery technologies regarding their capacitance and efficiency, have already made possible the first fully battery depended ships, employing both high energy efficiency and zero emissions. However, battery cost and limited capacitance still pose barriers which have to be overcome. An interesting solution which aims to combine the proven availability and operational efficiency of conventional propulsion manner, and the benefits of novel technologies is Hybrid Propulsion and Energy Conversion. Hybrid propulsion is an option where one or more modes of powering the ship can be utilized to optimize performance for economic, environmental or operational reasons.

A common hybrid configuration is that the different powering modes feed a common electrical bus bar from which power can be drawn for various purposes. This, however, is not necessarily the case since many examples of mechanical linkages between independent power sources have been designed and operated in ships, both past and present [5]. The key factor in order to achieve respectable higher efficiency is the control strategy. For instance, studies have shown that a 10-35 % fuel and emission reduction is possible in battery deployment and intelligent use of DC configurations by implemented appropriate control strategies.

In this context it is clear that the control methodology implemented on a hybrid propulsion system plays a crucial role if a notable efficiency improvement is the target. The current thesis examines the implementation of methodologies from the artificial intelligence spectrum in particular that of reinforcement learning . Advancements in the field of reinforcement learning has made ,the once disregarded as a purely theoretical concept , a powerful representative of the capabilities of modern machine learning methods in control applications . Control methodologies from a machine learning scope will be implemented on a hybrid diesel - electric power plant facility (HIPPO-2) with the main focus being the optimal power split between the two power sources in order to achieve good tracking control with disturbance rejection and minimization of the fuel consumption and NO_x production of the system . Before introducing the concept of reinforcement learning and presenting the experimental facility in the premises of the Laboratory of Marine Engineering (LME) a summary of previous work done on hybrid propulsion plants in general and on the HIPPO-2 test bed specifically will be examined .

1.2 Previous work

It is a well known fact that the majority of control concepts and methodologies that are eventually applied to marine hybrid propulsion stem from advancements mainly from the automotive sector . Throughout the years various concepts have been implemented successfully . Regarding the problem of the optimal energy management of hybrid powertrains the main concepts that tackle this optimization problem revolve around dynamic programming (DP) , stochastic dynamic programming (SDP) , the Pontryagin's minimization principle (PMP) , the equivalent fuel consumption strategies (ECMS) and model predictive control MPC/NMPC (linear/non linear) . ECMS is based on the concept that in charage sustaining vehicles the battery is used only as an energy buffer and all the energy ultimately comes from fuel . According to this notion the battery is considered to be a refillable auxiliary fuel tank that can only be refilled by energy produced from the system itself and not any other source outside of the main plant of the vehicle . Among the advanced control design methodologies, MPC proved to be the most promising due to its ability to optimize online a performance metric by generating a series of future projections about the time evolution of the system . Its capability to handle simultaneously multivariable processes , satisfy system

constraints and deal with long time delays while utilizing knowledge for plant disturbance response made MPC and its derivatives very appealing to the HEV industry.

In maritime applications one could choose to implement a hybrid propulsion plant with the main propulsive system being the common diesel engine with an electric motor also being connected to the main shaft line. The idea is that in high loads and speeds when the efficiency of a diesel engine is high the power will be produced by it . On the other hand, during operations that are characterized by lower speeds for example during maneuvering when the diesel engine has low efficiency the electric motor will provide the power needed . Today the most common use of electrical components in vessels is the use of shaft generators. In partial loads the diesel engine increases its efficiency by providing extra power to the shaft generator which in turn covers various electrical supply demands of the vessel .

Another common approach is the use of a hybrid power supply . In this case , a battery is added to the plant which operates as an auxiliary power source . The battery could either be charged offshore or recharged by the plant itself during operation . The second scheme is based on the ECMS strategy which seeks to find the optimal power split in order to recharge the battery online and at the same time utilize it wisely , namely at partial loads where the diesel engine is the least efficient and has the highest soot and emissions production while also caring for a reduction in the fuel consumption . This optimization problem is addressed in the works of [2,3] where the framework of non linear MPC control is implemented in an energy management and emissions minimization control system (EMEMS) and in [3,4] where indirect ICE control through fuel rate control and direct ICE control with battery considetation was applied. Both of them where experimentally tested on the HIPPO-2 testbed.

Chapter 2

Reinforcement Learning

2.1 General

Reinforcement learning is simultaneously a problem , a class of solution methods that work well on the problem and in general the field that studies this problem and its solution methods. The concepts of reinforcement learning bear resemblance to the way leaving organisms learn via interactions with the environment and the corresponding outcomes of that interaction.

The general concept of learning by interacting with a previously unknown environment is formalized in Reinforcement Learning with ideas from dynamical systems theory , specifically as the optimal control of incompletely known Markov decision processes. The basic idea revolves around an agent whose goal is to learn (the learning goal depends on the problem) by interacting with the environment over time . The specific terminology utilized in the RL context will be analyzed in a subsequent section.

Reinforcement learning algorithms belong to the broad area of AI and Machine Learning (ML) . Machine learning algorithms are categorized 2 basic types · Supervised Learning and Unsupervised Learning . Reinforcement learning differs from supervised learning which contains algorithms that support learning from a training set of labeled examples provided by a knowlegdable external supervisor . The object of this type of learning is to extrapolate/infer and generalize so as to respond well in situations not present in the training set. In interactive problems ,similar to those that RL methods seek to tackle, it is often impractical to capture first a set of examples of desired behavior that are both correct and representative of all the situations in which the agent has to act . In situations the agent has never encountered before he should be able to learn through its own experience .

Reinforcement learning differs also from unsupervised learning . The concept of unsupervised learning is typically finding hidden structures in collections of unlabeled data. Reinforcement learning and unsupervised learning share the fact that both of their methods do not rely on a labeled data set to extrapolate a behavior (as in supervised learning

) . However the algorithms' goals in each case are different , because RL methods try to maximize a signal generally called reward instead of trying to reveal a hidden pattern among unlabeled data . Uncovering those hidden structures can surely be beneficial but is not the main goal of RL methods which is solely the maximization of the reward signal.

Reinforcement learning algorithms especially those that belong to the more specific area of Deep reinforcement learning methods (where ML technics are used for approximation) have been successfully implemented in many engineering applications that have to do with optimal control and operations research .

2.2 Reinforcement Learning Elements and Formulation

In this section the basic elements needed to describe and analyze reinforcement learning algorithms are discussed.

One can identify five main elements that can define a reinforcement learning system : an agent , an environment (optionally its model) , a policy , a reward signal , a value function . As mentioned before the goal of an agent is to seek way (by interacting with his environment) to maximize his reward . Such problems are usually formulated with the help of the Markov Decision Process scheme . Starting from a given state,the agent which is the decision maker and the learner interacts with the environment by selecting actions and moves to another state . The interaction with the environment produces a reward signal that the agent recognises during his transitions between each possible state . The figure below encapsulates the afore mentioned concept .

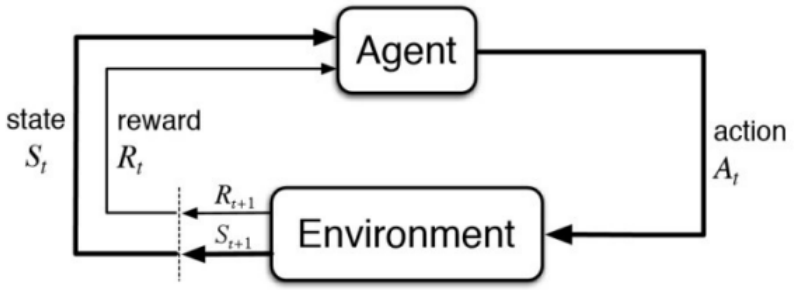

Figure 2.1: The agent - environment interaction

A Markov decision process (MDP) is defined as the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{R}_a, \mathfrak{p}\}$ where :

- $\mathcal{S}$ is the state space - the set of all possible states that the agent might encounter
- $\mathcal{A}$ is the action space - the set of all possible actions that the agent can take
- $\mathcal{R}_a$ is the reward that agent gets by taking action a

- $\mathfrak{p}(s', r|s, a)$ is the probability that if the agent takes action a from state s will end up in state s' granted a reward r

$\mathfrak{p}$ is the function that defines the dynamics of the systems described as a MDP with the probability of each possible value depending on the immediately preceding state and action . $\mathfrak{p}$ is an ordinary deterministic function with 4 parameters $\mathfrak{p} : \mathcal{S} \times \mathcal{R} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$

$$\mathfrak{p}(s', r|s, a) \doteq \mathcal{P}r(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{2.2.1}$$

For each choice of s and a $\mathfrak{p}$ defines a probability distribution

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} \mathfrak{p}(s', r|s, a) = 1, \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.2.2}$$

Return : The purpose of the agent is formalized in terms of the reward which is passed from the environment to the agent during each transition from a state to another . The reward signal is communicating to the agent what needs to be achieved but does not provide any information on how that should be achieved . The agent seeks to maximize the expected return :

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{2.2.3}$$

The additional concept of discounted returns is introduced . The agent now tries to maximize the discounted return which is defined as follows ,

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + G_{t+1} \tag{2.2.4}$$

Policy and Value Functions : The policy of an agent is essentially a mapping from the state space to probabilities , meaning that a policy is a probability function $\pi(a|s) : S \times A \to [0, 1]$ that expresses the probability that action a will be selected from the agent while he is at state s .

Value functions are functions of states (or state - action pairs) that estimate "how goo" it is for the agent to be in a given state . For MDPs one can identify 2 basic value functions .

- **State-Value function** $V_\pi(s)$ which is the expected return that the agent will gain starting from s and following policy π .

- **Action-Value function** $Q_\pi(s,a)$ which is the expected return that the agent will gain starting from s , taking action a and following policy π

$$\begin{aligned} V_\pi(s) &\doteq \mathbb{E}\left[\, G_t \mid S_t = s \,\right] \\ Q_\pi(s,a) &\doteq \mathbb{E}\left[\, G_t \mid S_t = s, A_t = a \,\right] \end{aligned} \tag{2.2.5}$$

A fundamental property of value functions used throughout reinforcement learning and dynamic programming is that they satisfy recursive relationships . For any policy π and any state s, the following consistency condition holds between the value of s and the value of its possible successor states . This recursive property is known as Bellman Equation

$$V_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) \sum_{s'\in\mathcal{S}, r\in\mathcal{R}} \mathfrak{p}(s', r|s, a)\left[r + \gamma V_\pi(s')\right] \tag{2.2.6}$$

The Bellman equation expresses a relationship between the value of a state and the values of its successor states by averaging over all the possibilities weighting each one by its probability of occurring .

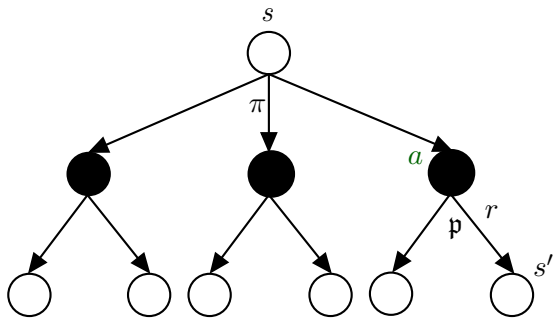

Figure 2.2: Backup Diagram - Bellman Equation

Another essential concept for the reinforcement learning theory is that of optimal policy and value functions .

Definition: A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states .

$$\pi \geq \pi' \; if \; and \; only \; if \; V_\pi(s) \geq V_{\pi'}(s) \; \forall \; s \in \mathcal{S} \tag{2.2.7}$$

From the definition above one can naturally define the optimal state - value and action - value functions corresponding to the optimal policy π_*

$$V_*(s) = \max V_\pi(s) \ \forall s \in \mathcal{S} \tag{2.2.8}$$
$$Q_*(s,a) = \max Q_\pi(s,a) \ \forall s \in \mathcal{S} \tag{2.2.9}$$
$$V_*(s) = \max Q_*(s,a) \ over \ all \ a \in \mathcal{A}(s) \tag{2.2.10}$$

From equations 2.2.8 Belmman optimality equations derives

$$V_\pi(s) = max_a \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} \mathfrak{p}(s', r|s, a) \left[r + \gamma V_{\pi *}(s')\right] \tag{2.2.11}$$
$$Q_\pi(s,a) = max_a \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} \mathfrak{p}(s', r|s, a) \left[r + \gamma Q_{\pi *}(s', a)\right] \tag{2.2.12}$$

Equations 2.2.11 and 2.2.12 are systems of equations for each state , meaning that with n states there are n equations in n unknowns . For a known $\mathfrak{p}(s', r|s, a)$ the system could be solved with methods for non-linear systems . Dynamic programming (DP) which refers to a collection of algorithms that can be used to compute optimal policies essentially solve the optimal Bellman system of equations . However (DP) is generally limited in RL due to the important assumption of a perfect model i.e a model whose MPD dynamic $\mathfrak{p}(s', r|s, a)$ is completely known . This is an ideal assumption that in most real-world applications is not true .

2.3 Basic RL Concepts

Now that the basic foundation of RL has been layed , the various RL algorithms implemented to handle many real world problems will be discussed . Before taxonomizing the different RL algorithms that have been developed there are some basic concepts that govern them and should be addressed in order to obtain a common understanding of the RL "lingo" .

Exploration vs Exploitation One of the common challenges that arise in reinforcement learning is the trade off between exploration and exploitation. Agent's goal is to maximize its reward received from the environment when taking an action . For this reason , the agent will tend to prefer actions that it has already tried and produced good results in terms of the reward it was granted . This tendency of the agent is referred to as *exploitation* . The agent exploits its experience and prefers

only the actions that produce good return . However , the agent also needs to *explore* new possibilities while learning with the hope of discovering better routes to achieve its goal in the long run . in general , neither of those can be pursued extensively without failing a task and this is the reason that a balance between them is (for now) the key to obtain a well trained agent that can generalize .

On policy vs Off policy learning The exploration - exploitation trade - off introduced before paves the way for the introduction of another important concept that divides RL learning methods . The goal of the agent is learning about the optimal policy while behaving according to an exploratory policy . The agent needs to seek learning action values subsequent to optimal behavior while also behaving non optimally in order to explore all actions . To achieve that we first introduce the concepts of on policy and off policy methods . On policy methods sample actions from the same policy (behavior policy) that they try to optimize (target policy) . Off policy methods on the other hand separate the target policy (i.e the policy that it they try to optimize) from the behavior policy (i.e the policy that they sample actions from). Later the concept of importance sampling will be introduced , a general technique for estimating expected values under one distribution given samples from another .

2.4 RL algorithms - Taxonomy and Comparison

Sutton and Barto in their well renowned book *"Reinforcement Learning : An Introduction "* they approach RL algorithms by introducing categories , the **tabular** solution methods and the **approximation** methods. Tabular methods are used in problems in which the action and the state space are small enough for the approximate value functions to be represented as arrays or tables . Approximation methods come into play in cases where the state space is large. A large state space does not only mean excessive memory for large tables that fit them but also large time and data to fit them accurately . Approximation methods are basically an extension to tabular methods for large state spaces .

In general an important first branching point in a RL algorithm answers the question *Does the agent have access to (or learns) a model of the environment ?* . The key word here is the model of the environment which , as already mentioned , is a function that predicts state transitions and the corresponding rewards . So one can make a distinction between :

- **Model - free** methods that do not rely on the knowledge of an transition function .
- **Model - based** methods that do rely on a transition model .

Having a transition model of the environment would be ideal because it would allow the agent to plan ahead by simulating what would happen for a range of possible choices and explicitly deciding between those options. Unfortunately , in most real life applications a model that accurately captures the dynamics of the system is not available to the agent . For this reason , model - free algorithms are to this day the ones mostly used and developed .

The next common branching point in the RL algorithms taxonomy concerns *what the agent is learning* during its training . We will focus mainly on the model - free spectrum because the algorithms that will be used in this study stem from it . There are two main approaches to training a model - free RL agent :

- **Q-Learning**
- **Policy Optimization**
- **An interpolation between those**

Q-Learning methods that belong to this family guide the agent to learn an approximator of the state-action value function $Q_\theta(s, a)$ for the optimal state-action value function $Q_*(s, a)$. The objective function used is typically based on the Bellman Equation (2.2.6) . This optimization is mostly performed **off-policy**. The actions taken by the Q-learning agent are given by

$$a(s) = argmax_a Q_\theta(s, a) \tag{2.4.1}$$

And the $Q(s, a)$ are updated via :

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R + \gamma max_a Q(s', a) - Q(s, a)] \tag{2.4.2}$$

Q-Learning is also known as Off-policy Tempordal Difference (TD) control . TD methods update estimates of the optimal state - action value functions based in part on other learned estimates (bootstrapping) . Q-Learning methods as well as Policy Optimization methods and their combinations do not rely on the knowledge of the MDP model of the environment because they learn from raw experience .

Policy Optimization methods represent a policy $\pi(a|s)$ that is parameterized by some parameters ϑ , $\pi_\theta(a|s)$. These parameters are optimized either directly bu gradient ascent on the performance objectives $J(\pi_\theta(a|s))$, or indirectly by maximizing local approximations of $J(\pi_\theta(a|s))$. This optimization is almost always performed

on-policy . Policy optimization also usually involves learning an approximator $V_{\varphi}(s)$ for the on - policy value function $V_{\pi}(s)$ which gets used in figuring out how to update the policy. Policy Optimization will be analyzed in detail later as the PPO algorithm , a famous policy optimization algorithm , will be used to tackle the control problem of the hybrid marine propulsion plant .

Policy Optimization and Q-Learning trade offs. The primary strength of policy optimization methods is that they directly seek to optimize the policy directly . By targeting the policy optimization directly the tend to be more stable and reliable . Q - learning methods , on the other hand , approximate the optimal policy by optimizing the state - value function that is indirectly . For this reason Q - learning methods tend to be less stable and require extended hyperparameter tuning to achieve convergence . However these methods tend to be also more sample efficient meaning that they can reuse data more effectively than policy optimaztion techniques .

Model - Free RL		
Policy Optimization	**Q - Learning**	**Interpolation Between Them**
Policy Gradient	DQN	DDPG
A2C/A3C	C51	SAC
PPO / TRPO	QR-DQN	TD3

Table 2.1: Model - Free reinforcement learning algorithms

Chapter 3

Propulsion Plant Description & Modeling

In this chapter, the experimental powertrain facility Hybrid Intergrated Propulsion POwertrain *HIPPO-2* that belongs to the laboratory of marine engineering (LME) at the school of naval architecture and marine engineering (NTUA) is presented. The work of this thesis is based on the modeling of the various components of HIPPO-2 that resulted from data driven analysis resulting from data acquired from a series of physical system integrated sensors installed by the engine manufacturer as well as virtual sensors developed for the prediction of the engine emissions . The mathematical models that approximate the system's behavior are the results of the work described analytically in [2,3] . These models are characterized by high accuracy and are deemed sufficient enough to be directly used in the current project .

3.1 HIPPO-2 Testbed

The facility is composed of three major components, Internal Combustion Engine (ICE), Electric Motor/Generator (EM) and Electric Brake (EB), which applies the load torque to the system. In addition, a virtual Battery (B) was also considered, which was simulated in parallel during the experimental test via the control platform.The HIPPO-2 hybrid diesel-electric power plant consists of a internal combustion engine (ICE) in serial connection to an electric motor (EM). In this configuration, the rotational speed of the ICE and the EM are identical and the supplied torques add together to maintain the total torque demand applied by a electric motor brake (EB).

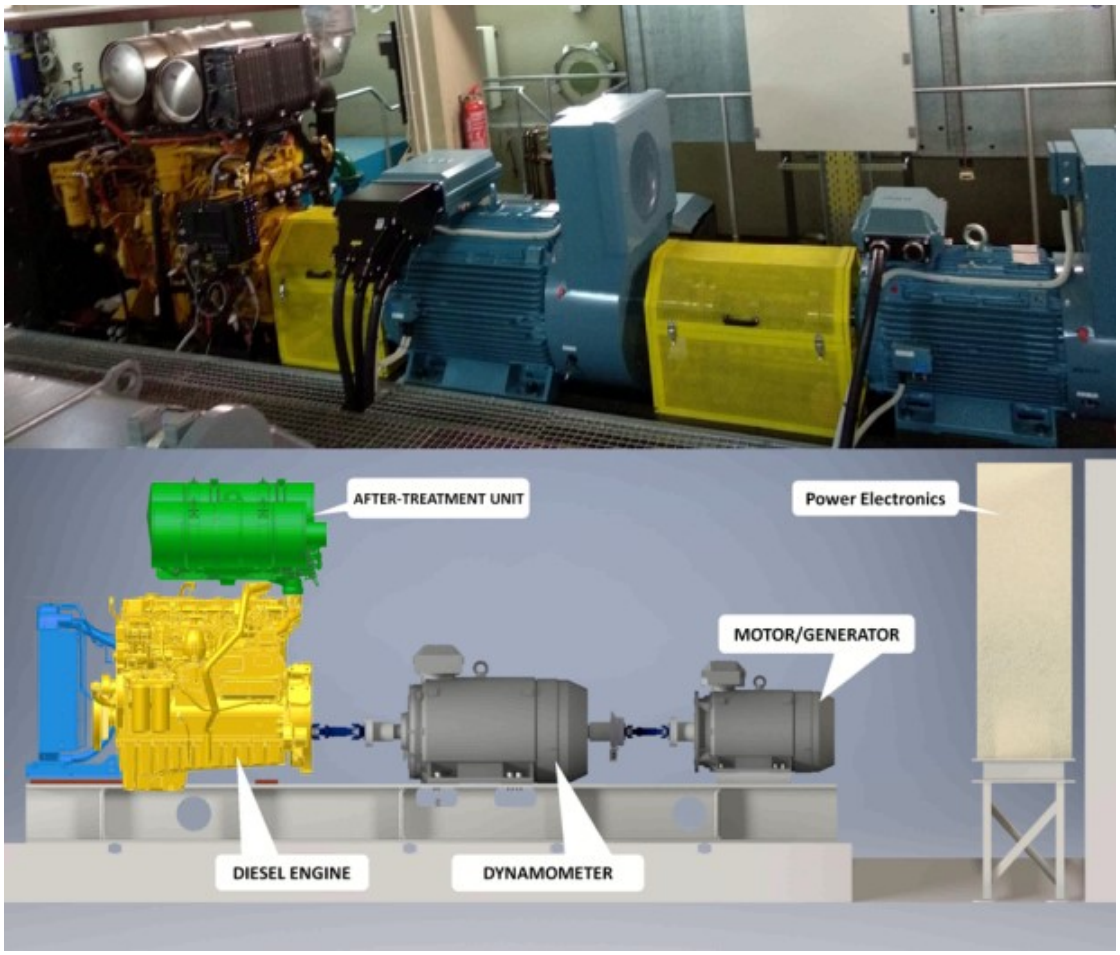

Figure 3.1: The HIPPO-2 hybrid diesel-electric testbed of LME

HIPPO-2 Integration

The ICE is a turbocharged CATERPILLAR R 6-cylinder 9.3-liter 4-stroke industrial diesel engine, model C9.3 ACERTTM, producing 261 kW at 2200 rpm and maximum torque 1596 Nm at 1400 rpm. The loading diagram of engine is shown in (Rating C). Accordingto the speed reference and the deviation of speed measurement, the electronic control unit (ECU) of the ICE controls the fuel injection in the cylinders in closed loop control, using controller in the form of look-up tables. The engine is designed to meet U.S. EPA Tier 4 Final, EU Stage IV emission standards. Exhaust Gas Recirculation (EGR) and Selective Catalyst Reducer (SCR) systems for NOx reduction, are also incorporated, along with a Diesel Particulate Filter (DPF).

The EM is a standard AC induction 3-phase motor, with a rated power of 90 kW at 1483 rpm, type M3BP 280SMB 4 IMB3/IM1001, manufactured by ABB R. The EM can operate both as motor, to assist the engine, and as generator to store energy. The EB is a standard AC induction 3-phase motor manufactured also by ABB R , type M3BP 355SMB 4 IMB3/IM1001, with 315 kW load capacity, operating at 1488 rpm The 3 motors are connected in series, thus the operating speed range of HIPPO-2 is from 600 to 2200 rpm, with maximum load of 341 kW (ICE and EM combined power).

3.2 Plant Components Analysis & Modeling

3.2.1 Diesel Engine

General

The main attributes of diesel engines and consequently the reason of their dominance as the main powering device in industry, over the past century, is their relatively high power/weight ratio and their relatively high thermal efficiency [17]. Four stroke turbocharged Diesels can reach efficiency of approximately 40 % .Two key factors are responsible for the above.

The first is the increased compression ratio. When the working fluid is compressed, its temperature rises, leading to increased thermal efficiency. Since, the compressed fluid is consisted only from oxidizer (air), there is no self-ignition problem. The fuel ejects and the mixture ignites, at the desired crankshaft angle. The second is that diesel engine can operate with lean mixtures of air and fuel in cylinder, such that throttling of the intake air can be completely avoided, something which is possible due to extremely hot air in cylinder. Consequently, the high thermal efficiency is maintained to a certain degree for part load operations.

The operation of diesel engines is associated with two major drawbacks [18].

The first is related to the low power-density they exhibit. This occurs from the fact that the mixture inside the combustion chamber is always lean, and thus less fuel can be burned in atmospheric conditions inside a cylinder. Moreover, engine maximum speed is relatively low due to mechanical limitations. This problem is sufficiently addressed with super or turbocharging the engine, namely, compressing the air before enters the cylinder, allowing more fuel to be burnt, in this way. The second disadvantage refers to issues of the exhaust gas purification. Apart of the ideal products of combustion, which are water (H2O) and carbon dioxide (CO2), several by products are also produced. A part of them are harmful to environment or cause health problems to humans. Therefore, numerous legislation, aiming to reduce the above effects, have been applied since the early 1970s for the automotive, and the late 1990s for marine industry, implementing limits to their concentration in the exhaust gases. The main pollutants that the above limits apply are nitrogen oxides (NOX), unburned hydrocarbons (HC), carbon monoxide (CO) and soot. Key factors for the concentration of above is ratio of air-fuel compered to stoichiometric (λ) and the cylinder temperature.

Although, Diesel engines have lower raw emissions than Otto , their working principle (i.e. lean operation), exclude the solution of the the three-ways-catalysts, since this system requires stoichiometric air-fuel ratios. Consequently, other technologies have developed, in order to restrict the above emissions, continually lowering fuel consumption and optimizing performance at the same time. These options after-treat the exhaust gas (e.g. Selective Catalyst Reducer - SCR), or affect the operation of engine itself (e.g. Exhaust Gas Recirculation - EGR).

Transient operations behavior

Most of the engine-oriented literature is focused on steady state operation, although transient applications represent a large portion of the engine operating patterns (e.g. maneuvering conditions for ships), or even the majority of operations (e.g. automotive vehicles). In recent years, due to the latest regulatory framework regarding the engine emissions, more attention is given in regard to this operation mode. According to literature [19], during transient loading profiles, gaseous and noise emissions typically exceed their acceptable values following the extreme, nonlinear and non-steady-state conditions experienced during dynamic engine operation. For instance, 50 % of NOx emissions from automotive engines during the European Driving Cycle stem from periods of acceleration, whereas instantaneous particulate matter and NOx emissions during load increase transients have been measured to be 1 to 2 orders of magnitude higher than their respective quasi-steady values.

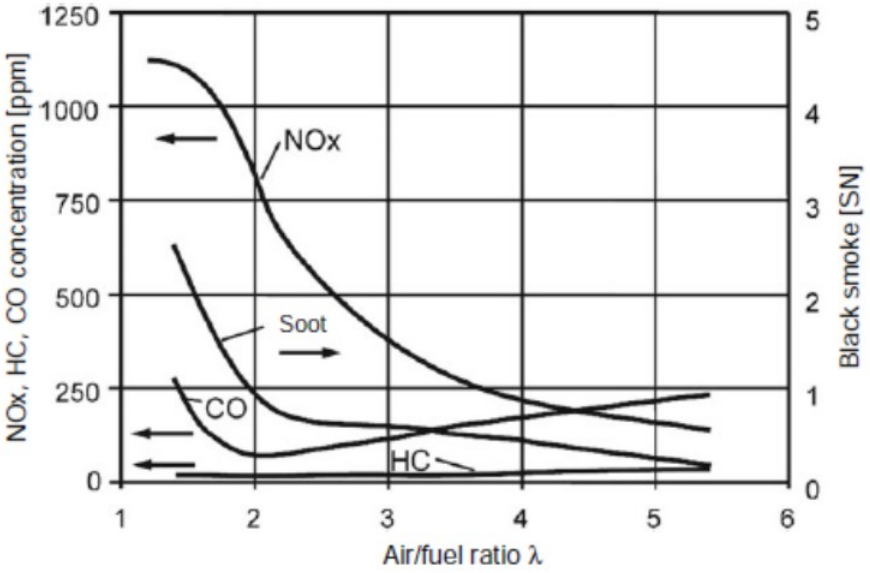

Figure 3.2: Byproducts' relation to Air/Fuel ratio

[Handbook of Diesel Engines]

3.2.2 Diesel Engine Modeling

The dominance of the diesel engine as the main powering device in the heavy industry has contributed to the development of various modeling methodologies that opt for optimizing their use . Depending on the application there is a vast majority of modeling techniques that could be used , each of which can approximate the operation of a diesel engine to a specific accuracy . In [6] the diesel models are classified by the dependencies of the model parameters namely into lumped models which are described by ordinary differential equations and distributed systems the parameters of which are described by partial differential equations . One can also distinguish the various models between crank-angle resolution models or cycle averaged models and their formulation as mean - value models or as discrete events models . These types of models described can usually get too complex to be used for control applications. For control oriented applications linear low - order models are generally preferred. In the current thesis for the modelling of the engine dynamics such models are used derived from fitting polynomials of a certain degree to data acquired from the experiments . For example a parameter P depended on variables x and y is modeled by a linear combination of mononomials of a given degree d

$$P = \theta^T \cdot f_d(x,y) \quad where,$$
$$f_d(x,y) = [1 \;\; x \;\; y \;\; xy \;\; x^2 \;\; y^2 \;\; ... \;\; x^d y \;\; y^d x \;\; x^d \;\; y^d]$$

3.2.3 Electric Motor-Generator

In hybrid propulsion plants, the electric machines are a key component, and usually they are reversible (i.e. the can operate both as motor and generator). The operation of these machines, according to [17], can be distinguished into three modes, one motoring and two generating, which are: (1) to convert the electrical power from the battery into mechanical power to drive the vehicle, (2) to convert the mechanical power from the engine into electrical power to recharge the battery, or (3) to recuperate mechanical power available at the drive train to recharge the battery (regenerative braking). Of course, in a marine hybrid plant the latter is not so common during operation, except maneuvering or special cases (e.g. during propeller ventilation the electric part can absorb and store to battery a portion of kinetic energy in order to reduce over-speeding). Desirable characteristics, for electric machines operating in a hybrid propulsion plant are [17]: high efficiency, low cost, high specific power, good controllability, fault tolerance, low noise, and uniformity of operation (low torque fluctuations).

The electric machines which are used in propulsion plants are rotating machines, with two major components, the stator and the rotor. The later is connected to the moving part of the machine (output shaft), in which the acting torque is applied. These electric machines are categorized into two major types according to current supply, direct current (DC) motors and alternating-current (AC) motors. For each category there are others sub-categories. At the experimental test-bed HIPPO 2, the motor is an AC asynchronous squirrel cage motor, and therefore only this motor type will be further described.

In AC motors in general, AC voltage is applied to stator, creating a rotating magnetic field in the stator wingdings, which are, for three phase motors, one or more sets of three.This magnetic field changes its orientation according to sign of current flowing in the wingdings, which is continuously varying, and consequently the orientation of the magnetic field keeps varying, resulting in a rotating magnetic field. The speed of the rotating magnetic field is called the synchronous speed. It equals the pulsation of the three-phase AC voltage divided by the number of poles. In asynchronous AC machines (also called induction motors), the rotor does not rotate with the same speed of the magnetic field when load to shaft is applied. The rotor usually hosts a set of conductors with end rings, an arrangement known as "squirrel cage". Electromotive force and thus current is induced in the rotor windings by the

interaction of the conductors with the rotating magnetic field the rotor becomes an electromagnet with alternating poles, attracted by those of the rotating magnetic field of the stator.

Control of these motors is conducted with sophisticated electronics (inverters), and various control schemes have been applied in the past, such as the Scalar Control (V/f Control), the Field Oriented Control, Sliding Control Mode (SMC) and the Direct Torque Control (DTC). The industrial Drive which controls the AC motor of HIPPO 2 is using the DTC control scheme. This type of control is based on the mathematical approach of induction machines, and therefore various parameters, such as stator resistance, mutual inductance, saturation co-efficiency, etc. are required. The control variables are output torque and stator magnetic flux. DTC is able to control more accurate and has the fastest response time, does not need feedback devices and has reduced mechanical failure. The disadvantage is that due to the inherent hysteresis of the comparator, higher torque and flux ripple exist

AC Induction Motor Modeling

Modeling approaches in general are categorized into quasi-static and dynamic modeling . Dynamic approach refers to the complete and pretty accurate approximation of a dynamical system usually by a number of differential equations (ordinary or even partial).Dynamical systems are of great use for system that are used for simulations that try to depict the real progress of a specific phenomenon and are usually characterized by high computational cost (memory and time-wise).For this reason detailed dynamic models are rarely used in control engineering . Quasi - static modeling ,on the other hand, refers to the construction of models that ignore the dynamic responses of the system and are applied in problems that under logical assumptions that is possible . For example , in problems where the dynamic response of the system is significantly lower in magnitude or in the time needed to return to a steady state condition , than the sample time of the physical system's controller the quasi static model's simplicity with an acceptable accuracy trade off is preferred.

HIPPO-2 AC motor is control via a DTC scheme . According to the manufacturer [21], the dynamic response of a DTC driven AC motor to 100 % torque step is typically 1–5 milliseconds (ms), which approaches the motor's physical limit. The sample time of the RL-based controller of the thesis is set to be 100 ms and for this reason the AC induction motor dynamics can be completely disregarded without impacting the controller's behavior. A quasi - static model was then utilized to describe the electric part of the HIPPO-2 Testbed .

Willans Approach

Willans approach is simple quasi static approach that can be generally used in energy converters where the input and output powers are considered to be linearly dependent .

$$P_2(t) = e \cdot P_1(t) - P_0 \tag{3.2.1}$$

where ,

- P_1 is the power need in one form of energy to get P_2 power in the other form of energy
- P_0 the power losses occurring after the energy conversion (friction , heat losses ,etc.)
- e is defined as the maximum efficiency namely the efficiency that can be obtained when P_0 is zero . Thus e represents the efficiency of the energy conversion process only (in particular electrical to mechanical and vice versa).

For an electric motor this approach takes the form of

$$Q_{em}(t) \cdot \omega_{em}(t) = e \cdot P_{em}(t) - P_0 \ , \qquad P_{em} \geq 0 \qquad (generating) \tag{3.2.2}$$

$$Q_{em}(t) \cdot \omega_{em}(t) = \frac{P_{em}(t)}{e} - P_0 \ , \qquad P_{em} < 0 \qquad (motoring) \tag{3.2.3}$$

where $P_{em}(t)$ is the input power , $\omega_{em}(t)$ is the rotational speed and $Q_{em}(t)$ is the output torque . In contrast with other quasi-static approaches, this model can describe the effect of the "idle" losses P_0 for small values of power . In this circumstance, it may occur that $P_{em} > 0$ while being $Q_{em}(t) \cdot \omega_{em}(t) < 0$, that is, even if some mechanical power is available from the downstream powertrain, still the energy source must supply a certain amount of electric power .

Moreover, it is suggested that parameters e and P_0 are dependent on motor speed $\omega_{em}(t)$. However, good average approximations can be found and used without significant error. In order to validate that, a Willans model was fitted to HIPPO's 2 AC motor data and then its compared to experimental results. The data used for fitting was derived from the manufacturer's data-sheets for operating points 25 %, 50 %, 75 %, and 100 % of maximum load under nominal voltage. The result fitting

of Willans model and its comparison with the experimental results [3] is shown at Figs 3.3,3.4.

Considering the above figures, it is clear that model can predict the required Power demand satisfactory enough, for low and high loads for both motoring and generating modes.

Figure 3.3: Willans model fitting results , compared to test data from manufacturer [3]

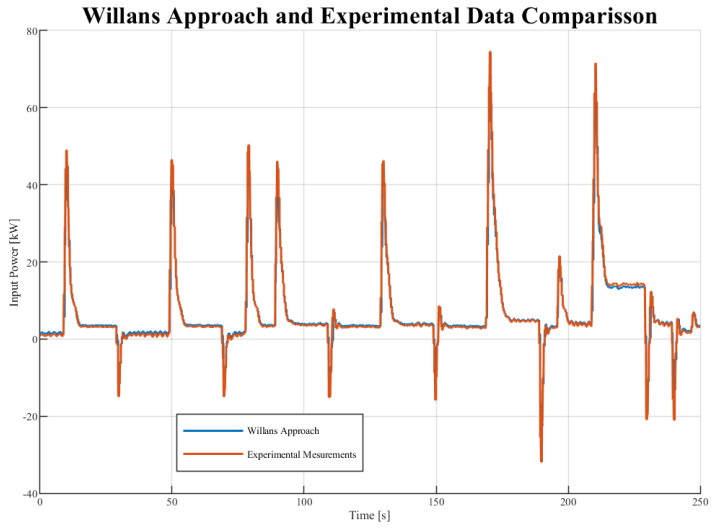

Figure 3.4: Willans model comparison with experimental data [3]

3.2.4 Battery modeling

Electrochemical batteries are key components of Hybrid Diesel - Electric Ships and Vehicles (HEV), in general. The main function of these devices, in a hybrid propulsion plant, is to transform and store electrical energy in chemical form, and then re-transform it back to electricity, in order to be used by the electric motors, when it is required.

Each battery cell is characterized by the maximum power it can provide to the propulsion plant, and the nominal capacity. The first refers to the rate of energy a battery can provide to the plant, and it is the product of current and voltage. The latter, defines the amount of electricity a battery can supply, in terms of Coulombs (Ampere-seconds, As) or more often in Ampere-hours, Ah. Also, a dimensionless parameter, the State of Charge (SoC), describes the remaining capacity of the battery, and it is expressed as percentage or fraction of the nominal capacity.

$$SoC(t) = \frac{Q(t)}{Q_{nom}} \tag{3.2.4}$$

Battery charge, and therefore SoC, is difficult to be measured directly. Subsequently, it is calculated indirectly, from the charge equilibrium which is expressed from

$$\dot{Q}(t) = -I_b(t) \tag{3.2.5}$$

It is common, in case of battery charge, a parameter to be taken into account, the coulombic or charging efficiency nc [22–24] in order to model the fact that a fraction current is not transformed into charge of the battery current due to irreversible, parasitic reactions taking place in the battery. Therefore, the above equation takes the form

$$\dot{Q}(t) = -n_c \cdot I_b(t) \tag{3.2.6}$$

Furthermore, it should be noted that not the whole capacity of the battery can be used in practice [17]. There is SoC window, whose limits define the maximum SoC that can be achieved during charging, and the minimum SoC that can be reached during discharging, in order to maximize battery life. This feature is expressed by the specific energy of battery.

Moreover, a group of attributes of a battery pack, operating in a hybrid power plant, are usually required [17], most important of which are high specific energy, high specific power, long calendar and cycle life,low initial and replacement costs, high reliability, wide range of operating temperatures and high robustness.

Quasi-static model for controller design

The model that will be used in the thesis for the approximation of the battery behavior (charging / discharging) derives from the Equivalent Circuit Method (ECM) . This method approximates the battery charge/discharge cycle well enough in the region between 10-20% to 80-90% for the SoC . Beyond those limits the battery behavior in respect to its voltage presents non linear characteristics and thus during the simulation the SoC will be bounded by the aforementioned limits . A basic physical model of a battery can be derived by considering an equivalent curcuit of the system such as the one shown in figure 3.5

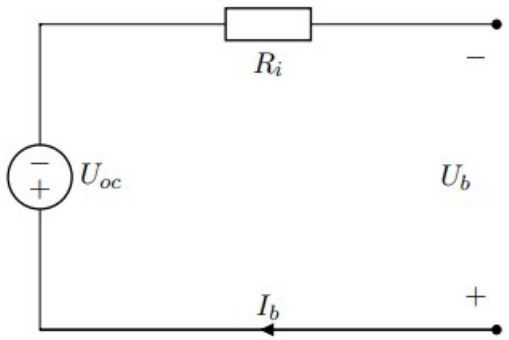

Figure 3.5: EC of the battery for the quasi static model [17]

In this curcuit , the battery is represented by an ideal open - curcuit voltage source in series with an internal resistance. Kirchhoff's voltage law for the equivalent circuit yields the equation

$$U_{oc}(t) - R_i(t) \cdot I_b(t) = U_b(t) \tag{3.2.7}$$

U_{oc} is the open source voltage and is the equilibrium potential of the battery . Since this quantity depends on the charge level , it is parameterized using the following affine relationship

$$U_{oc}(t) = k_2 \cdot Q(t) + k_1 \tag{3.2.8}$$

The constants k_1 and k_2 depend only on the battery construction and the number of cells but not on operative variables and thus can be considered constant with time

. The linearity expressed in equation 3.2.8 is equivalent to a constant voltage source in series with a capacitor with constant capacitance . By considering that $k_2 = \frac{C_b}{Q_{norm}}$ where C_b is a bulk capacitance Equation 3.2.8 can be rewritten as

$$U_{oc}(t) = k_2 \cdot SOC(t) + k_1 \tag{3.2.9}$$

As for the internal resistance R_i , it takes into account several phenomena. In principle, it is the combination of three contributions. The first is the ohmic resistance R_o, i.e., the series of the ohmic resistance in the electrolyte, in the electrodes, and in the interconnections and battery terminals. The second contribution is the charge-transfer resistance R_{ct}, associated with the "charge-transfer" (i.e., involving electrons) reactions taking place at the electrodes. The third contribution is the diffusion or concentration resistance R_d, associated with the diffusion of ions in the electrolyte due to concentration gradients. Thus, in principle, the internal resistance of a battery is calculated as the sum of those resistances

$$R_i(t) = R_d + R_{ct} + R_o$$

Internal resistance's components are approximated in the literature by various models that can contain a large number of parameters . For the purpose of this thesis an internal resistance modeled by an affine combination as for the OC voltage is satisfactory

$$R_i(t) = k_4 \cdot SOC(t) + k_3 \tag{3.2.10}$$

In the present work , the battery modeling is intended for the application of control methodologies on a propulsion system . For this reason the input variable for the estimation of the SoC is wise to be the required power $P_b(t)$. Obviously the input power $P_b(t)$ is related to the current and the terminal voltage in the following manner

$$I_b(t) = \frac{P_b(t)}{U_b(t)} \tag{3.2.11}$$

Plugging equation 3.2.11 into equation 3.2.7 a quadratic equation with respect to the terminal voltage is obtained

$$U_b^2(t) - U_{oc}(t) \cdot U_b(t) + P_b(t) \cdot R_i(t) = 0 \tag{3.2.12}$$

which has the solution

$$U_b(t) = \frac{U_{oc}(t)}{2} + \sqrt{\frac{U_{oc}^2(t)}{4} - P_b \cdot R_i(t)} \tag{3.2.13}$$

Substituting the terminal voltage into equation 3.2.11 and multiplying with the conjugate of the denominator the battery current and the input power take the form

$$I_b(t) = \frac{U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4P_b(t) \cdot R_i(t)}}{2R_i(t)} \tag{3.2.14}$$

$$P_b(t) = \frac{-U_b^2(t) + U_b(t) \cdot U_{oc}(t)}{R_i(t)} \tag{3.2.15}$$

Finally by using equation 3.2.14 and the definitions 3.2.4 and 3.2.5 a differential equation for the state of charge is formulated :

$$\frac{dSOC}{dt} = -\frac{100}{Q_{norm}} \cdot \frac{U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4P_b(t) \cdot R_i(t)}}{2R_i(t)} \tag{3.2.16}$$

Battery Operating Limits

The operating limits of the above model , can be derived from the equations of the previous section . For the discharging case , the limitations applied are

1. $P_b \geq 0$ where the equality stands for $U_b = 0$ *or* $U_b = U_{oc}$
2. $U_b < U_{oc}$

The maximum power can be found either by differentiating the input power equation or by noticing from equation 3.2.13 that the domain of definition of that function requires that

$$\frac{U_{oc}^2}{4} - P_b \cdot R_i(t) \geq 0 \Longrightarrow Pb \leq \frac{U_{oc}^2}{4R_i(t)} \Longrightarrow Pb_{max}(t) = \frac{U_{oc}^2}{4R_i(t)} \tag{3.2.17}$$

The maximum is achieved at

$$U_{b,P_{max}} = \frac{U_{oc}(t)}{2} \ , \ I_{b,Pmax}(t) = \frac{U_{oc}}{2R_i(t)}$$

Notice that the maximum power depends on OCV and by extension to the SoC , which means that the maximum power is actually time varying .

Also, in case of control schemes involving battery cell components, additional constraints regarding the battery function are applied, such as rate of SoC alteration, maximum current and voltage, etc. [36]. These constraints refer to battery health, and the are usually defined by the manufacturer. In this work, the battery component is virtual i.e. its model running on the software during experimental tests, and therefore, no further analysis is conducted .

3.3 Summary of HIPPO-2 Modeling

In this section a summary of the HIPPO-2 modeling is laid out by introducing the equations that govern the system of and the mathematical models that where used to approximate the function of each component of the HIPPO-2 powertrain.The models introduced in this section are directly cited from the work of [2,3,4] who used them for the control of HIPPO-2 using the nonlinear model predictive control methodology (NMPC) . The models are deemed accurate enough and suitable for the modeling of the reinforcement learning environment and for this reason the production of a different set of models was considered beyond the scope of the current thesis .

3.3.1 Equations that govern the system

Rotational shaft dynamics

The rotational dynamic behavior of the power plant derives from the application of Newton's second law for rotation on the system .

$$\frac{d\omega_{eng}}{dt} = \frac{1}{J_{system}}(Q_{ice} + Q_{em} - Q_{load}) \tag{3.3.1}$$

where ω_{eng} is the engine shaft rotational speed, J_{system} is the powertrain moment of inertia at the engine side, Q_{ice} is the brake torque of the engine delivered at the shaft, Q_{em} is the output torque of the electric motor/generator (positive if the EM is motoring), and Q_{load} is the torque load which is applied to the powertrain at the engine side of the gearbox.

Diesel engine control-oriented model

In section 3.2.2 it was mentioned that for control oriented applications high order models that tend to be more computationally costly are generally avoided . For this

reason low order models are preferred and will be used in this work . The engine brake torque output Q_{ice} and the fuel consumption $\dot{m}_f$ are modelled by the following manner [2]

$$Q_{ice} = \boldsymbol{\vartheta}_Q \cdot \left[1 u_{ice}\ N_{eng}\ N_{eng}^2\right]^T \tag{3.3.2}$$

$$\dot{m}_f = \rho_f(f, T_f) \cdot \boldsymbol{\vartheta}_f \cdot \left[1\ u_{ice}\ N_{eng}\ N_{eng} u_{ice}\ u_{ice}^2\ N_{eng}^2\right]^T \tag{3.3.3}$$

The polynomial models above contain the electronic fuel index u_{ice} which is fed to the engine ECU and the engine speed in rpm N_{eng} . The vectors $\boldsymbol{\vartheta}_Q, \boldsymbol{\vartheta}_f$ are the regressors that weight each component of the polynomials vector suitably to fit the experimental data acquired from HIPPO-2 . In equation 3.3.2 the terms containing the engine speed refer to torque losses due to shaft frictions . Finally , ρ_f is the measured fuel density (depended on the fuel quality and its temperature T_f). The aforementioned models where produced and used in [2] and the fitting accuracy achieved is 0.994 and 0.995 respectively .

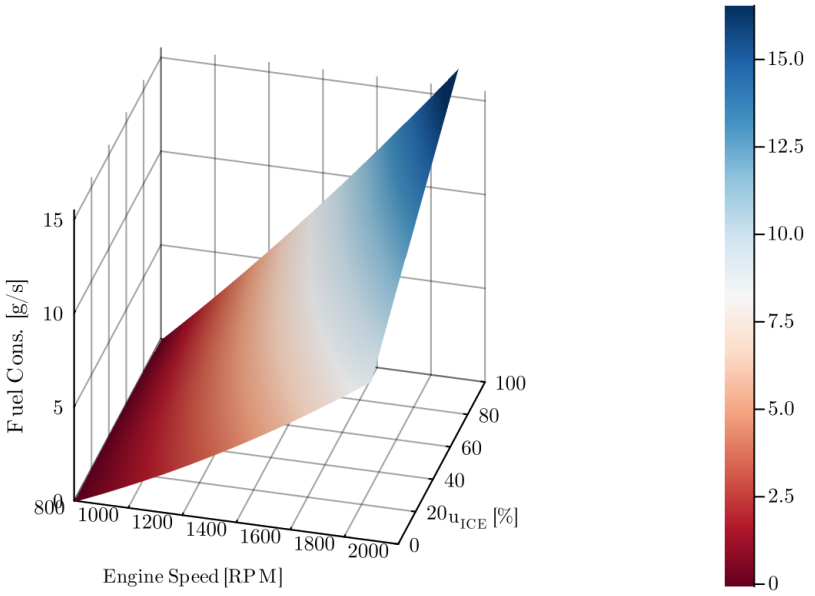

Figure 3.6: Diesel Engine Fuel Consumption model Eq.3.3.3

NOx emissions modeling As far as the NOx emissions of the diesel engine are concerned, the modeling problem becomes more complicated. The NOx production depends not only on the engine operating point but also on the operation of a high-pressure Exhaust Gas Recirculation (EGR) system. However, in the HIPPO-2 testbed, the EGR actuator is controlled by the engine Electronic Control Unit (ECU) and it cannot be regulated with an external input signal. Besides, the control law of the EGR system cannot be approximated due to the lack of specific measurements such as the intake manifold air mass flow rate which is not available. As such, it was decided to be excluded from the NOx model . The model created is based on sigmoid functions that are generally capable of capturing non-linear patterns . The model produced by [2] is the following :

$$\dot{m}_N = b_{N_1}[a_{N_1} - a_{N_2}\sigma_1(z_1)]u_{ice}^{a_{N_4}} N_{eng} - b_{N_2}\sigma_2(z_2)\sigma_3(z_3)u_{ice}N_{eng} \tag{3.3.4}$$

where $\sigma(z_i)$ is the sigmoid function and

- $z_1 = a_{N_3}(N_{eng} - N_{eng,N_0})$
- $z_2 = a_{N_5}(u_{ice} - u_{ice,N_1})$
- $z_3 = a_{N_6}(N_{eng} - N_{eng,N_1})$

$N_{eng,N_0}, N_{eng,N_1}, u_{ice,N_1}$ are specific values that influence the threshold of the sigmoid functions and are chosen so as to capture the trend of the NOx production function around the local minima observed when the NOx production function is plotted w.r.t u_{ice}, N_{eng} . The parameters a_{N_i}, b_{N_i} are fitted to the measured data with the model achieving an accuracy of 0.953 .

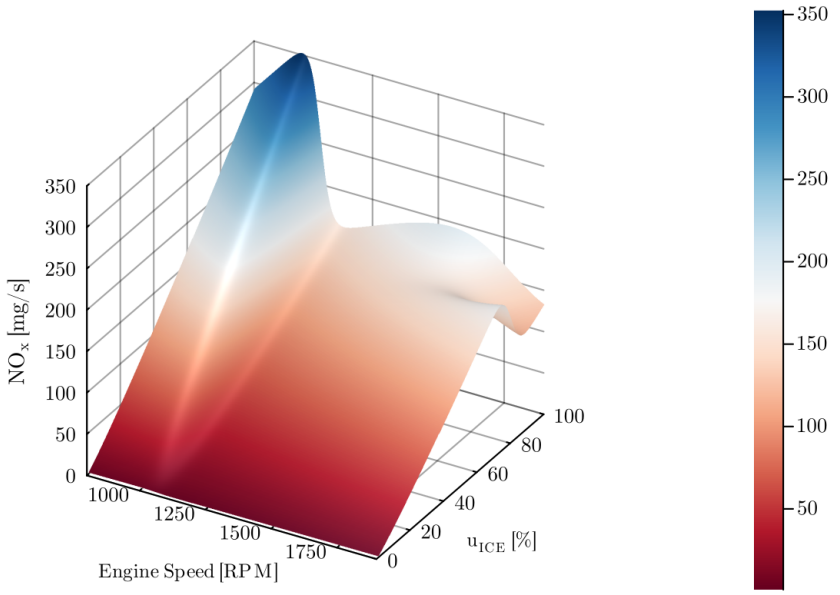

Figure 3.7: Diesel Engine NOx emissions behavior Eq. 3.3.4

Battery and Motor

In sections 3.2.3 and 3.2.4 the modeling of the motor and the (virtual) battery of the HIPPO powertrain was thoroughly analyzed . Here the main equations derived from that analysis are gathered . Firstly , the differential equation that capture the state of charge dynamics 3.2.16 is

$$\frac{dSOC}{dt} = -\frac{100}{Q_{norm}} \cdot \frac{U_{oc}(t) - \sqrt{U_{oc}^2(t) - 4P_b(t) \cdot R_i(t)}}{2R_i(t)} \tag{3.3.5}$$

where the battery voltage $U_{oc}(t)$ is

$$U_{oc}(t) = k_2 \cdot SOC(t) + k_1 \tag{3.3.6}$$

and finally the power consumption of the electric motor that stems from the application of the Willans approach 3.2.2 is ,

$$P_b = Q_{em} \cdot \frac{2\pi N_{eng}}{60} \cdot e_w^{K(u_{em})} + P_0 \tag{3.3.7}$$

where

- $K(u_{em}) = \frac{2}{1+2.7183^{(2u_{em})}} - 1$
- e_w, P_0 are the Willans coefficients fitted to the motor .
- Q_{em} is the torque output/input (motoring/generating) .

The EM torque is modeled with the following linear relation

$$Q_{em} = c_{em} \cdot u_{em} \tag{3.3.8}$$

where u_{em} is the torque command as a percentage of the maximum torque which is fed to the drive and c_{em} expresses the transformation to torque units

3.3.2 System Constraints

Now the constraints of the system will be cited that if violated the episode will be terminated because the problem will be considered infeasible

Engine Torque Command Limits

These limits refer to the potential of the engine to follow up the intermediate torque control input

$$\begin{aligned} u_{ice_{lim1}} &= \frac{1}{15} \cdot SE - \frac{2.5}{3} \\ u_{ice_{lim2}} &= -\frac{1}{50} \cdot SE - 120.91 \end{aligned} \tag{3.3.9}$$

$$u_{ice} \leq min(u_{ice_{lim1}}, u_{ice_{lim2}}) \tag{3.3.10}$$

Battery Limits

The battery limit can actually be derived from 3.2.17 and is the following

$$\frac{U_{oc}^2}{4R_i} - P_b \geq 1000 \tag{3.3.11}$$

Chapter 4

Theoretical Framework

In this section policy optimization methods are discussed . In particular , the mathematical framework of policy gradient methods will be analyzed . Once that has been completed further analysis will be conducted concerning arguably the most used policy optimization algorithm - proximal policy optimization (PPO).

4.1 Policy Gradient Methods-Introduction

Policy optimization methods in general refer to methods that try to learn a parameterized policy.Its parameters are optimized in a way such that the agent can select actions without consulting a value or action-value function . The value functions might still be used for optimization but not for the action selection process . In particular , policy gradient methods optimize this parameterized policy using the gradient of a scalar performance measure , usually denoted as $J(\boldsymbol{\vartheta})$ where $\boldsymbol{\vartheta} \in \mathbb{R}^b$ is the policy's parameter vector . If a methods also uses a learned value function as well the the weight vector of that function is denoted by $\mathbf{w} \in \mathbb{R}^d$, $V(s, \mathbf{w})$.

$$\pi(\alpha|s, \boldsymbol{\vartheta}) = \mathfrak{p}\left(\mathcal{A}_t = \alpha|\mathcal{S}_t = s, \boldsymbol{\vartheta}_t = \theta\right) \tag{4.1.1}$$

The policy parameters $\boldsymbol{\vartheta}$ are updated using the scalar performance measure :

$$\boldsymbol{\vartheta}_{t+1} = \boldsymbol{\vartheta}_t + h \cdot \nabla \hat{J}(\boldsymbol{\vartheta}_t) \tag{4.1.2}$$

Methods that update the policy in such manner are known as *policy gradient* methods . Methods that learn approximations of both the policy and value functions are called *actor - critic* methods . In policy gradient methods the policy can be parameterized in any way assuming that it is differentiable with respect to its parameters , namely the gradient of $\pi(\alpha|s, \boldsymbol{\vartheta})$ exists and is finite .

4.2 REINFORCE algorithm

At first a scalar performance measure needs to be introduced that can communicate to the agent if the policy that it currently follows produces the results needed . A first performance measure that one can use is a measure of the expected reward that the agent will receive by following a certain policy . This can be formulated by introducing the following :

- $\tau = (s_o, \alpha_o, r_o, ..., s_{T-1}, \alpha_{T-1}, r_{T-1}, s_t)$ which is a state - action trajectory
- $\mathcal{R}(\tau) = \sum_{t=0}^{T} \mathcal{R}(s_t, \alpha_t)$ which is the sum of rewards for a trajectory τ

Using those one can define the performance measure $J(\boldsymbol{\vartheta})$ as :

$$J(\boldsymbol{\vartheta}) = \mathbb{E}\left[\sum_{t=0}^{T} \mathcal{R}(s_t, \alpha_t); \pi(\alpha|s, \boldsymbol{\vartheta}) \right] = \sum_{\tau} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) \tag{4.2.1}$$

where $\mathcal{P}(\tau; \boldsymbol{\vartheta})$ is the probability of following a trajectory τ following policy $\pi_{\boldsymbol{\vartheta}}$. The objective is to find the policy parameters $\boldsymbol{\vartheta}$ that create a trajectory τ that maximizes the expected rewards

$$arg\max_{\theta} J(\boldsymbol{\vartheta}) = arg\max_{\theta} \sum_{\tau} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) \tag{4.2.2}$$

Equation 4.1.2 describes the update rule of the policy parameters $\boldsymbol{\vartheta}$ via gradient ascent on the performance measure . Now that a first performance measure has been established its gradient is computed as follows :

$$\begin{aligned} \nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) &= \nabla_{\boldsymbol{\vartheta}} \sum_{\tau} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) = \sum_{\tau} \nabla_{\boldsymbol{\vartheta}} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) = \\ &= \sum_{\tau} \frac{\mathcal{P}(\tau; \boldsymbol{\vartheta})}{\mathcal{P}(\tau; \boldsymbol{\vartheta})} \nabla_{\boldsymbol{\vartheta}} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) = \sum_{\tau} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \nabla_{\boldsymbol{\vartheta}} \log(\mathcal{P}(\tau; \boldsymbol{\vartheta})) \mathcal{R}(\tau) \end{aligned} \tag{4.2.3}$$

The derivation 4.2.3 can be used to calculate the gradient of the objective function J . However the last sum of equation 4.2.3 can be written as an expectation with respect to $\mathcal{P}$ as p.d.f of the random variable that formulates the probability that a certain trajectory is followed when using policy $\pi(\alpha|s, \boldsymbol{\vartheta})$

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) = \mathbb{E}\left[\, \log \nabla_{\theta} \mathcal{P}(\tau; \boldsymbol{\vartheta}) \mathcal{R}(\tau) \,\right] \tag{4.2.4}$$

The policy gradient can be represented as an expectation. That means that sampling can be used to approximate it (of course all possible trajectories are generally not available to calculate analytically).

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) \approx \frac{1}{m} \sum_{i=1}^{m} \mathcal{R}(\tau_i) \nabla_{\boldsymbol{\vartheta}} \log(\mathcal{P}(\tau_i; \boldsymbol{\vartheta})) \tag{4.2.5}$$

$\mathcal{P}(\tau_i; \boldsymbol{\vartheta})$ refers to the probability of following a trajectory τ_i when taking actions following policy $\pi_{\boldsymbol{\vartheta}}$. In their book Sutton & Barto provide a really helpful -yet pretty logical- decomposition of this p.d.f .

$$\mathcal{P}(\tau_i; \boldsymbol{\vartheta}) = \mu(s_o) \cdot \prod_{t=0}^{T-1} \mathfrak{p}(s_{t+1}^i | s_t^i, \alpha_t^i) \cdot \pi_{\theta}(\alpha_t^i | s_t^i) \tag{4.2.6}$$

- $\mu(s_o)$, is the initial state probability
- $\mathfrak{p}(s_{t+1}^i | s_t^i, \alpha_t^i)$, is the MDP model

By "logging" both sides of equation 4.2.6 and differentiating with respect to $\boldsymbol{\vartheta}$one can easily derive that

$$\nabla_{\boldsymbol{\vartheta}} \log \mathcal{P}(\tau_i; \boldsymbol{\vartheta}) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_{\theta}(\alpha_t^i | s_t^i) \tag{4.2.7}$$

The right side of equation 4.2.7 is often called *Score function*. Taking into account equation 4.2.7 the objective function becomes :

$$\begin{aligned} \nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) &\approx \frac{1}{m} \sum_{i=1}^{m} \mathcal{R}(\tau_i) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_{\theta}(\alpha_t^i | s_t^i) \\ \nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) &\approx \frac{1}{m} \sum_{i=1}^{m} \left(\sum_{t=0}^{T-1} r(s_t^i, \alpha_t^i) \right) \left(\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_{\theta}(\alpha_t^i | s_t^i) \right) \end{aligned} \tag{4.2.8}$$

Intuition of the objective function The score function in equation 4.2.8 measures the likelihood of the observed data , namely how likely a certain trajectory is under the current policy . By multiplying the score function with the total reward of the trajectory the likelihood of a policy that yields high reward is increased .

Equation 4.2.8 produces a first simple policy gradient algorithm which is generally refered to as **REINFORCE** . This algorithm uses the so called Monte Carlo rollouts which means that a complete trajectory (i.e complete episode) is played out before calculating the total reward and updating the objective function. Simple as it is , REINFORCE algorithm has its drawbacks . The main drawback is the same that concerns all the Monte Carlo methods , that is high variance which in turns leads to bad convergence properties . The stochasticity of the policy drives the agent to select different actions throughout the episode . Small changes in the agent's decisions can cause completely different results . Moreover consider the case where an agent during episode (A) takes all the low reward actions except maybe from the last 2-3 when it picks actions with extremely high reward . Then the agent replays an episode (B) during which it chooses always mediocre actions (i.e neither low neither high rewards) . For the reason that Monte Carlo rollouts take into consideration only the final reward achieved by the agent and not how it achieved it , episode (A) and (B) would be considered equally "good". However episode (B) is actually better , during which the agent displayed a mediocre yet stable behavior contrary to episode (A) where even by luck it managed to pick the right last few actions to match the total reward of episode (B).

Another issue with the Monte-Carlo rollouts is the poor data efficiency because one has to wait until a full episode is completed before actually updating the policy.

4.3 Possible Policy Parametrizations

Before exploring the different ways that the REINFORCE algorithm can be enhanced the most common policy parametrizations will be discussed in order to obtain a more tangible understanding of a parameterized policy. The manner in which the policy will be parameterized depends on the model of the action space $\mathcal{A}$, namely if it is continuous or discrete . Continuous action spaces are characterized those whose actions can take all real values within a given domain and discrete those that the actions are modelled by discrete values . For instance , in a autonomous car problem the agent can turn the wheel to an angle ranging from $[-\pi/2, \pi/2]$ (continuous action space) , while in the classic benchmark problem of the inverted pendulum the action space is the discrete set $\{-1, 1\}$ (-1 to move the pendulum to the left and 1 to move to the right). The two most common representations currently used (one for each case) are presented below :

Discrete action space : The most common parametrization of a policy concerning a discrete action space is the softmax policy . The actions are weighted using linear combinations of features $\phi(s,a)^T \cdot \boldsymbol{\vartheta}$. Using the softmax function the parameterized policy is represented as :

$$\pi(a|s,\boldsymbol{\vartheta}) = \frac{e^{\phi(s,a)^T \cdot \boldsymbol{\vartheta}}}{\sum_a e^{\phi(s,a)^T \cdot \boldsymbol{\vartheta}}} \tag{4.3.1}$$

The softmax function is essentially the generalization of the sigmoid function used in neural networks for normalization . The utilization of a softmax function entails the normalization of the output , in this case the output is the probability of an action i.e the policy , to a probability distribution .

Continuous action space : Generalizing the afore mentioned concept the parame trization of the policy function using a Gaussian is natural . In this case , assuming that the values of the actions are Gaussian distributed , the mean of the Gaussian can be defined as the ouput of a neural network with the state space as the input with the variance being fixed or also modeled in a similar manner .

$$\pi(a|s,\boldsymbol{\vartheta}) = \mathcal{N}(\mathcal{F}_{NN_{\boldsymbol{\vartheta}}}(s_t), \Sigma) \tag{4.3.2}$$

Having this Gaussian parametrization of the action one can with relatively simple algebra compute the score function and update the parameters with the REINFORCE algorithm .

4.4 REINFORCE algorithm modifications

Now that some variants of parameterized policies have been proposed , it is time to detail a series of improvements on the REINFORCE algorithm that can tackle the variance issues described on paragraph 4.2 . Monte Carlo rollouts that characterize the REINFORCE algorithms are noisy (high variance) and introduce an unbiased estimate of the gradient. The solutions to these issues are the use of **baselines** and **temporal structures** . Note that increasing the batch size before each update could tackle the problem of high variance, but increasing it too much can lead to significant sample inefficiency .

At first , a modification needs to be made in equation 4.2.8 . Interpreted as it in its current version future actions impact past decisions because the time index t in both the reward and the score function in equation 4.2.8 start at the same time

instant . What should happen instead is summing the rewards from the time instant a certain action was taken .

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) \approx \frac{1}{m} \sum_{i=1}^{m} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_\theta(\alpha_t^i | s_t^i) \sum_{t'=t}^{T-1} r(s_{t'}^i, \alpha_{t'}^i) \right] \tag{4.4.1}$$

Recall that for a particular trajectory $\tau \sum_{t'=t}^{T-1} r(s_{t'}^i, \alpha_{t'}^i) \doteq G_t$ (which can be also discounted reward).

Baselines : Introducing the use of a baseline $b(s)$ that is a term not related to $\boldsymbol{\vartheta}$an unbiased and less noisy gradient estimator is achieved . The baseline is subtracted from the reward term of equation 4.4.1 . An initial baseline can be the value function . The objective function then becomes

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) \approx \frac{1}{m} \sum_{i=1}^{m} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_\theta(\alpha_t^i | s_t^i) \left(\sum_{t'=t}^{T-1} r(s_{t'}^i, \alpha_{t'}^i) \; - \; V(s_t^i) \right) \right] \tag{4.4.2}$$

This modification can be justified by recalling the definition of the state value function (2.2.5) . The latest version of objective function essentially increases the log probability of choosing an action α_t proportionally to the how much the actual returns are better than the expected . The expected returns are quantified by the state value function $V(s_t)$.

Temporal Structures : Apart from subtracting a baseline from the return G_t to reduce variance , another modification that has been generally been proposed is changing the *target* . Until now the target of the policy gradient algorithms was the return G_t which is an unbiased estimation of the value function at a state s_t from a single Monte Carlo rollout . The variance can be further reduced by introducing bias via bootstrapping and also by value function approximation . That is when the **Actor-Critic** methods come into play . The Actor is the policy , the agent, that selects the actions during the episode . The Critic estimates whether those actions where rewarding enough by estimating either of the state value functions (u, q). Choosing the target to be the state - action value function $q(s, a)$ equation 4.4.2 becomes

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta}) \approx \frac{1}{m} \sum_{i=1}^{m} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\vartheta}} \log \pi_\theta(\alpha_t^i | s_t^i) \left(Q(s_{t'}^i, \alpha_{t'}^i, \mathbf{w}) \; - \; V(s_t^i) \right) \right] \tag{4.4.3}$$

where **w** is the parameter vector of the estimated state-action value function .

The second factor of the product in equation 4.4.3 is defined as the Advantage function

$$A_\pi(s,a) \doteq Q_\pi(s,a) - V_\pi(s) \tag{4.4.4}$$

The critic can select any blend between Temporal Difference and Monte Carlo structures for the target . That is

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}\left[\, r_0 + \gamma r_1 + \gamma^2 r_2 + ... \mid s_0, a_o \,\right] && (MC) \\ &= \mathbb{E}\left[\, r_0 + \gamma V(s_1) \mid s_0, a_o \,\right] && (1-step\ TD) \\ &= \mathbb{E}\left[\, r_0 + \gamma r_1 + \gamma^2 V(s_2) \mid s_0, a_o \,\right] && (2-step\ TD) \\ &= \mathbb{E}\left[\, r_0 + \gamma r_1 + ... + \gamma^{n-1} r_{n-1} + \gamma^n V(s_n) \mid s_0, a_o \,\right] && (n-step\ TD) \end{aligned}$$

The (MC) and the various (TD) structures can be combined to form the Generalized Advantage Estimation (GAE).

$$\begin{aligned} A_t^{(\kappa)} &= r_t + \gamma r_{t+1} + ... + \gamma^{\kappa-1} r_{t+\kappa+1} + \gamma^\kappa V(s_{t+1}) - V(s_t) \\ A_t^{GAE}(\gamma,\lambda) &= (1-\lambda)\cdot\left(A_t^{(1)} + \lambda A_t^{(2)} + \lambda^2 A_t^{(3)} + ...\right) \end{aligned} \tag{4.4.5}$$

The GAE depends on the hyperparamaters λ,$\gamma \in (0,1)$. Hyperparameter γ is the already established pararameter that downweights the rewards corresponding to delayed effects . Hyperparameter λ essentialy controls the blend of the various structures . Both γ and λ introduce bias however λ introduces much lass than γ . Their values need tuning depending on the problem to achieve the desirable agent behavior .

4.5 TRPO and PPO

Now that the foundations of policy gradient methods have been laid it is time to dive a little bit deeper on the two most cited papers concerning policy gradient methods . Complete proofs of some of the mathematical derivations will be omitted because they deviate from the scope of this thesis . However the curious reader is encouraged to study the corresponding papers due to the beautiful insights that they provide on the topic . These papers concerning the TRPO and PPO algorithms emerged from the need to meet some desired properties of a policy gradient algorithm . These are

- Quick convergence to local optima
- Each policy update to be a monotonic improvement
- During policy search alternate between policy evaluation and policy improvement

The basic concepts that the TRPO & PPO papers introduce that opt for satisfying those demands are the utilization of the *Minorize-Maximization (MM) algorithm*,*importance sampling* and *Trust Region* optimization .

4.5.1 MM algorithm

The Minorize - Maximization algorithm is an iterative optimization method which exploits the convecity of a function to finds its optima . This method essentially ,constantly maximizes via iteration the lower bound function which approximates the objective function of a problem . In order to do that it is assumed that the lower bound function that approximates the objective function is simpler to optimize .

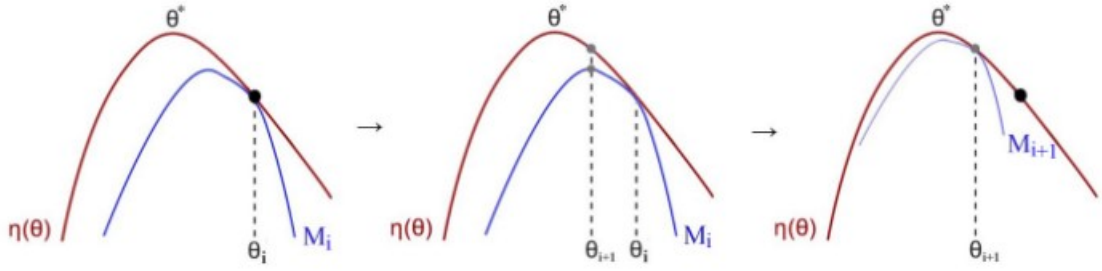

Figure 4.1: Graphic representation of the MM algorithm

$M(\theta)$ is called the minorized version of the objective function if

$$\begin{cases} M(\theta;\theta_i) \leq \eta(\theta) \quad \forall\ \theta \\ M(\theta_i;\theta_i) = \eta(\theta_i) \end{cases}$$

Then M is maximized instead of $\eta(\theta)$ and the next point where equality holds is

$$\theta_{i+1} = arg\max_{\theta} M(\theta;\theta_i)$$

By applying many iterations of that manner $\eta(\theta)$ will eventually converge to a local optima or saddle point .

4.5.2 Trust Region Optimization

The two most renowned optimization methods are line search and trust region . Gradient ascent/descent which is the method used for the update of the parameters in the most simple policy gradient algorithm ,the REINFORCE algorithm , is a case of line search . The ascending direction of the objective function is determined and a step towards it is taken . This method poses some problems . It uses the first order derivative and approximates a flat surface . If the objective function however is characterized by intense curvature fluctuations each step is not guaranteed to lead to convergence . Too large of a step can lead to a significant deviation from the best local optima while too small of a step is leads to really slow learning .

The trust region optimization method takes care of those issues . In this kind of optimization the maximum step size of exploration is determined at first and then an optimal point is located within the trust region that the maximum step creates.

$$\begin{aligned} &\max_{t \in \mathbb{R}^n} M_k(t) \\ &s.t \quad ||s|| \ \leq \ \delta \end{aligned} \tag{4.5.1}$$

where $M_k(t)$ is the approximation of the objective function. The constraint on the step size δ can be dynamically adjusted throughout the run according to the curvature of the surface .

4.5.3 Importance Sampling

Another really useful technique used widely in the reinforcement learning and machine learning in general field of study is the concept of importance sampling. It is a very useful technique that allows the evaluation of properties of one distribution given only samples of another distribution different from the one given . Due to its extensive mathematical background attention will be focused on the result of that technique .

Given a random variable x and a function of that random variable $f(x)$ importance sampling calculates the expected value of $f(x)$ where x has data distribution p. Importance sampling offers the choice of not sampling the value of $f(x)$ from p but instead use the sample data from another distribution q and use the probability ration between them to calibrate the result .

$$\mathbb{E}_{x \sim p}\left[f(x)\right] \overset{\text{sample from q}}{\longrightarrow} \quad \mathbb{E}_{x \sim q}\left[\frac{f(x)p(x)}{q(x)}\right] \tag{4.5.2}$$

with a variance of

$$\frac{1}{N}\left(\mathbb{E}_{x\sim p}\left[\frac{p(x)}{q(x)}f(x)^2\right]-\mathbb{E}_{x\sim p}\left[f(x)\right]^2\right) \tag{4.5.3}$$

Importance sampling is used in reinforcement learning problems where off policy learning can be useful . In policy gradient methods for example once the policy is updated the samples of the old policy are not reusable which means that standard policy gradient methods are not sample efficient . Using importance sampling the objective function can be rewritten in a manner that is sample efficient , meaning that data from a previous policy can be used to calculate the new policy gradient

4.6 PPO algorithm

Now that the basic concepts that concern the PPO algorithm have been discussed , it is time to apply them to describe some of the details of the algorithm that will be used to tackle the continuous control problem of this thesis. The most commonly used gradient estimator is that calculated in equation 4.4.3 which is rewritten as follows :

$$\nabla_{\boldsymbol{\vartheta}} J(\boldsymbol{\vartheta})=\mathbb{E}_t\left[\nabla_{\boldsymbol{\vartheta}}\log\pi_\theta(\alpha_t^i|s_t^i)\hat{A}_t\right] \tag{4.6.1}$$

where $\hat{A}_t$ is the estimator of the advantage function and the expected value represents the average over a finite batch of samples in an algorithm that alternates between optimization and sampling . The objective function of the policy gradient method itself (symbolized with L to follow the original PPO paper) is

$$L^{(PG)}(\boldsymbol{\vartheta})=\mathbb{E}_t\left[\log\pi_\theta(\alpha_t^i|s_t^i)\hat{A}_t\right] \tag{4.6.2}$$

Now take a step back and recall the initial objective of the whole conversation.That objective was to find a policy that maximizes the expected reward when following that policy . This can be represented as the η function used to analyze the MM algorithm in section 4.5.1 . So

$$\eta(\boldsymbol{\vartheta})=\mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\vartheta}}}\left[\mathcal{R}(\tau)\right]=\mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\vartheta}}}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] \tag{4.6.3}$$

That is the function that needs to be optimized and in this case maximized . To apply the MM algorithm , a surrogate function , i.e a function that approximates η locally needs to be found . Combining the MM algorithm with the trust region optimization technique in the TRPO paper it is proven that an objective function that achieves **monotonic** improvement to begin with is the following :

$$\begin{aligned} \underset{\vartheta}{maximize} \quad & \mathbb{E}_t \left[\frac{\pi_\theta(\alpha_t|s_t)}{\pi_{\theta_{old}}(\alpha_t|s_t)} \hat{A}_t \right] \\ s.t \qquad & \mathbb{E}_t \left[\mathbf{KL}(\pi_\theta(\alpha_t|s_t), \pi_{\theta_{old}}(\alpha_t|s_t)) \right] \leq \delta \end{aligned} \tag{4.6.4}$$

The hard constraint on 4.6.4 can be transformed into a penalty with a coefficient β . TRPO algorithm uses the hard constraint because it is difficult to choose a certain penalty β . Directly cited from the PPO paper : "*..., to achieve our goal of a first-order algorithm that emulates the monotonic improvement of TRPO, experiments show that it is not sufficient to simply choose a fixed penalty coefficient β and optimize the penalized objective Equation 4.6.5 with SGD; additional modifications are required*"

$$\underset{\vartheta}{maximize} \quad \mathbb{E}_t \left[\frac{\pi_\theta(\alpha_t|s_t)}{\pi_{\theta_{old}}(\alpha_t|s_t)} \hat{A}_t \right] - \beta \cdot \mathbb{E}_t \left[\mathbf{KL}(\pi_\theta(\alpha_t|s_t), \pi_{\theta_{old}}(\alpha_t|s_t)) \right] \tag{4.6.5}$$

Equation 4.6.5 is the M function that will be optimized instead of the η function according to the MM algorithm .The first term of that function contains expected advantage function and it is estimated with importance sampling using the old policy (i.e the policy before the update of the parameters) . $KL(\cdot, \cdot)$ represents the KL divergence of 2 probability distributions . A hard constraint (TRPO) or a penalty (PPO) is applied on the KL divergence of the new-updated policy and the old policy . This ensures that the updated policy is not much different from the old policy and for this reason the variance of the estimation from the importance sampling remains at low levels (equation 4.5.3) .

The modifications proposed in the PPO paper are

- PPO with adaptive KL penalty coefficient
- PPO with clipped surrogate objective

4.6.1 Adaptive KL penalty coefficient

This approach refers to the adaptive change of parameter β in order to achieve a specific value of the KL divergence during each policy update. The simplest form of this algorithm is performed following the steps below in each policy update

1. Optimize the KL-penalized objective

$$L^{KLPEN}(\vartheta) = \mathbb{E}_t \left[\frac{\pi_\theta(\alpha_t|s_t)}{\pi_{\theta_{old}}(\alpha_t|s_t)} \hat{A}_t - \beta \cdot \mathbf{KL}(\pi_\theta(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t)) \right] \tag{4.6.6}$$

2. Compute the average KL divergence $d = \mathbb{E}_t \left[\mathbf{KL}(\pi_\theta(\cdot|s_t), \pi_{\theta_{old}}(\cdot|s_t)) \right]$

   - If $d < d_{target}/1.5$ *then* $\beta \leftarrow \beta/2$
   - If $d > 1.5 d_{target}$ *then* $\beta \leftarrow 2\beta$

The updated β is then used for the next policy update . The initial value of parameter β is also assumed to be a hyperparameter by the authors of the PPO paper , though not that crucial in practice because the algorithm adjusts it quickly.

4.6.2 Clipped Surrogate Objective

The most used version of the PPO algorithm is that which utilizes a clipped surrogate objective . The `ReinforcementLearning.jl` package of `Julia` programming language uses this specific version of the PPO algorithm . A more detailed description of the `ReinforcementLearning.jl` package will be analyzed in a subsequent chapter .

Theprobability ratio of the two policies , the old and the new one , is denoted as $r_t(\vartheta)$. In the TRPO paper the surrogate objective to be maximized is

$$L^{CPI}(\vartheta) = \mathbb{E}_t \left[\frac{\pi_\theta(\alpha_t|s_t)}{\pi_{\theta_{old}}(\alpha_t|s_t)} \hat{A}_t \right] = \mathbb{E}_t \left[r_t(\vartheta) \hat{A}_t \right] \tag{4.6.7}$$

CPI stands for conservative policy iteration where this objective was first introduced by *S. Kakade* and *J. Langford*. PPO algorithm differentiates itself from that of TRPO by introducing a "clip" in the probability ratio between the old and the new policy .The objective proposed is the following

$$L^{CLIP}(\vartheta) = \mathbb{E}_t \left[min(r_t(\vartheta)\hat{A}_t, \texttt{clip}(r_t(\vartheta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t) \right] \tag{4.6.8}$$

Equation 4.6.9 is different from 4.6.7 but there is some common ground between them . The first term in the $min(\cdot,\cdot)$ function in equation 4.6.9 is the surrogate objective of the TRPO paper . The second term clips , namely enforces an upper and a lower bound on the probability ratio of the updated policy and its predecessor . The minimum between the unclipped and clipped objective is a lower bound on the unclipped objective (i.e the CPI objective). The clipped surrogate objective function can be enhanced by adding an entropy bonus that ensures sufficient exploration and advantage function estimators as objectives that help reduce the variance .

$$L^{CLIP+VF+S}(\vartheta) = \mathbb{E}_t \left[L^{CLIP}(\vartheta) - c_1 L^{VF}(\vartheta) + c_2 S[\vartheta](s_t) \right] \tag{4.6.9}$$

where S is the entropy bonus and $L^{VF}(\vartheta)$ is a squared-error loss $(V_\vartheta(s_t) - V_t^{targ})^2$.

Algorithm 1 PPO algorithm pseudocode

Input : Initialize policy parameters θ_0 and value function parameters φ_0
for Iteration=0,1,2... **do**
  for actor = 1,2,...,N_{actors} **do**
    Run policy π_ϑ until episode termination (final timestep T).
    Compute the advantage estimates $\hat{A}_1, ..., \hat{A}_T$
  end for
  for 1:N_{epochs} **do**
    for 1:$N_{minibatches}$ **do**
      Compute the objectives and the entropy bonus $L^{CLIP}(\vartheta)$, $L^{VF}(\vartheta)$, $S[\vartheta](s_t)$
      Optimize the objective $L^{CLIP+VF+S}(\vartheta)$ via Adam SGD
    end for
  end for
  Update $\theta_{old} \leftarrow \theta$
end for

In algorithm 1 there are N_{actors} running in parallel the same policy π_ϑ and collect data at every time step.The information collected is the ones needed to construct the surrogate loss function $L^{CLIP+VF+S}(\vartheta)$ and it is then optimized via Adam minibatch SGD.

Chapter 5

Basic Controller Implementation

5.1 Classic Control through the RL scope

The thesis opts for introducing a rather academic approach on an optimal control problem that has already been tackled by means of standard optimization methodologies such as the MPC and non linear MPC [3,2] . This approach as already mentioned stems from the field of machine learning and particularly refers to reinforcement learning methodologies . In this section , it is deemed vital that a side by side comparison between the two approaches is presented so as to fathom the implementation of data based methodology on a control problem and discuss the possible drawbacks that it may include.

Beginning by presenting the common ground among the two methodologies mentioned , both methods obviously refer to ways in which one can approach solving an optimal control problem or generally an optimization problem . In other words , MPC as well as reinforcement learning attempt to solve a sequential decision making problem to determine the actions that optimize a performance objective .

Model Predictive Control Model predictive control draws at every time step $\mathbf{k}$ a vector of measurements and an observer estimates the state vector $\hat{\mathbf{x}}_\mathbf{k}$ which describes fully the controller model at current time. Then the state $\mathbf{x}$ and input $\mathbf{u}$ trajectories are optimized for a finite prediction horizon Δt_h . The MPC scheme is also characterized by the following quantities .

- $\mathcal{J} = \int_{t=t_k}^{t_k+\Delta t_h} l(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{u}(t), \mathbf{y}(t), \mathbf{z}(t), \mathbf{d}(t), \mathbf{p})dt$ is an explicit representation of the objective function where
  - $\mathbf{y}(t)$ are the model outputs
  - $\mathbf{z}(t)$ are algebraic variables

  - $\mathbf{d}(t)$ are the disturbances
  - $\mathbf{p}(t)$ are time-varying variables

- **F** a controller model
- **H** constraints

The reinforcement learning scheme was described analytically in previous sections and thus a summarised depiction of that scheme is presented to readily delineate the side by side comparison of the two optimization schemes. Reinforcement learning depends on the concept of the Markov Decision Processes (MPD) where the decision making model is described by the state space $\mathcal{S}$, the action space $\mathcal{A}$, the rewards $\mathcal{R} \subseteq \mathbb{R}$ and a transition function of the environment $\mathbf{f}$. The agent interacts with the environment during a sequence of discrete-time steps. Every time step k the RL agent obtains an observation of the state space $S_k \in \mathcal{S}$ and a reward R_k that expresses how good the action was towards achieving the agent's goal . The environment is defined by $\mathcal{E} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{S} \times \mathcal{R}$ and ultimately the objective of the RL is to infer an optimal control policy $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ that maximizes the cumulative return G when the agent acts according to that policy .

Equivalence of elements The two schemes described above might seem to be distinctively different yet a more meticulous observation reveals equivalence among concepts and quantities used to describe each one . The most obvious similarities can be tracked among elements such as the control input $\mathbf{u}$ and the action $a \in \mathcal{A}$, the plant and environment and of course the controller and agent . Although there is significant correspondence between the aforementioned elements one should be cautious also about the particularities that each of them entails in its definition . To illustrate starting with the plant and the environment , in the MPC context the plant is limited to the representation of the system process while the environment with which an RL agent interacts is extended to provide information about the states and the rewards as perceived by the agent. Furthermore, attention should focused on the definition of the state provided by the two schemes . For the control community , the state vector $\mathbf{x}$ designates only internal properties of the system while for the machine learning community the state vector usually denoted by $\mathbf{s}$ refers to the environment's condition which may include internal as well as external system variables , namely forecast of measurements and/or previous measurements.

An analogy also can be detected between the objective function $\mathbf{J}$ and the cumulative return $\mathbf{G}$. The control community tries to minimize the objective function for a given problem whereas the machine learning community maximizes the cumulative return.The relation between both can be formalized through the immediate reward.

In the deterministic setting , the scalar immediate reward r_{k+1} can be related to the objective function as follows

$$r_{k+1} = -(J_{k+1} - J_k) \tag{5.1.1}$$

Optimality As far as optimality is concerned the MPC's solution fidelity is largely depended on the accuracy of the controller model , which is simplified in general for the sake of reduction of the overall computational cost . On the other hand methods that rely on the MDP scheme such as Dynamic Programming ensure optimality via the Bellman principle . Their downside is the curse of dimensionality namely , the exponential complexity growth with the increase of the size of the state-action space. However as mentioned in section 2.2 DP is generally limited in RL due to the important assumption of a perfect model. For this reason the collection of methods that approximate any element of the MDP (i.e RL or the so called approximate dynamic programming) can only guarantee optimality under certain conditions , when using linear function approximations or Monte Carlo learning.The reinforcement learning methodologies that do not guarantee optimality have nevertheless been able to tackle real complex problems where exact methods may become infeasible.

Use of Models Models are generally used both in MPC and RL methodologies , yet their usage is different in each case . In the MPC scheme the model used represents the system and is generally called the controller model.That model encapsulates the dynamics of the plant and is acquired via means of system identification or even with the application of supervised learning methodologies based on log data provided by the plant's operation. The controller model is most of the times simplified , sacrificing fidelity and performance for the sake of convergence ,low computational cost and consequently fast response. On the other hand,RL models in general refer to models that approximate the policy or the value function quantities that do not necessarily represent the system dynamics in the way the controller model does . Note , however ,that system models, namely models that represent to some extent the system dynamics or the environment dynamics are used in **simulation based reinforcement learning** .

In this thesis,the approach of simulation based reinforcement learning is examined with the model of the whole environment and its inherent dynamics being described in sections 3.2.2-3.3.1. The reasons why simulation based RL is in some aspects superior to directly applying the RL controller on the real system , in this case on HIPPO-2 , are presented concisely in the following section .

5.2 Real-World Reinforcement Learning

In [Challenges of Real-World RL] the challenges of direct implementation of reinforcement learning algorithms on real world systems are condensed into the nine most common challenges that researchers should confront in order to allow for the productionisation of RL in real world problems. These are directly cited from [Challenges of Real-World RL] below :

1. Off-line training from an external behavior policy
2. Learning on the real system from physical samples
3. High dimensional continuous state and action spaces
4. Safety constraints that should never or at least rarely be violated.
5. Tasks that may be partially observable, alternatively viewed as non-stationary or stochastic.
6. Reward functions that are unspecified, multi-objective, or risk-sensitive
7. System operators who desire explainable policies and actions
8. Inference that must happen in real-time at the control frequency of the system
9. Large and/or unknown delays in the system actuators, sensors, or rewards.

The challenges mentioned can sometimes be a strong incentive for a researcher to stick to simulated environments , when that is possible , due to the fact that a simulated environment is characterized by unlimited data for training , trivial consequences for poor action selection from the agent (worst case scenario a PC that might crash) and system dynamics that are clearly defined via mathematical equations . In this case , the main reasons that a simulated environment for the agent's training was used mostly relate to challenges 2,3,4 . Moreover,considering that HIPPO-2 contains an ICE engine its control would require its constant operation something that is extremely costly and the agent's wrong action selections during the training process could potentially damage it .

5.3 Julia Implementation

This sections aims to describe some of the reinforcement learning capabilities of the Julia programming language by introducing the main key features that pertain to reinforcement learning algorithms and experiments . The explicit package widely

used in the Julia community for reinforcement learning experiments is the ReinforcementLearning.jl package , which is a newly developed yet really promising package for anyone opting for experimenting with various RL methodologies and environments . This section serves also as a brief guideline to anyone trying to grasp the basic concepts of the aforementioned package.

5.3.1 Basic Features

Experiments generally refer to any simulated environment in the Julia jargon.In order to run a simulation ,which is essentially a method, needs to define beforehand the four attributes of the experiment method which are

- Environment
- Agent
- Stop Condition
- Hook

Environment: The environment in Julia is defined as a separate data structure called AbstractEnv and every environment created is a subtype of that data structure . The necessary steps that one needs to do in order to create a custom environment is define the action space, the state space, the reward and the observation states.Furthermore, three basic functions are essential to define completely a custom environment which are the `reset!()` ,`step!()` and `action()` methods on the environment data structure.

The names of the methods are actually self descriptive however , the `reset!()` function initializes the necessary quantities at the end of each episode , the `step!()` method includes the dynamics of the environment and is responsible for the state transition .The `action()` checks that the selected action belongs to the action space and feeds the action to the `step!()` method to complete the state transition for the agent according to the selected action .

Agent: The agent is a data structure that accepts as attributes a *policy* and a *trajectory*. The policy entails any approximator used and other hyperparameters while the trajectory is responsible for storing the necessary data during each episode at a given rate defined by the user .

Stop Condition & Hook : The stop condition is also a self descriptive characteristic of the environment and defines when the experiment is completed . The usual ways to terminate an experiment is either after a predefined number of episodes (usually preferred for single environments) or after a predefined number of state transitions , namely steps (usually used in the so-called MultiThreadEnviroments used in algorithms where data need to be sampled from the simultaneous run of numerous identical enviroments is,like the PPO)

The hook facilitates the option for the user to extract data from the experiment during any stage of the run , pre-episode , post-episode etc.One can create his own custom hook with the most commonly used one being the one that collects the total reward during each episode.

Chapter 6

Energy Management & Emissions Minimization Control System

In the previous section the optimal torque/power split for the control of HIPPO-2 was examined resembling the operation of a ship's hybrid propulsion plant during transient loading . In this section the Energy Management and Emissions Minimization Control System (EMEMS) [2] is introduced and the problem formulation is also presented . The application of the EMEM system has already been implemented on the HIPPO-2 and the control scheme used was that of MPC [2] . In this thesis,the goal was to try to implement the EMEM system with reinforcement learning methodologies and compare the results produced by the corresponding simulation.

6.1 Problem Formulation

The control problem objectives are revisited , this time with an additional goal regarding the minimization of emissions and the fuel consumption .

- **Speed tracking** : Following the desired speed reference profile.
- **State of charge control** : Maintain the SoC levels around a predefined level independently of the load profile
- **Robustness** : Rejecting external load disturbances with swift convergence to a non-oscillatory behavior
- **Minimization of energy consumption and emissions production**: Fuel consumption and emissions production become variables of the equivalent fuel consumption (m_{ec}) that is essentially a combination of those two quantities.

Equivalent Consumption

The equivalent consumption model was introduced by Paganelli [ECMS] and is a strategy based on the concept that in self sustaining vehicles the battery used plays the role of an energy buffer , namely a unit in the system that stores energy in the form of electric charge , energy readily available to the system.The key note in this concept is the fact that the recharge of the battery does not happen from an external source rather internally from the system itself using fuel that is already used from main power source , usually as well as in this case an ICE engine . The battery is seen for this reason as an auxiliary reversible fuel tank .

In both charge and discharge phases , a virtual fuel consumption can be associated with the use of electrical energy and can be summed to the actual fuel consumption to obtain the instantaneous equivalent fuel consumption .

$$\dot{m}_{ec}(t) = \dot{m}_f(t) + \dot{m}_{batt}(t) = \dot{m}_f(t) + \frac{s}{Q_{lhv}} P_{batt}(t) \cdot p(x) \tag{6.1.1}$$

where ,

- $\dot{m}_f(t)$ is the instantaneous fuel consumption
- $\dot{m}_{batt}(t)$ is the virtual fuel consumption associated with
  - Q_{lhv} fuel's lower heating value
  - $P_{batt}(t)$ the battery power
  - $p(x)$ is a correction that takes into account the deviation of the nominal SOC from the reference
  - s is the equivalence factor

The equivalent consumption in the current thesis was applied in the following manner,

$$\dot{m}_{ec}(t) = (1 - A)\dot{m}_f + A\lambda_N \dot{m}_N + \lambda_{ecems} \cdot \lambda_{SOC} \frac{P_b}{Q_f} \tag{6.1.2}$$

where λ_{ecems} is a tuning parameter which is determined in simulation to derive the same initial and final battery state of charge. In this way, the savings regarding the hybrid operation can be evaluated in relation to the achieved energy efficiency and battery energy conservation . The parameter λ_{SOC} refers to the penalty enforced by the extensive usage of the battery . In general the goal for the SOC is to remain

around 50 % which set to be the reference point and large deviation from it is penalized with λ_{SOC} . λ_N is a scaling parameter to ensure that $\dot{m}_f$ and $\dot{m}_N$ are on the same scale when considering the equivalent consumption and is equal to the ratio of the maximum fuel rate to the maximum NO_x production rate. Finally **A** represents the relative weight between the fuel consumption and the NO_x production.

$$\lambda_{SOC} = \frac{|SOC - SOC_{ref}|^3}{10^3} \tag{6.1.3}$$

RL agent quantities

To tackle the problem mentioned a policy gradient method and in particular proximal policy optimization (PPO) will be used , the theory of which was analysed in section 4. The agent will have access to a two dimensional action space that will also be continuous .The actions of the agent , namely the control variables of the controller , will be the derivative of the electronic fuel indexes of the ICE and the electric motor in %.

Action & Action Space

$$\begin{aligned} \alpha &= [\dot{u}_{ICE}, \dot{u}_{em}]^T \\ \mathcal{A} &= [\dot{u}_{ICE_{min}}, \dot{u}_{ICE_{max}}] \times [\dot{u}_{em_{min}}, \dot{u}_{em_{max}}] \end{aligned} \tag{6.1.4}$$

States & State Space

$$\begin{aligned} s &= \left[N_{eng}, \dot{N}_{eng}, N_{error}, SOC, \dot{SOC}, SOC_{error}, \dot{m}_{ec}\right]^T \\ \mathcal{S} &= [s_{1_{min}}, s_{1_{max}}] \times [s_{2_{min}}, s_{2_{max}}] \times \ ... \ \times [s_{7_{min}}, s_{7_{max}}] \end{aligned} \tag{6.1.5}$$

where

$$N_{error} = N_{eng} - N_{ref} \ and \ SOC_{error} = SOC - SOC_{ref}$$

A critical step that needs to be done is the normalization of the states . State normalization is of utmost importance in methods that depend on neural networks to predict Q, V values or in the case of PPO to directly predict the optimal action (via the policy representation as a normal distribution). Normalization ensures that all states are scaled on the same level , thus scale effects are depleted . In other words , using the machine learning jargon , a feature scaling is performed before these are provided to the actor - neural network that will predict the desirable quantities .

The following function is used to perform the normalization of the states

$$\aleph(s) = tanh\left(2 \cdot \frac{s - \min(s)}{\max(s) - \min(s)} - 1\right) \tag{6.1.6}$$

By applying that function on the state space the normalized state space is engendered (for the sake of symbol fluency the normalized quantities will be symbolized with a n on the subscript).

$$s_n = \aleph(s) = \left[N_{eng.n}, \dot{N}_{eng.n}, N_{error.n}, SOC_n, \dot{SOC}_n, SOC_{error.n}, \dot{m}_{ec.n}\right]^T \tag{6.1.7}$$

Reward Function

The reward function of the RL scheme could be considered one of its ,if not the, most essential parts . As already stated , an agent's action selection process is massively depended on the effect that the chosen action will have . The reward function assimilates that effect and informs the agent on the suitability of his choice on a given state transition . For this reason , the reward function shaping is an exigent process that , unfortunately most of the times its final representation is established via trial and error , as different problems require different reward functions and explicit methodologies describing the reward function design do not exist .

The reward function for the problem discussed in this section is a linear combination of a number of functions

$$\begin{aligned} &\mathcal{R} = \overrightarrow{w} \cdot \overrightarrow{f},\ where \\ &\overrightarrow{w} = [w_N, w_S, w_{\dot{m}}, w_{\dot{S}}, w_{\dot{N}}, w_t] \\ &\overrightarrow{f} = \left[f_N(N_{error}), f_S(SOC_{error}), f_{\dot{m}}(\dot{m}_{ec}), f_{\dot{S}}(\dot{SOC}), f_{\dot{N}}(\dot{N}_{eng}), f_t(t_{motor})\right] \end{aligned} \tag{6.1.8}$$

w_i represent the weight factors assigned to each sub-function of the reward function.Those relative weights play a significant role as hyperparameters in the training process and their values where determined mainly as a combination of intuition behind the system's operation and trial and error (with the use of hyperparameter tuning tools),mainly due to the lack of an explicit methodology.In each experiment described on the training results section the corresponding values of each weight are mentioned.

As described from equation 6.1.8 the reward function is designed to be a multidimensional function with variables

- N_{error} which is the tracking error
- SOC_{error} which represents the deviation from SOC_{ref}
- m_{ec} which is the equivalent consumption
- $\dot{SOC}$ the derivative of the SOC
- $\dot{N}_{eng}$ the speed derivative

The sub-functions used are again hyperbolic functions and in particular $tanh(\cdot)$ in order to scale each function in the [0,1] domain (they are also squared) .

$$
\begin{aligned}
f_N(N_{error}) &= 1 - tanh\left(2 \cdot \frac{N_{error} + 15}{30} - 1\right)^2 \\
f_{\dot{N}}(\dot{N}_{eng}) &= 1 - tanh\left(2 \cdot \frac{\dot{N}_{eng} + 5}{10} - 1\right)^2 \\
f_S(SOC_{error}) &= 1 - tanh\left(0.03 \cdot exp(||SOC_{error}| - 1|\right)^2 \\
f_{\dot{S}}(\dot{SOC}) &= 1 - tanh\left(2 \cdot \frac{\dot{SOC} + 5}{10} - 1\right)^2 \\
f_{\dot{m}}(\dot{m}_{ec}) &= 1 - tanh\left(2 \cdot \frac{\dot{m}_{ec} + 8.5}{17} - 1\right)^2
\end{aligned}
\tag{6.1.9}
$$

In the following page the corresponding plots of the sub-functions that constitute the reward function are presented . The sub-function regarding the SoC error reward is presented separately. In these plots one should notice when the reward approaches zero . The values around that area are essentially the bounds imposed in the quantity of the sub-function . For instance in the speed error sub function if the speed error exceeds 50 rpm the agent gains no reward from that part (with the episode not necessarily being terminated) .

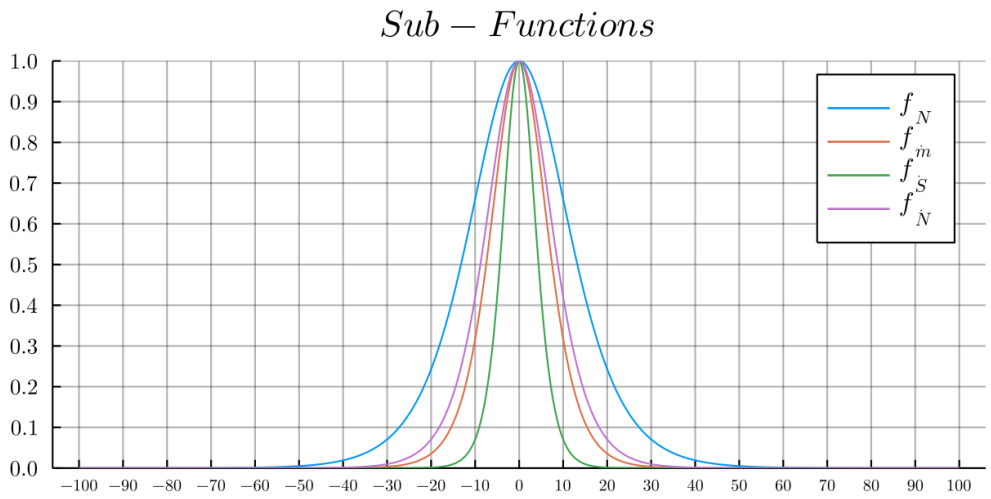

Figure 6.1: Speed Error , $\dot{SoC}$, Equivalent fuel Consumption & $\dot{N}$ sub-functions

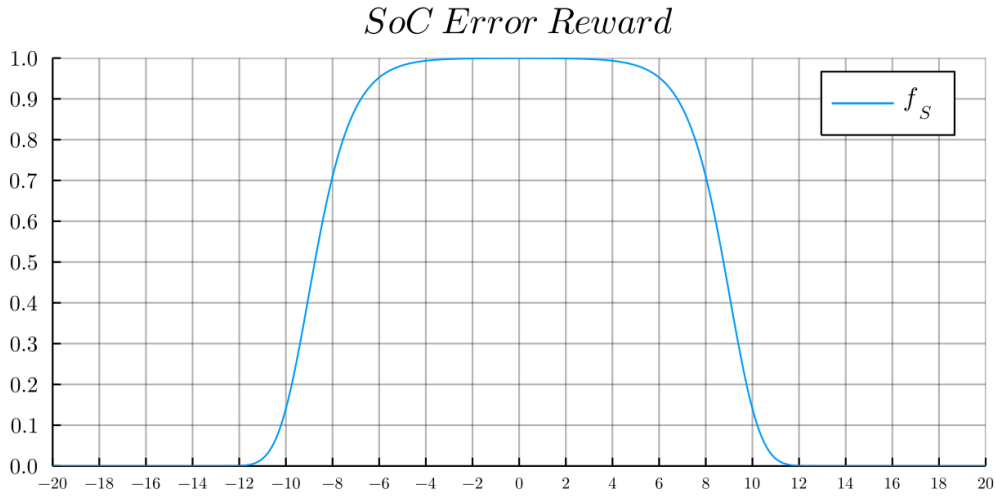

Figure 6.2: SOC_{error} sub-function

The next figure presents the whole plant-controller setup via the reinforcement learning scope by establishing the controller - plant interaction dynamic that the environment - agent pair entails .The environment is consisted of the consumption,battery,ICE and electric motors models as well as an external environment model which essentially represents an approximation of the sea environment and provides the necessary references for the operation of the plant,in this case the HIPPO-2.

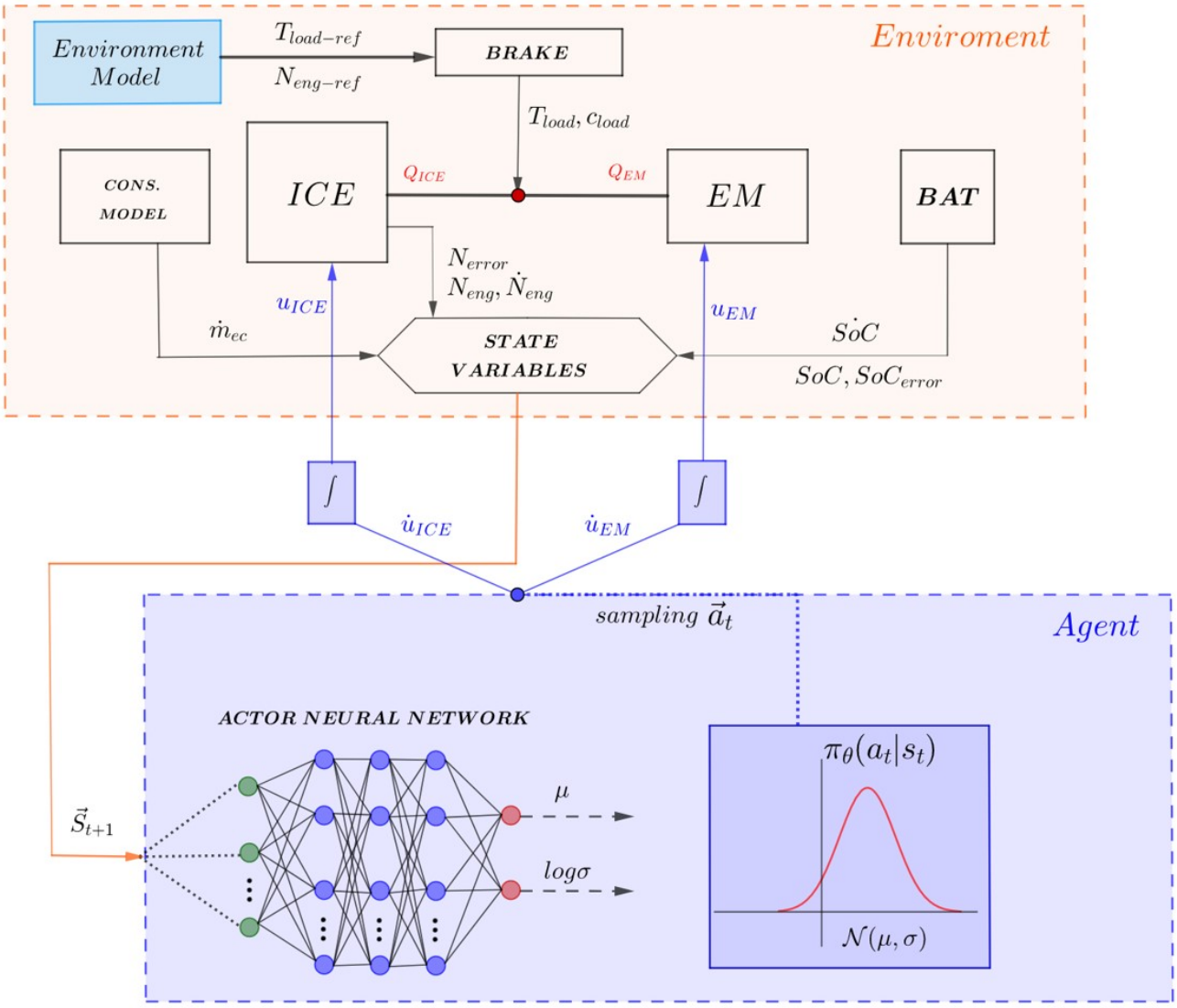

Figure 6.3: Environment (Plant)- Agent (Controller)

Furthermore ,the agent's action selection part is depicted which is the actor neural network.The inputs of that NN are the state variables provided by the environment and the outputs are a mean action and its deviation that comprise the normal distribution according to which an action is sampled and is fed back to environment, namely to the ICE and the electric motor .

6.2 Training Results

The agent's training sessions are categorized according to the value of the weight factor of the equivalent consumption. The agent's training was done with a specific relative weight factor A . The parameters of interest are presented in table 6.1. There the command rates of the ICE and the electric motor are presented (i.e the action space of the agent) as well as the constraints regarding the SoC limits and the engine's revolution speed . The internal battery model and the equivalent consumption parameters are also presented.

Parameter	**Symbol**	**Value**
SoC Constr.	SOC_{hard}	[20 80] %
N_{eng} Constr.	$N_{eng,soft}$	[700 2000]
EM cmd rate	$\dot{u}_{em}$	[-50 50] %/s
ICE cmd rate	$\dot{u}_{ICE}$	[-20 10] %/s
Equivalent Cons Parameters	λ_N	0.067
	λ_{ecems}	1.56
Battery Internal Model Parameters		
Open Source Voltage Coefficient	kV_1	696 V
	kV_2	1.022 V/%SOC
Internal Restistance	R_i	0.0640 Ω
Nominal Capacity	Q_{nom}	27.84 kWh

Table 6.1: Control Scheme Parameters

In table 6.2 the parameters concerning the PPO algorithm execution are presented . The agent was trained from data stemming from his interaction with 8 identical environments with the complete episode having a duration of 500s (5000 steps with sampling rate $dt = 0.1s$) . The values that pertain to the PPO algorithm such as the critic & actor loss weights where set to the values seen in table 6.2 according to the commensurate values used in other continuous control applications .

Parameter	Value
N_{env}	8
Update Frequency	2048
Max Steps	5000
Reward Discount γ	0.99
GAE coeff. λ	0.96
Clip Range	0.2
Critic Loss Weight	1.0
Actor Loss Weight	1.0
Entropy Loss Weight	0.01
Actor - Critic NN Structure	
μ	$[n_s$ 64 64 $n_a]$
$\log \sigma$	$[n_s$ 64 64 64 $n_a]$
Critic	$[n_s$ 64 64 64 1]
Epochs	10
Optimizer (ADAM)	$3 \cdot 10^{-4}$
Microbatch Size	32

Table 6.2: PPO learning algorithm parameters

6.2.1 Training Scheme

The training scheme via which the agent learned , opted for good generalization characteristics . In particular , the agent was trained in order to be able to cope with an unknown simulation environment significantly different to that of training. This was achieved by introducing randomness into the training methodology mainly in two ways .

1. Changing the load coefficient c_{load} randomly at each step following a uniform distribution in $[2.3 \cdot 10^{-5}, 2.7 \cdot 10^{-5}]$.
2. Expose the agent to a simple yet , random speed tracking profile at the end of each episode . A suitable function was created that served the purpose of generating random speed profiles during after the end of each episode .

Random load demand : The purpose of the implementation of the random load demand was the introduction of the agent to the randomness that the sea environment disturbances entail . The torque demand in training was derived used the propeller law according to which the power needed to be produced to move the ship

at a certain speed is proportional to the cube of the engine's rotational speed . Thus the torque provided is of magnitude ,

$$T_{load} = c_{load} \cdot N_{eng}^2 \tag{6.2.1}$$

By randomly changing the c_{load} parameter in each step the agent gets accustomed to the nature of the sea environment disturbances . While the range of the c_{load} value domain is not particularly large , it is satisfying because it introduces a sufficient amount of complexity that allows the agent to converge eventually to a desirable policy.It also approximates well enough the propeller law coefficient values of the propulsion plant model according to which the final simulation is based [2] .

Random Speed Profile : In order for the agent to be able to generalize to various speed profiles , but also allow for convergence to a sufficient policy , the agent was trained on various step-like speed profiles . Some potential speed profiles are presented below

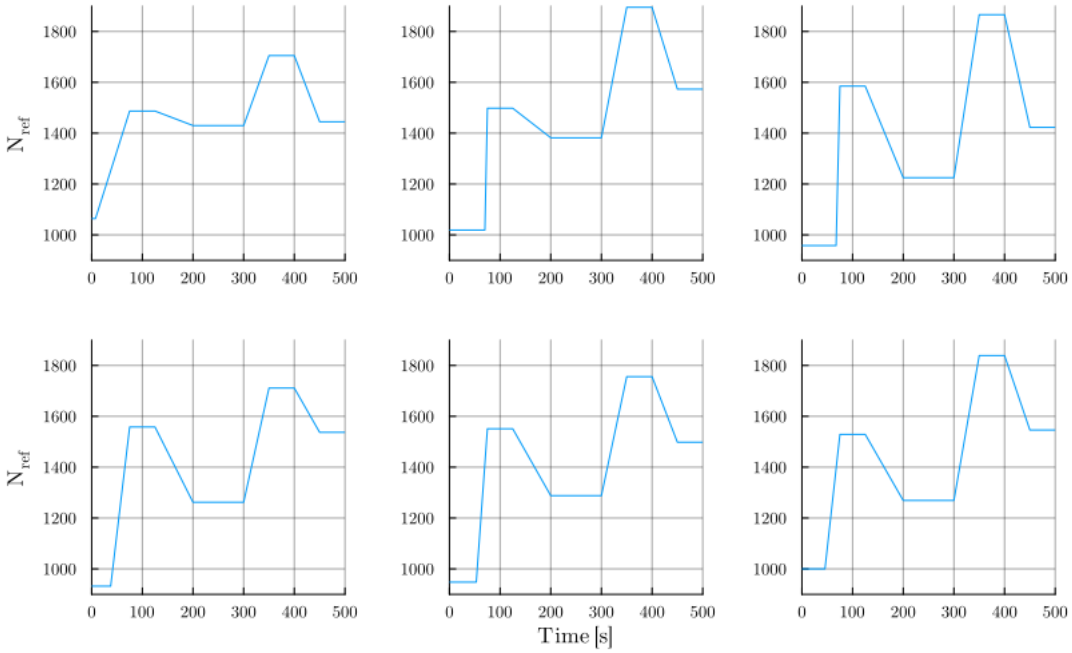

Figure 6.4: Various speed profiles produced

These levels where produced by a custom function that generates speed profiles by randomly assigning the value of the speed and the time in which the transition

from a speed value to another occurs. In the figure above one can notice in each speed profile 5 constant speed sectors . The speed values and the corresponding time duration of those are the following

- Sector 1 : $N_{ref} \in [900, 1100]$, $t_{ref} \in [0, 0.15 \cdot t_{ep}]$
- Sector 2 : $N_{ref} \in [1400, 1600]$, $t_{ref} \in (0.15 \cdot t_{ep}, 0.25 \cdot t_{ep}]$
- Sector 3 : $N_{ref} \in [1200, 1500]$, $t_{ref} \in (0.25 \cdot t_{ep}, 0.5 \cdot t_{ep}]$
- Sector 4 : $N_{ref} \in [1700, 1900]$, $t_{ref} \in (0.5 \cdot t_{ep}, 0.75 \cdot t_{ep}]$
- Sector 5 : $N_{ref} \in [1400, 1600]$, $t_{ref} \in (0.75 \cdot t_{ep}, t_{ep}]$

The speed profile generator assigned a reference speed value for each sector . The duration of some of the sectors was also random , namely sectors 1,2 and 5 where in some experiments assigned random duration.

6.2.2 Training results A = 0

Now that the general methodology of the training has been established the results of the training are presented starting with the case where the relative weight coefficient A is set to 0 . Among various runs the reward sub-functions' weights where determined for this case to be the following

$$\overrightarrow{w}_{A=0} = [w_N \ w_S \ w_{\dot{m}} \ w_{\dot{N}} \ w_t] = [3 \ 12 \ 1.05 \ 1 \ 0.1]$$

The training was performed for $250 \cdot 10^3$ steps producing the following rewards

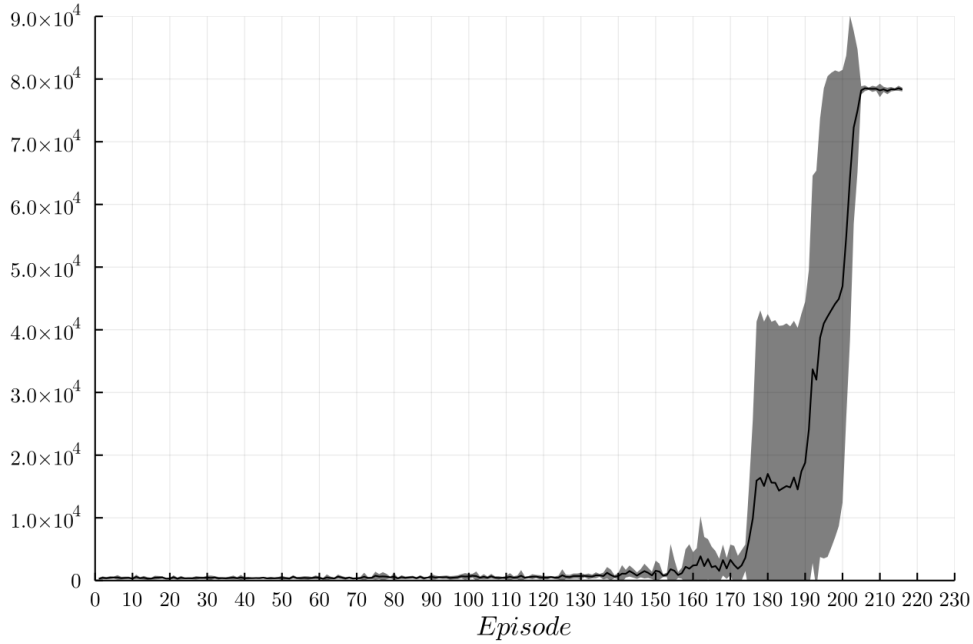

Figure 6.5: Average Reward per Episode (A=0) : The continuous line represents the mean reward among the N_{env} and the grey portions represent the deviation

The agent achieves convergence in the last episodes of the experiment and at that point completes the 5000 steps of the episode with a normalized score (ratio of the score attained to maximum possible score) around 0.93. The results of the last episode are almost identical for every environment (from the graph deviation is minimal) and are the following

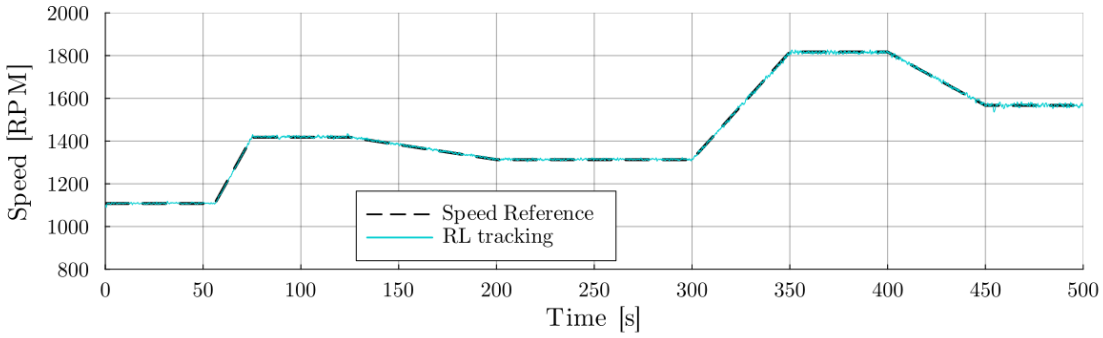

Figure 6.6: Final Episodes' speed profile and tracking speed (A=0)

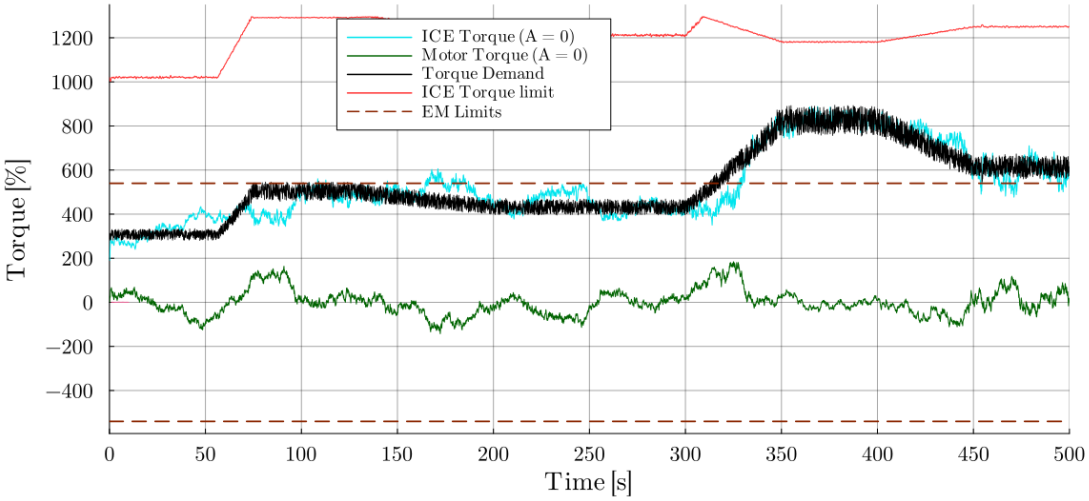

Figure 6.7: Loading and power split for the last training episodes (A=0)

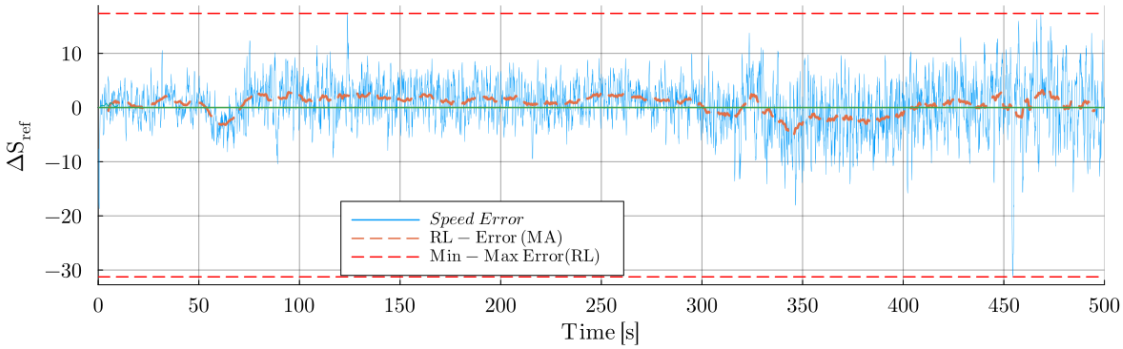

Figure 6.8: Error in tracking (A=0)

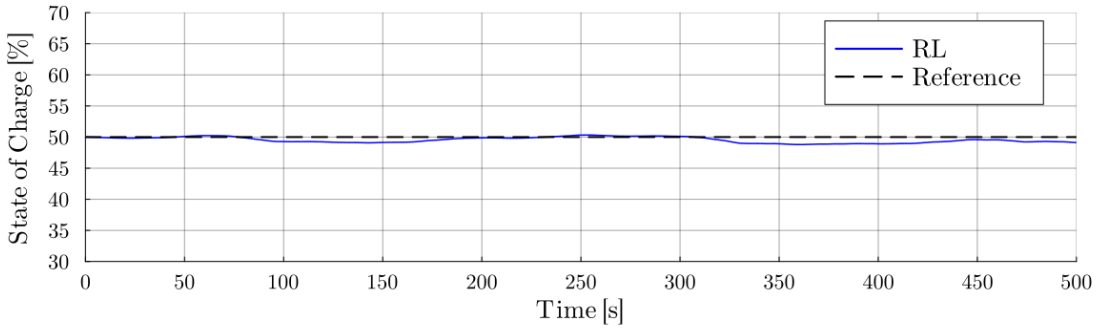

Figure 6.9: The corresponding state of charge (A=0)

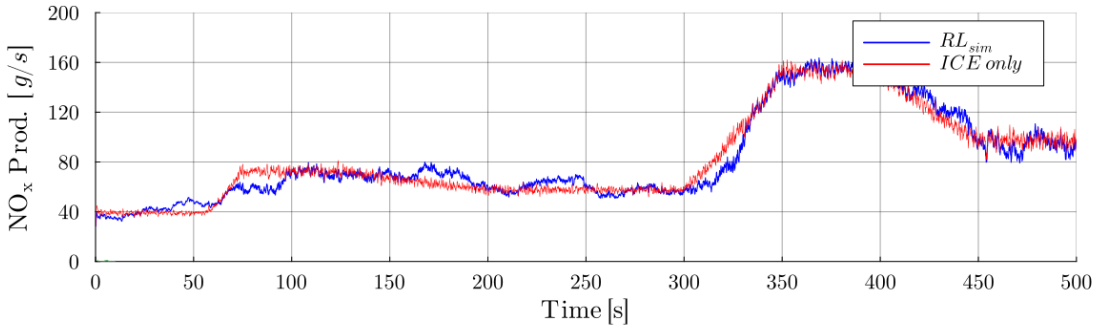

Figure 6.10: ICE's NO_x production (A=0)

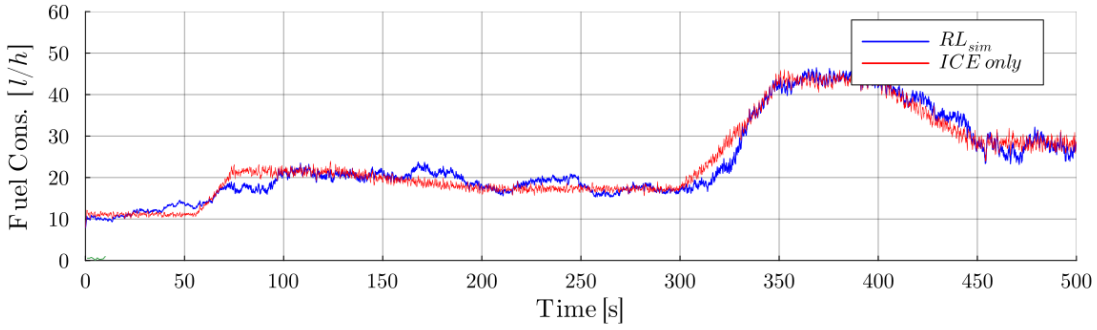

Figure 6.11: ICE's fuel consumption (A=0)

6.2.3 Training results A = 0.5

Now the corresponding results from training the environment with the relative weight coefficient set at 0.5 . The reward sub-functions' weights where determined for this case to be the following

$$\overrightarrow{w}_{A=0.5} = [w_N \; w_S \; w_{\dot{m}} \; w_{\dot{N}} \; w_t] = [3 \; 12 \; 0.85 \; 1 \; 0.1]$$

The training was performed for $250 \cdot 10^3$ steps producing the following rewards

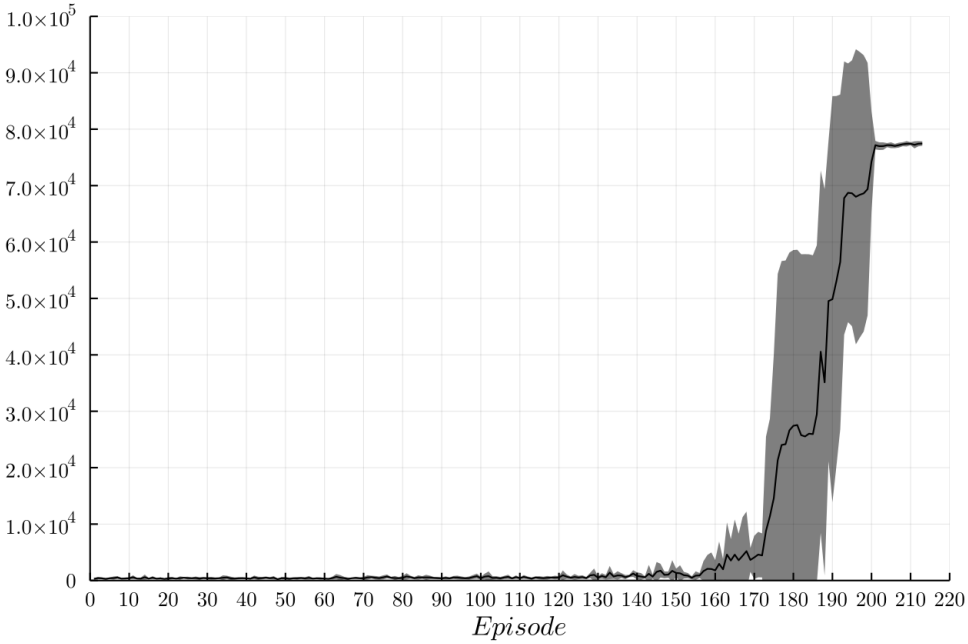

Figure 6.12: Average Reward per Episode (A=0.5) : The continuous line represents the mean reward among the N_{env} and the grey portions represent the deviation

Again the agent achieves convergence in the last 10 episodes of its training with the deviation among the environments being minimal . The normalized score at those last episodes is around 0.92 meaning that the managed to minimize all the quantities of interest . A comment about both rewards plots is that the training needs to be long enough for the agent to converge at a desirable point.

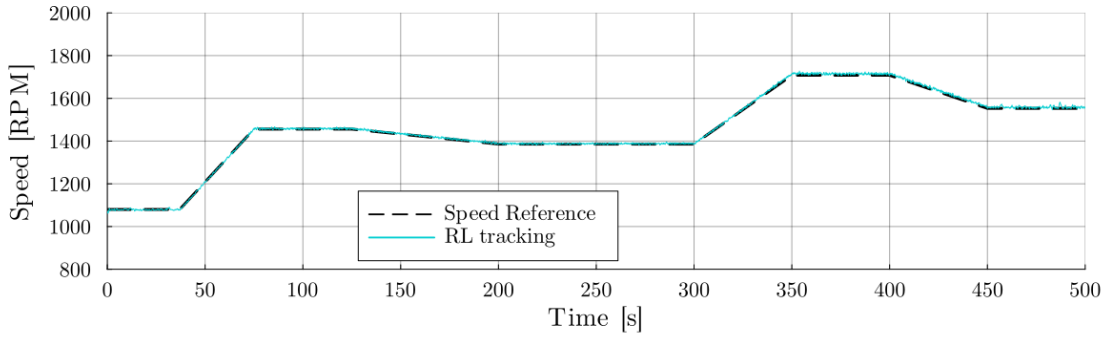

Figure 6.13: Final Episodes' speed profile and tracking speed (A=0.5)

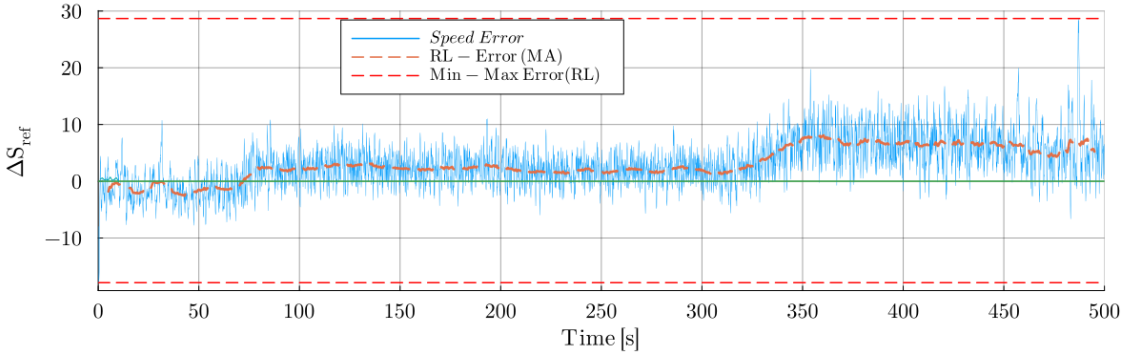

Figure 6.14: Error in tracking (A=0.5)

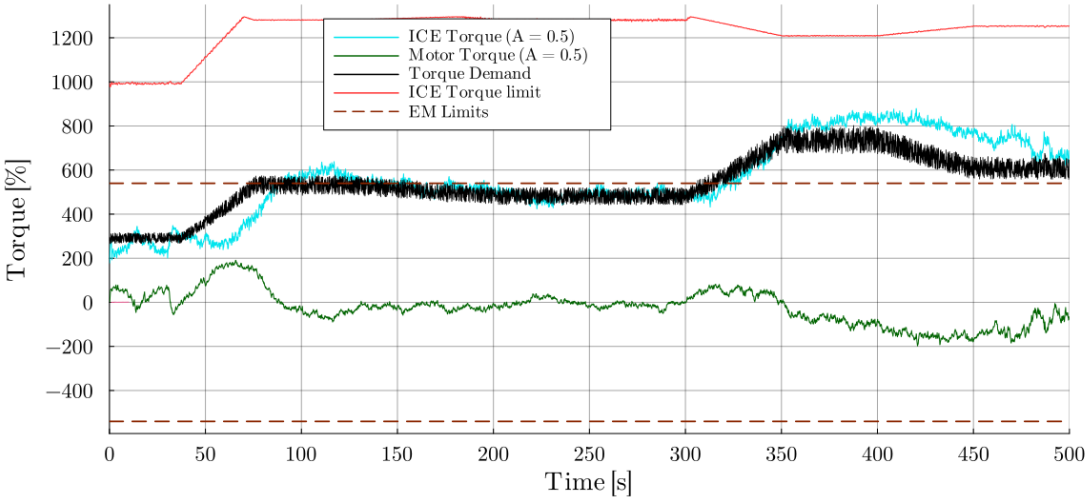

Figure 6.15: Loading and power split for the last training episodes (A=0.5)

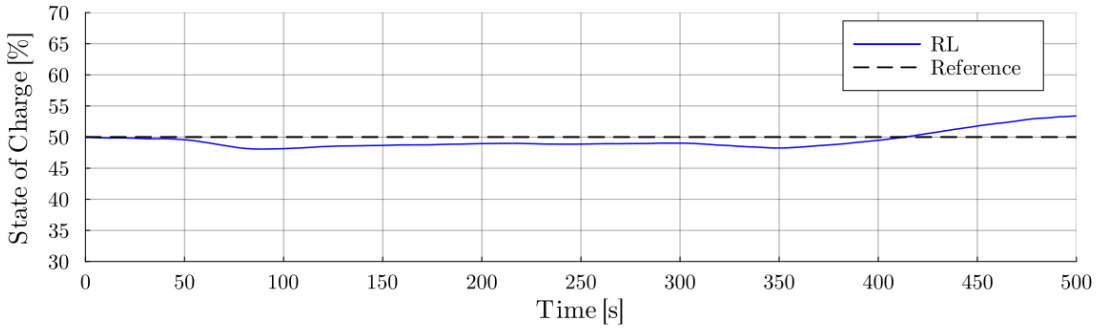

Figure 6.16: The corresponding state of charge (A=0.5)

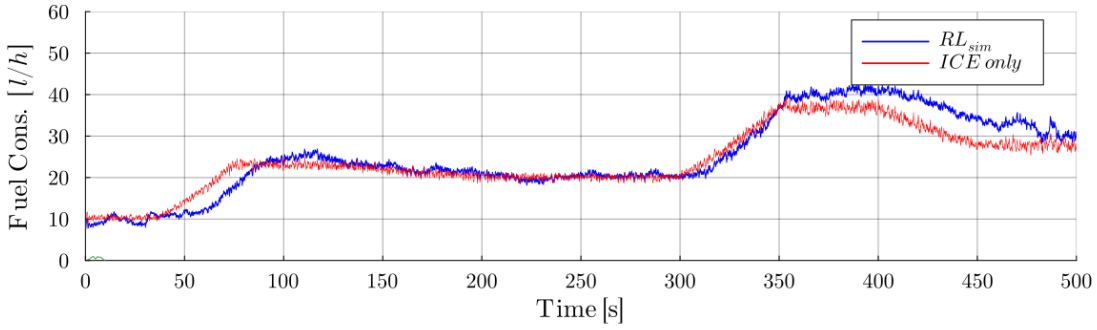

Figure 6.17: ICE's fuel consumption (A=0.5)

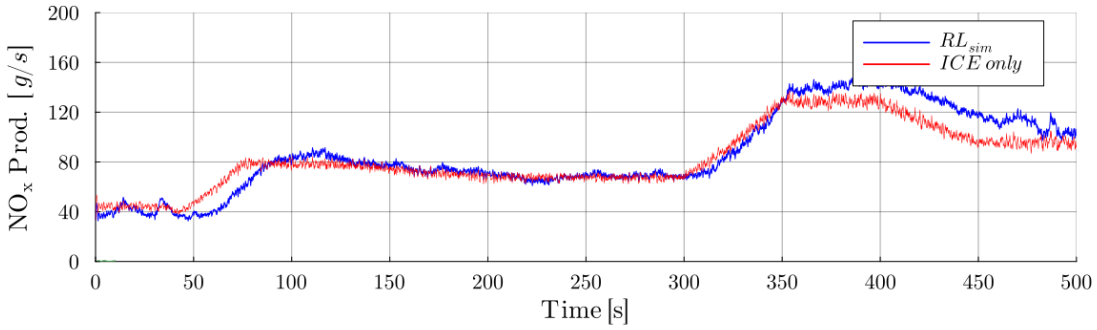

Figure 6.18: ICE's NO_x production (A=0.5)

6.2.4 Comments on Training results

In both training sessions with different fuel to weight ratio the agent managed to obtain a desirable (sub) optimal policy that enabled him afterwards to tackle the unknown load disturbances while closely following a given speed profile . After a certain number of steps in both cases the agent converged and in every environment achieved almost identical performance with minimal deviation . In both cases (A=0,A=0.5) the corresponding performance of the ICE engine is observed . The fuel consumption in each case turned out to be really close to that,resulting by operating the ICE on its own . The quantities of interest from the last of episode of its training session is presented below . The NOx production in the case of A = 0.5 turned out to be a little bit larger than the case where the ICE is solely operated and that occurred because the agent recharged the battery above the reference level of $SoC = 50$.

Weight A (Fuel to NOx)	Cumulative Fuel [l]	Cumulative NO_x [g]	SOC Difference [%]
A=0.0	3.29	41.13	-1.76
ICE only	3.33	41.18	-
A=0.5	3.48	43.26	6.78
ICE only	3.34	41.44	-

Table 6.3: Training results concerning the fuel and NOx production for each training scenario

6.3 Simulation Results

The training procedure essentially calibrated the agent's actor neural network so as to produce a desirable action when faced with a previously unknown state . The now "experienced" agent is set to handle the following scenario . A more sophisticated level where the agent needs to calculate the optimal control commands for a more complex speed profile . That speed profile is selected from a number of load cycles provided in [2] which was used to experimentally test the NMPC controller on HIPPO-2 . The foresaid cycle is the following

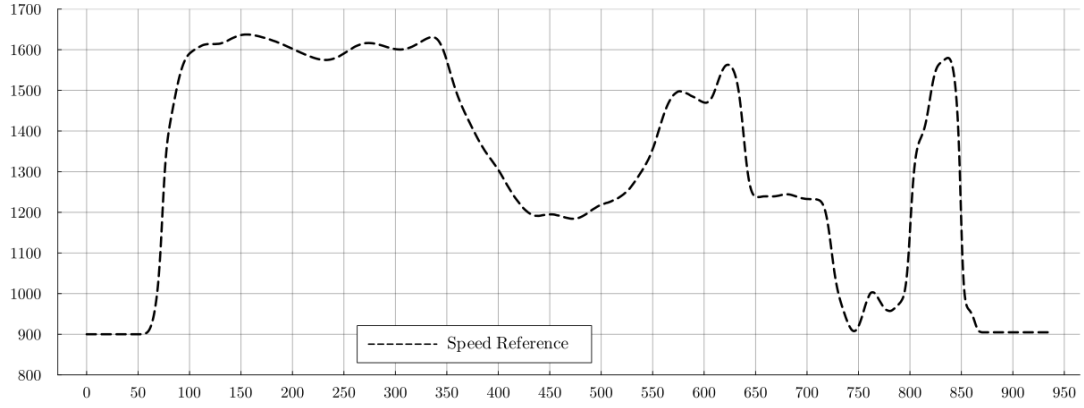

Figure 6.19: Speed reference profile used in simulation

The selection of that cycle is the appropriate test to examine the agent's ability to generalize its learning and apply them to unknown instances . The aforementioned speed profile is consisted of numerous speed variations that cover the whole spectrum of the engine speed with curt transitions among them that enable the testing of the agent's response to such transient conditions .

6.3.1 Weight Coefficient A = 0

During the simulation the same values for the control scheme parameters where implemented . The values used are the same used for the NMPC simulations in order to set a common basis for comparison . Starting with the A=0 case the agent's performance is the following .

It should be noted that the following results for the NMPC refer to experimental results and not simulation meaning that the transient response seen for instance in the beginning of the experiment is not characteristic of the NMPC's performance in a corresponding simulation .

The agent's performance regarding the speed tracking is really good considering that the maximum speed error is around 40 rpm while the majority of the speed error values lie within the [-10,10] range . That result is encouraging because it indicates that the agent managed to generalize and apply the experience gained from the training session into the previously unknown and rather complex speed profile . One can also observe from the SOC diagram that in general the agent chose a different path to satisfy the requirements of the simulation namely reducing the NOx production and the corresponding fuel consumption .

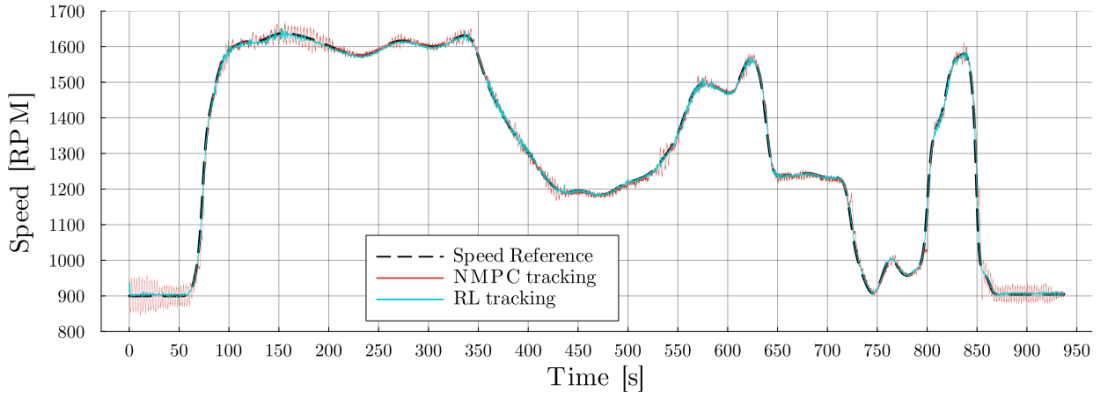

Figure 6.20: Speed Tracking A = 0

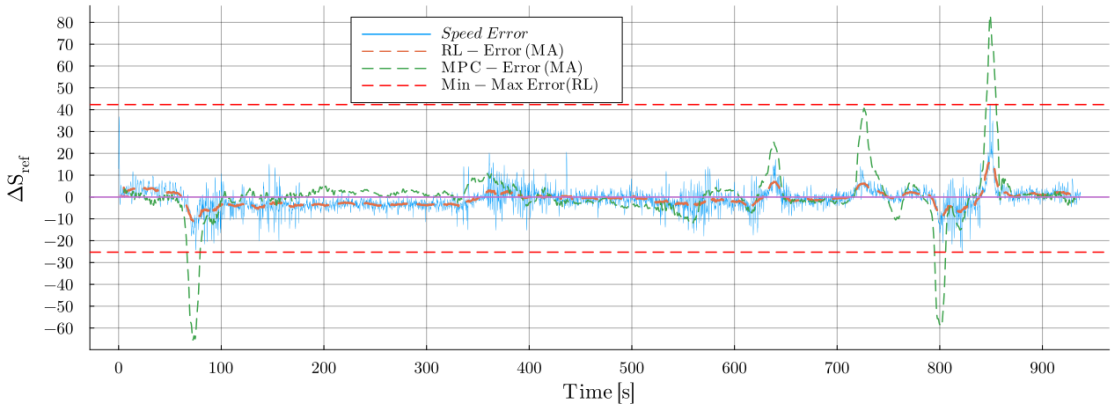

Figure 6.21: Speed Error A = 0 RL - NMPC

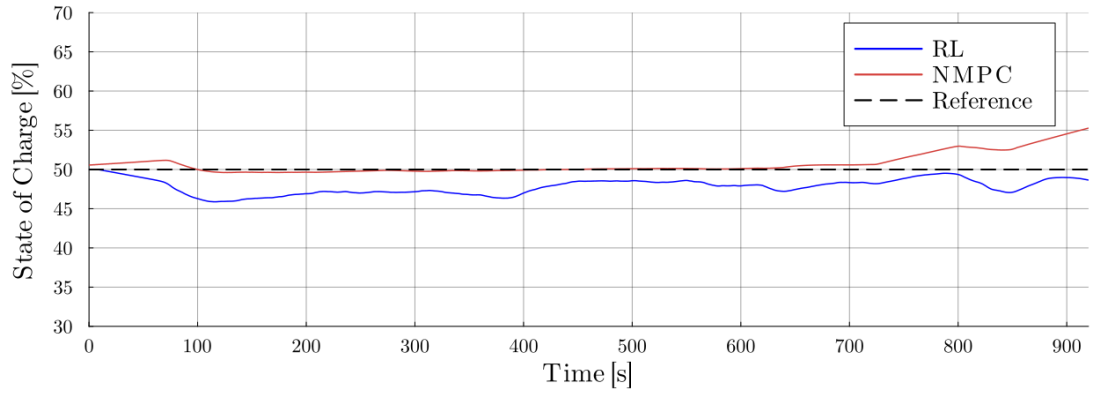

Figure 6.22: SOC trajectory A = 0 RL - NMPC

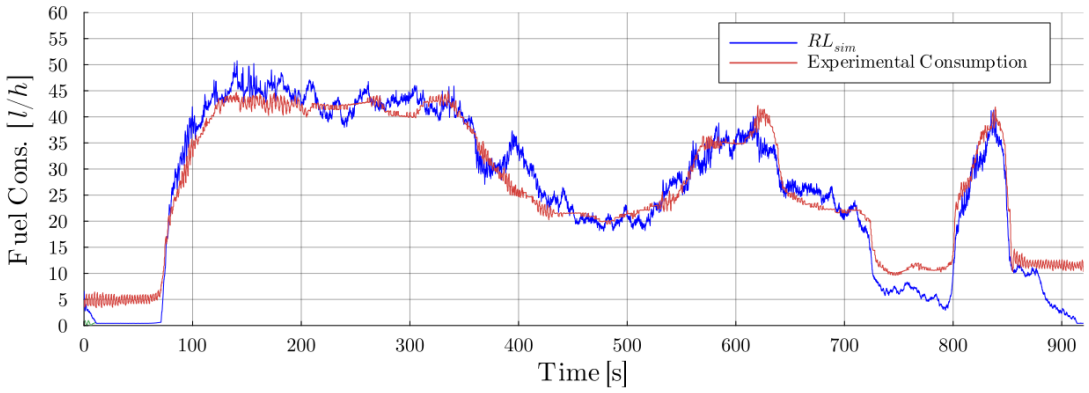

Figure 6.23: RL-NMPC comparable Fuel Consumption (A=0)

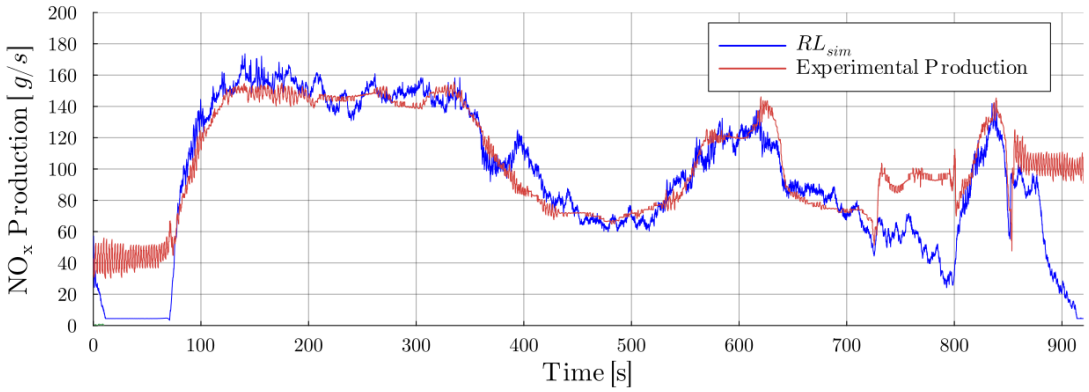

Figure 6.24: RL-NMPC comparable NOx Production (A=0)

6.3.2 Weight Coefficient A = 0.5

The corresponding results for the A=0.5 case are presented below . Again the agent had no issue dealing with a completely unknown speed reference profile while also satisfying the load demand .

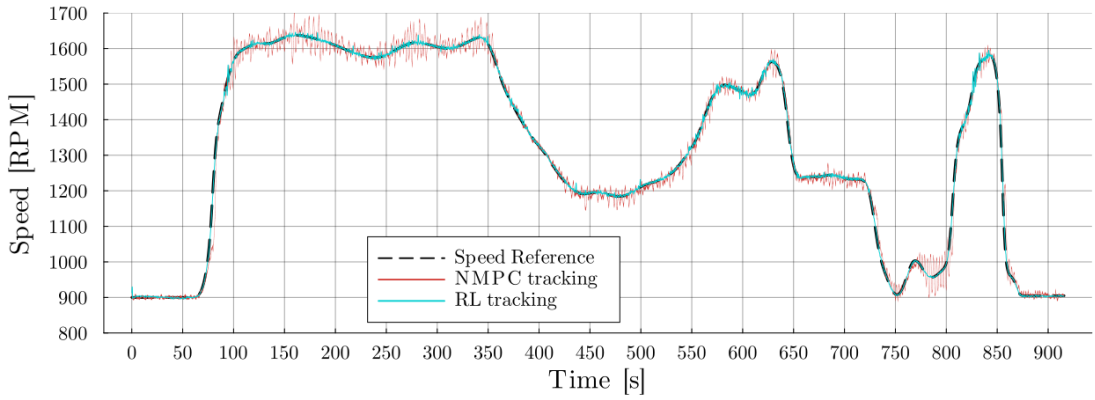

Figure 6.25: Speed Tracking A = 0.5

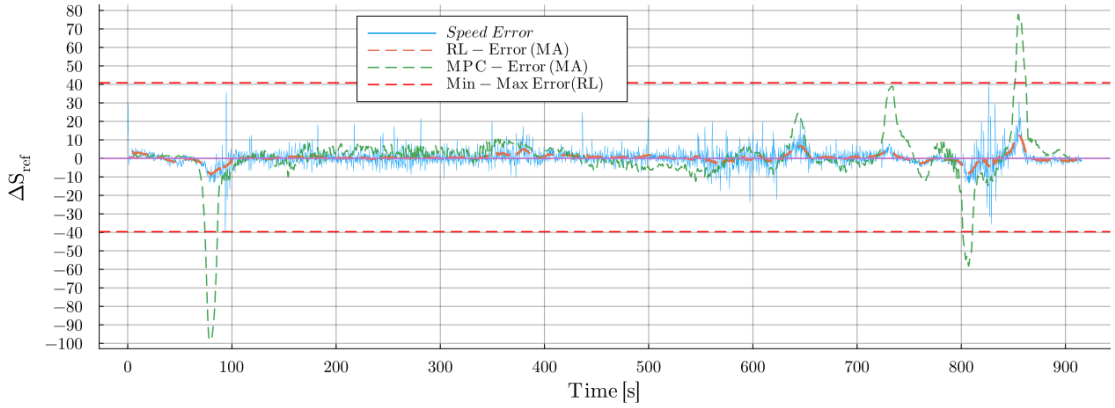

Figure 6.26: Speed Error A = 0.5 RL - NMPC

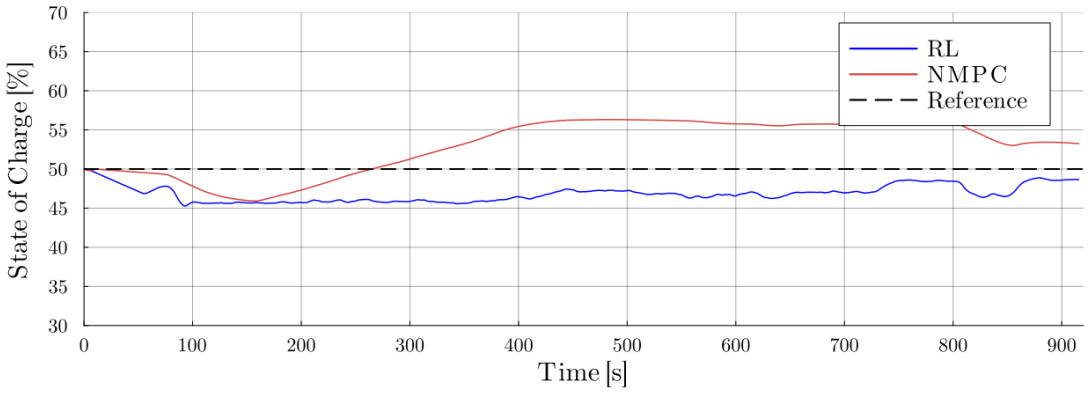

Figure 6.27: SOC trajectory A = 0.5 RL - NMPC

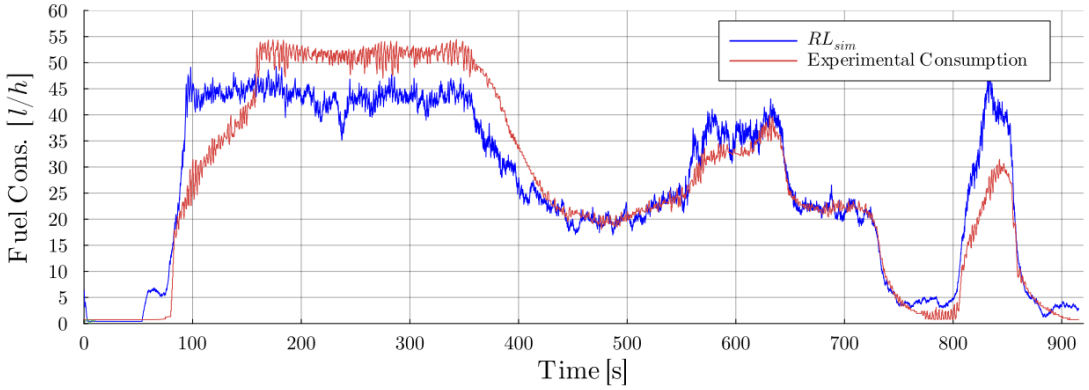

Figure 6.28: RL-NMPC comparable Fuel Consumption (A=0.5)

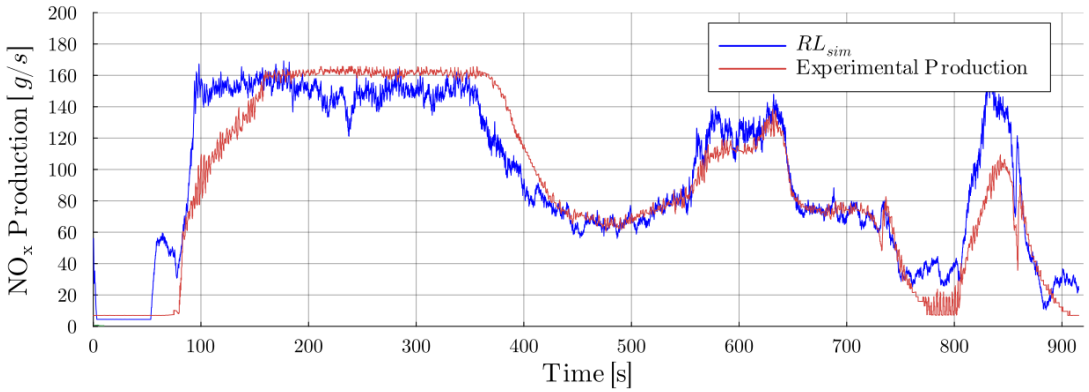

Figure 6.29: RL-NMPC comparable NOx Production (A=0.5)

6.3.3 Weight Coefficient A = 0.8

A final case is also presented where the weight coefficient A was set to 0.8 . In this mode the agent chose a slightly different splitting strategy . The corresponding graphs are provided below

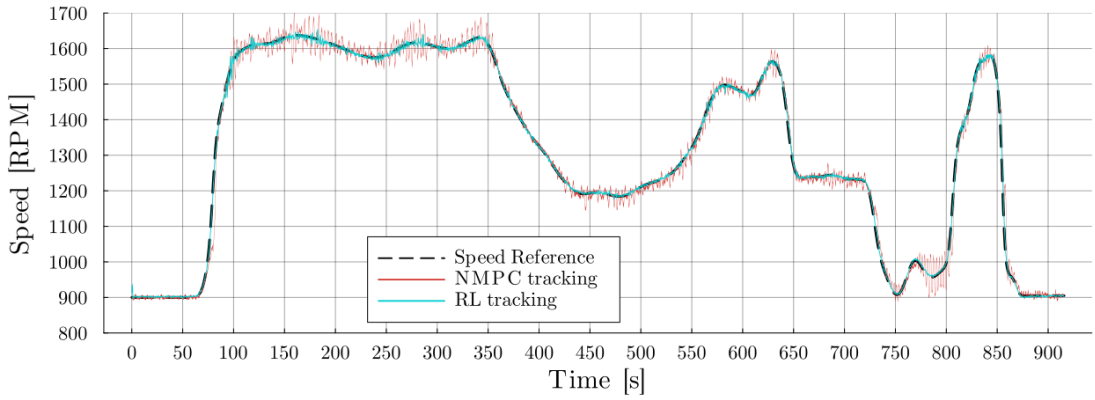

Figure 6.30: Speed Tracking A = 0.8

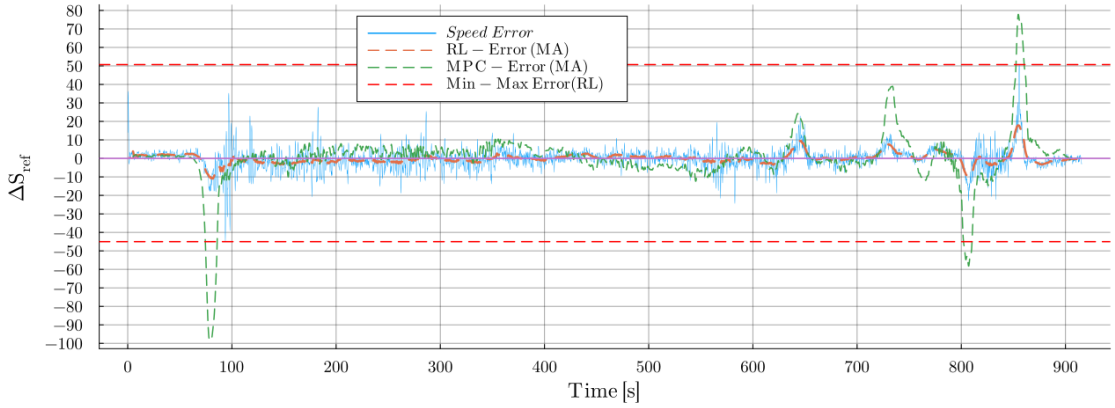

Figure 6.31: Speed Error A = 0.8 RL - NMPC

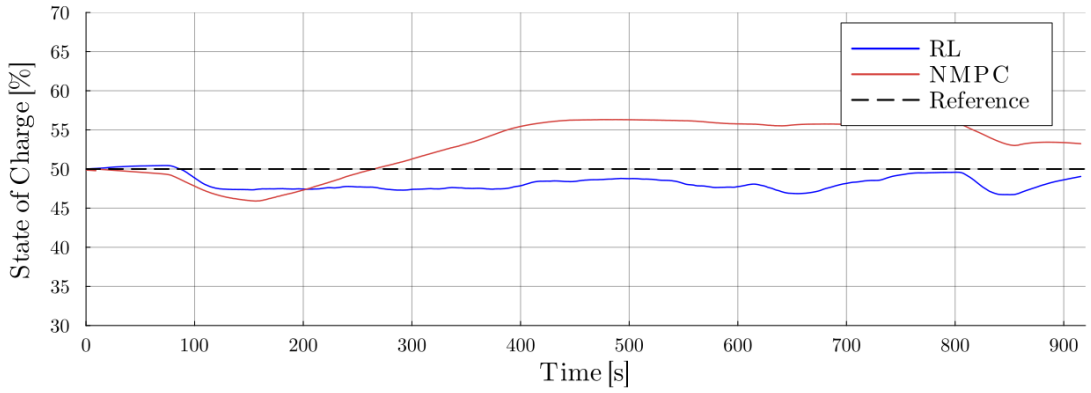

Figure 6.32: SOC trajectory A = 0.8 RL - NMPC

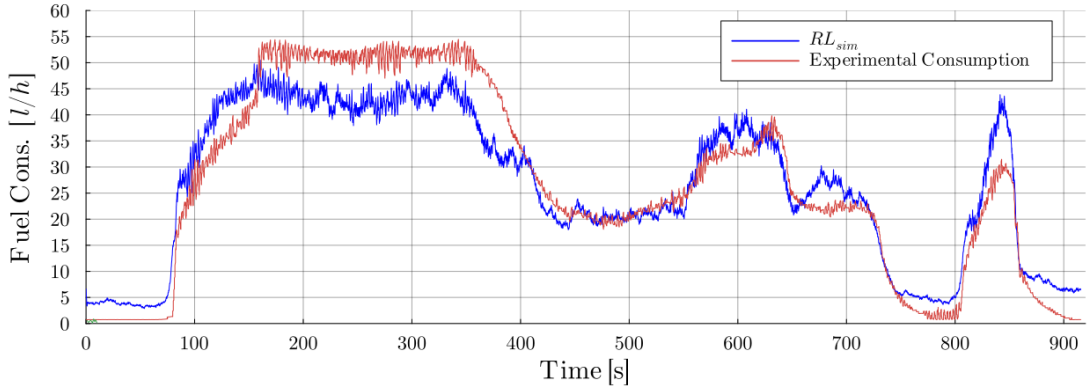

Figure 6.33: RL-NMPC comparable Fuel Consumption (A=0.8)

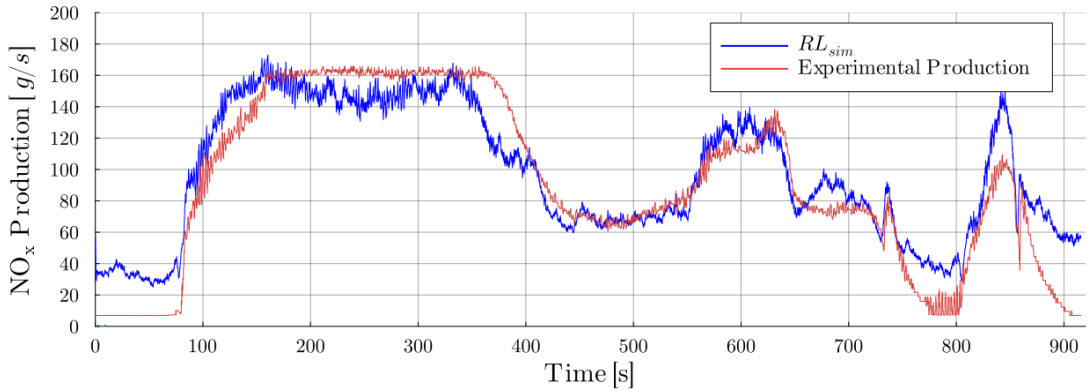

Figure 6.34: RL-NMPC comparable NOx Production (A=0.8)

6.3.4 Cases Comparison

In this section a brief comparison of the behavior of the agent is discussed while also the final results of the simulations are presented . Generally , in both cases the agent managed to achieve the basic goal of tracking the predefined speed profile while also keeping NOx production and fuel consumption within lower levels .

The nature of the training however did not allow for significant alterations in the behavior of the agent according to weight coefficient A . Indeed the fuel ration A = 0 forced the agent to follow a path that minimized fuel consumption while A = 0.5 allowed for a compromise between fuel consumption and the total NOx production which was abridged in this case . Nevertheless , in both cases the agent had an inclination towards choosing a path that made good use of the auxiliary machine mostly in motoring mode . The reason for that is probably the fact that the reward function depended on the rates of fuel consumption and NOx production and not their cumulative value . This essentially means that the agent opted for minimizing the rates which tend to be better for tracking purposes rather than the whole equivalent consumption .

In the NMPC scheme a final stage penalty can easily be enforced to penalize the excessive cumulative equivalent fuel consumption . On the RL scheme something like that poses a number of difficulties considering that the agent relies on reward during each state transition ; solving those issues was considered to be beyond the scope of this thesis.Note also that in the case of the NMPC the controller was optimized to minimize the cumulative weighted consumption $m_{ec} = Am_f + (1 - A)m_{NOx}$ while **ignoring** the term involving the battery (3.3.3).

The results regarding the quantities of interest are presented in the following table ,

Weight A (Fuel to NOx)	Cumulative Fuel [l]	Cumulative NO_x [g]	SOC Difference [%]
RL (A=0)	6.72	87.18	3.56
NMPC (A=0)	6.71	94.7	2.82
RL (A=0.5)	6.76	86.9	2.7
NMPC (A=0.5)	6.83	91.8	3.41
RL (A=0.8)	6.75	88.86	1.90
NMPC(A=0.8) sim.	6.85	94.1	3.41
ICE Only	6.62	100.4	-

Table 6.4: Comparative Simulation Results

The results seem encouraging for the agent as in both simulated cases the cumulative fuel as well as the cumulative NOx production are around the same level with

the NMPC controller or even lower . In both cases one can also observe a significant reduction in the cumulative NOx production relative to that of the NMPC case and to that of the ICE-only operation.

The following diagramms present the comparative plots for the SOC , fuel consumption rate and NOx production rate as well as the diagram presenting the power split decisions to the enforced load demands .

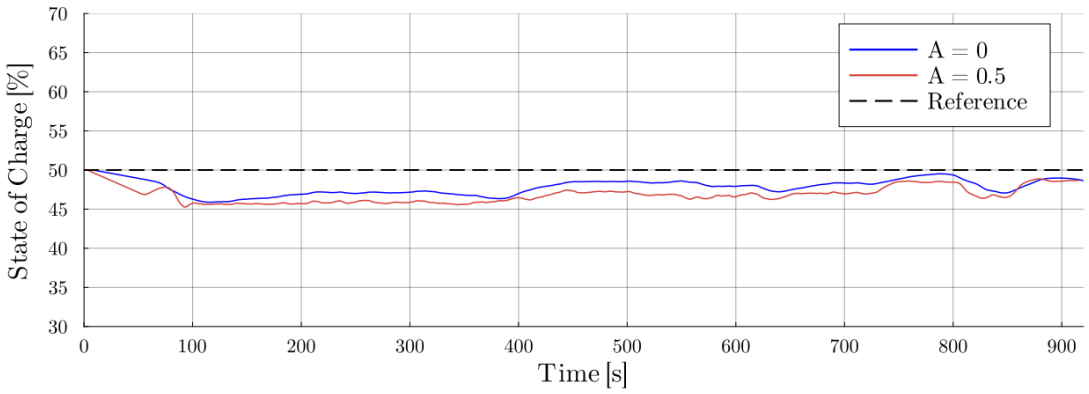

Figure 6.35: Comparative SOC trajectories

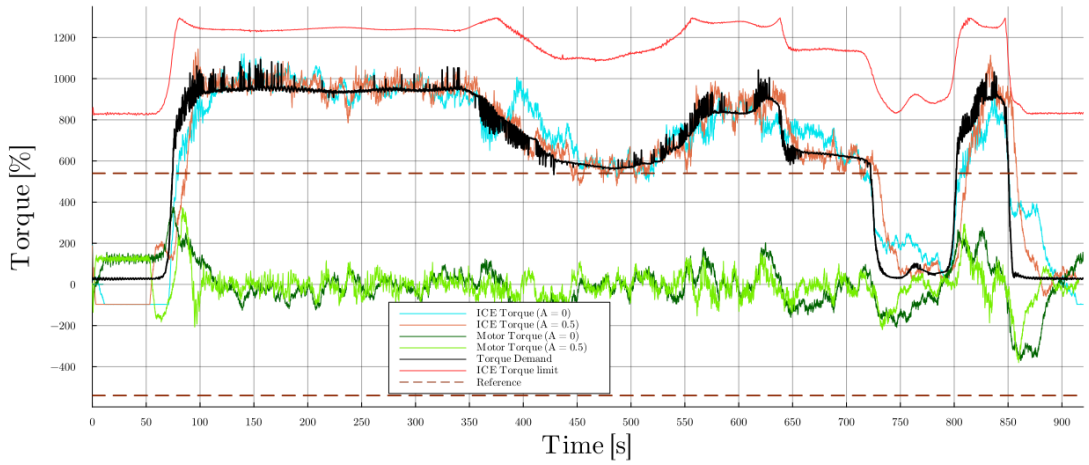

Figure 6.36: Power split and load demand A=0 - A=0.5

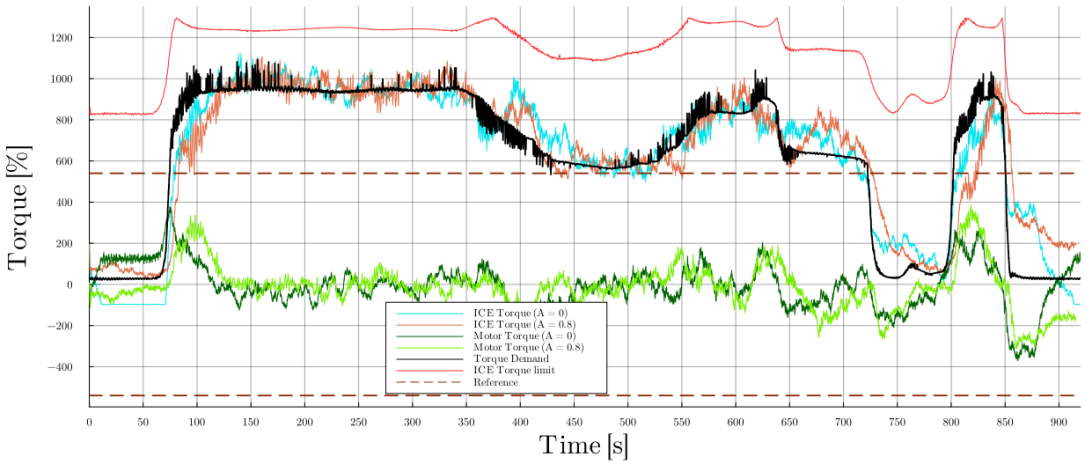

Figure 6.37: Power split and load demand A=0 - A=0.8

Some final comments pertinent to the load diagram , is that in this case the agent was able to satisfy the load demands and its irregular fluctuations . Remember that during training the load provided was following the propeller law with the coefficient c_{load} not surpassing a value of $2.5 \cdot 10^{-5}$. It is noteworthy the fact that during the simulation the agent was regularly exposed to c_{load} values significantly higher than those the agent was "familiar" with during its training . This indicates that overfit was essentially avoided , enabling the agent to generalize .

The agent's strategy in splitting the power is mainly trying to maintain the SOC level around the desired reference while also taking advantage of the auxiliary motor when needed for example during power sudden load increase ($t_1 \approx 75s \; and \; t_2 \approx 800s$) . During small load transitions or seemingly constant load the agent decides to keep the SOC levels around the same levels . The reason that happens is because the goal is having a SOC level close to 50 at the end of the load cycle and not have a fully discharged battery . In the case where A was set to 0.8 it can be observed from figure 6.37 that the agent initially utilizes the motor in generating mode increasing the battery's charge in order to take advantage of it afterwards . The agent then chooses again the fast transients around t_1 *and* t_2 to utilize the battery while the rest of the time maintaining the SOC levels relatively constant .

Finally , the comparative plots for the $\dot{m}_f$ and $\dot{m}_{NOx}$ quantities are presented

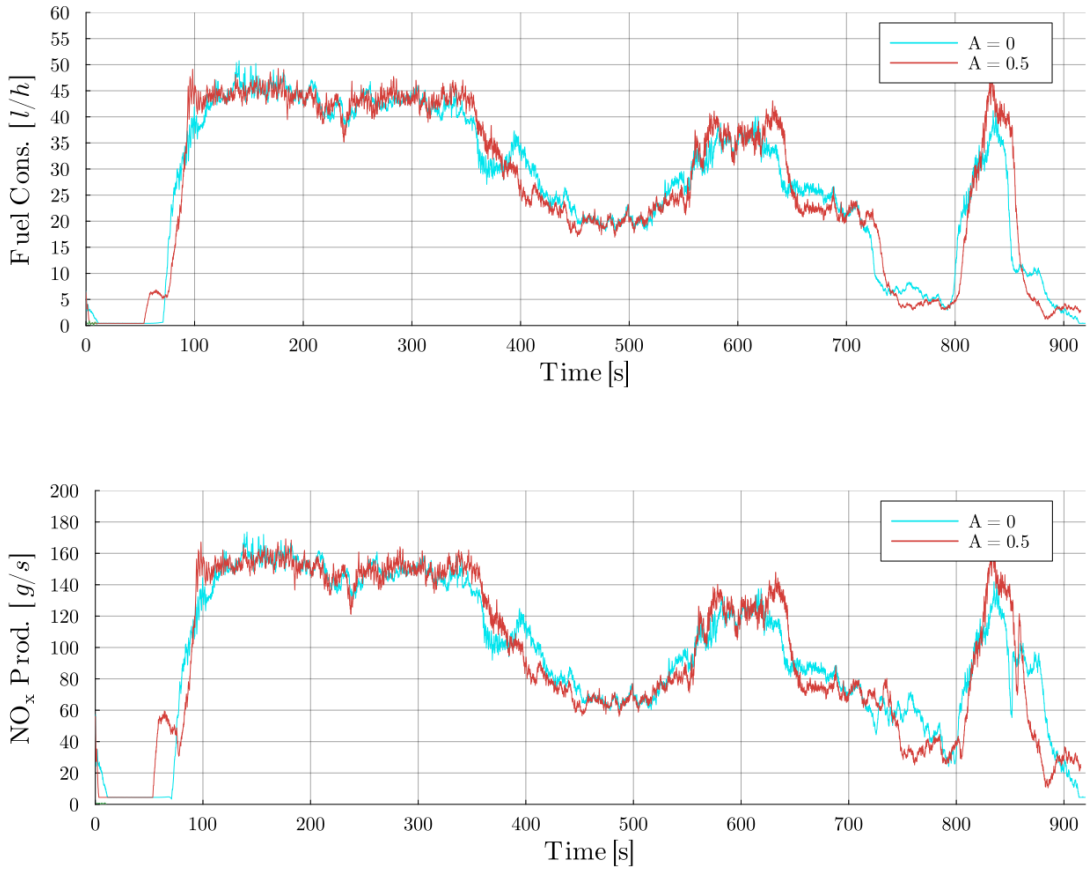

Figure 6.38: Comparative engine performance for the two different weighting coefficients

Chapter 7

Conclusion

This is the conclusion chapter in which the work is recapitulated and the some of the emerging results are discussed . Potential future work ideas are also provided for individuals who want to base their project on this work or even enhance it .

7.1 Work Summary and Conclusions

The main goal of this work is implementing the newly emerged reinforcement learning methodologies into real life applications and in particular the energy management of the HIPPO-2 hybrid diesel electric propulsion plant . The main task of this thesis was to tackle the power split optimization problem of HIPPO-2 with the application of a rather inchoate control scheme that pertains to reinforcement learning. This project was largely influenced by the work of Vasilios Karistinos [2] and Nikolaos Planakis [3] and other contributors who the Non linear model predictive control to develop a real time Energy Management and Emissions Minimization System (EMEMS). That system is reproduced in this work via the use of policy gradient methods namely the proximal policy optimization (PPO) , with the workflow being the following .

Initially , the differential equations governing the environment were updated with models designed in the laboratory of marine engineering (LME) . Once the environmet's dynamics were developed the implementation of the (PPO) algorithm took place with the necessary calibrations . In general, only policy gradient methodologies were selected with the main reason for that being the fact that policy gradient methodologies are the ones that can handle continuous action and state spaces more efficiently .

The main objective was to create an agent who would essentially play the part of the classic controller and decide on the optimal power split actions to minimize

emissions while also satisfying other constraints that pertain to load demands , speed profile tracking and maintaining a valid SOC level.The main idea behind EMEMS is the use of a quantity called equivalent fuel consumption that simultaneously takes into account the fuel consumption , NOx production and the electric energy usage from the battery.In order to do that , the agent was trained at first in various simple yet sufficient environments that would eventually allow the agent to generalize into more complex conditions . Random speed tracking profiles with random transitions in combination with a variable load that abided by the propeller law (varying the coefficient) were effective for that cause . The training results were evaluated by observing the convergence of the reward function relative to the maximum reward that the agent could attain as well by collating the results with the scenario in which only the diesel engine was operating. The agent was trained on two scenarios with different fuel to NOx weight coefficient (A=0 & A=0.5) .

Finally,once the agents training was successful the performance of the agent was tested in simulations using load cycles and speed profiles identical to those used in [2] in order to have a common basis for a basic result's evaluation . The results of the simulation where encouraging because the agent proved able to cope with unknown inputs. The agent during training managed to develop a certain logic that enabled it to generalize and perform well with respect to the constraints and the fuel economy and emissions reduction goal . As already mentioned in the corresponding section the simulated results indicated that the agent could perform equally well or even better in some cases with a more robust control scheme like the NMPC .

To summarize, the thesis is aligned with the general attitude towards the application of reinforcement learning in the field of control and automation . The results produced in this work indicate that an RL agent could potentially substitute the classic controller in various control related applications that require real time optimization . Nevertheless, considerable work needs to be done in the field of reinforcement learning to tackle the various drawbacks surrounding the methodologies of this scheme mainly those regarding stability , efficient data manipulation and the black-box dynamics of the methods that entail the use of neural networks.

7.2 Future Work Suggestions

This thesis provides the necessary impetus for anyone considering to further develop the application of reinforcement learning methods into control related projects .

First, while the agent presented the desirable performance in simulation , experimental tests need to be performed in order to solidify the general capabilities of that scheme . Moreover , although the models used in this thesis and the training

methodologies where considered adequate there is always room for improvement by introducing higher fidelity models . The training process could also be enhanced by introducing the concept of levels cited in the paper ACCEL [13] where the agent is facing levels with difficulty to complete that is proportional to its performance at a given instance . It is a really interesting concept that is worthwhile exploring . Furthermore , in this thesis the reward functions used were those that provided the desirable results . However , further investigating the dynamic of the reward function as well as the effect of other hyperparameters of the RL scheme provide an interesting topic overall . Finally , other RL methodologies could be implemented combined with different control objectives .

Bibliography

[1] Reinforcement Learning: An Introduction R. Sutton, and A. Barto. The MIT Press, Second edition, (2018)

[2] N.PLanakis Power-Split Strategies for Hybrid Marine Propulsion Plants in Transient Loading Conditions for Optimal Energy Management and Emissions Reduction

[3] V. Karystinos, "Nonlinear model predictive control of a hybrid diesel-electric marine propulsion plant," 2019

[4] Dimitri P. Bertsekas, Reinforcement Learning and Optimal Control Massachusetts Institute of Technology

[5] Rushikesh Kamalapurkar, Patrick Walters, Joel Rosenfeld, Warren Dixon, Reinforcement Learning for Optimal Feedback Control A Lyapunov-Based Approach

[6] N. Planakis, G. Papalambrou, and N. Kyrtatos, "Predictive power-split system of hybrid ship propulsion for energy management and emissions reduction," Control Engineering Practice, vol. 111, p. 104795, 2021.

[7] N. Planakis, G. Papalambrou, and N. Kyrtatos, "Integrated Load-Split Scheme for Hybrid Ship Propulsion Considering Transient Propeller Load and Environmental Disturbance," Journal of Dynamic Systems, Measurement, and Control, vol. 143, 10 2020. 031004.

[8] N. Planakis, V. Karystinos, G. Papalambrou, and N. Kyrtatos, "Nonlinear model predictive control for the transient load share management of a hybrid diesel-electric marine propulsion plant," in 2020 American Control Conference (ACC), pp. 1955–1960, 2020.

[9] N. Planakis, V. Karystinos, G. Papalambrou, and N. Kyrtatos, "A predictive energy management system for a hybrid diesel-electric marine propulsion plant," in 2020 European Control Conference (ECC), pp. 693–698, 2020.

[10] G. Sulligoi, S. Castellan, M. Aizza, D. Bosich, L. Piva, and G. Lipardi, "Active frontend for shaft power generation and voltage control in fremm frigates integrated power system: Modeling and validation," in International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion, pp. 452–457, June 2012.

[11] S. K. Topaloglou, G. Papalambrou, K. Bardis, and N. Kyrtatos, "Transient load share management of a diesel electric hybrid powertrain for ship propulsion," International Journal of Powertrains, vol. 7, p. 341, 01 2018.

[12] N. Planakis, G. Papalambrou, and N. Kyrtatos, "Predictive control for a marine hybrid diesel-electric plant during transient operation," pp. 989–994, 04 2018.

[13] Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, Tim Rocktäschel , Evolving Curricula with Regret-Based Environment Design