Optimal maintenance of deteriorating systems integrating deep reinforcement learning and Bayesian inference

MSc Thesis
Civil Engineering
Structural Engineering track

Student

Christos Lathourakis | 5143616

Assessment committee

Dr. C. P. Andriotis (*Daily Supervisor*)
Dr. A. Cicirello (*Chair*)
Dr. A. A. Nunez Vicencio

“Wir müssen wissen. Wir werden wissen”
David Hilbert (1930)

ABSTRACT

An issue of utmost significance constitutes the maintenance of engineering systems exposed to corrosive environments, e.g. coastal and marine environments, highly acidic environments, etc. The most beneficial sequence of maintenance decisions, i.e. the one that corresponds to the minimum maintenance cost, can be sought as the solution to an optimization problem. Owing to the high complexity of this sequential decision optimization problem, traditional methods such as threshold-based approaches, fail to arrive at an optimal strategy, while at the same time the commonly used offline knowledge about the environment can not capture efficiently the stochastic way in which an engineering system deteriorates. Over the last few years, Deep Reinforcement Learning (DRL) has been proven a promising tool to tackle such problems, being often limited though by the dimensionality curse and the implications caused by large state and action spaces, an issue which leads to simplifications like their discretization. Bayesian principles and model updating are the most widely used tools to model accurately systems with high uncertainty, by incorporating data acquired through monitoring devices and thus improving the knowledge about the stochastic system.

This research proposes an integrated framework that aims to determine an optimal sequence of maintenance decisions over the lifespan of deteriorating engineering systems, combining the aforementioned core concepts of Deep Reinforcement Learning (DRL) and Bayesian Model Updating (BMU). More specifically, it investigates different Deep Reinforcement Learning (DRL) algorithms, namely Double Deep Q-Network (DDQN), Advantage Actor Critic (A2C), and Proximal Policy Optimization (PPO), while the updating of the uncertain parameters is performed through sampling, i.e. No-U-Turn Sampler (NUTS). All these tools will be first applied to elementary problems for the sake of verification and validation, while the culmination of this research is the application of the framework on a more realistic and complicated, multi-component structure. The obtained results are compared with benchmark performances to properly showcase the efficiency and the weaknesses of the tool.

ACKNOWLEDGMENTS

I always considered the coupling between computer science and civil engineering fascinating. Back in November, I had a solid background in structural engineering and Python as tools in my hands, and zero experience on Deep Reinforcement Learning (DRL) and Bayesian inference. I was willing to dive into unknown fields, gain new practical and theoretical knowledge, and expand my skill set. My mentors seemed more than certain that I could pull this off, and I am grateful for their trust and glad for proving them right seven months later. Of course, this amazing experience, and unique opportunity to learn and grow, would not have been the same without the contribution of many people whom I would like to thank.

First of all, I would like to express my gratitude to my mentors Charalampos Andriotis and Alice Cicirello for their tremendous assistance and support during this project. Their passion for the field is an ever-motivating factor that was and keeps inspiring me. They provided me with incredible insight and feedback, and their total guidance was a key factor in completing this project. I want to also thank Alfredo Nunez Vicencio for his valuable input and his constructive comments that improved significantly aspects of the project. Lastly, I am grateful for the genuine help of Ziead Metwally, who was always willing to discuss with me various challenging issues and provide me with valuable insight.

Last but not least, I would like to thank my friends and family, both back in Greece and the Netherlands. In particular, I am thankful to Arte for being a friend and supporter of mine for longer than I can remember, Lisa and Matteo, for the unique trips, dinners, and brunches, Giorgos and Maria, for the countless movie and board game nights we had, my good friend Kostas, for continuing to be the best candidate to discuss any new project I set my hands on, and many others I would like to include, for being part of my life and supporting me in their unique way. Special thanks to my dear friend and partly roommate Leo for being part of so many beautiful experiences, but this time also for the extra computational power he generously granted me during this project. I could not quantify the contribution of my partner Anastasia to every little or big achievement of my adult life. This one could not differ, with her support in every daily aspect and her encouragement for me to keep pushing forward, played a vital part in me completing this journey. Finally, I would like to thank my brother Nikos and my mother Zoi, who always supported and believed in me, and trying to make them proud has always been a driving factor.

Christos Lathourakis
Delft, July 2022

Contents

Acronyms **ix**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Research Question . . . 3
1.4 Research Methodology . . . 4
1.5 Thesis Structure . . . 7

2 Literature Review **8**
2.1 Markov Decision Process (MDP) . . . 8
2.2 Partially Observable Markov Decision Process (POMDP) . . . 10
2.3 Reinforcement Learning (RL) . . . 11
2.4 Deep Reinforcement Learning (DRL) . . . 12
2.4.1 Deep Q-Network (DQN) . . . 13
2.4.2 Double Deep Q-Network (DDQN) . . . 14
2.4.3 Advantage Actor Critic (A2C) . . . 14
2.4.4 Proximal Policy Optimization (PPO) . . . 15
2.4.5 Other Deep Reinforcement Learning (DRL) algorithms . . . 16
2.5 Bayesian Model Updating (BMU) . . . 16
2.6 Deterioration Processes . . . 18
2.7 Research gap . . . 20
2.8 Conclusions . . . 21

3 Methodology **22**
3.1 General Framework . . . 22
3.2 Sampling Algorithm . . . 25
3.3 Deep Reinforcement Learning (DRL) algorithms . . . 31
3.3.1 Double Deep Q-Network (DDQN) . . . 32
3.3.2 Advantage Actor Critic (A2C) . . . 33
3.3.3 Proximal Policy Optimization (PPO) . . . 34
3.4 Conclusions . . . 35

4 Verification, Validation and Benchmarking **36**
4.1 Toy Problem . . . 36
4.1.1 Problem Description . . . 36
4.1.2 Discrete case . . . 40
4.1.3 Continuous case . . . 49
4.2 Validation . . . 57
4.2.1 Bayesian Inference . . . 57

4.2.2 Deep Reinforcement Learning (DRL) algorithms . . . 59
4.3 Benchmarking . . . 60
4.4 Results . . . 61
4.4.1 Discrete case . . . 61
4.4.2 Continuous case . . . 63
4.5 Conclusions . . . 69

5 Case Study **71**
5.1 Problem Description . . . 71
5.1.1 Modelling . . . 71
5.1.2 Actions . . . 73
5.1.3 Deterioration Model . . . 74
5.1.4 Probability of failure . . . 79
5.2 Framework . . . 83
5.3 Benchmarking . . . 88
5.4 Results . . . 91
5.5 Conclusions . . . 98

6 Discussion - Conclusion **99**
6.1 Discussion of results . . . 99
6.2 Limitations . . . 101
6.3 Future Development . . . 102

A Appendices **107**
A.1 Appendix 1 - Case Study . . . 107
A.1.1 Load Case . . . 107
A.1.2 Deterioration . . . 110

List of Figures

1 Problem Statement - General Framework . . . 2
2 Research Methodology flowchart . . . 5
3 Core `Python` dependencies[1] . . . 6
4 The agent-environment interaction in an Markov Decision Process (MDP) [15] . . . 9
5 Non stationary Gamma process describing the damage evolution over 70 years . . . 20
6 Problem conceptual breakdown - Motivation for the selected tools . . . 22
7 Graphical Model of the employed POMDP . . . 23
8 Proposed Framework . . . 25
9 Actor-critic Deep Neural Network (DNN) architectures [2] . . . 31
10 Problem conceptual breakdown - Motivation for the selected tools and algorithms . . . 35
11 Single Degree of Freedom (SDOF) oscillator . . . 36
12 Repair by reducing the damage $D(\tau)$ . . . 38
13 Repair by reducing the deterioration rate τ . . . 38
14 Repair by reducing both the damage and the deterioration rate . . . 39
15 Belief vector, $\underline{b}$ for discrete case . . . 41
16 Transition matrix, $\mathbb{P}_\tau$ for discrete case . . . 42
17 Transition matrix, $\overline{\overline{\mathbb{P}_\tau}}$ for discrete case explanation . . . 43
18 Double Deep Q-Network (DDQN) DNN architecture for the discrete toy problem . . . 44
19 Actor-critic DNN architecture for the discrete toy problem . . . 45
20 Framework flowchart for the toy problem . . . 50
21 Failure Probability calculation . . . 52
22 DDQN DNN architecture for the continuous toy problem . . . 53
23 Actor-critic DNN architecture for the continuous toy problem . . . 54
24 Posterior distributions of parameters A, B after 5 iterations of Bayesian Inference and No-U-Turn Sampler (NUTS) . . . 57
25 Posterior distributions of parameters A, B during 20 iteration of Bayesian Inference and NUTS . . . 58
26 All three algorithms on the CartPole-v0 environment . . . 59
27 DDQN and Proximal Policy Optimization (PPO) on discrete SDOF environment . . . 62
28 DDQN and PPO policy realizations on discrete SDOF environment . . . 63
29 DDQN and PPO on continuous SDOF environment . . . 64
30 DDQN and PPO policy realizations on continuous SDOF environment . . . 65
31 Probability of failure for 50 policy realizations, for both DDQN and PPO . . . 66
32 Probability of failure for 50 policy realizations, for both DDQN and PPO . . . 66
33 Updating of parameter A for nine (9) of the episodes . . . 67
34 Updating of parameter B for nine (9) of the episodes . . . 68
35 Probability of failure for 50 policy realizations . . . 69
36 Structural system employed for the Case Study . . . 71
37 First three eigenmodes of the frame . . . 72

38 Degradation of the IPE cross-section . . . 74
39 Cross section properties deterioration . . . 75
40 Belief updating flowchart . . . 76
41 Case Study Bayesian Inference . . . 77
42 Linearisation of the Limit State Surface (LSS) $M(\underline{U}) = 0$ at the design point U^* in the uncorrelated standard normal random variables $\underline{U}$ space . . . 80
43 Comparison between First Order Reliability Method (FORM) and Monte Carlo (MC) 82
44 PPO architecture - Centralized states and actions . . . 84
45 Case Study complete flowchart . . . 85
46 Policy realization for all components - CBM Benchmark . . . 89
47 TBM benchmark costs over replace intervals . . . 90
48 PPO applied on case study . . . 92
49 Policy realization for all components (constrained) . . . 94
50 Policy realization for all components for different training episodes - Left side . . . 96
51 Policy realization for all components for different training episodes - Right side . . . 97
52 Policy realization for all components (constrained) - 3D . . . 100
53 PPO architecture - Centralized states and decentralized actions . . . 103
54 PPO architecture - Centralized states and decentralized actions - Floor variation . . . 104
55 PPO architecture - Component's ID . . . 105
56 Elastic response spectrum . . . 108
57 Simplified model of the frame . . . 109
58 I-beam cross-section . . . 110

List of Tables

1 Action-space for toy problem . . . 39
2 Rewards (costs) for the toy problem . . . 40
3 Toy problem input data . . . 40
4 Counters used for DDQN . . . 46
5 Counters used for PPO . . . 46
6 Parameters of the stochastic deterioration model . . . 49
7 Benchmark maintenance thresholds and costs - Toy Problem . . . 60
8 DDQN hyperparameters - Toy problem . . . 61
9 PPO hyper-parameters - Toy problem . . . 61
10 Deep Reinforcement Learning (DRL) algorithms' performance on continuous Toy Problem . . . 64
11 Case study frame geometry and properties . . . 73
12 Rewards (costs) for the case study . . . 74
13 Case study input data . . . 79
14 CBM Benchmark maintenance thresholds and costs - Case Study . . . 88
15 PPO hyper-parameters - Case Study . . . 91

16 Benchmark and DRL performance on the Case Study 93
17 Values of the parameters describing the recommended Type I elastic response spectra [62] 107
18 Elastic quasi-static forces based on EN1998-1 [62] 109

List of Algorithms

1 Störmer-Verlet ("leapfrog") integrator 26
2 Hamiltonian Monte Carlo (HMC) 27
3 BuildTree[3]recursive function 28
4 Heuristic for choosing an initial value of ϵ 29
5 No-U-Turn Sampler (NUTS) with Dual Averaging 30
6 Double Deep Q-Network (DDQN) 32
7 Advantage Actor Critic (A2C) 33
8 Proximal Policy Optimization (PPO) [1] 34
9 Double Deep Q-Network (DDQN) - Discrete Toy Problem 47
10 Proximal Policy Optimization (PPO) - Discrete Toy Problem 48
11 Proximal Policy Optimization (PPO) agent training - Toy problem 49
12 Deterioration model parameters updating - Toy Problem 51
13 Double Deep Q-Network (DDQN) - Continuous Toy Problem 55
14 Proximal Policy Optimization (PPO) - Continuous Toy Problem 56
15 Deterioration model parameters updating - Case Study 78
16 First Order Reliability Method (FORM) geometric interpretation 81
17 Proximal Policy Optimization (PPO) - Case Study 86
18 Proximal Policy Optimization (PPO) agent training - Case study 87

Acronyms

A2C	Advantage Actor Critic ii, iv, viii, 14, 15, 31–33, 35, 44, 45, 54, 59, 70, 83, 99
AI	Artificial Intelligence 22
ANN	Artificial Neural Network 13
BMU	Bayesian Model Updating ii, iv, 1, 3, 16, 22–24, 55–57, 73, 86, 99, 102
CBM	Condition-Based Maintenance vii, 88, 89, 92, 93, 98, 99
CDF	Cumulative Distribution Function 80
CNN	Convolutional Neural Network 13
CV	Coefficient of Variation 49, 79
DCMAC	Deep Centralized Multi-agent Actor Critic 16, 84
DDMAC	Deep Decentralized Multi-agent Actor Critic 16, 102
DDPG	Deep Deterministic Policy Gradient 16
DDQN	Double Deep Q-Network ii, iv, vi–viii, 14, 31, 32, 35, 44–47, 52–55, 59, 61–67, 69, 70, 83, 99
DL	Deep Learning 1, 13
DNN	Deep Neural Network vi, 12–14, 31, 44, 45, 52–54, 83
DOF	Degree of Freedom 72
DP	Dynamic Programming 10, 11
DQN	Deep Q-Network iv, 13, 14, 20, 31
DRL	Deep Reinforcement Learning ii–v, vii, viii, 2, 3, 11–14, 16, 21–24, 31, 32, 44, 51, 52, 57, 59, 61, 62, 64, 73, 81, 83, 86, 93, 99–102, 106
FE	Finite Element 73, 76, 78, 81
FEM	Finite Element Method 72
FORM	First Order Reliability Method vii, viii, 79, 81, 82, 86

HCRL Hierarchical Coordinated Reinforcement Learning 21

HMC Hamiltonian Monte Carlo viii, 17, 25–28

KDE Kernel Density Estimation 51, 78

KL Kullback–Leibler 15

LSF Limit State Function 79

LSS Limit State Surface vii, 79–81

MC Monte Carlo vii, 17, 79, 81, 82

MCMC Markov Chain Monte Carlo 17, 25, 43

MDP Markov Decision Process iv, vi, 8–11

NUTS No-U-Turn Sampler ii, vi, viii, 18, 24–26, 28, 30, 35, 43, 51, 57, 58, 78

OMA Operational Modal Analysis 2, 24, 36, 37, 72, 77

PDF Probability Density Function 19, 79

POMDP Partially Observable Markov Decision Process iv, vi, 8, 10, 11, 18, 20, 23, 43

PPO Proximal Policy Optimization ii, iv, vi–viii, 15, 16, 31, 32, 34, 35, 44–46, 48, 49, 54, 56, 59, 61–67, 69, 70, 78, 83, 84, 86–88, 91–93, 99, 103–106

RC Reinforced Concrete 37

ReLU Rectified Linear Unit 61

RL Reinforcement Learning iv, 1, 2, 11–14

RNN Recurrent Neural Network 13

RVD Random-Variable Degradation 18

SDOF Single Degree of Freedom vi, 36, 46, 60, 62–65, 73, 88

SHM Structural Health Monitoring 1, 24
SLS Serviceability Limit State 72, 99
SMC Sequential Monte Carlo 17
SMDP Semi Markov Decision Process 8
SPD Stochastic-Process Degradation 18

TBM Time-Based Maintenance vii, 88, 90, 92, 93
TD Temporal Difference 11
TMCMC Transitional Markov Chain Monte Carlo 17
TRPO Trust Region Policy Optimization 15, 70

VoI Value of Information 16

1 Introduction

1.1 Motivation

An ever-important issue regarding civil infrastructure or in general engineering systems is their deterioration over time. Deterioration is a serious concern since it often leads to the reduction of structural capacity as well as the reliability and service life of a system. Many instances showcase the importance of structural degradation, with structures located in coastal and marine environments being the most common cases, as well as structures subjected to cyclic loading, hence fatigue. Typical examples of deteriorating systems constitute bridges [2], [3], [4], offshore platforms [5], [6], [7], and wind turbines [8], [9], [10]. In particular, in cases of structural steel, it is most likely that galvanic corrosion will occur due to the atmospheric exposure in a marine environment, whereas the deterioration of reinforced concrete elements takes place in the form of corrosion of the reinforcement and/or spalling of the concrete. Lastly, fatigue can also lead to significant degradation of the structure, being responsible for the formation and the propagation of cracks.

However, these degradation processes are highly stochastic, and their prediction often requires a probabilistic analysis, having first expressed quantitatively the uncertainties, which are involved in these physical procedures. Therefore, the maintenance of a deteriorating system constitutes a complex sequential decision-making problem under uncertainty, for which it is often intractable to find closed-form solutions concerning the optimal actions that accomplish a plethora of life-cycle objectives. Additionally, the existence of multiple components, their interaction, and their ability to mitigate one another's failure, contribute to the enhancement of the uncertainty and the difficulty to define an optimal sequence of actions that will fulfill long-term goals.

To define the actual degradation of a system, a common practice is to employ new information derived from monitoring devices, to update the prior knowledge of a system's parameters. This non-destructive damage assessment, i.e. incorporating observations based on Structural Health Monitoring (SHM), can reflect the actual deterioration, leading to a decreased variability of the system's current condition and structural capacity, hence to more realistic and accurate modeling of it, allowing the decision-maker to proceed with more informed and rational decisions. The majority of these updating techniques rely on Bayesian principles and the notion of Bayesian Model Updating (BMU).

As far as the optimal maintenance policy is concerned, due to the significant amount of deterioration states, possible maintenance actions, and the decision steps under consideration, analytical solutions for determining the wanted optimal sequence of actions are more often than not computationally heavy. The so-called curse of dimensionality has been alleviated with the use of Reinforcement Learning (RL) techniques, which were able to provide approximate solutions for optimal maintenance policies for engineering systems. Even further progress was achieved when, in 2014, DeepMind patented an application of RL techniques to Deep Learning (DL), with

the scope of playing Atari games better than human experts [11]. This new approach, namely Deep Reinforcement Learning (DRL), overcame the limitations that traditional RL had, and was proved to be a promising tool for finding near-optimal control policies.

1.2 Problem Statement

The scope of this thesis is the development of an integrated framework that will combine DRL techniques with Bayesian Inference, with the former tackling the sequential decision optimization, while the latter one would deal with the accurate modeling of the stochastic deterioration process. The ultimate goal of this tool is to determine the optimal sequence of maintenance decisions that result to the minimum cost throughout the service life of an engineering system. To elaborate further on the interaction between the various elements that will be used in this framework, response quantities of the system, that are contaminated with noise, will be fed into an Operational Modal Analysis (OMA) procedure, in order to obtain modal characteristics (also including some uncertainty, through additional noise). Furthermore, the transition of the system to its new state will be described by a stochastic deterioration model, which will be constantly more accurate by incorporating observations during each decision step. A generic schematic representation of the aforementioned problem is depicted in Figure 1.

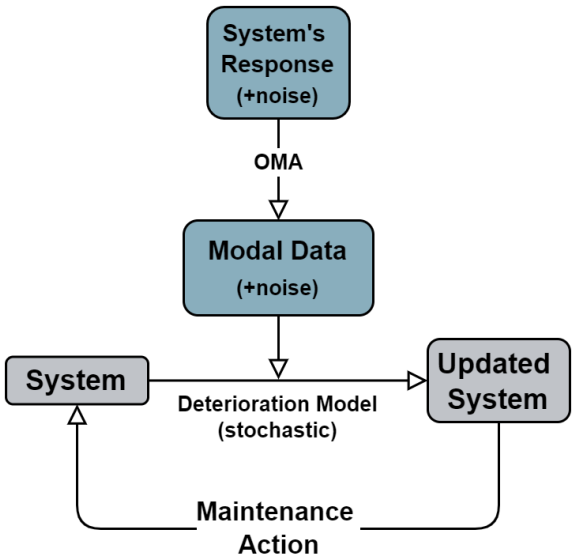


Figure 1: *Problem Statement - General Framework*

The necessary **inputs** are:

- Initial (prior) distribution for system parameters (e.g. stiffness, mass, deterioration parameters, etc)
- Stochastic deterioration model over time
- Possible actions (e.g. do nothing, repair, replace, etc)
- Cost definition (e.g. action costs, risk of failure)
- Noise interfering in observations' monitoring
- Time window of interest

The **ouput** of the proposed framework will be the optimal sequence of actions, that minimize the cost function, over the system's service life.

It should be mentioned, that such a coupling of these two core concepts, namely DRL and BMU, has not been done yet in the existing research (as will be thoroughly presented in Chapter 2), especially in the field of infrastructure maintenance, a fact that highlights the innovation of the proposed framework. Nevertheless, owing to the limited assumptions and simplifications that will be considered for the sake of accuracy, it is likely that considerable challenges and obstacles may be posed regarding the computational costs.

1.3 Research Question

The research question can be formulated as follows:

> *"How to develop an integrated framework that will efficiently couple Deep Reinforcement Learning (DRL) algorithms and Bayesian Model Updating (BMU) when it comes to structural systems' life cycle optimization, using vibration data/observations? "*

In order to efficiently reach the answer to the main research question, it is further broken down into the following sub-questions:

- How can this framework be applied in a simplified yet representative case (toy problem)?
- Having produced sound results for the simple case, which DRL algorithm performs better?
- Due to the significant computational cost of BMU, how can it be integrated in a more time-efficient way?
- Can this framework be scaled up to more complicated cases? (e.g. bigger action spaces, complex multi-component structures, etc)

1.4 Research Methodology

The research of the current problem can be structured in four main sections, as illustrated also in Figure 2. The research framework constitutes the first part, including the motivation, the definition of the problem statement and a clear formulation of the research question. The second part focuses on the literature review, according to which the research objectives and questions might be refined. Having laid the theoretical foundation, and built an informed picture of the existing research, the tool development follows, as well as the formulation of the applications on which the proposed tool will be tested. In the current project, two problems of different complexity will be addressed, with the minor one acting also as a validation and verification test for the proposed framework. The culmination of the aforementioned steps will be the evaluation of the results, accompanied by the corresponding discussions and considerations about future developments, based on the showcased strengths and weaknesses of the tool.

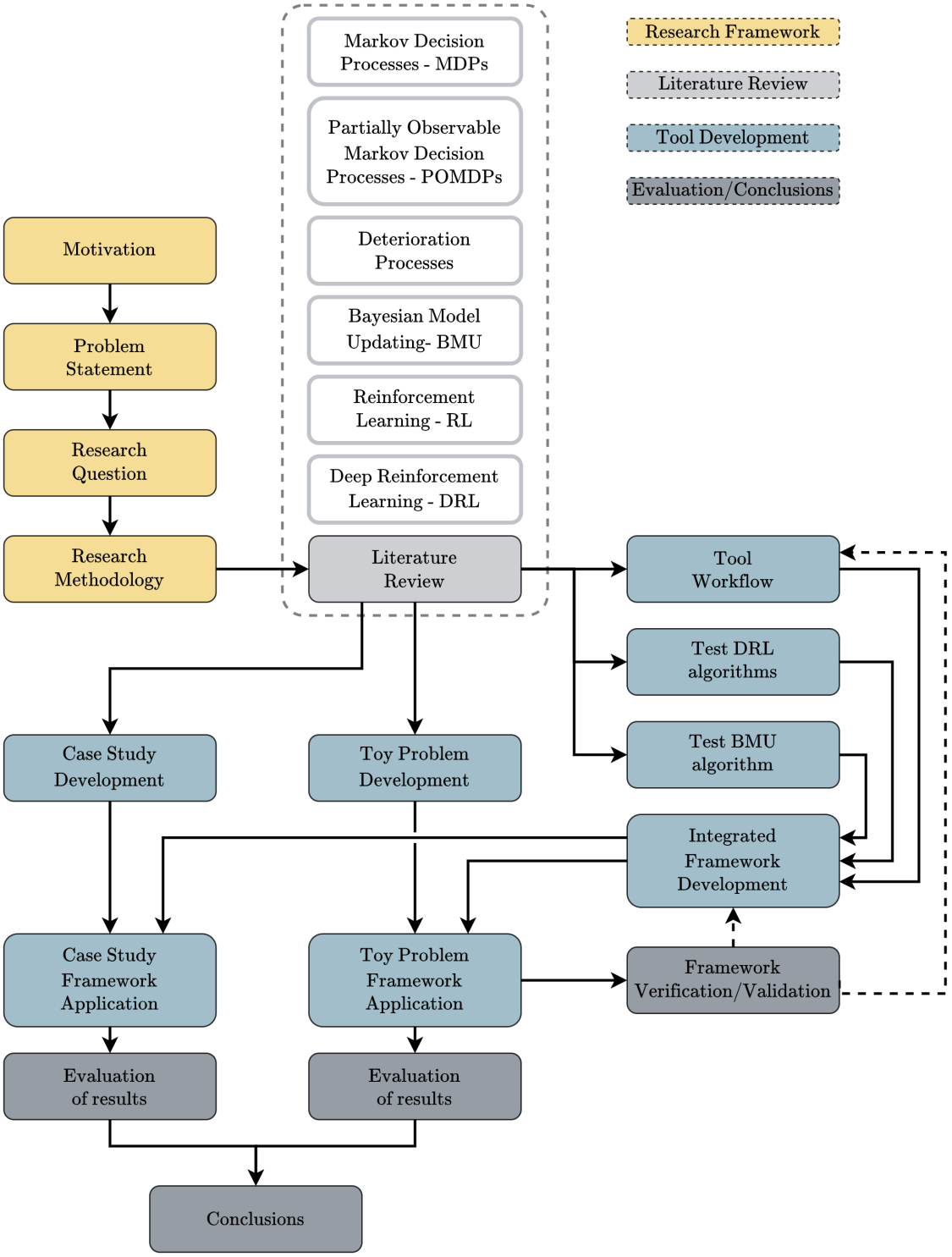


Figure 2: *Research Methodology flowchart*

The sheer amount of mathematical operations and algorithms that are required for the current project, will be handled with `Python` programming language. A summary of the core libraries/packages that will be needed, is displayed in Figure 3.

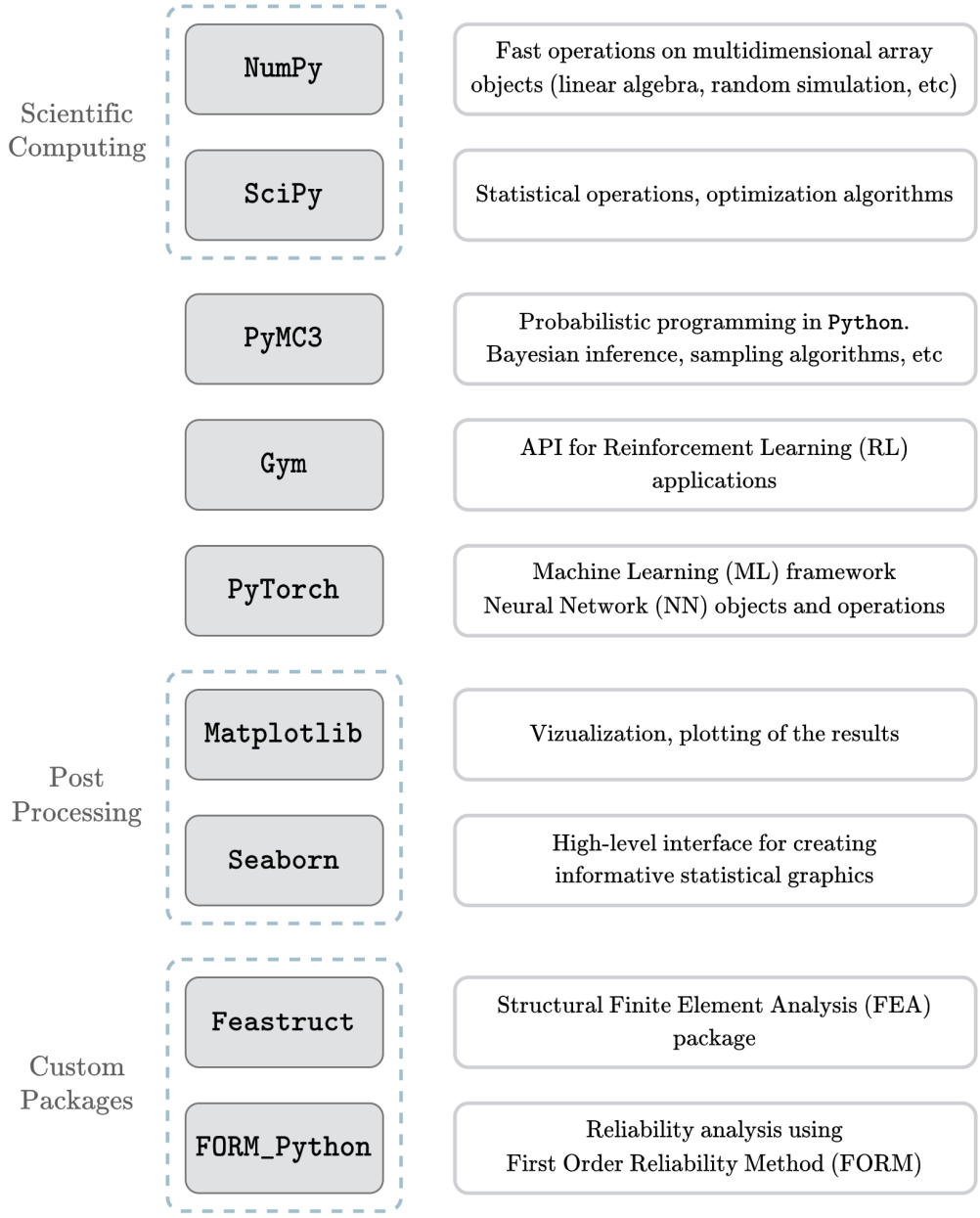


Figure 3: *Core* `Python` *dependencies*[4]

[4] Apart from the Python libraries, that are widely used for similar projects, there will be use of other, custom ones, that were available open-source, online [12], [13]

1.5 Thesis Structure

In concluding this introductory chapter, the content of the following ones will be briefly described, further elaborating along with the research methodology flowchart (Figure 2) on the structure of the whole thesis. In the coming chapter, the literature review is introduced (Chapter 2), followed by the description of the methodology of the proposed framework in a generic fashion (Chapter 3). A simple toy problem is presented in the following chapter, where the aforementioned tool is applied (Chapter 4). Then, the developed framework is further applied to a more complicated and realistic case study (Chapter 5), leading to the final chapter, where the main conclusions are drawn, along with a reflection on the advantages and disadvantages of the proposed method, as well as topics for further discussion and future work (Chapter 6).

2 Literature Review

2.1 Markov Decision Process (MDP)

The problem of optimal stochastic control is usually handled using an MDP, which provides a principled mathematical methodology that can address the uncertainties of planning optimum inspection and maintenance strategies, including uncertain action outcomes and exact observations. The theory behind MDPs, together with further concepts that will not be covered in the current work, like Semi Markov Decision Processes (SMDPs), State Augmentation, etc, has been analyzed in depth in many papers, e.g. [14]. However, a basic elaboration on MDPs and Partially Observable Markov Decision Processes (POMDPs) will be made, in order to build a foundation for the more advanced concepts in the following sections.

The basic components of an MDP are the Environment and the Agent, while it can fully described by the tuple:

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$$

where,

$\mathcal{S}$: the finite set of states
$\mathcal{A}$: the finite set of actions
$\mathcal{P}$: the state probability matrix
$\mathcal{R}$: the reward function
γ : a discount factor for the future rewards

At each decision step t, the decision-maker, i.e. the agent, observes the current state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$, receives a reward $R_t(s_t, a_t) \in \mathcal{R}$ and moves to the next state $s_{t+1} \in \mathcal{S}$ based on the environment's transition dynamics. This procedure is displayed in Figure 4. According to the so-called Markov property, the next state, s_{t+1}, depends only on the current state, s_t, and the chosen action, a_t, regardless from the preceding history of states and actions. The sequence of actions followed by the agent, defines its policy, π, that can be either deterministic or stochastic, meaning that a policy can map states to actions or states to probability mass (or density) functions, respectively.

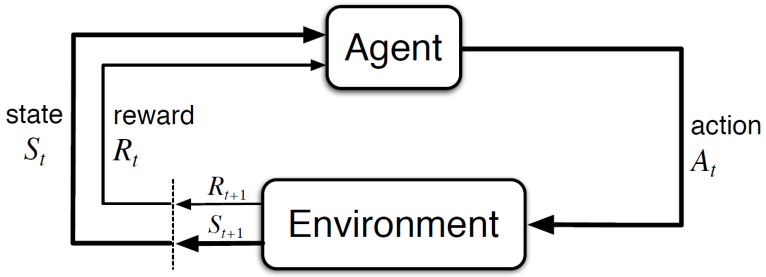


Figure 4: *The agent-environment interaction in an MDP [15]*

The ultimate goal of an optimal policy is to maximize the sum of the discounted rewards, i.e. the total return, G_t, of the chosen actions, which is given by the following formula:

$$G_t = R(s_t, a_t) + \gamma R(s_{t+1}, a_{t+1}) + \gamma^2 R(s_{t+2}, a_{t+2}) + \ldots = \sum_{i=t}^{T} \gamma^{i-t} R(s_i, a_i) \tag{1}$$

It should be noted that the total return is not deterministic, owing to the problem being stochastic. Thus, accounting for all future state-action pairs, the action-value function, $Q^{\pi}(s_t, a_t)$ is defined as the expected return[5]over both $\mathcal{S}$ and $\mathcal{A}$ sets (Equation 2).

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_t, a_t}[G_t \mid s_t, a_t] \tag{2}$$

It should be noted that the notation $\mathbb{E}$ corresponds to the expected value, i.e. the mean, while the variables over which it is computed, are denoted as the subscripts.

Decomposing Q^{π} into the immediate reward plus the discounted value of the successor state, leads to the following recursive form:

$$Q^{\pi}(s_t, a_t) = R(s_t, a_t) + \mathbb{E}_{s_{t+1}, a_{t+1}}\left[\gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t, a_t\right] \tag{3}$$

The state-value function $V^{\pi}(s_t)$ is defined in a similar way, corresponding to the expected return starting from state s_t and following policy π.

$$V^{\pi}(s_t) = \mathbb{E}_{a_t}[G_t \mid s_t] \tag{4}$$

Since the state-value function is stochastically defined for all possible actions of policy π, it takes the form:

$$V^{\pi}(s_t) = \mathbb{E}_{a_t}\left[Q^{\pi}(s_t, a_t)\right] \tag{5}$$

For any MDP there exists an optimal policy π_* that achieves the optimal state and action-state value function. These are formulated through the Bellman Optimality Equation as follows:

$$V^*(s_t) = \max_{a_t \in \mathcal{A}} \left[R(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathsf{P}(s_{t+1} \mid s_t, a_t)\, V(s_{t+1}) \right] \tag{6}$$

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} \mathsf{P}(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{7}$$

The symbol $\mathsf{P}(\cdot)$ corresponds to the probability of a quantity; a notation that will be used for the rest of the thesis.

When the optimal action-state value function is known, the optimal policy is known, thus the MDP is solved. Since the Bellman Optimality Equation is non-linear, it is often solved using Dynamic Programming (DP) (value iteration, policy iteration), or algorithms like Q-learning and SARSA (both will be presented in Section 2.3).

A considerable disadvantage of using MDPs for a sequential decision problem is the curse of dimensionality, as with an increase in the state-space, the computational cost grows exponentially. Therefore, in the existing literature, there are mostly attempts that choose discrete deterioration states instead of continuous, e.g. [16], [17].

2.2 Partially Observable Markov Decision Process (POMDP)

A significant aspect in the exploitation of the ever-increasing available observation data is the degree of confidence the decision-maker has in the received input. This issue is undoubtedly connected with uncertainties regarding various environmental and loading conditions, modelling errors, inefficient measuring systems or inaccuracies in the information transmission network [18]. This is why, more often than not, models based on MDPs are limited by the need of perfect observations, leading to their extension to Partially Observable Markov Decision Processes (POMDPs), which take into consideration the partial observability of the systems' information in order to approach the optimal sequence of maintenance decisions based on uncertain structural data. In these cases, when the states are not fully observable, at each decision step the agent gets a belief, $\boldsymbol{b}$, over the states of the system. Since the Markov property still holds, this probability distributions over $\mathcal{S}$ is sufficient to describe the history of actions and observations. Starting with an initial belief, $\boldsymbol{b_t}$, the agent takes an action, a_t, but instead of receiving the new state, gets an observation $o_{t+1} \in \Omega$, with which the belief is updated, using Bayesian principles.

$$\begin{aligned} b(s_{t+1}) &= \mathsf{P}(s_{t+1} \mid o_{t+1}, a_t, \boldsymbol{b_t}) \\ &= \frac{\mathsf{P}(o_{t+1} \mid s_{t+1}, a_t)}{\mathsf{P}(o_{t+1} \mid \boldsymbol{b_t}, a_t)} \sum_{s_t \in \mathcal{S}} \mathsf{P}(s_{t+1} \mid s_t, a_t)\, b(s_t) \end{aligned} \tag{8}$$

In a similar notion with MDPs and assuming that the beliefs are the states of this environment, Equation 6 can be rewritten for POMDPs.

$$V^*(\boldsymbol{b_t}) = \max_{a_t \in \mathcal{A}} \left[\sum_{s_t \in \mathcal{S}} b(s_t)\, R(s_t, a_t) + \gamma \sum_{o \in \Omega} \mathsf{P}(o \mid \boldsymbol{b}, a)\, V(\boldsymbol{b_{t+1}}) \right] \tag{9}$$

POMDPs have been used in many cases for infrastructure maintenance in the existing literature ([19], [20]) often along with point-based algorithms ([21], [22], [23], [24], [25]) demonstrating that they can model more efficiently complex decision problems outperforming heuristic-based policies. However, they face limitations when it comes to the solution of large action- and state-spaces, and even greater ones in the case of continuous state spaces. The most significant difficulty lays on the updating of the belief, with Equation 8 (which corresponded to discrete spaces) taking the following form:

$$b'(s_{t+1}) = \mathsf{P}(s_{t+1} \mid o_{t+1}) \propto \mathsf{P}(o_{t+1} \mid s_{t+1}, a_t) \int_{s_t} \mathsf{P}(s_{t+1} \mid s_t, a_t)\, b(s_t) \mathrm{d}s_t \tag{10}$$

Because the integral of Equation 10 can not be calculated in a closed form, many researchers have attempted to use other mathematical ways that work around this issue ([26], [27], [28]). This updating of the belief can be broken down into two steps, the *transition step*, during which the belief propagates in time according to a predefined conditional probability distribution [24]:

$$b(s_{t+1}) = \int_{s_t} \mathsf{P}(s_{t+1} \mid s_t, a_t)\, b(s_t) \mathrm{d}s_t \tag{11}$$

and the *estimation step*, with the belief now updating based on obtained evidence by means of Bayes' rule:

$$b'(s_{t+1}) = \mathsf{P}(s_{t+1} \mid o_{t+1}) \propto \mathsf{P}(o_{t+1} \mid s_{t+1}, a_t)\, b(s_{t+1}) \tag{12}$$

These drawbacks and difficulties of POMDPs led to the need of developing an RL or DRL methodology, capable of tackling them.

2.3 Reinforcement Learning (RL)

RL is often an advantageous technique to deal with sequential decision optimization, especially when knowledge about the system is uncertain or unknown. Indeed in the existing literature there are even examples that tackle optimal maintenance planning in the absence of a deterioration model [29]. The agent interacts directly with the environment by taking actions and adjusts the policy based on the information received back, aiming to identify the optimal one.

RL algorithms are divided into two main categories, model-based and model-free. In the former group, e.g. DP (policy iteration, value iteration), the model must provide state transition probabilities and expected rewards for every state-action pair in order to identify the optimal policy, whereas algorithms in the latter category, such as Temporal Difference (TD) learning, SARSA, Q-learning,

rely on real samples from the environment, which is a feature that makes them applicable in various different cases. Both model-based and model-free approaches were tested along with Q-learning in [30], for a problem of maintenance optimization, with the system's discrete state transitions being defined through the assumed degradation model.

A second major distinction between RL algorithms, is the one among on-policy and off-policy. In on-policy learning the $Q(s, a)$ function is learned through actions of the current policy, π, while in off-policy learning, it is learned while taking different/random actions. Namely, SARSA is an on-policy algorithm, that updates the state-action value function as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \tag{13}$$

where a_{t+1} is the action taken according to policy π.

On the other hand, Q-learning is an off-policy learning algorithm, that updates the state-action value function in the following way:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(R(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \tag{14}$$

where a_{t+1} can be any of the possible actions.

In Q-learning, the value functions Q for any state-action pair are stored in a table, and they are updated by interacting with the environment, i.e. taking actions and receiving rewards. In order to explore every possible action, hence, every unknown area of the Q-table, usually an ϵ-greedy policy is used, in order to balance out exploration and exploitation. More specifically, the agent will exploit the already known Q-values, or explore, picking an action at random, based on the following rule:

$$a_t = \begin{cases} \max_{a_t \in \mathcal{A}} Q(s_t, a_t) & \text{with probability } 1 - \epsilon \\ \text{random } a_t \in \mathcal{A} & \text{with probability } \epsilon \end{cases} \tag{15}$$

In [31], a customized version of Q-learning is introduced, namely "*safe Q-learning*", that includes a model-based safe exploration for near-optimal management of infrastructure, in order to decrease the variance induced by choosing random actions (exploring).

However, there are two main limitations in the classic Q-learning approach, which are (1) the curse of dimensionality when the state- and action-space are large, and (2) the inability of this algorithm to visit all states, hence, to estimate the Q-values for the whole table.

2.4 Deep Reinforcement Learning (DRL)

To tackle the aforementioned limitations, the Q-function is approximated through a Deep Neural Network (DNN). This way, the state-action value function is reparameterized, and is now expressed

in terms of parameters θ, in order to alleviate the computational cost and instabilities that large state- and action-spaces cause.

2.4.1 Deep Q-Network (DQN)

The state, s_t is given as an input to the DNN, and by using a suitable number of inner layers and activation functions, it outputs the action-state value function, $Q(s_t, a_t \mid \theta)$, for every $a_t \in \mathcal{A}$. The parameters, θ, i.e. the Q-network weights, are adjusted in every iteration of the agent's training in order to minimize a sequence of loss functions that are given by the following equation:

$$L(\theta) = \mathbb{E}_{s_t, a_t}\left[\left(y_t - Q(s_t, a_t \mid \theta)\right)^2\right] \tag{16}$$

with y_t being the target for decision step t,

$$y_t = \mathbb{E}_{s_{t+1}}\left[R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} \mid \theta^-) \mid s_t, a_t\right] \tag{17}$$

In order to stabilize the learning process, tuples of $(s_t, a_t, R(s_t, a_t), s_{t+1})$ are being stored in a replay buffer and are then used in batch training the Deep Q-Network (DQN) according to the following gradient:

$$\nabla_\theta L(\theta) = \mathbb{E}_{s_t, a_t}\left[\left(R_t(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q(s_{t+1}, a_{t+1} \mid \theta^-) - Q(s_t, a_t \mid \theta)\right) \nabla_\theta Q(s_t, a_t \mid \theta)\right] \tag{18}$$

Each tuple of the experience replay is potentially used in many weight updates, which leads to the use of non-consecutive uncorrelated samples, hence, to the reduction of the variance through the updates. Additionally, as it is shown in Equations 17, 18, a target network is used, with parameters θ^-. This target network takes the values of the original one with a delay, which contributes to the stability of the training.

Various examples exist in the literature, using DQN including different DNNs to approximate the value functions. In the initial coupling of RL and DL by DeepMind [11], Convolutional Neural Networks (CNNs) were used in order to decompose the Atari's screen into a rectangular grid that was fed into the network. Similarly, a DRL framework for optimal maintenance, again with a CNN is developed in [32]. Moreover, in both [33] and [34], Recurrent Neural Networks (RNNs) are used for the policy optimization while in a partially observable environment. Finally, an Artificial Neural Network (ANN) is used in [35] in order to tackle the two-dimensional state-space limitation, existing in cases when CNNs or RNNs were chosen instead. This approach was followed in [36], too, as well as in [37] where an ANN was used both for the multi-component system's maintenance but also for the creation of a surrogate model.

2.4.2 Double Deep Q-Network (DDQN)

It is known that standard DQN suffers from an overestimation problem for the action values in noisy environments, thus, an even more stable algorithm is DDQN that uses both the target and the original network in the calculation of y_t, in particular:

$$y_t = R(s_t, a_t) + \gamma\, Q\left(s_{t+1}, \arg\max Q\left(s_{t+1}, a_{t+1} \mid \theta\right) \mid \theta^{-}\right) \tag{19}$$

In [38] a DDQN is used for the maintenance planning of a stochastically deteriorating system, accounting also for the dependency between its multiple components. DDQN is thoroughly explained in [39].

2.4.3 Advantage Actor Critic (A2C)

Another common approach in both RL and DRL is to instead of interacting with value functions, change directly the policy, π. In the case of DRL, when a DNN is used to approximate π, the training is done through the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t, a_t}\left[\sum_{t \geq 0} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\, Q(s_t, a_t)\right] \tag{20}$$

where $J(\theta)$ is the objective function, i.e.

$$J(\theta) = \sum_{t \geq 0} \gamma^t\, R(s_t, a_t) \tag{21}$$

One way to reduce the variance and improve policy gradient methods is by subtracting a baseline from the Q-function. This is why the advantage value is being introduced:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{22}$$

corresponding to how much better a specific action is, compared to the average, general action at the given state.

In order to compute all terms in Equation 20, the policy gradient requires also an estimation of a value function. This issue led to the creation of the so-called Actor-Critic methods, which use a value approximator (critic) to train the parameters of the policy approximator (actor). Two DNNs are employed, one for each of the two approximators (actor-network, critic-network), with the latter being used for the V-function estimation, since Equation 22 can be rewritten through the Bellman equation as follows:

$$A(s_t, a_t) = R(s_t, a_t) + \gamma\, V(s_{t+1}) - V(s_t) \tag{23}$$

This leads to the Advantage Actor Critic (A2C) algorithm, which aims to the minimization of:

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{s_t,a_t}\left[\sum_{t\geq 0}\nabla_\theta \log\pi(a_t \mid s_t,\theta)\left(R(s_t,a_t)+\gamma\, V(s_{t+1})-V(s_t)\right)\right] \\ &= \mathbb{E}_{s_t,a_t}\left[\sum_{t\geq 0}\nabla_\theta \log\pi(a_t \mid s_t,\theta)\, A(s_t,a_t)\right] \end{aligned} \tag{24}$$

2.4.4 Proximal Policy Optimization (PPO)

In order to avoid destructively large policy updates in policy gradient algorithms, the trust region methods were developed. In particular, the Trust Region Policy Optimization (TRPO) algorithm, aims to maximize the "surrogate" objective function, under the constraint that the Kullback–Leibler (KL) divergence between the old and new policy is less than a constant δ.

$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}A_t\right] \tag{25}$$

$$\text{with}\quad \mathbb{E}_t\left[\mathrm{KL}\left[\pi_{\theta_{\text{old}}}(\cdot \mid s_t),\pi_\theta(\cdot \mid s_t)\right]\right] \leq \delta \tag{26}$$

or, using a penalty term instead of a constraint:

$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}A_t-\beta\,\mathrm{KL}\left[\pi_{\theta_{\text{old}}}(\cdot \mid s_t),\pi_\theta(\cdot \mid s_t)\right]\right] \tag{27}$$

An improvement to TRPO is the PPO algorithm. Although, an in depth description of this method is presented in [40], a brief reference to its main steps/equations will be also included here.

Denoting the probability ratio,

$$r_t(\theta)=\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{28}$$

the TRPO "surrogate" objective becomes:

$$L^{CPI}(\theta)=\mathbb{E}_t\left[\frac{\pi_\theta\,(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}\,(a_t \mid s_t)}A_t\right]=\mathbb{E}_t\left[r_t(\theta)A_t\right] \tag{29}$$

The superscript CPI refers to conservative policy iteration, while the maximization of L^{CPI} would cause large policy updates. Therefore, the objective needs to be adjusted so as to penalize policy changes that move $r_t(\theta)$ away from 1. Thus, the clipped surrogate objective is considered instead:

$$L^{CLIP}(\theta)=\mathbb{E}_t\left[\min\left(r_t(\theta)A_t,\mathrm{clip}\left(r_t(\theta),1-\epsilon,1+\epsilon\right)A_t\right)\right] \tag{30}$$

The term, $\text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) A_t$, modifies the surrogate objective by clipping the probability ratio, not allowing it to move outside the interval $[1-\epsilon, 1+\epsilon]$.

PPO is used to optimize the maintenance of renewable energy systems in [41].

2.4.5 Other Deep Reinforcement Learning (DRL) algorithms

Regarding different DRL algorithms, Deep Deterministic Policy Gradient (DDPG) is used in [42], which is an actor-critic algorithm that enables the modeling of a continuous deterioration state-space. Additionally, the development of a new algorithm, namely Deep Centralized Multi-agent Actor Critic (DCMAC), is presented in [43] that aims for the optimal maintenance in multi-component systems with high dimensional action- and state-spaces. Lastly, in order to address challenges regarding stochastic optimal control, such as the curse of dimensionality in large spaces, the curse of history, and the environment uncertainties/stochasticity, Deep Decentralized Multi-agent Actor Critic (DDMAC) is introduced in [44].

2.5 Bayesian Model Updating (BMU)

As it has already been mentioned, in the current project, the problem of optimal stochastic control will be combined with the notion of BMU. To be more precise, model updating constitutes an inverse problem, meaning that, instead of knowing beforehand the exact parameters of a model to calculate its response, observations of the system's behavior are used in order to update or calibrate the unknown system properties. This technique can efficiently tackle the fact that many system parameters are not deterministic, but stochastic. Additionally, in most cases the deterioration of an existing structure can not be modeled accurately, with a Gamma process e.g. like in [38], [43], [45] (this type of deterioration will be covered in depth in Section 2.6) or generally a relationship of the form $D(t) = A\, t^B$ for the damage, $D(t)$, over time, t, e.g. like in [46]. Therefore, an updating procedure can be useful in defining its properties during its whole service life. An extensive review on model updating about damage assessment, including BMU is presented in [47].

It should be mentioned that, a detailed and instructive example on how to account for a structure's deterioration through BMU is presented in [46]. In that research, aiming to quantify the Value of Information (VoI), periodic inspections are made, and the obtained observation are used in the updating of the system's parameters as well as its structural reliability. A heuristic based approach was followed regarding the life-cycle optimization.

From a more technical standpoint, Bayesian Inference is performed using Bayes' Theorem, which, assuming that $\boldsymbol{\theta}$ are the parameters of interest and $\boldsymbol{D}$ are the observations, takes the form:

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{D}) = \frac{\mathrm{P}(\boldsymbol{D} \mid \boldsymbol{\theta})\,\mathrm{P}(\boldsymbol{\theta})}{\mathrm{P}(\boldsymbol{D})} \tag{31}$$

where,

$\boldsymbol{\theta}$	:	the vector of the parameters of interest
$\boldsymbol{D}$	:	the vector of observations
$\mathrm{P}(\boldsymbol{\theta})$	:	the prior distribution
$\mathrm{P}(\boldsymbol{D} \mid \boldsymbol{\theta})$	:	the likelihood function of the parameters $\boldsymbol{\theta}$
$\mathrm{P}(\boldsymbol{D})$	:	the evidence
$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{D})$	:	the posterior distribution of $\boldsymbol{\theta}$

Analyzing each factor involved in Equation 31:

- **Prior distribution**
  The prior distribution $\mathrm{P}(\boldsymbol{\theta})$ corresponds to the initial hypothesis about the system's parameters. It is an uninformed estimation of them, e.g. if only the upper and lower bounds are known, a Uniform distribution among these bounds will be used.
- **Likelihood function**
  The likelihood function describes the degree of agreement between the observations $\boldsymbol{D}$ and the output/result of the actual model, computed deterministically using the existing knowledge for parameters $\boldsymbol{\theta}$
- **Evidence function**
  The evidence function acts as a normalizing constant in Bayes Theorem. This way the integral of the posterior distribution sums up to 1. Because the evidence is independent from parameters $\boldsymbol{\theta}$, it does not affect the shape of the posterior distribution, hence, it can be written:
  $$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{D}) \propto \mathrm{P}(\boldsymbol{D} \mid \boldsymbol{\theta})\,\mathrm{P}(\boldsymbol{\theta}) \tag{32}$$
- **Posterior distribution**
  The posterior distribution is the updated distribution of parameters $\boldsymbol{\theta}$ with the use of the observed response data.

A challenge lays on the sampling of the posterior distribution, because it can not be expressed in a closed form, but only implicitly, point-wise, using a MC approach. To overcome this obstacle, many sampling methods have been developed, in order to approximate $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{D})$. An in-depth overview of three popular sampling methods, namely Markov Chain Monte Carlo (MCMC), Transitional Markov Chain Monte Carlo (TMCMC) and Sequential Monte Carlo (SMC) is presented in [48]. Two of the most well-known MCMC algorithms are Metropolis-Hastings and Gibbs sampling. However, they often fail to converge to the posterior distribution, especially for continuous model parameters. Therefore, more efficient algorithms have been developed and are widely used, such as Hamilto-

nian Monte Carlo (HMC), and an even more advanced variation of it, No-U-Turn Sampler (NUTS). NUTS tends to be significantly efficient, thanks to its stability and the fact that it does not need hand-tuning of hyper-parameters by the user. A brief elaboration on NUTS is presented in Section 3.2, whereas an in-depth presentation of it, and its many variations, is included in [49].

Concluding, the stochastic nature of this problem, and the inability to know the exact value for the parameters of interest at every decision step, comes to highlight the need of POMDP, where the belief, $\boldsymbol{b_t}$ at every t, will correspond to the posterior distribution, after having incorporated the observations D.

2.6 Deterioration Processes

As elaborated thoroughly in [50] and [51], the uncertainty associated with the evolution of degradation over time is an important consideration for the optimisation of maintenance. The commonly used Random-Variable Degradation (RVD), where the rate of degradation[6]is random, can not capture the temporal variability of the degradation, which is why a Stochastic-Process Degradation (SPD) model is usually adopted. An efficient modelling option would be the Brownian motion with a drift. A stochastic process that has been applied successfully in a plethora of fields (e.g. exchange value of shares, movement of small particles in fluids, etc), however in the current application fails to perform due to the fact that it can alternately increase and decrease, which is not the case for a structure's performance, which monotonically decreases. Therefore, more often than not, the stochastic process chosen for the modelling of an engineering system's deterioration is the Gamma process. It is a stochastic process with independent non-negative increments that have a gamma distribution with identical scale parameter and time-dependent shape parameter. This choice has been proven suitable to model gradual damage monotonically accumulating over time applicable to a variety of problems e.g. wear, fatigue, corrosion, crack growth, erosion, consumption, creep, swell, etc [50].

[6]For a degradation in the form of $A\,t^B$, linear parameter A is known as the rate of degradation, while B is the non-linear trend of the degradation law. Usually, A reflects the variability in a large population of similar components.

At every step t, the damage d follows a Gamma distribution with shape $v(t) > 0$ and scale $u > 0$. Mathematically, its Probability Density Function (PDF) is given by:

$$\mathrm{Ga}(d \mid v(t), u) = \frac{u^{v}(t)}{\Gamma(v(t))} d^{v(t)-1} \mathrm{e}^{(-u\,d)} \mathrm{I}_{(0,\infty)}(d) \tag{33}$$

where,

$$\mathrm{I}_A(d) = 1 \quad \text{for} \quad d \in A$$
$$\mathrm{I}_A(d) = 0 \quad \text{for} \quad d \notin A$$

assuring positivity of d, and,

$$\Gamma(\alpha) = \int_{t=0}^{\infty} t^{\alpha-1} \mathrm{e}^{-t} \,\mathrm{d}t$$

The following properties hold for a Gamma process:

$$d(0) = 0 \quad \text{with probability} \quad 1$$
$$d(\tau) - d(t) \sim \mathrm{Ga}(v(\tau) - v(t), u) \quad \text{for all} \quad \tau > t \geq 0$$
$$d(t) \quad \text{has independent increments}$$

Its first two statistical moments, i.e. the mean and the variance, are:

$$\mathbb{E}\big(d(t)\big) = \frac{v(t)}{u}$$
$$\mathrm{Var}\big(d(t)\big) = \frac{v(t)}{u^2}$$

Empirical studies show that the expected deterioration at time t is proportional to a power law:

$$v(t) = c\, t^{b} \quad \text{with} \quad c > 0\,,\, b > 0 \tag{34}$$

which, as already mentioned, is a simplified deterioration model employed in many projects in the existing literature [46].

A typical arbitrary (in terms of shape and scale parameters) example of such a deterioration process is plotted in Figure 5.

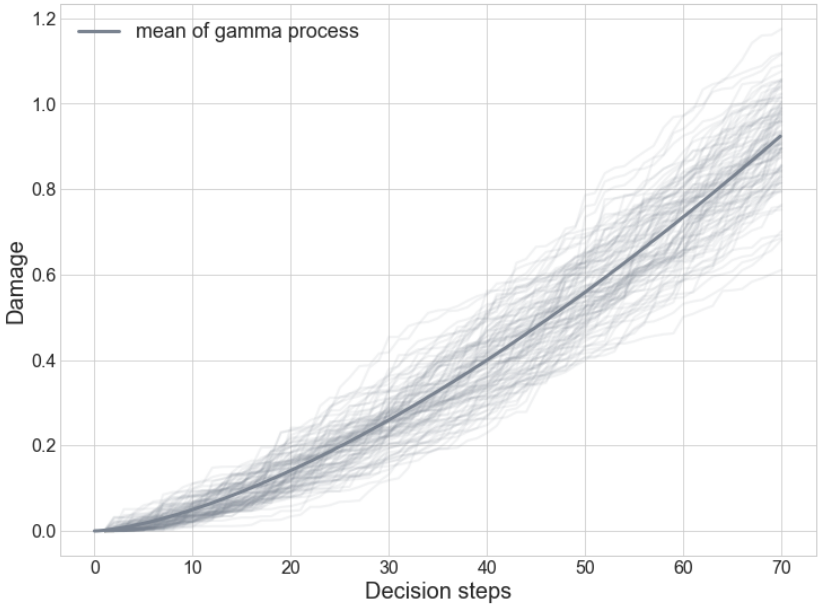


Figure 5: *Non stationary Gamma process describing the damage evolution over 70 years*

2.7 Research gap

Throughout the literature review, a plethora of obstacles and features that needed improvement was noted, enriching the existing research gap and shaping the final research question and problem formulation.

As a general observation, the majority of the examined papers, considers only discrete deterioration states, owing to the computational complexity that is induced in continuous or large state-spaces. This necessity for an efficient maintenance framework for large/continuous state spaces is highlighted in [17], [21].

What is more, a common assumption in many papers is full observability of the environment, and consequently not accounting for measurement errors. This issue is being mentioned as a future improvement in [17], [31] and [36]. Additionally, while the framework proposed in [33] considers a POMDP, its efficiency is low when observability is low, emphasizing the need of an efficient integration of Bayesian Inference.

One of the few cases when a large and continuous state space was considered, along with DQN, is in [36]. However, the authors stressed out the need of including partial observability, in the sense of noisy observations, as well as model updating.

Lastly, in many papers in which POMDPs were considered for stochastic optimal control, it was pointed out that a more efficient algorithm needs to be used. To be more precise, in [23] it is

mentioned that instead of point-based algorithms, a more advanced learning technique should be used, making DRL the suitable choice. In [24], where Bayesian Networks and point-based solvers are used, it is also stated that a DRL approach to solve Bayesian Updating is necessary. Lastly, in [45], Hierarchical Coordinated Reinforcement Learning (HCRL) is proposed for the maintenance of large-scale multi-component systems, however, it also refers to the development of a DRL algorithm for maintenance as a future improvement.

2.8 Conclusions

Concluding this chapter, along with the introduction of various scientific fields, a thorough investigation of the existing literature was presented. Through the findings and comments of other researchers, the advantages and disadvantages of different algorithms were better understood, while at the same time assisted to the choice of the necessary tools for the current project, which will be presented in detail in the coming chapter (Chapter 3) as parts of the general methodology. A significant finding of the literature review constitutes the definition of the existing gap, which was also elaborated extensively, as it further compliments the objective of this thesis and its contribution.

3 Methodology

3.1 General Framework

Conducting the literature review, it can be observed that, although Artificial Intelligence (AI), and in particular DRL, has a huge potential regarding the life-cycle optimization and maintenance of engineering systems, there are still considerable limitations. These limitations gave rise to the scope of the current project, and subsequently justify its future contribution.

To be more precise, the goal of the considered framework is to couple Bayesian Inference with DRL algorithms, aiming to find the optimal sequence of maintenance actions for a stochastically deteriorating engineering system. The thought process, thus the motivation, behind the choice of these two core concepts is illustrated in Figure 6.

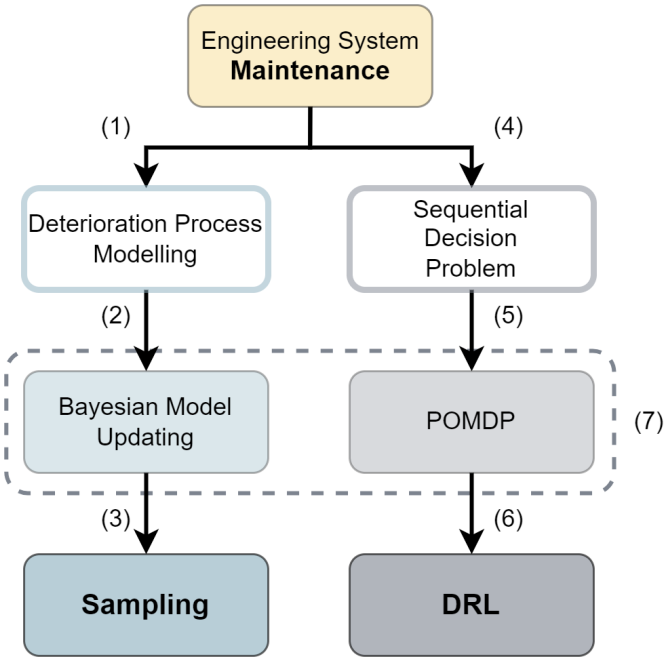


Figure 6: *Problem conceptual breakdown - Motivation for the selected tools*

In order to explain better this workflow, each logical step is numbered and further elaborated:

(1) A key concept interfering with any engineering system's maintenance, is its deterioration. As already stated, the way in which a system ages is not straightforward, since many uncertainties are involved in the physical degradation processes. Therefore, an important sub-problem constitutes the deterioration process modelling.

(2) An efficient way to tackle the underlying stochasticity in the deterioration processes is to incorporate the ever-increasing available observed/measured data, in order to update the knowledge about the system's parameters; a key element in the BMU concept.

(3) As explained in Section 2.5, applying Bayes Theorem (Equation 31), can be cumbersome, due to the inability to calculate the normalizing constant in the denominator, i.e. the so-called evidence $P(\boldsymbol{D})$. This is the case especially in the inference of continuous variables, which justifies the choice of a sampling algorithm to tackle this obstacle.

(4) Returning to the general problem, the maintenance of an engineering system constitutes a sequential decision problem, since the sought strategy is defined by the optimal sequence of maintenance actions.

(5) More often than not, POMDPs are utilized since they provide a principled mathematical methodology for stochastic optimal control under uncertain action outcomes and observations [24].

(6) As elaborated in Chapter 2, the most efficient way to solve a POMDP, is DRL, since it can handle multi-dimensionality, and even continuous state and action spaces, leading eventually to the wanted optimal strategy.

(7) It should be mentioned that both BMU and POMDPs rely on the same principles, employing Bayesian rules to update the system's parameters using observations in the former case, and update the state probability distribution, i.e. the so-called belief, in the latter one.

The interaction between the different elements of the POMDP for the current framework, i.e. beliefs, actions, rewards and observations, during each decision step, is depicted in Figure 7.

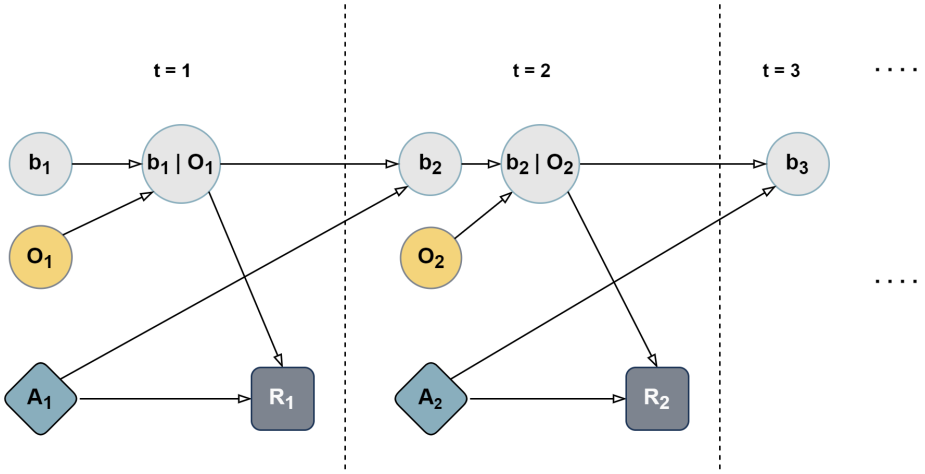


Figure 7: *Graphical Model of the employed POMDP*

Elaborating in each of the components displayed in Figure 7:

- $\boldsymbol{b_t}$ is the unknown deterioration state. To be in accordance with the theory presented in Section 2.2, it represents the belief, meaning a probability distribution about the deterioration of the system. On the contrary to the biggest part of the existing literature, where the

deterioration space is discretized, in this project a continuous state-space is considered, hence, the belief is a continuous probability distribution.

- $\boldsymbol{O_t}$ is the observation that is obtained through a SHM system periodically, i.e. in every decision step. This observation (possibly the acceleration at the location of the sensors) is fed into an OMA scheme, in order to derive modal data, e.g. eigenfrequencies, eigenmodes, etc. It should be mentioned, that the OMA step of this procedure will not be considered for the current project, since more emphasis is going to be given on the DRL and BMU parts. Therefore, the needed modal data will be generated directly, taking into account that they are contaminated with noise.
- $\boldsymbol{A_t}$ is the maintenance action that is taken at the decision step, t. The action space is assumed to be discrete.
- $\boldsymbol{R_t}$ is the reward received for taking the action A_t when in state S_t. As displayed in Sections 2.1 and 2.2, the expected return of the sum of these rewards, including also a discounting factor γ for future rewards, is the quantity that needs to be maximized (Equation 1). In the current project, the rewards correspond to the costs, thus, the goal is the minimization of the rewards. At any given state, the reward/cost can be decomposed into two sub-costs, i.e. the cost of the taken action and the cost associated with the risk of failure.

$$R_t = C_t = C_{A_t} + C_{\text{risk}} \tag{35}$$

  The risk of failure cost is calculated as the product of the probability of failure times the failure cost, i.e. $C_{\text{risk}} = P_f \cdot C_F$. An important matter constitutes the calculation of this probability for every deterioration state.
- $\boldsymbol{b_t \mid O_t}$ is the updated deterioration state, having included the information of the observation O_t. This means that at every decision step, a Bayesian Inference is performed in order to improve the knowledge, i.e. the probability distribution, about the deterioration parameters of interest. The updating is executed using the NUTS method, which will be briefly introduced in the coming section (Section 3.2).

The proposed general framework is illustrated in more detail in Figure 8.

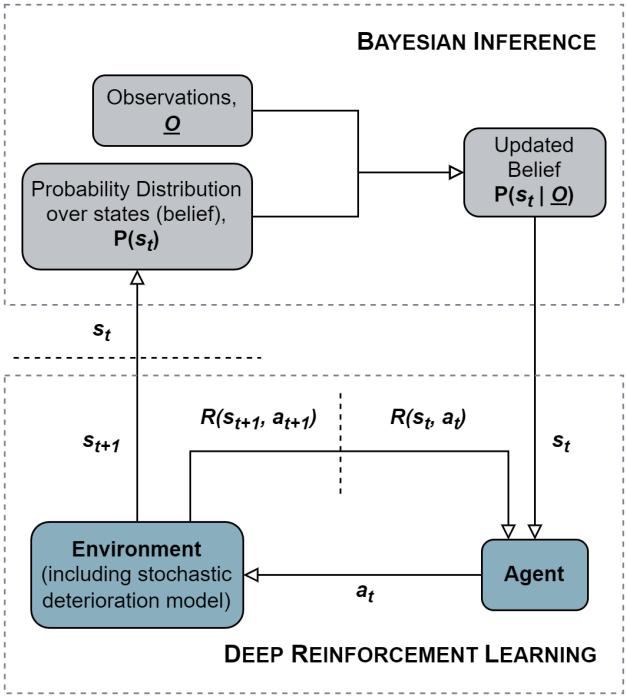


Figure 8: *Proposed Framework*

3.2 Sampling Algorithm

For the sake of completeness and transparency regarding the proposed framework, a brief walk-through the sampling procedure used, is presented in this section. To be more precise, the chosen sampling method is NUTS, which will be applied through the probabilistic programming `Python` package of `PyMC3` [52].

As already mentioned, in bayesian inference problems the posterior distribution is usually intractable. The commonly used MCMC methods, approach the target distribution by drawing a series of correlated samples. However, in complicated problems with many parameters, widely applicable methods such as random-walk Metropolis and Gibbs sampling, could be computationally expensive to achieve convergence, because of the random walks with which they explore the parameter space. This is why, in applications with continuous model parameters HMC has been proven more efficient, shifting from a problem of sampling to a problem of simulating Hamiltonian dynamics, as elaborated in [53].

Nevertheless, in order to apply HMC there are two tuning parameters that the user needs to calibrate, i.e. the step size ϵ and the number of steps L for which the simulated Hamiltonian system is ran. Determining these parameters is usually a cumbersome and time-efficient task, that requires

also some experience, which is the reason why HMC is not widely used. Nonetheless, it provides the foundation for NUTS, which is a self-tuning algorithm that eliminates the need to choose the problematic number-of-steps parameter L. At the same time, the version of NUTS which will be used through `PyMC3`, includes a dual averaging scheme, introduced in [54], which automatically tunes the step size parameter ϵ, too.

As it exceeds the scope of the current research, more information about the exact mathematical formulation of both HMC and NUTS, as well as the step by step algorithms, are presented in [49]. However, the main algorithms are presented briefly along with the core principles, as elaborated on the aforementioned paper.

In HMC an auxiliary momentum variable r_d is introduced for the model variables θ_d, which are usually drawn independently from the standard normal distribution, leading to their joint density being:

$$\mathrm{P}(\theta, r) \propto \exp\left\{\mathcal{L}(\theta) - 0.5\, r \cdot r\right\} \tag{36}$$

where $\mathcal{L}$ is the logarithm of the joint density of the values of interest θ, while $r \cdot r$ denotes the inner product between the momentum vectors.

A fictitious Hamiltonian system can be used instead of this augmented model, where each parameter gains a physical meaning. In particular, θ corresponds to the particle's position in the D-dimensional space, r denotes its momentum, $\mathcal{L}$ is the negative potential energy function for the given position, $0.5\, r \cdot r$ is the particle's kinetic energy and lastly $\log \mathrm{P}(\theta, r)$ is the negative energy of the particle. The evolution over time of this Hamiltonian system, is often simulated through the Störmer-Verlet ("leapfrog") integrator, which is presented in Algorithm 1.

Algorithm 1: Störmer-Verlet ("leapfrog") integrator

Leapfrog *(θ, r, ϵ)*:
- Set $\tilde{r} \leftarrow r + (\epsilon/2)\, \nabla_\theta \mathcal{L}(\theta)$
- Set $\tilde{\theta} \leftarrow \theta + \epsilon\, \tilde{r}$
- Set $\tilde{r} \leftarrow \tilde{r} + (\epsilon/2)\, \nabla_\theta \mathcal{L}(\tilde{\theta})$
- **return** $\tilde{\theta}, \tilde{r}$

Even though θ and r are vectors, they are not underlined (which would be consistent with the thesis' notation) in order to be in accordance with the original paper [49]

Having introduced the basic principles, the complete HMC algorithm is presented in Algorithm 2.

Algorithm 2: Hamiltonian Monte Carlo (HMC)

Given θ^0, ϵ, L, $\mathcal{L}$, M **for** $m = 1$ **to** M **do**
    Sample $r^0 \sim \mathcal{N}(0, I)$ `// I denotes the identity matrix`
    Set $\theta^m \leftarrow \theta^{m-1}$, $\tilde{\theta} \leftarrow \theta^{m-1}$, $\tilde{r} \leftarrow r^0$
    for $i = 1$ **to** L **do**
        Set $\tilde{\theta}, \tilde{r} \leftarrow$ **Leapfrog**$(\tilde{\theta}, \tilde{r}, \epsilon)$
    With probability $\alpha = \min\left\{1, \frac{\exp\left(\mathcal{L}(\tilde{\theta}) - 0.5\,\tilde{r} \cdot \tilde{r}\right)}{\exp\left(\mathcal{L}(\theta^{m-1}) - 0.5\,r^0 \cdot r^0\right)}\right\}$, set $\theta^m \leftarrow \tilde{\theta}$, $r^m \leftarrow -\tilde{r}$

where L is the number of steps, i.e. Leapfrog updates, and M is the amount of drawn samples.

As mentioned already, a crucial improvement on HMC is the self-tuning of the hyperparameters, ϵ and L, which is being done by the NUTS algorithm. To determine when the amount of leapfrog steps is sufficient, a recursive function is used, namely *Buildtree*, which is presented in Algorithm 3.

Algorithm 3: BuildTree[7] recursive function

BuildTree *(θ, r, u, v, j, ϵ, θ^0, r^0)*:
  if $j = 0$ **then**
    Base case - take one leapfrog step in the direction v
    $\theta', r' \leftarrow$ **Leapfrog**$(\theta, r, v\epsilon)$
    $n' \leftarrow \mathbb{I}\left[u \leq \exp\left\{\mathcal{L}(\theta') - 0.5\, r' \cdot r'\right\}\right]$
    $s' \leftarrow \mathbb{I}\left[u < \exp\left\{\Delta_{\max} + \mathcal{L}(\theta') - 0.5\, r' \cdot r'\right\}\right]$
    return $\theta', r', \theta', r', \theta', n', s', \min\left\{1, \exp\left\{\mathcal{L}(\theta') - 0.5\, r' \cdot r' - \mathcal{L}(\theta^0) + 0.5\, r^0 \cdot r^0\right\}\right\}, 1$
  else
    Recursion - implicitly build the left and right subtrees
    $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha \leftarrow$ **BuildTree**$(\theta, r, u, v, j-1, \epsilon, \theta^0, r^0)$
    if $s' = 1$ **then**
      if $v = -1$ **then**
        $\theta^-, r^-, -, -, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow$ **BuildTree**$(\theta^-, r^-, u, v, j-1, \epsilon, \theta^0, r^0)$
      else
        $-, -, \theta^+, r^+, \theta'', n'', s'', \alpha'', n''_\alpha \leftarrow$ **BuildTree**$(\theta^+, r^+, u, v, j-1, \epsilon, \theta^0, r^0)$
      With probability $\dfrac{n''}{n' + n''}$, set $\theta' \leftarrow \theta''$
      Set $\alpha' \leftarrow \alpha' + \alpha''$, $n'_\alpha \leftarrow n'_\alpha + n''_\alpha$
      $s' \leftarrow s'' \mathbb{I}\left[(\theta^+ - \theta^-) \cdot r^- \geq 0\right] \mathbb{I}\left[(\theta^+ - \theta^-) \cdot r^+ \geq 0\right]$
      $n' \leftarrow n' + n''$
    return $\theta^-, r^-, \theta^+, r^+, \theta', n', s', \alpha', n'_\alpha$

where $\mathbb{I}[\cdot]$ is a boolean operator, returning 1 if the expression inside brackets is true, and 0 if false.

[7] More information on the *BuildTree* function, its input, output and intermediate steps, can be found in [49].

As far as the choice of ϵ is concerned, the function used and the step by step procedure is presented in Algorithm 4.

Algorithm 4: Heuristic for choosing an initial value of ϵ

FindReasonableEpsilon *(θ)*:
    Initialize $\epsilon = 1$, $r \sim \mathcal{N}(0, I)$ `// I denotes the identity matrix`
    Set $\theta', r' \leftarrow$ **Leapfrog** (θ, r, ϵ)
    $a \leftarrow 2\mathbb{I}\left[\frac{\mathsf{P}(\theta', r')}{\mathsf{P}(\theta, r)} > 0.5\right] - 1$
    while $\left(\frac{\mathsf{P}(\theta', r')}{\mathsf{P}(\theta, r)}\right)^a > 2^{-a}$ **do**
        $\epsilon \leftarrow 2^a \epsilon$
        Set $\theta', r' \leftarrow$ **Leapfrog**(θ, r, ϵ)
    return ϵ

Moving towards the final algorithm which is followed by the `PyMC3` package, Algorithm 6 of [49], namely No-U-Turn Sampler (NUTS) with Dual Averaging, is presented in Algorithm 5 of the current project.

Algorithm 5: No-U-Turn Sampler (NUTS) with Dual Averaging

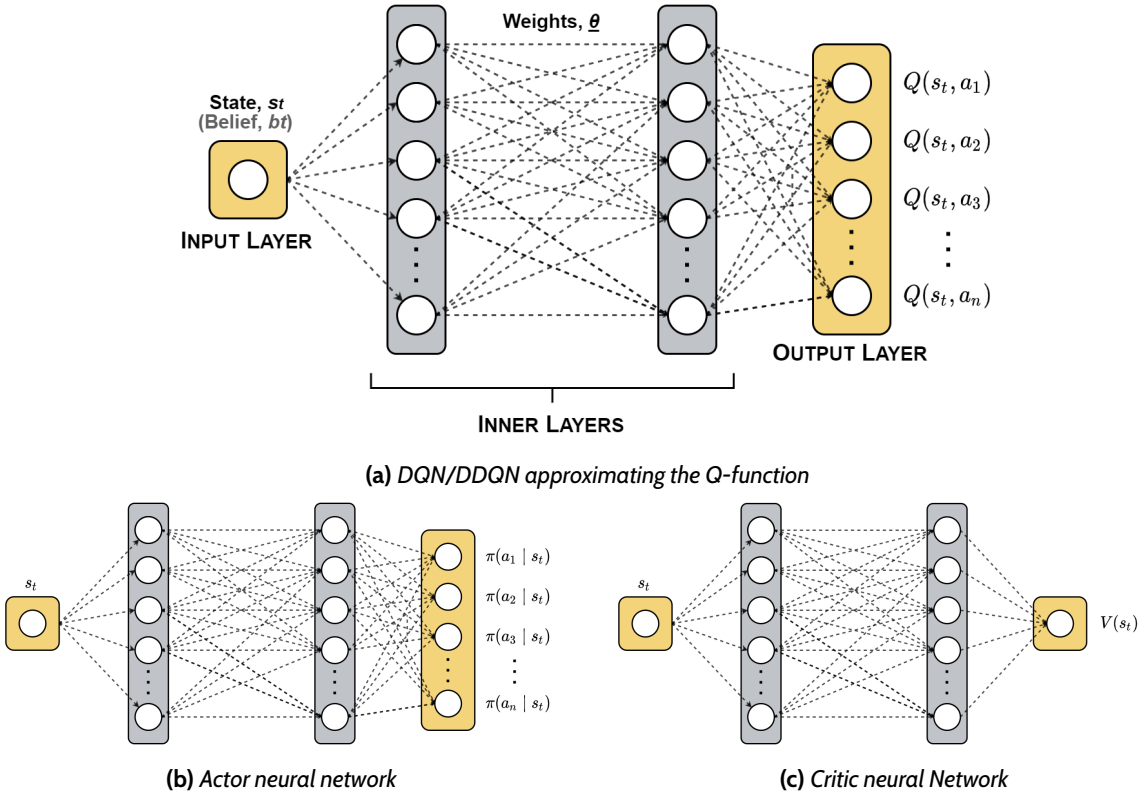
Given $\theta^0, \delta, \mathcal{L}, M, M^{\text{adapt}}$
Set $\epsilon_0 =$ **FindReasonableEpsilon**(θ), $\mu = \log(10\epsilon_0)$, $\bar{\epsilon}_0 = 1$, $\bar{H}_0 = 0$, $\gamma = 0.05$, $t_0 = 10$, $\kappa = 0.75$
for $m = 1$ **to** M **do**
    Sample $r^0 \sim \mathcal{N}(0, I)$
    Resample $u \sim \text{Uniform}\Big(\big[0, \exp\{\mathcal{L}(\theta^{m-1} - 0.5\, r^0 \cdot r^0)\}\big]\Big)$
    Inititalize $\theta^- = \theta^{m-1}$, $\theta^+ = \theta^{m-1}$, $r^- = r^0$, $r^+ = r^0$, $j = 0$, $\theta^m = \theta^{m-1}$, $n = 1$, $s = 1$
    while *s=1* **do**
        Choose a direction $v_j \sim \text{Uniform}(\{-1, 1\})$
        if $v_j = -1$ **then**
            $\theta^-, r^-, -, -, \theta', n', s', \alpha, n_\alpha \leftarrow$ **BuildTree**$(\theta^-, r^-, u, v_j, j, \epsilon_{m-1}, \theta^{m-1}, r^0)$
        else
            $-, -, \theta^+, r^+, \theta', n', s', \alpha, n_\alpha \leftarrow$ **BuildTree**$(\theta^+, r^+, u, v_j, j, \epsilon_{m-1}, \theta^{m-1}, r^0)$
        if $s' = 1$ **then**
            With probability $\min\{1, \frac{n'}{n}\}$, set $\theta^m \leftarrow \theta'$
        $n \leftarrow n + n'$
        $s \leftarrow s'\, \mathbb{I}\big[(\theta^+ - \theta^-) \cdot r^- \geq 0\big]\, \mathbb{I}\big[(\theta^+ - \theta^-) \cdot r^+ \geq 0\big]$
        $j \leftarrow j + 1$
    if $m \leq M^{adapt}$ **then**
        Set $\bar{H}_m = \Big(1 - \frac{1}{m + t_0}\Big)\bar{H}_{m-1} + \frac{1}{m + t_0}\Big(\delta - \frac{\alpha}{n_\alpha}\Big)$
        Set $\log\epsilon_m = \mu - \frac{\sqrt{m}}{\gamma}\bar{H}_m$, $\log\bar{\epsilon}_m = m^{-\kappa}\log\epsilon_m + (1 - m^{-\kappa})\log\bar{\epsilon}_{m-1}$
    else
        Set $\epsilon_m = \bar{\epsilon}_{M^{\text{adapt}}}$

where δ is the desired average acceptance probability of the samples and M^{adapt} is the number of iterations after which the adaption is stopped.

3.3 Deep Reinforcement Learning (DRL) algorithms

It has already been mentioned that in DRL the value functions Q, V as well as the policy π are approximated by a DNN, in order not only to capture efficiently their non-linear behaviour, but also, achieve a reparameterization, and express them in terms of the network's weights, so as to decrease the computational cost and instabilities.

An illustration of such networks is displayed in Figure 9. The input to the DNN is the state (or belief[8]), s_t, while the output layer includes the action-state value function for every possible action, $a_t \in \mathcal{A}$, in the case of DQN and DDQN. In a similar fashion for actor-critic algorithms like A2C and PPO, the actor and the critic neural networks are illustrated in Figures 9b, 9c. The former takes as input the state and yields as an output the probability to take each action given the state, $\pi(a_t \mid s_t)$, and the later using the same input, i.e. the state, returns the corresponding value state function, $V(s_t)$.



(a) *DQN/DDQN approximating the Q-function*

(b) *Actor neural network*

(c) *Critic neural Network*

Figure 9: *Actor-critic DNN architectures* [9]

[8] The used s_t notation represents the belief vector $\underline{b}$ along with any other input quantities (e.g. age) that together compose the state, and are passed as input to the neural networks.

[9] The amount of inner layers and neurons depicted is for the sake of a more clear and explanatory representation

The DRL algorithms that will be considered in the current project are:

- Double Deep Q-Network (DDQN)
- Advantage Actor Critic (A2C)
- Proximal Policy Optimization (PPO)

The step-by-step procedure for all three of them, as found in literature, is described in the following subsections and more specifically in Algorithms 6, 7, 8, respectively.

3.3.1 Double Deep Q-Network (DDQN)

Algorithm 6: Double Deep Q-Network (DDQN)

Initialize primary network weights θ
Initialize target network weights θ^-
Initialize replay buffer
Set target update time T_{update}
for $episode \leftarrow 1$ **to** M **do**
    for $t \leftarrow 1$ **to** T **do**
        Select action a_t according to ϵ-greedy method
        Collect reward $R(s_t, a_t)$, observe next state s_{t+1}
        Store tuple $(s_t, a_t, R(s_t, a_t), s_{t+1})$ in replay buffer
        Sample batch of tuples $(s_i, a_i, R(s_i, a_i), s_{i+1})$ from replay buffer
        if s_{i+1} *is terminal state* **then**
            $y_i = R(s_i, a_i)$
        else
            $y_t = R(s_t, a_t) + \gamma\, Q\left(s_{t+1}, \arg\max Q\left(s_{t+1}, a_{t+1} \mid \theta\right) \mid \theta^-\right)$
        Update parameters θ according to: $\nabla_\theta L(\theta) \simeq \sum \left[\left(Q\left(s_i, a_i \mid \theta\right) - y_i\right) \nabla_\theta Q\left(s_i, a_i \mid \theta\right)\right]$
        if T_{update} **then**
            $\theta^- = \theta$

3.3.2 Advantage Actor Critic (A2C)

Algorithm 7: Advantage Actor Critic (A2C)

Initialize policy parameters θ
Initialize value function parameters θ_v
for $Episode = 0, 1, 2 \ldots$ **do**
  for $t \leftarrow 1$ **to** T **do**
    Perform a_t according to policy $\pi(a_t \mid s_t, \theta)$
    Collect reward $R(s_t, a_t)$, observe next state s_{t+1}
  if *terminal* s_t **then**
    $R = 0$
  else
    $R = V(s_t \mid \theta_v)$// Bootstrap from last state
  Update θ according to:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_t, a_t}\left[\sum_{t \geq 0} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\left(R(s_t, a_t) + \gamma V(s_{t+1} \mid \theta_v) - V(s_t \mid \theta_v)\right)\right]$$

  Update θ_v according to:

$$\nabla_{\theta_v} J(\theta_v) = \mathbb{E}_{s_t, a_t}\left[\nabla_{\theta_v} V(s_t \mid \theta_v)\left(R(s_t, a_t) + \gamma V(s_{t+1} \mid \theta_v) - V(s_t \mid \theta_v)\right)\right]$$

3.3.3 Proximal Policy Optimization (PPO)

Algorithm 8: Proximal Policy Optimization (PPO) [1]

Initialize policy parameters θ_0

Initialize value function parameters ϕ_0

for $k = 0, 1, 2 \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment

Compute rewards-to-go R_t

Compute advantage estimates, A_t (using any method of advantage estimation) based on the current value function V_{ϕ_k}

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| \, T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta \left(a_t \mid s_t\right)}{\pi_{\theta_k} \left(a_t \mid s_t\right)} A^{\pi_{\theta_k}} \left(s_t, a_t\right), \quad g\left(\epsilon, A^{\pi_{\theta_k}} \left(s_t, a_t\right)\right) \right)$$

typically via stochastic gradient ascent with Adam

Fit the value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| \, T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi \left(s_t\right) - R_t \right)^2$$

typically via some gradient descent algorithm

3.4 Conclusions

Having elaborated on the selected algorithms of this project, an updated version of the conceptual breakdown flowchart (Figure 6) is illustrated in Figure 10.

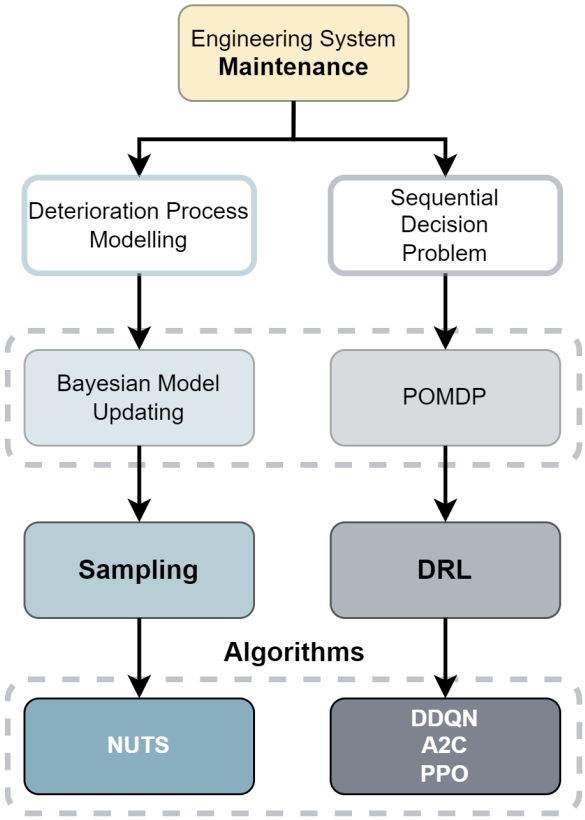


Figure 10: *Problem conceptual breakdown - Motivation for the selected tools and algorithms*

This flowchart acts both as a summary of the current chapter as well as the motivation and reasoning behind the choice of the specific algorithms. It moves from the general problem to be tackled, namely "Engineering System Maintenance", to the most efficient tools existent for every sub-task, i.e. NUTS, DDQN, A2C and PPO.

4 Verification, Validation and Benchmarking

4.1 Toy Problem

4.1.1 Problem Description

Now that the general framework of this project has been explained, a simple case, i.e. a "toy" problem, is chosen in order to further elaborate on the proposed algorithm in a more streamlined manner, but also to make a first assessment of its efficiency, drawbacks, and future issues that need to be adjusted. Therefore, a SDOF oscilator is selected, with its stiffness being the deterioration parameter. The mass of the oscilator, m, is assumed to be deterministic and constant, while the initial stiffness is denoted as K_0. The described system is illustrated in Figure 11.

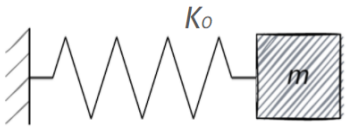


Figure 11: *SDOF oscillator*

The deterioration model employed is:

$$D(\tau) = A\tau^B \tag{37}$$

where A, B, are random variables, responsible for the uncertainty in this model. In particular, A corresponds to the deterioration rate, while B is related to the non-linearity effect in terms of a power law in time. This is a standard model, described by the rate equation above, often employed for an engineering system's deterioration, e.g. [46].

A clear distinction should be made between the decision step t, which is the running time variable of the system's lifespan, and the deterioration rate τ which describes the exposure time, or the age, of the system, and subsequently the degree to which the corrosive environment affects it. In a scenario where no maintenance action is performed during the time window of interest, t and τ coincide. However, as will be displayed in this section, the agent's actions can possibly reduce or even reset the deterioration rate τ while the decision step t will keep increasing with unit step.

The stiffness at any given state is calculated as follows:

$$K(\tau) = \frac{K_0}{1 + D(\tau)} = \frac{K_0}{1 + A\tau^B} \tag{38}$$

It is assumed that there is a monitoring system, whose noisy measurements are passed through an OMA scheme, which subsequently outputs modal data, in this case, the eigenfrequency, ω. The eigenfrequency, as known from basic structural dynamics theory, is calculated, hence related to the system's damage, through Equation 39.

$$\hat{\omega}(\tau) = \sqrt{\frac{K(\tau)}{m}} = \sqrt{\frac{K_0}{m\,(1 + D(\tau))}} = \sqrt{\frac{K_0}{m\left(1 + A\tau^B\right)}} \tag{39}$$

Since A and B are stochastic, $\hat{\omega}(\tau)$ represents the aforementioned noisy measurement. After passing it through the the OMA procedure, the yielded observation which is given to the agent can be expressed as:

$$\omega(\tau) = \hat{\omega}(\tau) + \varepsilon_{\text{oma}} \tag{40}$$

where ε_{oma} corresponds to the additional noise that is explicitly added during the OMA scheme.

For the case at hand, it is assumed that the additional noise ε_{oma} follows a Gaussian distribution with a zero mean and a standard deviation that is proportional to the noisy measurement.

$$\varepsilon_{\text{oma}} \sim \mathcal{N}\left(0,\, \epsilon_{\text{c}} \cdot \hat{\omega}(\tau)\right) \tag{41}$$

where ϵ_{c} is a coefficient describing the degree to which the OMA scheme contaminates the measurement with noise.

Therefore, the observation during each decision step is generated based on the following Gaussian distribution:

$$\omega(\tau) \sim \mathcal{N}\left(\hat{\omega}(\tau),\, \epsilon_{\text{c}} \cdot \hat{\omega}(\tau)\right) \tag{42}$$

The choice for the possible actions that the agent can take is a significant modeling decision. Apart from the "*do nothing*" and the "*total replacement*" actions, there is the need of a "*partial repair*" one, too. The way in which the rest of the parameters will be affected due to such a repair can vary depending on the materials of the structure, the type of repair, etc. Regarding the chosen deterioration model, i.e. $D(\tau) = A\tau^B$, there are three cases of partial repair that can be distinguished.

- Reduce only the caused damage $D(\tau)$, but the deterioration rate, τ, at which the environment affects the structure, stays the same. This would mean that the slope of the $D(\tau) - \tau$ curve will stay the same, and a vertical shift of the right-half curve will be observed, as displayed in Figure 12. This could be the case when restoring the damaged surrounding concrete of a Reinforced Concrete (RC) component, but no action is taken for the corrosion of the rebar, which will continue to develop.
- Reduce the deterioration rate, meaning that the environment will continue to affect the structure with a reduced intensity, but the existent damage that is already caused is not affected. Geometrically, there will be a horizontal shift to the left in order for the damage to continue developing in a less steep slope (Figure 13). For example, applying an epoxy painting on a steel member without repairing the existent damage, will slow down the effect of the corroding environment, but the damaged cross-section will remain as is.

- A combination of the two cases above, which means that both the damage $D(\tau)$ and the deterioration rate, τ, are reduced. This action equals a move back along the $D(\tau) - \tau$ curve as illustrated in Figure 14. For example, removing/restoring the corroded parts of a steel cross-section and applying a protective paint to protect it against the corroding environment.

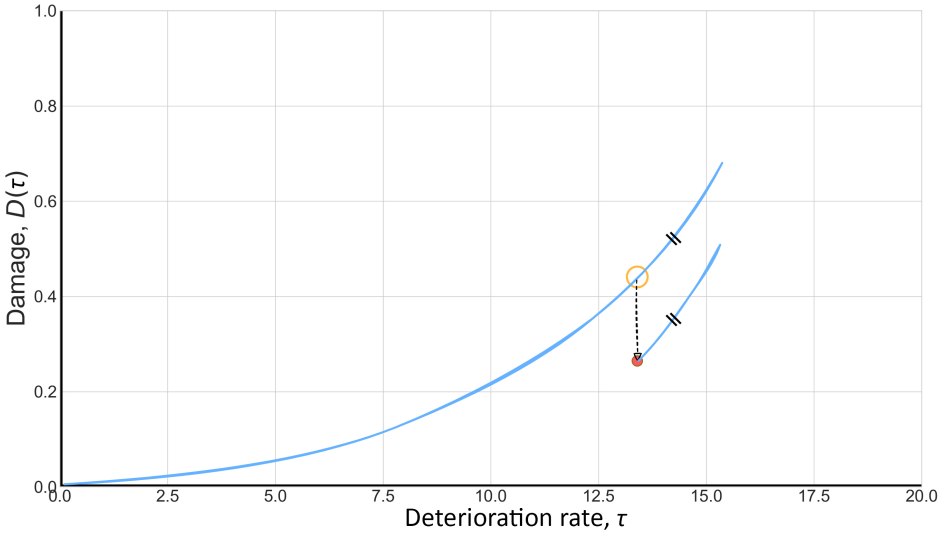


Figure 12: *Repair by reducing the damage* $D(\tau)$

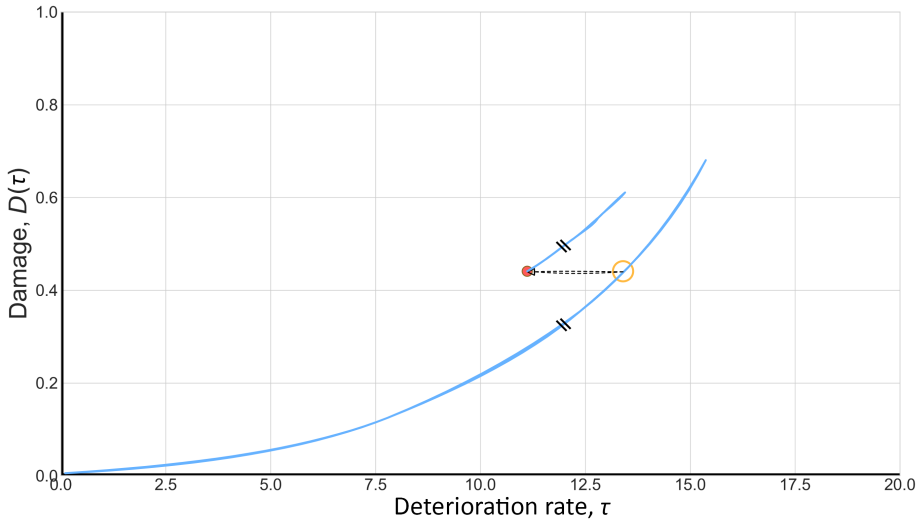


Figure 13: *Repair by reducing the deterioration rate* τ

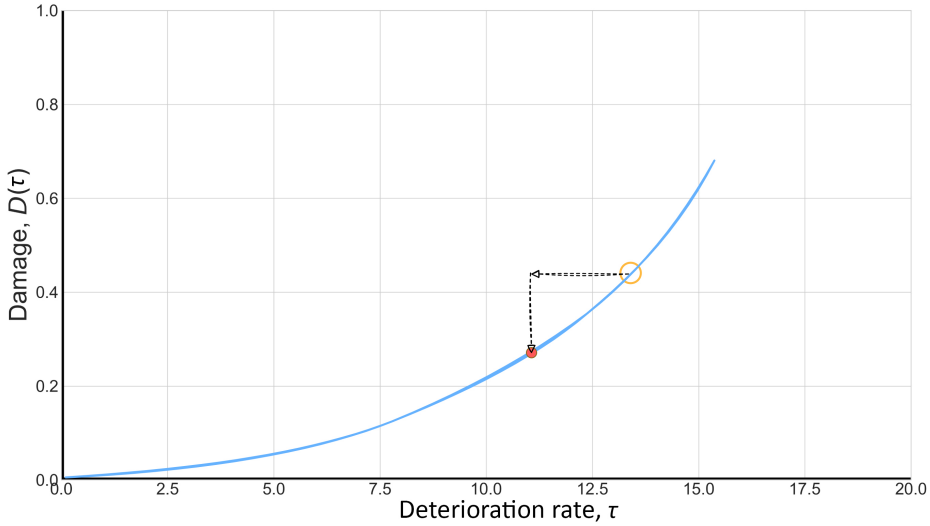


Figure 14: *Repair by reducing both the damage and the deterioration rate*

Having described the various approaches for modelling a repair action, the one depicted in Figure 13 is chosen for the current application. Thus, the three possible actions are listed in Table 1.

Table 1: *Action-space for toy problem*

Index	Action
0	do nothing
1	partial repair*
2	total replace

* Decrease the deterioration rate τ, i.e. rewind by two steps

Regarding the rewards, i.e. the costs of maintenance, a fixed amount is considered for the system's total replace (action 2), and every other cost is expressed as relatively to this value. The correlation between the costs is included in Table 2.

Table 2: *Rewards (costs) for the toy problem*

Description	Cost	Value	Factor
Total replacement	C_{R}	C_0 units	1
Partial repair	C_{M}	$0.5\,C_{\mathsf{R}}$	0.5
Failure	C_{F}	$2\,C_{\mathsf{R}}$	2
Risk of failure	C_{risk}	$P_f\,C_F$	$2\,P_f$

It can be observed that failure, which will cause a complete replacement of the component (system), has a higher cost than the actual replacement as an action. This is the case because of the sudden aspect of a structure failing, and the unpredicted consequences that this event might provoke financially.

The input data used for this application, such as deterministic quantities, starting values, etc, are gathered and displayed in Table 3.

Table 3: *Toy problem input data*

Quantity	Value	Units
Mass, m	10	[kg]
Initial stiffness, K_0	200	[N/m]
Replace cost, C_R	10000	[-]
Noise coefficient, ϵ_{c}	10%	[-]

4.1.2 Discrete case

Considering the stochastic parameters A, B, as well as the damage $D(\tau)$ to be continuous variables, increases significantly the computational cost. This is why, in order to scale up gradually the complexity in verifying the validity of the proposed methodology, a discrete version of the described toy problem is being tackled first.

In particular, the following discrete values are accounted for:

$$A = \begin{bmatrix} 6\mathrm{e}-4 & 8\mathrm{e}-4 & 10\mathrm{e}-4 & 12\mathrm{e}-4 & 14\mathrm{e}-4 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.4 & 1.6 & 1.8 & 2.0 & 2.2 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \end{bmatrix}$$

It should be noted that just for demonstration purposes in the coming figures, smaller discrete spaces will be used, particularly:

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$
$$B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$$
$$D = \begin{bmatrix} D_1 & D_2 & D_3 \end{bmatrix}$$

In each iteration the agent does not know the exact value of the damage, so it forms a belief $\underline{b}$, i.e. a vector which contains the probabilities of all possible damage states[10]. For the smaller scale representative discrete case, this vector has the form displayed in Figure 15.

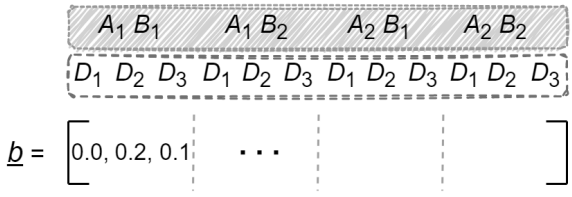


Figure 15: *Belief vector, $\underline{b}$ for discrete case*

The observation $\omega(\tau)$ in each decision step is generated as described already in Equation 42.

The main advantage of this simplification, compared to the continuous case, is the calculation of the belief vector in a closed-form. This is achieved using the so-called transition matrix $\underline{\underline{\mathbb{P}}}$, as well as the observation matrix $\underline{\underline{\mathbb{O}}}$. The former corresponds to the probability of shifting to a new state given the previous state and the chosen action, while the latter reflects the probability of an eigenfrequency ω to be observed given the current state. In math notation, they are defined as follows:

$$\underline{\underline{\mathbb{P}}} = \mathsf{P}(s_{t+1} \mid s_t, a_t) \tag{43}$$
$$\underline{\underline{\mathbb{O}}} = \mathsf{P}(o_t \mid s_t) \tag{44}$$

where s_{t+1} is the next state, s_t is the current state, a_t is the chosen action and o_t is the observation ω.

The dependency of the transition probability to the chosen action is dropped, since a_t is accounted for by modifying the deterioration rate. Additionally, in order to describe all possible transitions, a different transition matrix is considered for each deterioration rate τ, i.e. $\underline{\underline{\mathbb{P}_\tau}} = \mathsf{P}(s_{t+1} \mid s_t)$.

[10] The sum of all elements in the belief vector should sum up to 1, i.e. $\sum b_i = 1$.

A typical example of a transition matrix for a random deterioration rate is depicted in Figure 16.

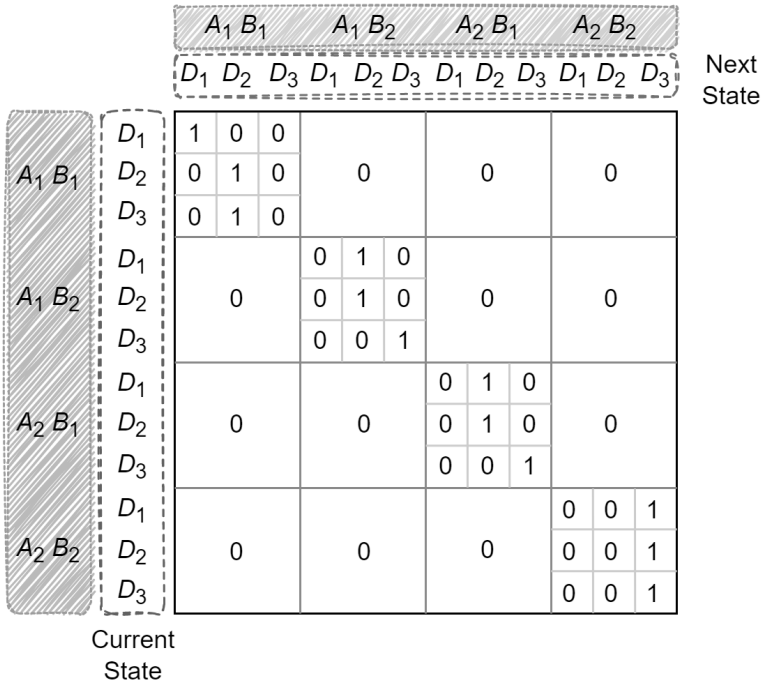


Figure 16: *Transition matrix, $\underline{\underline{\mathbb{P}_\tau}}$ for discrete case*[11]

To elaborate a bit further on the meaning of the entries in the transition matrix, the second row and second column are examined, as displayed in Figure 17.

[11]The values included in both Figures 15, 16 are arbitrary, for illustration purposes.

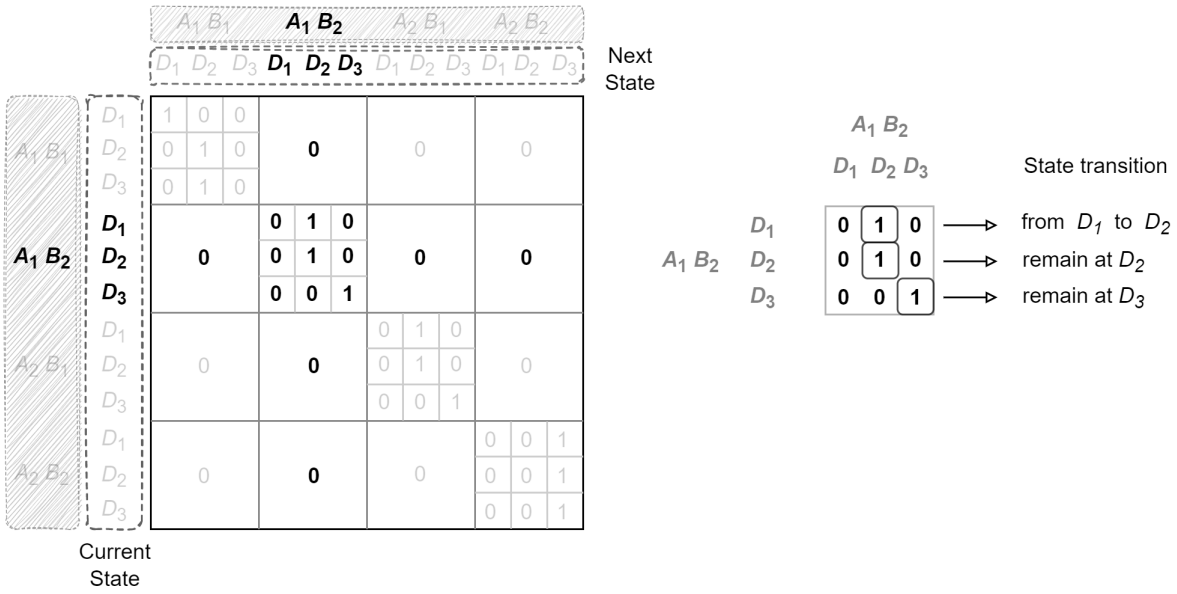


Figure 17: *Transition matrix, $\underline{\underline{\mathbb{P}}}_\tau$ for discrete case explanation*

In the examined part, knowing that the parameters A, B have the values A_1 and B_2 respectively, the damage state will be D_2. This means that the agent can either shift from state D_1 to D_2, or remain to state D_2, or in the case the prior damage has already reached D_3, it can not go back in a less damaged state, so it will remain at D_3. It is observed that only the 3 by 3 sub-matrix that corresponds to the same A, B values both in the row and the column indexing is populated with non-zero values, which is reasonable since the two parameters can not simultaneously be equal to different values. Lastly, an important property of the transition matrix is that each row needs to sum up to 1 (as noted also for the belief vector).

Having defined the necessary quantities, the belief vector can be found using Bayes Theorem, applied in POMDPs, avoiding time consuming sampling procedures like MCMC or NUTS. For a single entry of $b(s_{t+1})$ it holds:

$$b(s_{t+1}) = \frac{p(o_{t+1} \mid s_{t+1})}{p(o_{t+1} \mid \underline{b})} \sum_{s_t \in \mathcal{S}} p(s_{t+1} \mid s_t)\, b(s_t) \tag{45}$$

where the denominator is a normalizing constant, i.e. the so-called evidence in Bayes Theorem, which is equal to:

$$p(o_{t+1} \mid \underline{b}) = \sum_{s_{t+1} \in \mathcal{S}} p(o_{t+1} \mid s_{t+1}) \sum_{s_t \in \mathcal{S}} p(s_{t+1} \mid s_t) b(s_t) \tag{46}$$

Equations 45, 46 can be generalized and rewritten in matrix notation:

$$\underline{b}' = \frac{\underline{\underline{\mathbb{O}}}(o_{t+1}) \odot \left[\underline{\underline{\mathbb{P}}}^T \cdot \underline{b}\right]}{\left[\underline{\underline{\mathbb{O}}}^T(o_{t+1}) \cdot \left[\underline{\underline{\mathbb{P}}}^T \cdot \underline{b}\right]\right]} \tag{47}$$

As far as the DRL aspect is concerned, apart from the belief vector, $\underline{b}$, the exposure time of the component, in other words the deterioration rate τ, needs to be fed into the DNN, since the current case constitutes a time dependent problem. A time parameter is a necessary input for the DNN in order to define accurately the rate with which the system deteriorates at every given state, after any maintenance action.

To be more precise, for DDQN, the belief vector $\underline{b}$ and the deterioration rate τ are passed as input to the DNN, and after a forward pass the network yields the action-state value function for each action, $Q(s_t, a_i)$ for $i = 1, 2, 3$, which is interpreted as the reward of taking a specific action a_i given a state s_t. These three value functions constitute the knowledge based on which the agent will act, i.e. if the agent chooses to exploit what it already knows, the action with the highest Q-value (as derived from the DNN) will be chosen. A schematic representation of the described DNN architecture is displayed in Figure 18.

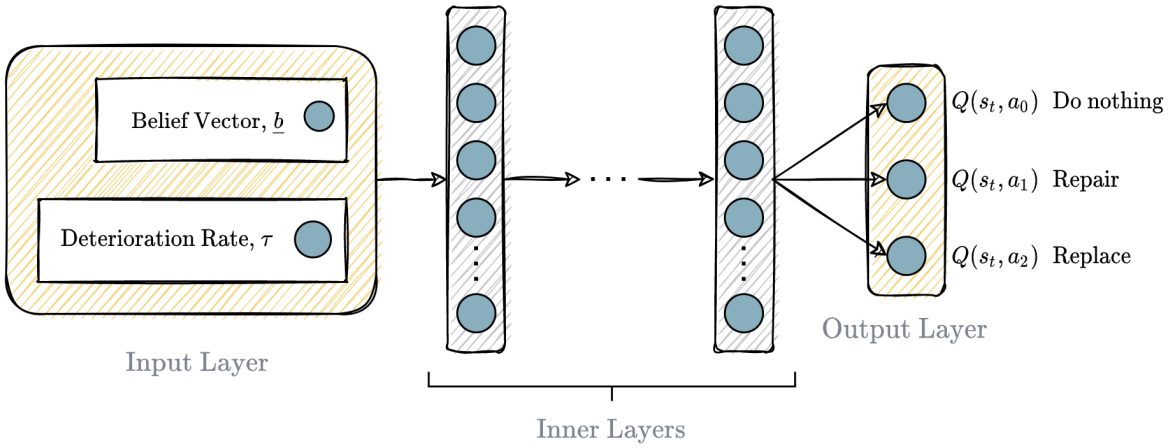


Figure 18: *DDQN DNN architecture for the discrete toy problem*

When it comes to A2C and PPO, which are actor-critic algorithms, the same input, $\underline{b}$ and τ, are passed to two different networks, namely the Actor and the Critic network. A forward pass of the former will yield directly the policy $\pi_\theta(a_i \mid s_t)$, i.e. the probability of choosing each action a_i when being at state s_t, while the latter one returns the state value function $V_\phi(s_t)$, which corresponds to the reward of being at the state s_t regardless from the chosen action. The aforementioned networks are depicted in Figure 19.

The symbol $\odot$ in Equation 47 denotes the Hadamar product, i.e. an elementwise matrix multiplication.

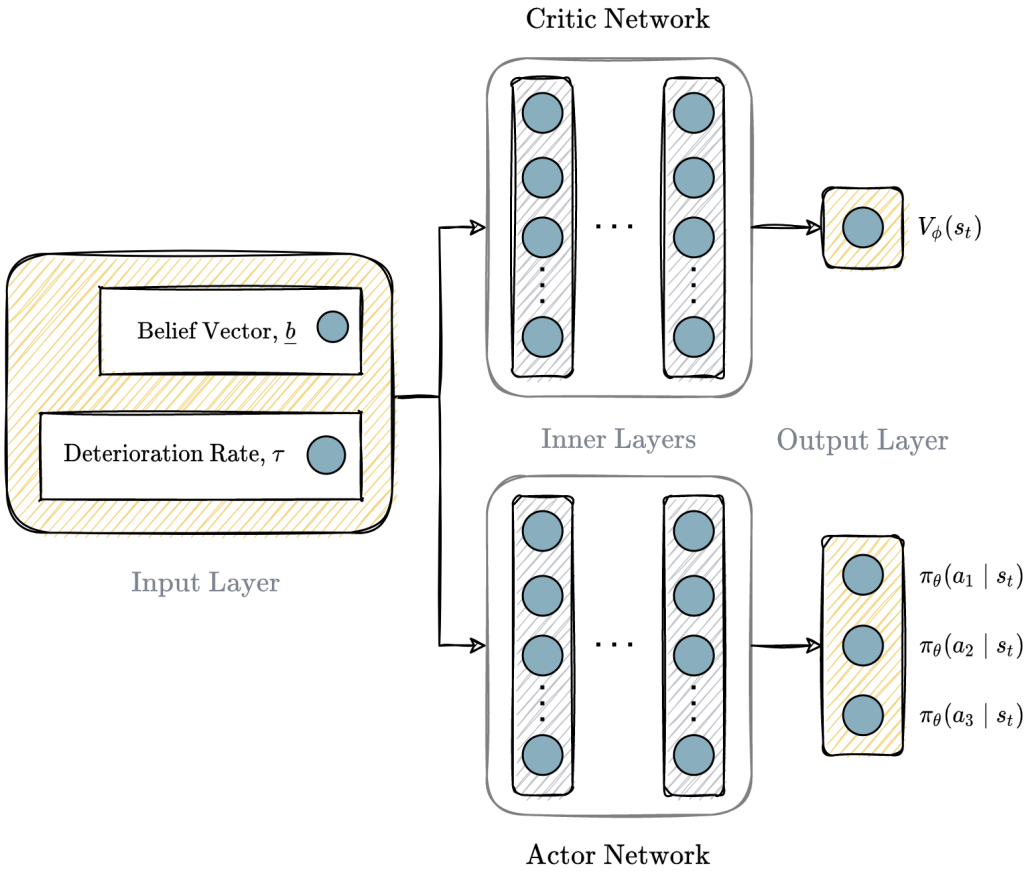


Figure 19: *Actor-critic DNN architecture for the discrete toy problem*

It should be noted that although three algorithms were initially chosen to be tested, namely DDQN, A2C and PPO, only two of them actually performed adequately. In particular, A2C failed to yield optimal solutions both for the discrete and the continuous variations of the toy problem. Therefore, the necessary steps of the proposed framework regarding the remaining two algorithms are listed in Algorithms 9 and 10 for DDQN and PPO respectively.

To produce more compact and readable algorithms, several counting parameters are defined, which are explained in Tables 4, 5, for DDQN and PPO respectively.

Table 4: *Counters used for DDQN*

Parameter	Name	Description
T	steps per episode	The amount of decision steps considered for the maintenance of the SDOF system
T_{update}	target network update	Every how many steps the parameters of the current neural network are passed to the target one
M	number of episodes	The number of episodes employed to train the DDQN agent. Each episode consists of T decision steps

Table 5: *Counters used for PPO*

Parameter	Name	Description
T	steps per episode	The amount of decision steps considered for the maintenance of the SDOF system
N	steps per epoch	The amount of decision steps employed for a single batch training of the PPO agent
M	number of epochs	The number of epochs used to train the PPO agent in total. Each epoch consists of N decision steps

Algorithm 9: Double Deep Q-Network (DDQN) - Discrete Toy Problem

Initialize primary network weights θ
Initialize target network weights θ^{-}
Initialize replay buffer
for $episode \leftarrow 1$ **to** M **do**
    $s_t \leftarrow$ reset environment // $\tau \leftarrow 0$, initialize belief vector $\underline{b}$ to zero damage
    for $t \leftarrow 1$ **to** T **do**
        $\tau \leftarrow \tau + 1$
        Calculate the next belief vector according to
        $\underline{b}' = \dfrac{\underline{\underline{\mathbb{O}}}(o_{t+1}) \odot \left[\underline{\underline{\mathbb{P}}}^{T} \cdot \underline{b}\right]}{\left[\underline{\underline{\mathbb{O}}}^{T}(o_{t+1}) \cdot \left[\underline{\underline{\mathbb{P}}}^{T} \cdot \underline{b}\right]\right]}$ // the transition matrix $\underline{\underline{\mathbb{P}}}$ depends on τ
        Choose Action *according to ϵ-greedy method*:
            Generate random number $rand \in [0,1]$
            if $rand < \epsilon$ **then**
                Sample a random action, $a_t \in \mathcal{A}$ // Explore
            else
                $a_t = \underset{a_t \in \mathcal{A}}{\text{argmax}} Q(s_t, a_t)$ // Exploit
        if a_t *is "replace"* **then**
            $\tau \leftarrow 0$
        else if a_t *is "repair"* **then**
            $\tau \leftarrow \max(\tau - 2, 0)$
        Calculate P_f for the belief vector $\underline{b}'$
        $R(s_t, a_t) \leftarrow C_{a_t} + P_f C_{\mathsf{F}}$
        Store tuple $(s_t, a_t, R(s_t, a_t), s_{t+1})$ in replay buffer // $s_t = \langle \underline{b}, \tau \rangle$, $s_{t+1} = \langle \underline{b}', \tau \rangle$
        $s_t \leftarrow s_{t+1}$
        Sample batch of tuples $(s_i, a_i, R(s_i, a_i), s_{i+1})$ from replay buffer
        if s_{i+1} *is terminal state* **then**
            $y_i = R(s_i, a_i)$
        else
            $y_t = R(s_t, a_t) + \gamma\, Q\left(s_{t+1}, \text{argmax}\, Q\left(s_{t+1}, a_{t+1} \mid \theta\right) \mid \theta^{-}\right)$
        Update parameters θ according to: $\nabla_\theta L(\theta) \simeq \sum \left[\left(Q\left(s_i, a_i \mid \theta\right) - y_i\right) \nabla_\theta Q\left(s_i, a_i \mid \theta\right)\right]$
        if T_{update} **then**
            $\theta^{-} = \theta$

Algorithm 10: Proximal Policy Optimization (PPO) - Discrete Toy Problem

Initialize policy (actor) network weights θ
Initialize value function (critic) network weights ϕ
for $episode \leftarrow 1$ **to** M **do**
    $s_t \leftarrow$ reset environment // $t \leftarrow 0, \tau \leftarrow 0$, initialize belief vector $\underline{b}$ to zero damage
    for $n \leftarrow 1$ **to** N **do**
        $t \leftarrow t+1, \tau \leftarrow \tau + 1$
        Calculate the next belief vector according to
        $\underline{b}' = \dfrac{\underline{\underline{\mathbb{O}}}(o_{t+1}) \odot \left[\underline{\underline{\mathbb{P}}}^T \cdot \underline{b}\right]}{\left[\underline{\underline{\mathbb{O}}}^T(o_{t+1}) \cdot \left[\underline{\underline{\mathbb{P}}}^T \cdot \underline{b}\right]\right]}$ // the transition matrix $\underline{\underline{\mathbb{P}}}$ depends on τ
        $\pi_\theta(a_t \mid s_t) \leftarrow$ **Actor Net** (s_t) // forward pass of the actor network
        $V_\phi(s_t) \leftarrow$ **Critic Net** (s_t) // forward pass of the critic network
        $a_t \leftarrow$ sample $\pi_\theta(a_t \mid s_t)$
        if a_t *is "replace"* **then**
            $\tau \leftarrow 0$
        else if a_t *is "repair"* **then**
            $\tau \leftarrow \max(\tau - 2, 0)$
        Calculate P_f for the belief vector $\underline{b}'$
        $R(s_t, a_t) \leftarrow C_{a_t} + P_f C_{\mathsf{F}}$
        Store tuple $\left(s_t, a_t, \pi_\theta(a_t \mid s_t), V_\phi(s_t), R(s_t, a_t)\right)$ in $\mathcal{D}_k$ // $s_t = \langle \underline{b}, \tau \rangle$
        $s_t \leftarrow s_{t+1}$ // $s_{t+1} = \langle \underline{b}', \tau \rangle$
        if $t = T$ **or** $n = N$ **then**
            if $t = T$ **then**
                $V_\phi(s_{t+1}) \leftarrow 0$
                $s_t \leftarrow$ reset environment // $t \leftarrow 0, \tau \leftarrow 0$, initialize belief vector $\underline{b}$ to zero damage
            else
                $V_\phi(s_{t+1}) \leftarrow$ **Critic Net**(s_t)
            Returns $\delta_t \leftarrow R(s_t, a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
            Advantages $A_t \leftarrow \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$
            Store δ_t, A_t in $\mathcal{D}_k$
    Train Agent $(\mathcal{D}_k)$

The function "*Train Agent*" is further elaborated in Algorithm 11.

Algorithm 11: Proximal Policy Optimization (PPO) agent training - Toy problem

Train Agent $(\mathcal{D}_k)$**:**

Update parameters ϕ, using the Critic cost function:

$$L^{\mathsf{VF}}(\phi) = \sum_{t=1}^{T} \left(V_\phi(s_t) - \delta_t \right)^2$$

Update parameters θ, using the Actor loss function:

$$L^{\mathsf{CLIP}}(\theta) = \sum_{t=1}^{T} \min \left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\mathsf{old}}}\left(a_t \mid s_t\right)} A_t\left(s_t, a_t\right), \mathsf{clip}\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\mathsf{old}}}\left(a_t \mid s_t\right)}, 1-\varepsilon, 1+\varepsilon \right) A_t \right)$$

via minibatch stochastic gradient ascent with Adam

4.1.3 Continuous case

Following the discrete case, the stochastic parameters A and B are now considered to be continuous variables. The assumed prior distributions are displayed in Table 6.

Table 6: *Parameters of the stochastic deterioration model*

Parameter	Distribution	Mean	Coefficient of Variation (CV)
A	Lognormal	$8.0E-03$	0.5
B	Normal	1.5	0.3

Since the continuous case of the toy problem does not impose any simplification as far as the proposed framework is concerned, the already presented flowchart (Figure 8) is now being updated and elaborated further, and is depicted in Figure 20.

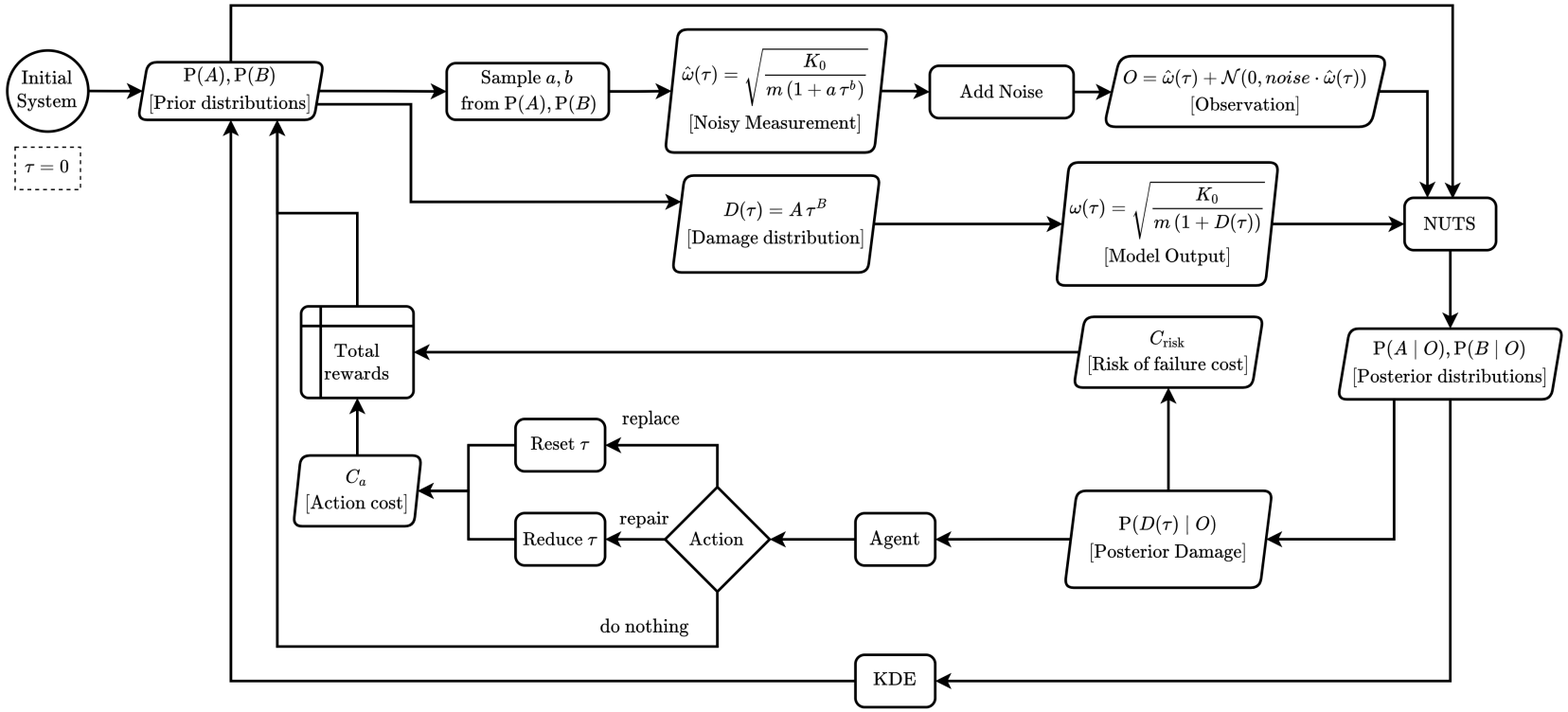


Figure 20: *Framework flowchart for the toy problem*

As displayed above, the parameters of interest that will be updated in every iteration are the distributions of A, B, hence, $\mathsf{P}(A), \mathsf{P}(B)$, which are used as priors for the Bayesian inference. These distributions are used both to define the damage distribution $D(\tau) = A\tau^B$, and to create the noisy measurement based on sampled a, b values. The measurement will be further contaminated with noise, as seen at the top part of the flowchart, while the damage distribution is used to compute the model output:

$$\omega = \sqrt{\frac{K_0}{m\big(1 + D(\tau)\big)}}$$

Then, the observation, the model output and the prior distribution are passed to the NUTS algorithm to yield the posterior distributions of A, B and subsequently $D(\tau)$. These posterior distributions, $\mathsf{P}(A \mid O), \mathsf{P}(B \mid O)$ are transformed into priors using a Kernel Density Estimation (KDE) scheme. The updated damage distribution $\mathsf{P}(D(\tau) \mid O)$ is used to calculate the risk of failure cost C_{risk} which is added to the stored total reward of the iteration, but it is also passed to the agent in order to choose an action based on it. If the agent chooses to perform a maintenance action, this would result to a modification of the deterioration rate while yielding also an additional action cost C_a which is added as well to the stored total reward. Before proceeding to the next iteration, the deterioration rate will be incremented by 1. This loop is being ran for 20 decision steps, and the quantity that needs to be optimized, i.e. minimized, is the total reward, thus the total maintenance cost.

A detailed description of the parameter updating procedure is presented in Algorithm 12.

Algorithm 12: Deterioration model parameters updating - Toy Problem

DeteriorationParametersUpdating *(*$\mathsf{P}(A), \mathsf{P}(B), mass, K_0, noise, T$*)*:
- $D(0) \leftarrow 0$
- $\tau \leftarrow 0$
- **for** $t \leftarrow 1$ **to** T **do**
  - $\tau \leftarrow \tau + 1$
  - $\omega_0 \leftarrow \sqrt{\frac{K_0}{mass\,(1 + D(\tau))}}$ `// mean ω`
  - Generate $\omega_{\text{obs}} \leftarrow \mathcal{N}(\omega_0, noise)$
  - **NUTS** *(*$\mathsf{P}(A), \mathsf{P}(B), \omega_{obs}$*)*:
    - **Output:** $\mathsf{P}(A \mid \omega_{\text{obs}}), \mathsf{P}(B \mid \omega_{\text{obs}}), \mathsf{P}(D(\tau))$
  - Choose action, a_t `// as explained in the DRL algorithm`
  - Adjust $D(\tau)$ distribution based on a_t
  - Sample A, B from $\mathsf{P}(A \mid \omega_{\text{obs}}), \mathsf{P}(B \mid \omega_{\text{obs}})$
  - $D(\tau) \leftarrow A\tau^B$ `// will be used to calculate mean ω`
  - $\mathsf{P}(A), \mathsf{P}(B) \leftarrow \mathsf{P}(A \mid \omega_{\text{obs}}), \mathsf{P}(B \mid \omega_{\text{obs}})$ `// posteriors become priors through KDE`

This algorithm focuses only on the Bayesian inference and the updating of the parameters. This is why the action part is covered abstractly.

The first term of the total cost during an iteration is the already mentioned risk of failure cost, which is being computed as the product of the probability of failure times the cost of failure, $P_f \cdot C_{\text{fail}}$. The probability of failure, P_f in the current simplified application is considered equal to the number of samples from the damage distribution that are located above the failure threshold, divided by the total number of samples (Figure 21).

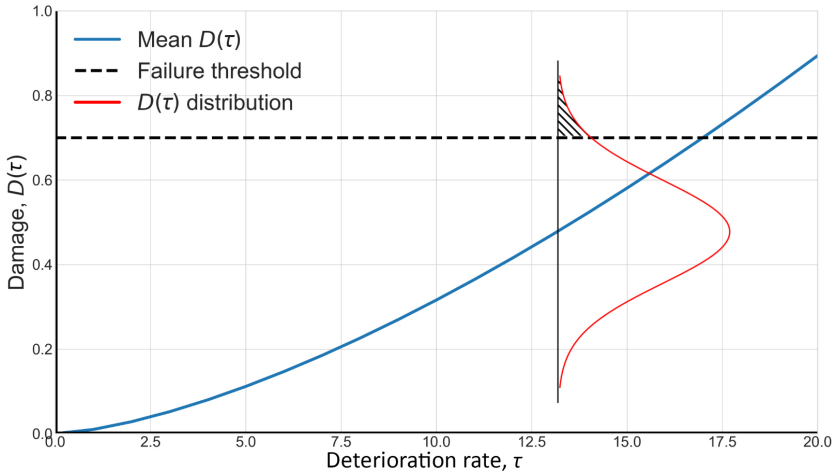


Figure 21: *Failure Probability calculation* [12]

Moving to the DRL part of the framework, the need to select a discrete number of features that will accurately describe each deterioration state has emerged, and subsequently will be fed into the DNN. For this purpose, the statistical moments of the $D(\tau)$ distribution were chosen, namely, the mean, the variance, the skewness and the kurtosis. As mentioned already for the discrete case, the examined problem is time dependent, meaning that the deterioration rate, τ of the structure needs also to be given as an input to the DNN. Regarding the neurons in the output layer, they correspond to the action state value functions for the three different actions when using the DDQN algorithm. The aforementioned characteristics of the DNN architecture are demonstrated in Figure 22.

[12] The curves (values and shape) illustrated in Figure 21 are arbitrary, for explanatory reasons.

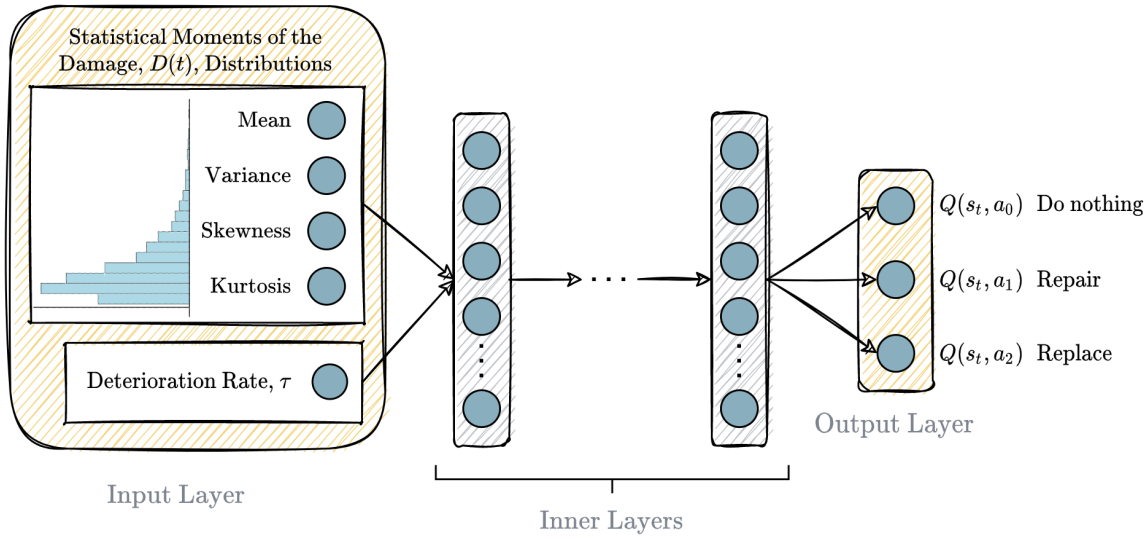


Figure 22: *DDQN DNN architecture for the continuous toy problem*

Accordingly, for actor-critic algorithms, hence, also actor and critic DNNs, the architecture is displayed in Figure 23.

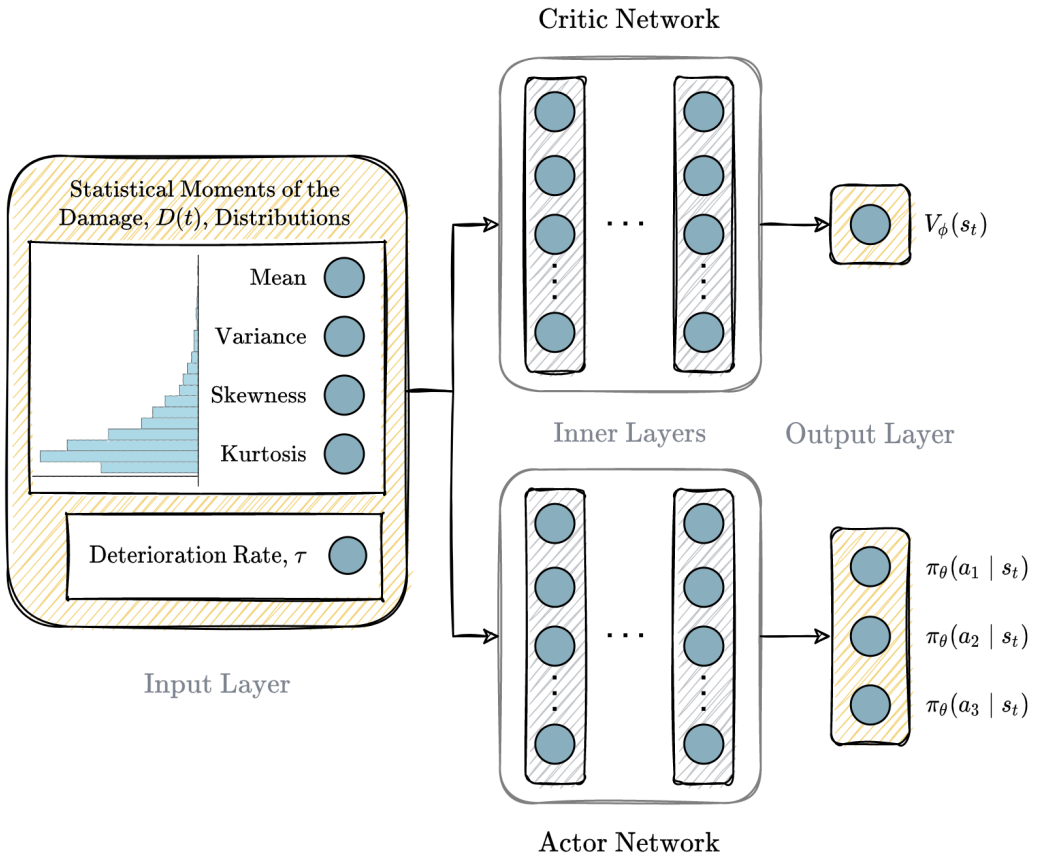


Figure 23: *Actor-critic DNN architecture for the continuous toy problem*

It has already been mentioned that only two out of the three tried algorithms managed to produce valuable results, with A2C being the one that under-delivered. Thus, the detailed procedure of the proposed framework concerning DDQN and PPO about the continuous variation of the toy problem, is presented in Algorithms 13 and 14 respectively. As with the discrete version of the problem, the counters used in the coming algorithms are presented in Tables 4 and 5.

Algorithm 13: Double Deep Q-Network (DDQN) - Continuous Toy Problem

Initialize primary network weights θ
Initialize target network weights θ^-
Initialize replay buffer
for $episode \leftarrow 1$ **to** M **do**
    $s_t \leftarrow$ reset environment // $\tau \leftarrow 0$, initialize A, B
    for $t \leftarrow 1$ **to** T **do**
        $\tau \leftarrow \tau + 1$
        BMU for params A, B // procedure shown in Algorithm 12
        Choose Action *according to ϵ-greedy method*:
            Generate random number $rand \in [0, 1]$
            if $rand < \epsilon$ **then**
                Sample a random action, $a_t \in \mathcal{A}$ // Explore
            else
                $a_t = \underset{a_t \in \mathcal{A}}{\text{argmax}} Q(s_t, a_t)$ // Exploit
        if a_t *is "replace"* **then**
            $\tau \leftarrow 0$
        else if a_t *is "repair"* **then**
            $\tau \leftarrow \max(\tau - 2, 0)$
        Calculate P_f for the $D(\tau)$ distribution
        $R(s_t, a_t) \leftarrow C_{a_t} + P_f C_{\mathsf{F}}$
        Observe next state s_{t+1} // the statistical moments of the $D(\tau)$ distribution
        Store tuple $(s_t, a_t, R(s_t, a_t), s_{t+1})$ in replay buffer // $s_t = \langle \texttt{stat.moments}, \tau \rangle$
        Sample batch of tuples $(s_i, a_i, R(s_i, a_i), s_{i+1})$ from replay buffer
        if s_{i+1} *is terminal state* **then**
            $y_i = R(s_i, a_i)$
        else
            $y_t = R(s_t, a_t) + \gamma\, Q\left(s_{t+1}, \arg\max Q\left(s_{t+1}, a_{t+1} \mid \theta\right) \mid \theta^-\right)$
        Update parameters θ according to: $\nabla_\theta L(\theta) \simeq \sum \left[\left(Q\left(s_i, a_i \mid \theta\right) - y_i\right) \nabla_\theta Q\left(s_i, a_i \mid \theta\right)\right]$
        if T_{update} **then**
            $\theta^- = \theta$

When resetting the deterioration rate, subsequently $D(0) = 0$ deterministically, which means that all the statistical moments of the damage distribution are zero.

Algorithm 14: Proximal Policy Optimization (PPO) - Continuous Toy Problem

Initialize policy (actor) network weights θ
Initialize value function (critic) network weights ϕ
for $episode \leftarrow 1$ **to** M **do**
    $s_t \leftarrow$ reset environment // $t \leftarrow 0$, $\tau \leftarrow 0$, initialize A, B
    for $n \leftarrow 1$ **to** N **do**
        $t \leftarrow t+1, \tau \leftarrow \tau+1$
        BMU for params A, B // procedure shown in Algorithm 12
        $\pi_\theta(a_t \mid s_t) \leftarrow$ **Actor Net** (s_t)
        $V_\phi(s_t) \leftarrow$ **Critic Net** (s_t)
        $a_t \leftarrow$ sample from $\pi_\theta(a_t \mid s_t)$
        if a_t *is "replace"* **then**
            $\tau \leftarrow 0$
        else if a_t *is "repair"* **then**
            $\tau \leftarrow \max(\tau - 2, 0)$
        Calculate P_f for the $D(\tau)$ distribution
        $R(s_t, a_t) \leftarrow C_{a_t} + P_f C_{\mathsf{F}}$
        Observe next state s_{t+1} // the statistical moments of the $D(\tau)$ distribution
        Store tuple $\left(s_t, a_t, \pi_\theta(a_t \mid s_t), V_\phi(s_t), R(s_t, a_t)\right)$ in $\mathcal{D}_k$ // $s_t = \langle \texttt{stat.moments}, \tau \rangle$
        $s_t \leftarrow s_{t+1}$
        if $t = T$ **or** $n = N$ **then**
            if $t = T$ **then**
                $V_\phi(s_{t+1}) \leftarrow 0$
                $s_t \leftarrow$ reset environment // $t \leftarrow 0$, $\tau \leftarrow 0$, initialize A, B
            else
                $V_\phi(s_{t+1}) \leftarrow$ **Critic Net**(s_t)
            Returns $\delta_t \leftarrow R(s_t, a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
            Advantages $A_t \leftarrow \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$
            Store δ_t, A_t in $\mathcal{D}_k$
    Train Agent $(\mathcal{D}_k)$

The "*Train Agent*" function is the one presented in Algorithm 11

4.2 Validation

4.2.1 Bayesian Inference

Prior to the coupling of BMU with DRL, each of these aspects has been tested in simple examples in order to eliminate possible errors in the final code of the integrated framework. Therefore, as far as Bayesian Inference is concerned, the deterioration model of the continuous case (described in section 4.1.3) will be used to perform the updating of parameters A, B (and subsequently $D(\tau) = A\tau^B$) using observations ω.

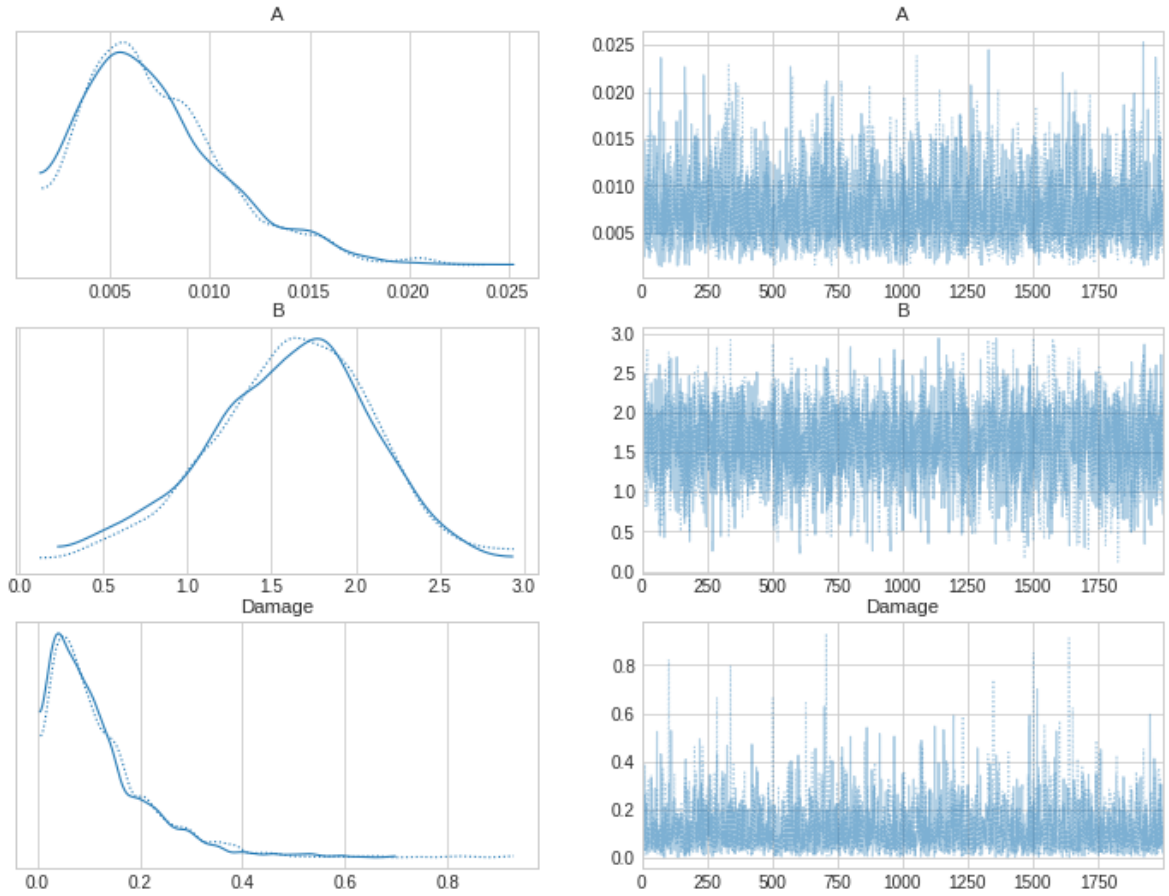
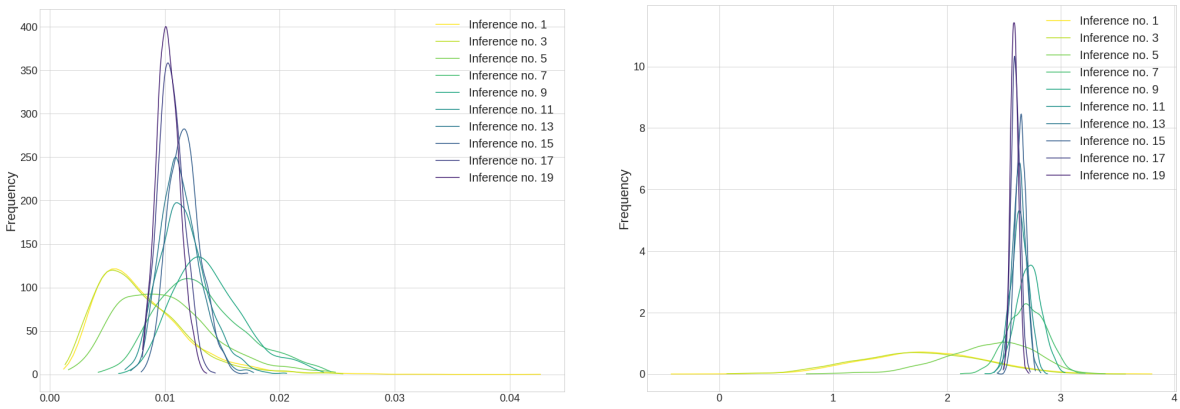


Figure 24: *Posterior distributions of parameters A, B after 5 iterations of Bayesian Inference and NUTS*

A typical example of NUTS is depicted in Figure 24, using two Markov chains and 4000 samples. What is more, the updating of the parameters' distribution through 20 iterations is illustrated in Figure 25, highlighting the effect of including observations in order to define more accurately the stochastic parameters of the deterioration model.



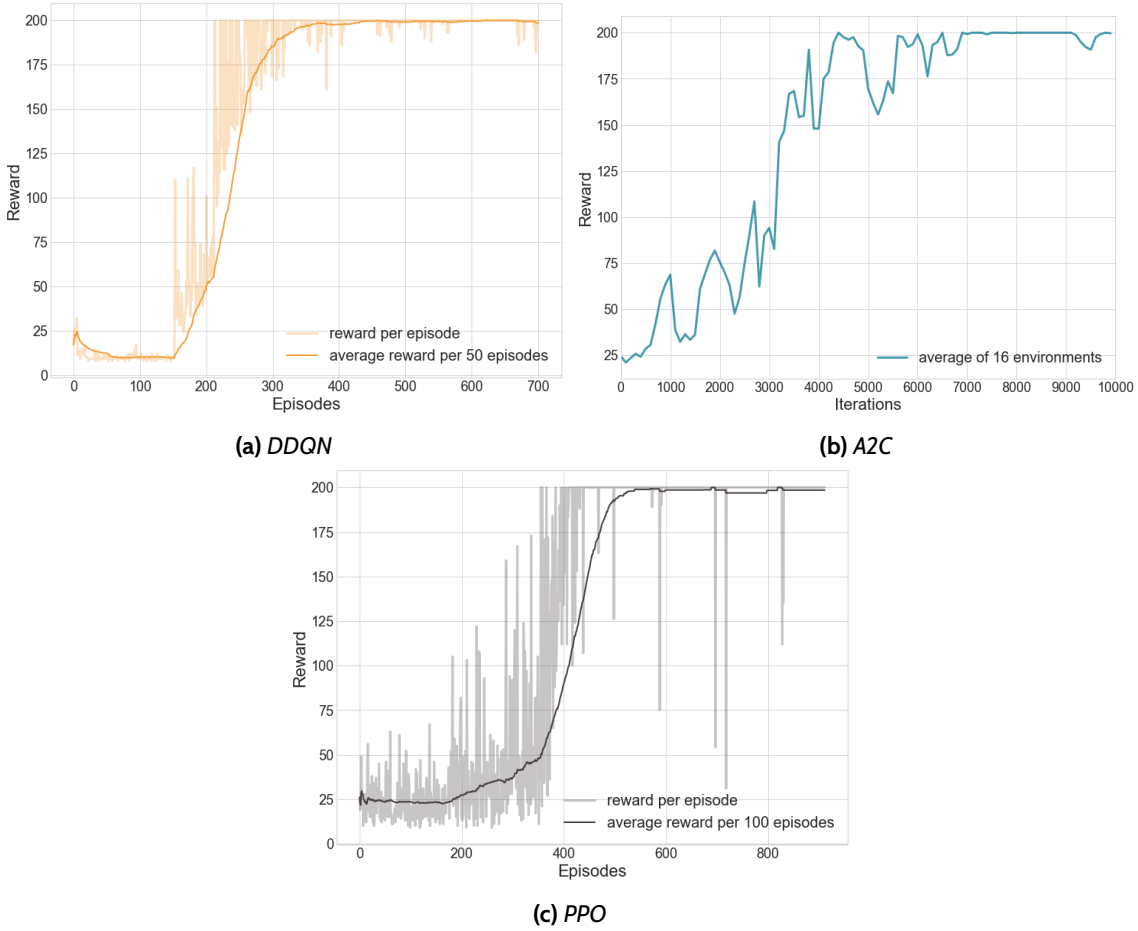
(a) *Posterior distribution of* A

(b) *Posterior distribution of* B

Figure 25: *Posterior distributions of parameters* A, B *during 20 iteration of Bayesian Inference and NUTS*

4.2.2 Deep Reinforcement Learning (DRL) algorithms

Before proceeding to more complicated cases, the three DRL algorithms, namely DDQN, A2C and PPO, will be tested and compared on the CartPole-v0[13] environment. The results, i.e. the rewards that the agent received over the episodes, during its training, are displayed in Figure 26.



(a) *DDQN*

(b) *A2C*

(c) *PPO*

Figure 26: *All three algorithms on the CartPole-v0 environment*

It should be noted that the Advantage Actor Critic (A2C) algorithm performed sufficiently in the CartPole-v0 environment this is why it is included in Figure 26. Unfortunately, this was not the case for the toy problem. Its stability is also ambiguous, since out of the three algorithms it was the slower one to reach the optimal CartPole-v0 reward.

[13] More information on the CartPole-v0 environment can be found here.

4.3 Benchmarking

Even if the algorithms converge to theoretically optimal values for the examined cases, the superiority of the proposed framework will be highlighted upon comparison with a benchmark value. Defining such a value is cumbersome and computationally expensive due to the stochastic nature of the problem (both discrete and continuous).

More often than not, a heuristic threshold based approach is being used, accounting for various control quantities, such as the maximum acceptable damage, the most beneficial periodic maintenance time interval or the maximum probability of failure allowed [55], [56], [57], [58]. For the current application, both in the discrete and in the continuous case, a fine grid of repair and replace damage values were tested, in order to determine when would be the most beneficial to intervene in the deterioration of the SDOF oscillator. For each combination of values, a plethora of episodes was ran, due to the high stochasticity. In the discrete case the variance was not significant, thus, only the expected value of the cost is included in the results. The obtained thresholds and costs are displayed in Table 7.

Table 7: *Benchmark maintenance thresholds and costs - Toy Problem*

	Optimal Thresholds				
	Repair	**Replace**	**Mean Cost**	**St. Dev.**	**Failure Damage**
Discrete	None	0.14	**21809.37**	-	0.5
Continuous	0.05	0.10	**80745.31**	22658.95	0.2

To further elaborate on the findings presented in Table 7, for the discrete case there was no scenario where it was beneficial to perform a repair, thus, only the replace value is relevant. It is worth mentioning that due to the parameter updating in a closed form that is possible in the discrete case, more decision steps were accounted for, thus, a higher damage failure value was considered.

4.4 Results

Prior to presenting the results for the toy problem, in order for them to be reproducible, the hyper-parameters used for both the discrete and continuous variations are displayed in Tables 8 and 9, for DDQN and PPO respectively.

Table 8: *DDQN hyperparameters - Toy problem*

Hyper-parameter	Discrete	Continuous
gamma, γ	0.99	0.99
learning rate	5.00E − 3	1.00E − 2
number of inner layers	2	2
size of inner layers	128	128
start epsilon, ϵ	1.0	1.0
batch size	64	128

Table 9: *PPO hyper-parameters - Toy problem*

Hyper-parameter	Discrete	Continuous
gamma, γ	0.99	0.99
clip ratio	0.2	0.1
lambda, λ	0.95	0.95
number of inner layers	2	2
size of inner layers	256	256
policy learning rate	1.00E − 4	1.00E − 3
value function learning rate	5.00E − 4	5.00E − 3

It should be noted that the number and the dimensions of the inner hidden layers were the same both for the actor and the critic network in the case of PPO. What is more, regarding the neural network activation function, for all networks of this project, and all layers, the Rectified Linear Unit (ReLU) function is chosen.

4.4.1 Discrete case

Combining the aforementioned aspects regarding the proposed framework and the toy problem, the DRL algorithms DDQN and PPO managed to yield optimal strategies that even outperform the benchmark solution. In Figure 27, the training of the agent is illustrated, by plotting the cost of the maintenance for a life cycle of 50 decision steps over the episodes ran during training.

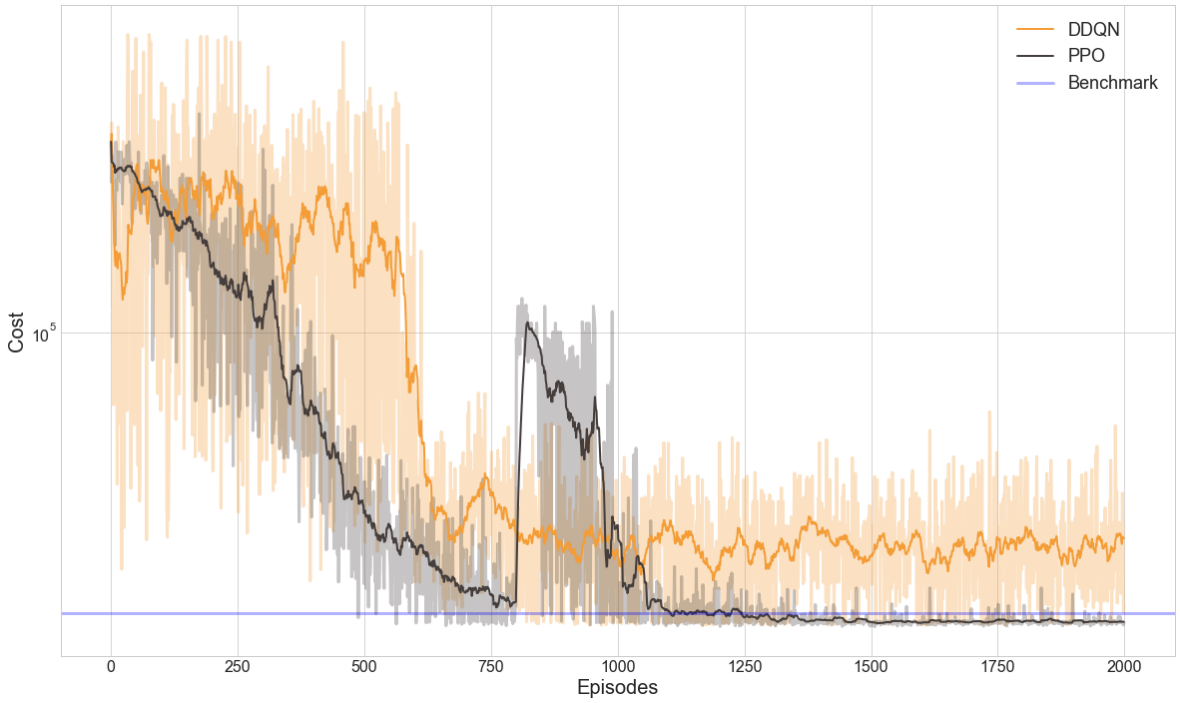


Figure 27: *DDQN and PPO on discrete SDOF environment*

It should be noted, that owing to the low complexity of this introductory application, it was probable that the DRL approach would not necessarily achieve a lower maintenance cost. However, it can be observed in Figure 27, that PPO performs slightly better. The superiority of PPO can be also complimented by its significantly lower variance, even though the environment is still stochastic. Another interesting finding for interpretation are the policies that were found by the agent. These policy realizations are plotted in Figure 28.

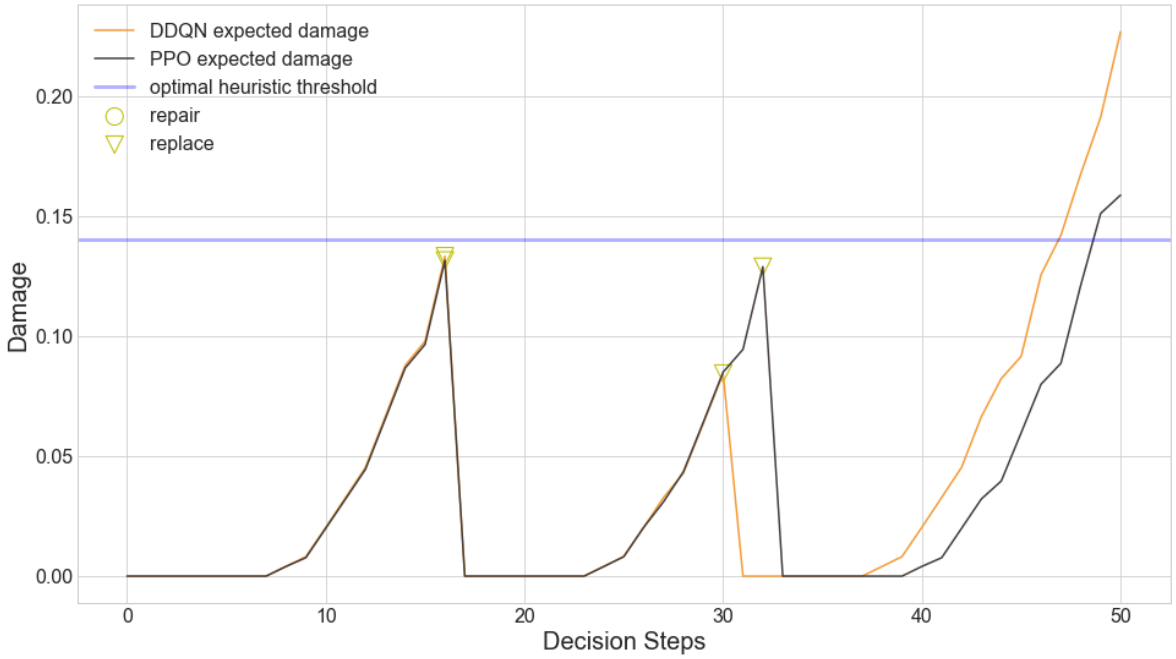


Figure 28: *DDQN and PPO policy realizations on discrete SDOF environment*

As it can be seen in the above plot, both the DDQN and PPO agents, managed to find a more suitable damage threshold to perform the replace action. The fact that DDQN fails to consistently choose when it is more beneficial to replace the component, as seen at decision step 30, leads to the slightly worse performance of the agent. On the other hand, PPO seems to be more stable and able to achieve lower maintenance costs, by performing a replace action when the damages reaches a value around 0.13.

Lastly, both the tested algorithms, avoid to perform a partial repair action, which is a fact backed up also by the benchmark runs. In this discrete setup of the toy problem, repairing the SDOF oscillator unarguably leads to higher maintenance costs.

4.4.2 Continuous case

Proceeding to the more accurate, from a modelling standpoint, continuous version of the toy problem, it is more evident that the proposed framework leads to optimal maintenance strategies in such stochastic environments. Prior to showcasing the performance of the tested algorithms, it should be mentioned that due to computational costs and time-consuming runs, the decision steps were reduced to 20 (instead of 50 for the discrete case), and the failure damage threshold was now assumed to be 0.2 (instead of 0.5 for the discrete case) as shown also in Table 3. The reduction in the damage threshold was made in order for the SDOF oscillator to deteriorate enough so as the cost linked to the probability of failure to be substantial. The training of the agent is plotted over the episodes in Figure 29.

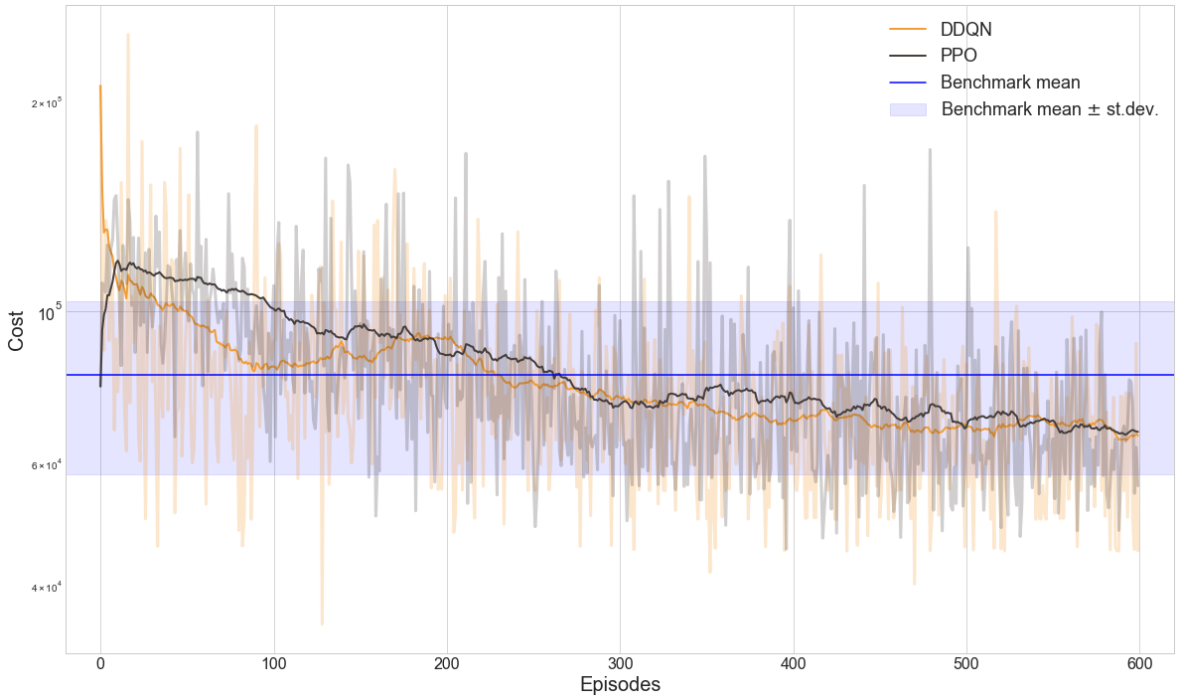


Figure 29: *DDQN and PPO on continuous SDOF environment*

It is evident that both the DDQN and the PPO algorithms outperform the benchmark approach. The exact details of this comparison are included in Table 10. Apart from the mean value and the standard deviation regarding the cost of each approach, the last column of the table contains the reduction in cost (as a percentage) compared with the traditional heuristic solution (benchmark). Additionally, one can observe that in the case of the benchmark, the variability in costs is significantly higher. This means that the stochasticity of the environment can lead to poor performance and higher costs, when following a threshold based policy, which is not the case when applying the proposed framework.

Table 10: *DRL algorithms' performance on continuous Toy Problem*

Last 50 Episodes			
DRL Algorithm	**Mean Cost**	**St. Dev. Cost**	**Cost Decrease**
Benchmark	80745.3	22659.0	-
DDQN	63776.0	14773.7	21.02%
PPO	64602.0	12992.6	19.99%

Owing to the extensively mentioned stochasticity of the environment, it is expected that each episode which was ran, will differ considerably from one another. Therefore, a single policy realization would not be a representative illustration, to fully understand the training of the

agent. Nevertheless, for the shake of comparison between the two algorithms and the benchmark thresholds, such realizations over the 20 decision steps are plotted in Figure 30.

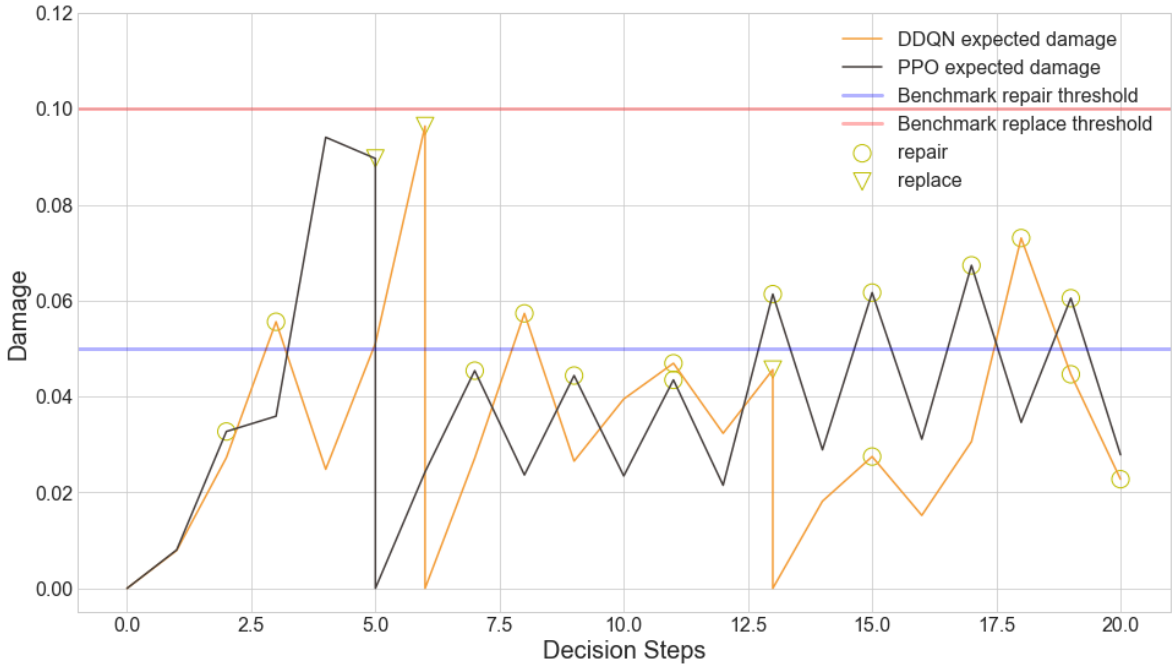


Figure 30: *DDQN and PPO policy realizations on continuous SDOF environment*

It is observed for both algorithms that the expected damage does not overcome the replace benchmark threshold of 0.10. Additionally the damage value when the agent chooses to perform a partial repair fluctuates around the heuristic benchmark value. More specifically, regarding PPO, when the damage increases in a more steep and unexpected way such as in decision step 4, the agent chooses to permit that and not proceed with a repair, letting the system deteriorate up to higher values and then performs a complete replacement. This is not strictly the case for DDQN, as it can be seen that for lower damage values like the one during decision step 13, the agent chose to perform a replacement action even though the damage was smaller than the repair benchmark threshold of 0.05. This constitutes an interesting finding, since both agents achieved almost identical costs, which is something that can be possible attributed to the high stochasticity of the corrosive environment.

More descriptive conclusion could be possibly drawn if more than one episodes, hence policies, were to be plotted. This is done for both DDQN and PPO in Figures 31 and 32 respectively.

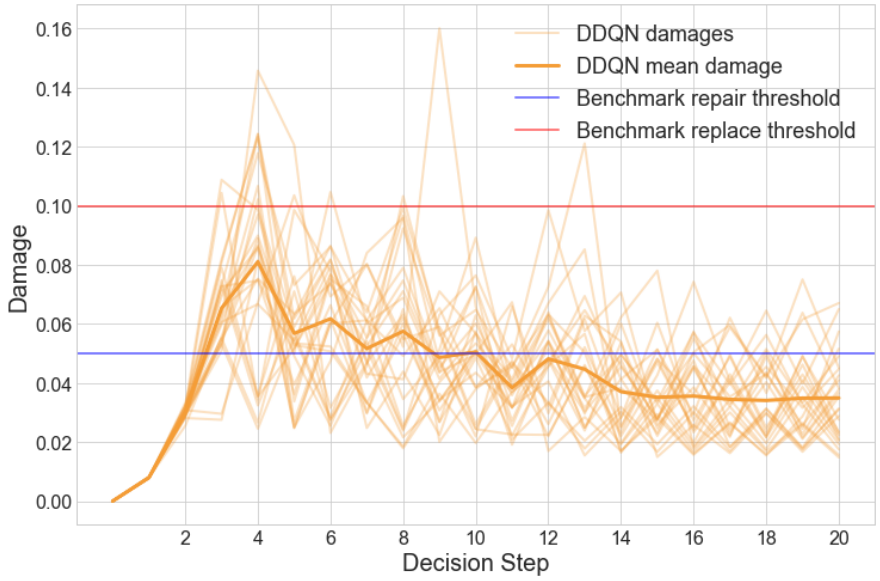


Figure 31: *Probability of failure for 50 policy realizations, for both DDQN and PPO*

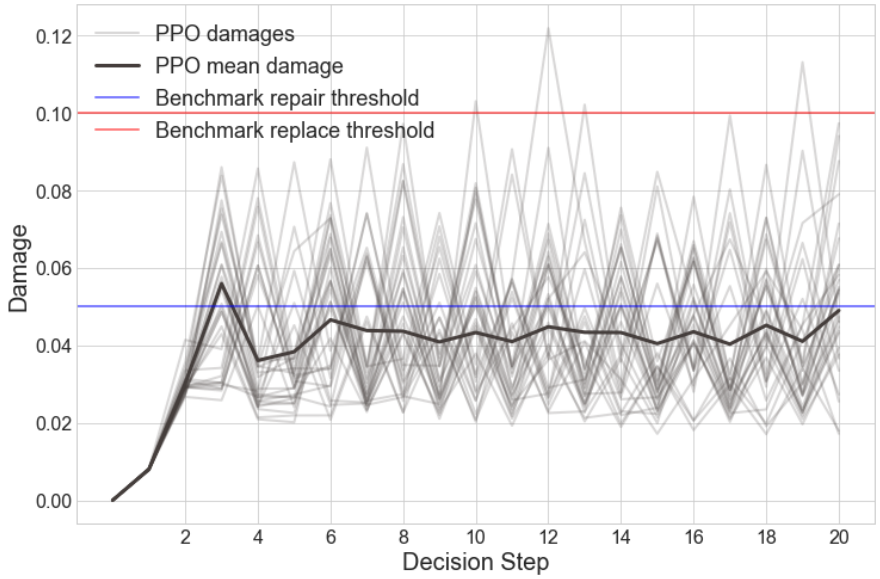


Figure 32: *Probability of failure for 50 policy realizations, for both DDQN and PPO*

In these plots, policy trends can be identified, highlighting once more the ability of the agent to diverge from the traditional heuristic actions and proceed in taking actions at unexpected stages of the deterioration. It should be mentioned that the plotted policies, even though they do not lead to the minimum of maintenance costs, they are all still smaller than the benchmark average one.

Elaborating further on the degree to which the obtained policies comply with the benchmark values, it can be stated that PPO chose actions in a considerably more consistent way compared to DDQN, with only limited policies passing the replace heuristic value, and the vast majority of the repair actions being performed for damages lower than the repair threshold. This is concluded from the steep peaks that have formed inside the band between 0.05 and 0.10, namely the repair and replace benchmark values. Even though these peaks could indicate a periodic pattern of maintenance, particularly replace ones, this is not the case, since they belong to different episodes. On the contrary, even though the DDQN agent restrict the damage mostly below the replace threshold, it is observed that the obtained policies are more stochastic, with many repair actions taking place even at times where the damages approaches 0.10, i.e. the replace heuristic value. An important issue, that is probably responsible for these differences among the two algorithms, is the way the agent chooses actions in each case. In DDQN the agent picks deterministically the action it considers the most beneficial, based solely on the action-state value functions $Q(a_t, s_t)$. On the other hand, the PPO agent, even if the damages has reached a worrying damage value, chooses the action based on a probability distribution, i.e. the policy $\pi(a_t \mid s_t)$ which makes every action, no matter how "good" or "bad" is, to still stand some chances of being picked.

Another interesting outcome of the proposed framework is the impact of the updating procedure, in case the "true" values of the parameters A, B are known. In Figures 33, 34, the evolution of A and B respectively, is plotted along the decision steps, for 9 different episodes.

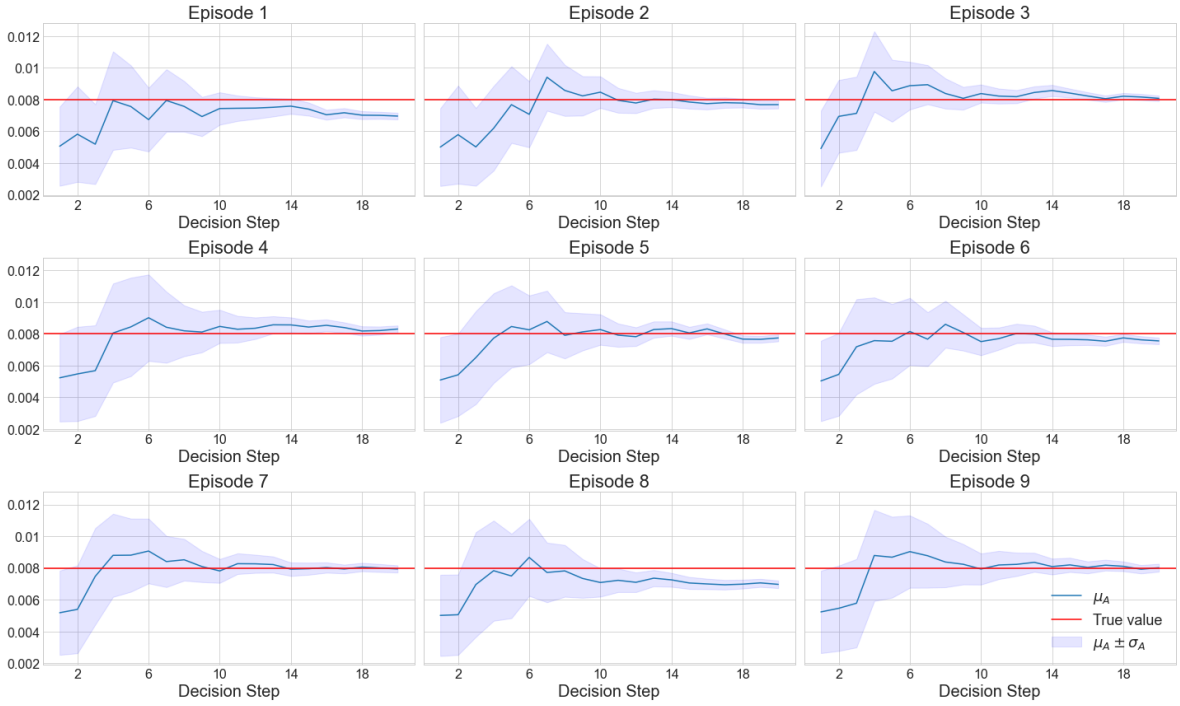


Figure 33: *Updating of parameter A for nine (9) of the episodes*

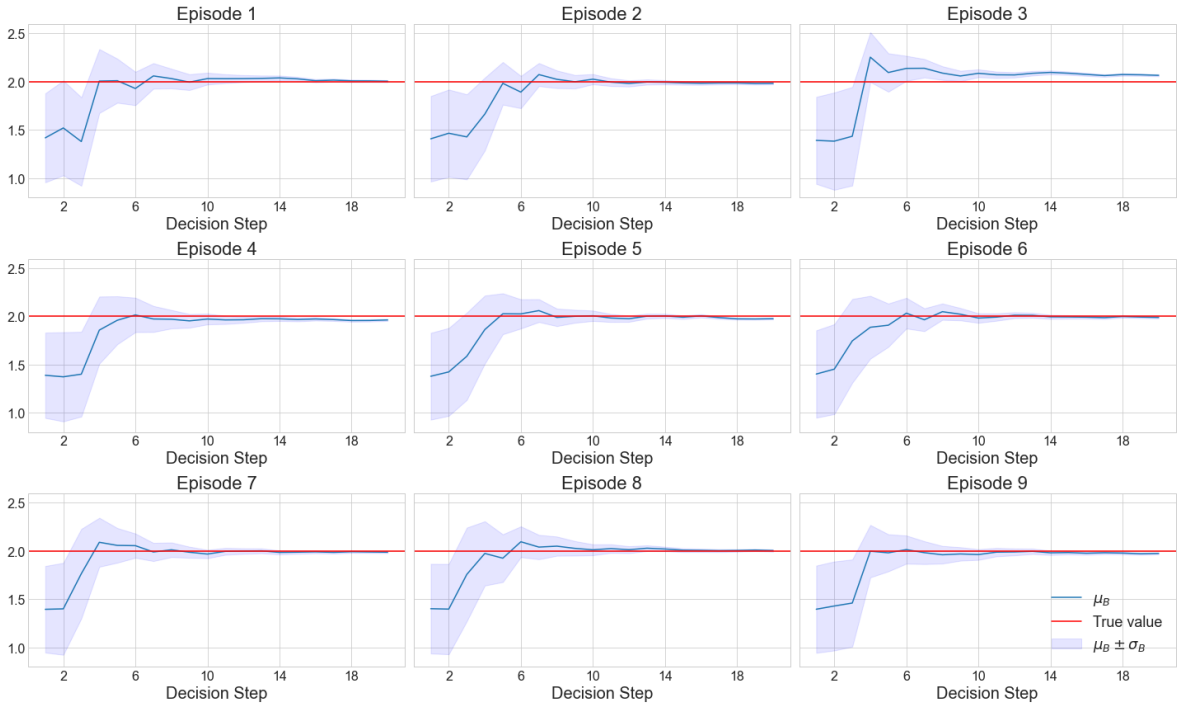
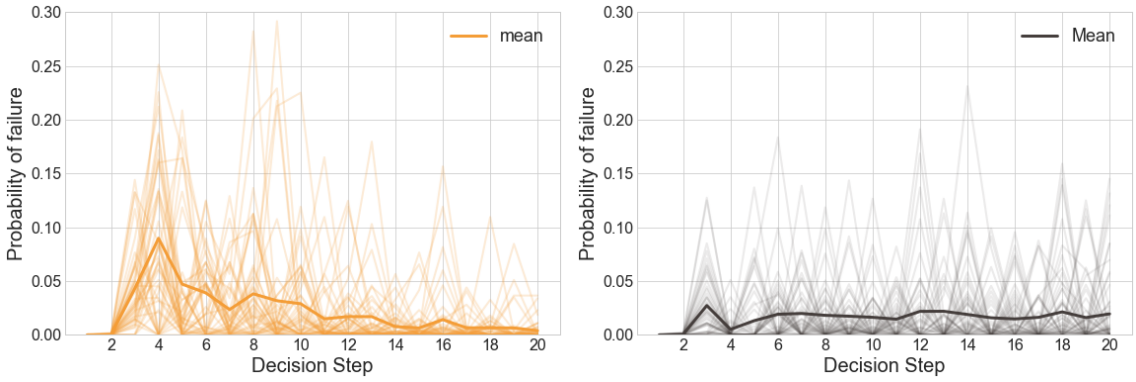


Figure 34: *Updating of parameter B for nine (9) of the episodes*

It can be safely concluded that incorporating more noisy observations, which have been generated using the assumed "true" values for A and B, i.e. 0.008 and 2.0, reduces the uncertainty, and the stochastic parameters indeed converge over time with an ever decreasing variance. It should be noted though that for limited cases the inferred value of the parameters seem to converge to a slightly offset value, such as in episodes 1, 8 in Figure 33, and in episode 3 in Figure 34. This anomaly could be justified by the limited amount of decision steps that was employed, expecting a better convergence to the expected "true" values if more updates were performed.

Lastly, since the cost linked to the risk of failure, and subsequently the probability of failure, have an important contribution to the total reward/cost of an episode, this probability is being plotted over the 20 decision steps for a plethora of episodes (50 in total) in Figure 35.



(a) *DDQN* **(b)** *PPO*

Figure 35: *Probability of failure for 50 policy realizations*

Similar conclusions that have already been drawn for the comparison between DDQN and PPO can be supported through the probability of failure plots, too. PPO appears to perform in a more controlled manner, not allowing the probability of failure, P_f, hence, the cost C_{risk} to grow excessively. Observing also the mean of these episodes, it can be deduced that the agent forces the risk of failure to stay approximately constant, especially for later steps (after the fifth one), and in total it does not allow P_f to overcome the value of 0.035. This is not the case for DDQN, as seen in Figure 35a, where especially for the early steps, the agent is not that strict, allowing even higher values for the cost associated with the risk of failure. Among the two algorithms, the maintenance strategies according to DDQN could be considered more reasonable, since taking early maintenance decision is counter-intuitive for a brand new engineering system. However, controlling the damage from an early stage, seems to work as well, leading to relatively stable results, as PPO showcases.

4.5 Conclusions

Concluding this chapter, a review of the obtained results for the toy problem will be made, while some summarizing comments and conclusions will be drawn.

Starting from the discrete case, due to the great simplifications in the modelling of the system's deterioration and the state and observation spaces, it was feasible to derive optimal maintenance strategies even with a heuristic damage threshold-based approach. Nonetheless, PPO managed to outperform the benchmark approach and arrived to a slightly better policy, showcasing the superiority of the proposed framework even for such trivial cases.

Moving to the more realistic continuous version of the toy problem, it can be safely concluded that the developed tool performed significantly better. Both DDQN and PPO managed to yield an optimal sequence of maintenance actions, which decreased the total cost over the system's lifetime b approximately 20% (the exact numbers/costs and the training of the agents can be found at Table 10 and Figure 29 respectively).

Unfortunately, the A2C algorithm was not able to perform adequately, this is why it is disregarded from the rest of the thesis. A possible reason for its poor performance can be its instability, especially in such a stochastic environment. After all, this is the main reason algorithms like TRPO and especially PPO were developed; to provide a more stable learning while still taking advantage of the benefits of a policy gradient algorithm [40].

5 Case Study

5.1 Problem Description

5.1.1 Modelling

Even though the results which were yielded for the toy problem confirm the capabilities of the proposed framework, its full potential will be better highlighted in the case of a more complicated engineering system. In particular, a statically indeterminate structure poses the perfect candidate, because it is often intractable for such systems to derive optimal maintenance strategies simply by using threshold-based heuristic approaches. The chosen structural system is a three storey two dimensional steel frame, which consists of linear elements, subjected to a lateral triangular load along its height. The exact geometry is illustrated in Figure 36.

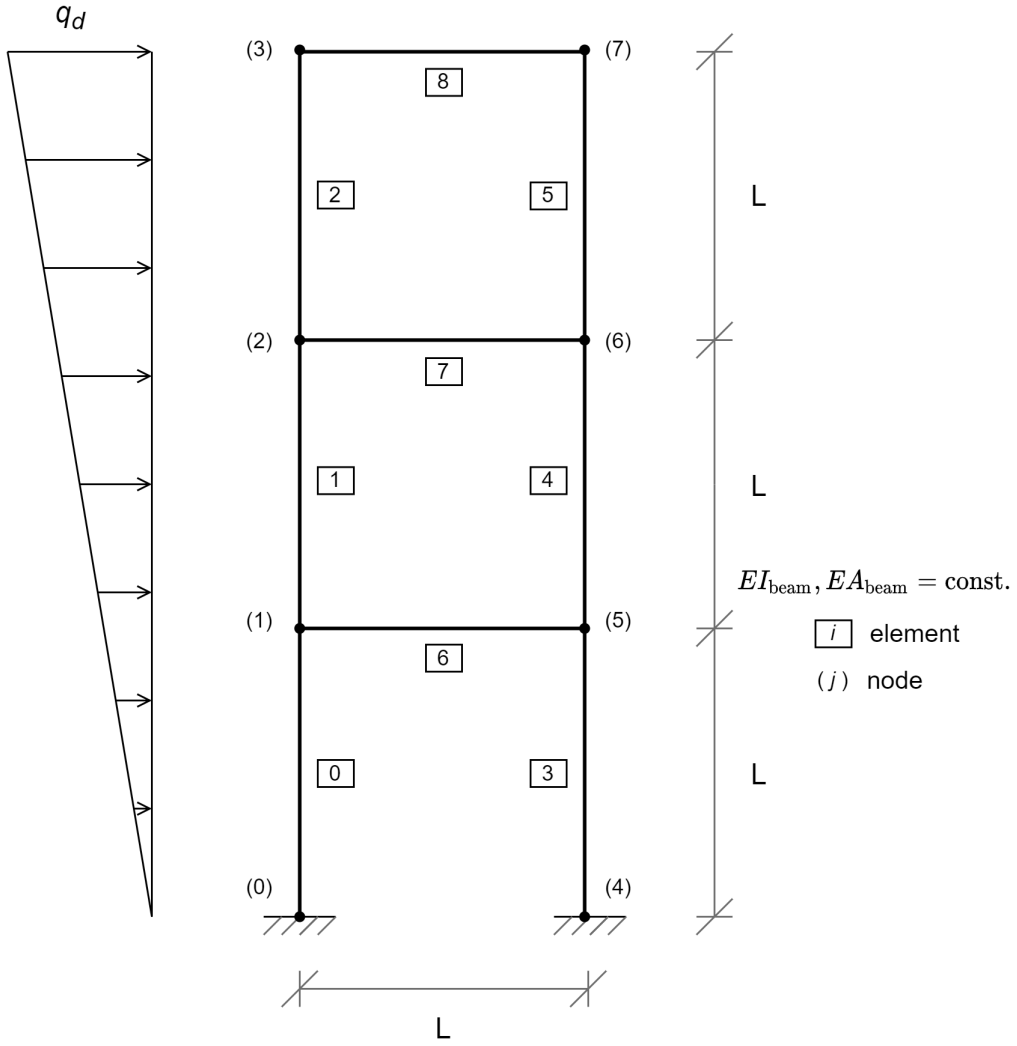
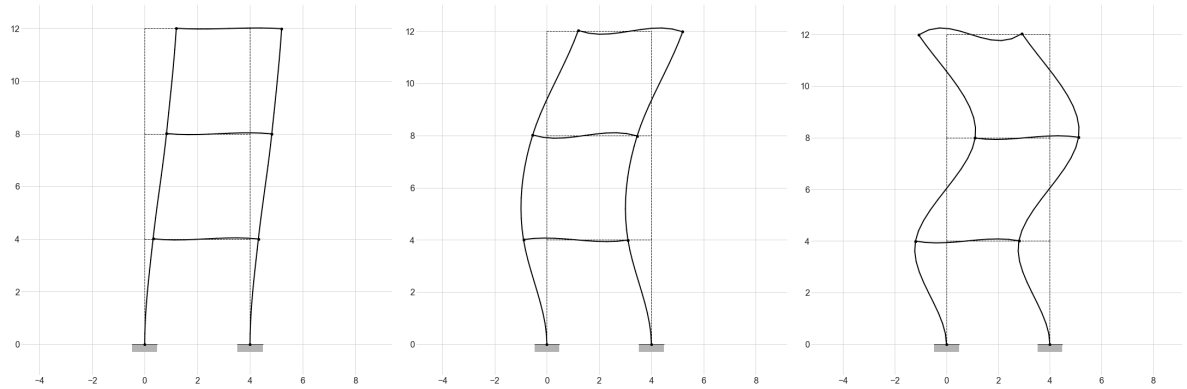


Figure 36: *Structural system employed for the Case Study*

The structure will be modelled using the Finite Element Method (FEM), and more specifically, both for the beams and the columns, linear Euler-Bernoulli beam elements will be used. Three Degrees of Freedom (DOFs) will be accounted for at the elements' nodes, i.e. two translational (horizontal and vertical) and one rotational, all defined in the displayed 2D plane. At this point it should be noted that the elements of interest regarding their degradation, are the columns. This is a realistic simplification, bearing in mind that the deterioration and the possible failure of the columns can lead to more significant consequences and possibly to a global failure of the frame. Hence, as seen also in Figure 36, both the axial and flexural stiffness of the beams are assumed to be constant.

In the same fashion as described in Section 4.1.1, noisy measurements, e.g. accelerations obtained through a monitoring system, are passed through an output-only OMA scheme, which yields modal data that are contaminated with additional noise, and are used as observations for the model updating procedure. In the current application, the first eigenmode is considered, meaning that a modal displacement will be observed for each of the eight (8) nodes, or to be more precise, for each of the six (6) nodes, since the two (2) bottom ones are fixed. This choice for the observed quantities serve for a better localization of the damage and the deterioration, which will eventually result in a more precise and beneficial maintenance strategy. For completeness reasons, the first three eigenmodes are illustrated in Figure 37.



(a) *Mode 1* **(b)** *Mode 2* **(c)** *Mode 3*

Figure 37: *First three eigenmodes of the frame*

It has already been mentioned that the reward, i.e. the maintenance cost, during each decision step, can be broken down into the cost of the taken action and the cost associated with the risk of failure (Equation 35). Therefore, in order to further elaborate on the concept of failure and quantify such a risk, and subsequently calculate the probability of failure, a Serviceability Limit State (SLS) check is employed. In particular, as stated in Clause 7.2.2 in EN 1993-1-1 [59] and the Dutch National Annex to EN 1990, cl. A1.4.3(7), the limit of the horizontal displacement of the top storey, i.e. the drift, is $u \leq H/500$, where H is the total height of the frame. Thus, for the structure at hand, failure has been reached if the drift of node (3) is greater than $H/500 = 12\,\mathrm{m}/500 = 0.024\,\mathrm{m} = 24\,\mathrm{mm}$. As far as the loading is concerned, a simple representation of seismic action is chosen. In particular, a

triangular static equivalent load of maximum magnitude, q_d, is accounted for, which serves as an approximation of the first eigenmode of the structure. This assumption will reduce significantly the computational cost that would be induced by a dynamic time history analysis. The calculation of the triangular load's maximum value is included in Appendix A.1.1. A more elaborate and realistic choice for the load (e.g. including the self weight of the transverse beams, etc) would not be beneficial for the scope of this project, since the assumed load case intervenes only in the calculation of the top storey drift and subsequently the probability and cost of failure. The DRL and BMU aspects of the problem are not affected by this modelling decision, which means that a simplified yet representative load will not affect the framework's accuracy.

The assumed material, cross-sections, dimensions and loads are summarized in Table 11.

Table 11: *Case study frame geometry and properties*

Beam cross section	IPE220
Column cross section	HEA300
Material	S355
L	4 m
q_d	3.6 kN/m

A simplified model in terms of discretization was chosen, i.e. one Finite Element (FE) per structural component, in view of reducing the computational cost. Owing to the relatively straightforward geometry, this decision does not cause any loss of accuracy, both for linear static and eigen- analyses.

5.1.2 Actions

In this multi-component application, a significant difference constitutes the fact that instead of a scalar action, there is an action *vector* $\underline{a_t}$, containing at each decision step, the different actions that will be performed at the same time on every component. This can have a serious impact on the dimension of the global action-space, since for n components and m different action, there are m^n possible action vectors in a combinatoric fashion. As a starting point, the same actions (3) presented in Section 4.1.1 and in Table 1 are accounted for, which means that for the six (6) components of the frame, there are $3^6 = 729$ possible action vectors.

Similarly to the toy problem, the costs for the actions are expressed relatively to one another, as stated in Table 2, with the failure cost this time being significantly bigger. In particular, for the SDOF oscillator, the failure cost was simply assumed twice as big compared to the replacement one, to account for the sudden nature of such an event and its consequences, but for the frame at hand, failure would mean a global collapse of the structure, so $C_\mathsf{F} = 6\, C_\mathsf{R}$[14]. The extra cost that would be

[14] six (6) is the number of the frame's components

considered for this event happening abruptly is assumed to be compensated by the fact that some of the relatively *undamaged* components could possibly be reused. Therefore, the global failure would cost the same amount as a complete replacement of all the components. The relative values of the costs are included in Table 12.

Table 12: *Rewards (costs) for the case study*

Description	Cost	Value	Factor
Component's total replacement	C_{R}	C_0 units	1
Component's partial repair	C_{M}	$0.5\,C_{\mathsf{R}}$	0.5
Structure's global failure	C_{F}	$6\,C_{\mathsf{R}}$	6
Risk of global failure	C_{risk}	$P_f\,C_F$	$6\,P_f$

5.1.3 Deterioration Model

As elaborated in Section 2.6, a common and efficient approach regarding the modelling of the structure's degradation is the use of a Gamma process. The damage $d(\tau)$ is defined as the ratio of the current, degraded, cross section area over the initial one, i.e. $A(\tau)/A_0$. It is assumed that the corrosion penetrates the steel cross section uniformly (radially) meaning that all parts of the cross-section are equally exposed to the corrosive environment. Denoting the width of the degradation layer as c, Figure 38 illustrates how the deterioration evolves on a cross-section scale.

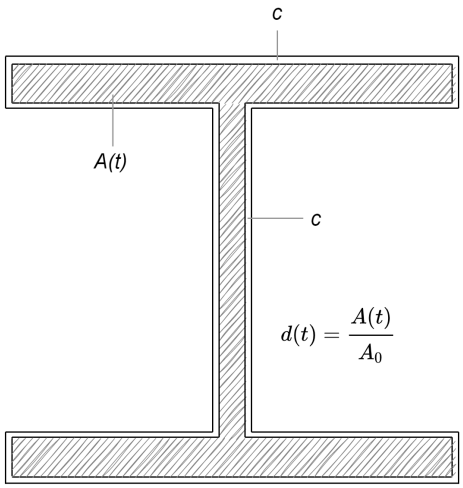


Figure 38: *Degradation of the IPE cross-section*

Since the examined case is a 2D problem, the properties of interest, that intervene both in the linear static and the eigen- analysis, are the axial stiffness, EA, and the flexural stiffness, EI. With Young's modulus E remaining constant, it is important to define the degradation of the moment of inertia, I, too. In Figure 39 the deterioration of the cross section's properties is being plotted

over the section loss percentage. The detailed calculations for the correlation between the two degradations (flexural and bending) are included in Appendix A.1.2.

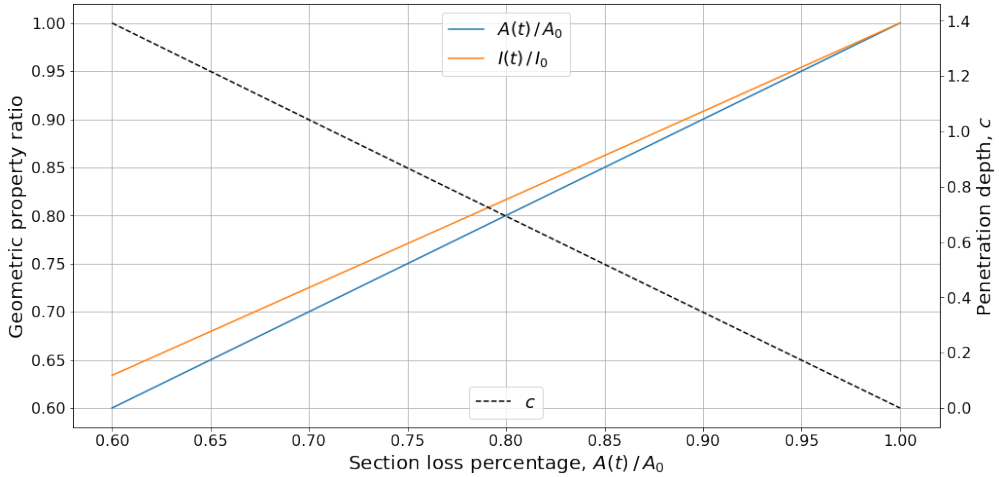


Figure 39: *Cross section properties deterioration*

Since the evolution of the deterioration follows a Gamma process, at every decision step the damage increment for each component is described by a Gamma distribution. Hence, as displayed also in Section 2.6:

$$\Delta D_i(\tau_i) \sim \text{Ga}(v(\tau_i) - v(\tau_i - 1), u) \quad \text{for} \quad i = 1, \dots 6 \tag{48}$$

where $v(\tau_i)$ is the shape of the Gamma distribution for the deterioration rate of the i component, and u is the scale factor, assumed to be constant for all components.

In order to calculate the scale factor, u, it is assumed that the corrosive environment is affecting the structural components in such a way that after 70 years, there is a mean section loss of 40% and a standard deviation Of 7.5%, as chosen also in [43]. This means that:

$$\left.\begin{aligned} \mathbb{E}\big(d(70)\big) &= \frac{v(70)}{u} = 0.40 \\ \text{Var}\big(d(70)\big) &= \frac{v(70)}{u^2} = 0.075 \end{aligned}\right\} \Rightarrow u = 71.11 \tag{49}$$

Regarding the shape of the Gamma distributions though, a similar calculation is not possible, since $v(\tau)$ is the term where the stochasticity of the deterioration is accounted for. In particular, it is defined as follows:

$$v(\tau) = A\tau^B \tag{50}$$

with A, B, being random variables. In a similar fashion with the toy problem, A, B constitute the uncertain parameters which are initially assumed (prior knowledge) and are being updated using

observations. Thus, during every decision step, a set of A_i, B_i values is being sampled from the distributions $\mathrm{P}(A), \mathrm{P}(B)$, for every component i, leading to a different Gamma distribution. Thus, the distribution of the total damage for each component at any given time/decision step is defined as the summation of all the Gamma distributions that describe the intermediate damage increments. It is known that the sum of gamma (v_i, u) random variables has a gamma $(\sum v_i, u)$ distribution, thus:

$$D_i^{\text{tot}} \sim \sum_{\tau=1}^{T} \mathrm{Ga}\left(\cdot \mid A_i^{\tau}\, \tau^{B_i^{\tau}} - A_i^{\tau}\,(\tau-1)^{B_i^{\tau}}, u\right) = \mathrm{Ga}\left(\cdot \mid \sum_{\tau=1}^{T} A_i^{\tau}\, \tau^{B_i^{\tau}} - A_i^{\tau}\,(\tau-1)^{B_i^{\tau}}, u\right) \tag{51}$$

where A_i^{τ}, B_i^{τ} are the values of A, B sampled for a deterioration rate τ for the i component[15].

The aforementioned details on the Gamma process, describe the so called *transition step* for the updating of the belief, hence, the probability distribution for the deterioration of the structure. The other aspect of such an updating is the *estimation step*, which is responsible for incorporating observations to improve the existing knowledge of the parameters of interest, i.e. A, B, with the use of a likelihood function. A graphical representation of the two, with the goal of defining the posterior distribution of the uncertain parameters is depicted in Figure 40.

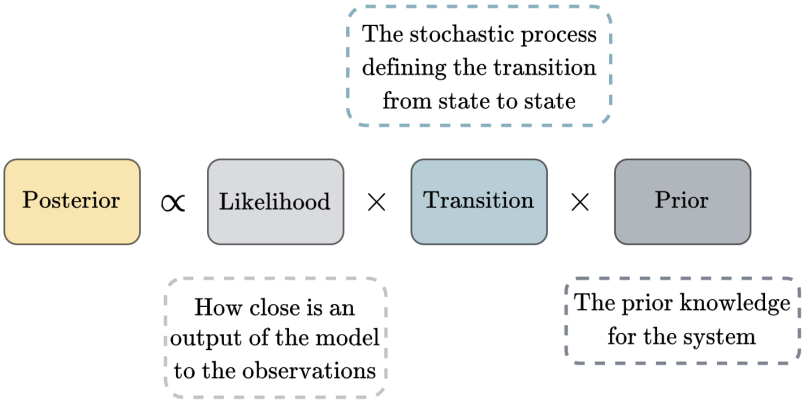


Figure 40: *Belief updating flowchart*

Denoting the system's parameters as $\theta = \langle A, B, \underline{D} \rangle$, and the observations, i.e. the modal displacements for the first eigenmode, as $\underline{O}$, the likelihood function is assumed to follow a normal distribution:

$$\mathrm{P}(\underline{O} \mid \underline{\theta}) = \left(\prod_{k=1}^{n} \frac{1}{\sigma_k \sqrt{2\pi}} \right) \cdot \exp\left[- \sum_{k=1}^{n} \frac{\left(O_k - \mathrm{M}(\underline{\theta})\right)^2}{2\sigma_k^2} \right] \tag{52}$$

In Equation 52, $\mathrm{M}(\underline{\theta})$ represents the modal displacements for the first eigenmode that are derived from the FE model, given the parameters θ, while σ_k is the standard deviation that describes the

[15] The Gamma distributions' shapes are dependent on the deterioration rate τ_i of each component, while the decision step counter, t, is global for the whole structure and it is evolving through unit increments until reaching the end of the episode, i.e. the time window of interest.

added noise from the OMA scheme. For each noisy measurement (modal displacement) m_k that is passed through the output-only OMA, the observation that is used in the likelihood function is $O_k \sim \mathcal{N}(m_k, \sigma_k)$, with $\sigma_k = m_k \cdot$ Noise.

A schematic representation of the updating process for the deteriorating structure at hand is demonstrated in Figure 41.

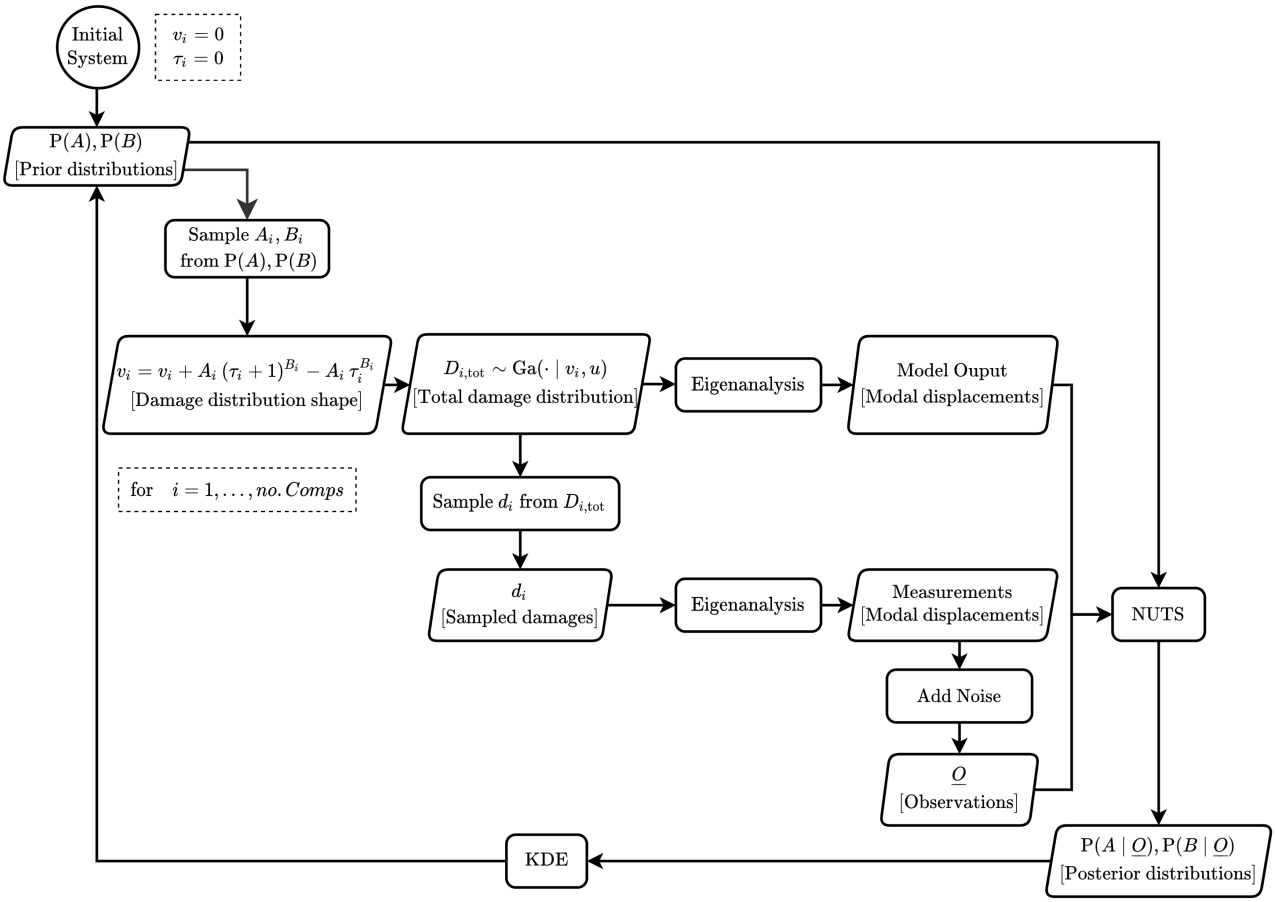


Figure 41: *Case Study Bayesian Inference*

The same procedure is presented in a more technical and formal way in Algorithm 15.

Algorithm 15: Deterioration model parameters updating - Case Study

DeteriorationParametersUpdating *(* $P(A), P(B)$*, FE model,* u, T *)*:
- Initialize $\mathrm{Ga}(\cdot)$ shapes $\underline{\upsilon}_{6x1} \leftarrow \underline{\mathbb{O}}_{6x1}$ `// 𝕆 is a vector of zeros`
- Initialize deterioration rates $\underline{\tau}_{6x1} \leftarrow \underline{\mathbb{O}}_{6x1}$
- **for** $t \leftarrow 1$ **to** T **do**
  - Increment deterioration rates $\underline{\tau} \leftarrow \underline{\tau} + \underline{\mathbb{I}}_{6x1}$ `// 𝕀 is a vector of ones`
  - $\underline{A}_{6x1}, \underline{B}_{6x1} \leftarrow$ sample from $\mathrm{P}(A), \mathrm{P}(B)$
  - $\underline{\upsilon} \leftarrow \underline{\upsilon} + \underline{A}\,(\underline{\tau} + 1)^{\underline{B}} - \underline{A}\,\underline{\tau}^{\underline{B}}$
  - Damage distribution of each component, $\underline{D}_{6x1} \sim \mathrm{Ga}(\cdot \mid \underline{\upsilon}, u)$
  - $\underline{d}_{6x1} \leftarrow$ sample from $\underline{D}$
  - $\underline{\Delta u}_{0\,6x1} \leftarrow$ **FE_eigenanalysis**$(\underline{d})$ `// mean modal displacements, for the first eigenmode`
  - Generate $\underline{\Delta u}_{\mathrm{obs}\,6x1} \leftarrow \mathcal{N}(\underline{\Delta u}_0,$ Noise$)$
  - **NUTS** *(* $P(A), P(B), \underline{\Delta u}_{obs}$ *)*:
    - **Output:** $\mathrm{P}(A \mid \underline{\Delta u}_{\mathrm{obs}}), \mathrm{P}(B \mid \underline{\Delta u}_{\mathrm{obs}})$
  - Choose action, $\underline{a}_{t\,6x1}$ `// according to the PPO agent`
  - **for** $i \leftarrow 1$ **to** 6 **do**
    - **if** a_t^i *is "repair"* **then**
      - $\tau_i \leftarrow \max(0, \tau_i - 2)$
    - **else if** a_t^i *is "replace"* **then**
      - $\tau_i \leftarrow 0$
      - $\upsilon_i \leftarrow 0$
  - $\mathrm{P}(A), \mathrm{P}(B) \leftarrow \mathrm{P}(A \mid \underline{\Delta u}_{\mathrm{obs}}), \mathrm{P}(B \mid \underline{\Delta u}_{\mathrm{obs}})$ `// posteriors become priors through KDE`

The dimensions of all the vectors are specified only at their first occurrence, i.e. $6x1$, and they always refer to the number of the structure's components

The (starting) values for the parameters of the model's deterioration are summarized in Table 13.

Table 13: *Case study input data*

Quantity	**Value**	**Units**	
Replace cost, C_R	10000	[-]	
Noise	0.1	[-]	
Failure drift	24	[mm]	
Parameter	**Distribution**	**Mean**	**CV**
A	Lognormal	0.1	0.5
B	Normal	1.8	0.2

5.1.4 Probability of failure

An important issue to be tackled is the calculation of the probability of failure, P_f, i.e. the drift of the top storey being $\geq 24\,\mathrm{mm}$, given the probability distributions of the components' damages. An MC sampling would be the ideal solution in terms of accuracy, however it demands a significant amount of computational time. Thus, FORM is chosen, and in particular, a geometric interpretation of it. To be more precise, as it has been thoroughly explained in [60], denoting $u_F = 24\,\mathrm{mm}$ i.e. the failure drift, and $\mu(\underline{D})$ as the drift of the frame given the damage distributions of the components, $\underline{D}$, the Limit State Surface (LSS)[16]is defined as:

$$M(\underline{D}) = u_F - \mu(\underline{D}) = 0 \tag{53}$$

The LSS separates the safe region, where $M(\underline{D}) > 0$, from the failure region, where $M(\underline{D}) < 0$, of the parameter space. The failure probability, P_f can be expressed as the integral over the domain $M(\underline{D}) < 0$:

$$P_f = \mathrm{P}(M(\underline{D}) \leq 0) = \int_{M(\underline{D})\leq 0} \mathrm{P}(\underline{D})\mathrm{d}\underline{D} \tag{54}$$

with $\mathrm{P}(\underline{D})$ being the joint PDF for the uncertain components' damages $\underline{D}$.

A computationally inexpensive way to calculate the integral of Equation 54 is through a geometric optimization analysis. The two necessary steps to do so, are:

1. Transform the uncertain variables, i.e. the damages, into independent normal basic variables $\underline{U}$ [17].

[16] In 2D problems, instead of a surface, there is the Limit State Function (LSF)

[17] The most popular methods for this task are the Rosenblatt and the Nataf transformations

2. Compute the minimum distance β, the so-called *reliability index*, of the LSS from the origin of the standard coordinate system $\underline{U}$ as displayed in Figure 42.

The point closest to the origin, U^*, is referred to as the design point, and is the point with the highest joint density on the LSS, meaning that it corresponds to the most probable combination of damages for the structure to fail.

The probability of failure can now be computed as:

$$P_f = \int_{M(\underline{U})\leq 0} \mathsf{P}(\underline{U})\mathrm{d}\underline{U} \approx 1 - \Phi(\beta) = \Phi(-\beta) \tag{55}$$

where Φ is the Cumulative Distribution Function (CDF) of a normally distributed random variable with zero mean and unit variance.

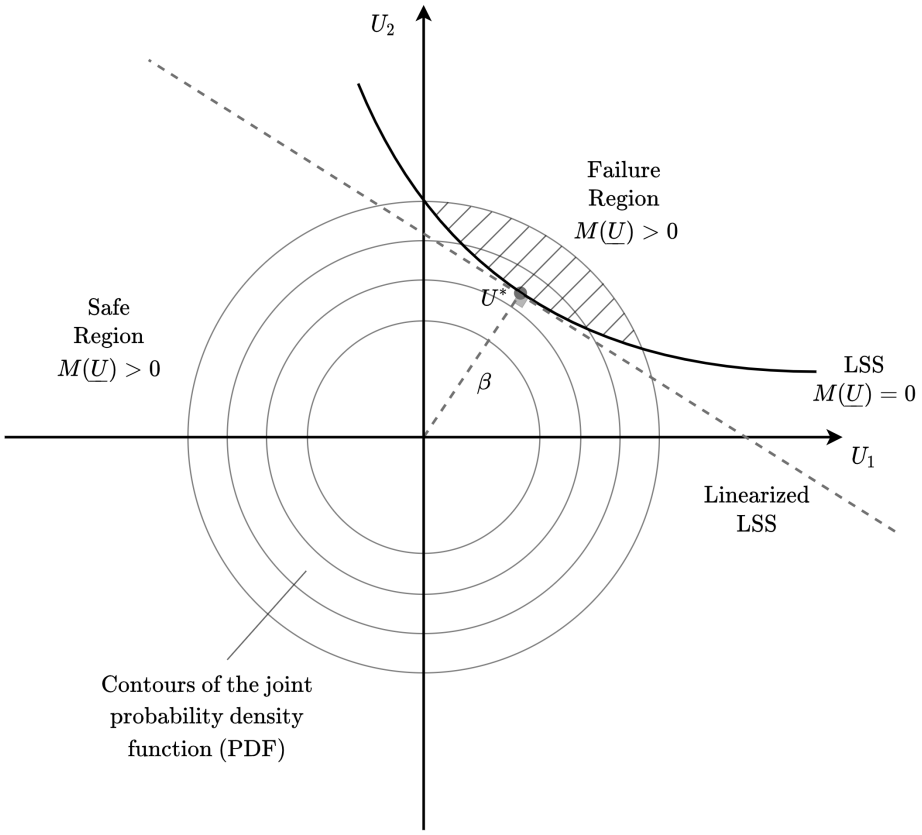


Figure 42: *Linearisation of the LSS $M(\underline{U}) = 0$ at the design point U^* in the uncorrelated standard normal random variables $\underline{U}$ space* [18]

[18] For clarity reasons, only two (2) standard normal variables are included in Figure 42. For the case at hand, there will be a 6-dimensional space and subsequently, 6-dimensional hyperplane as an LSS.

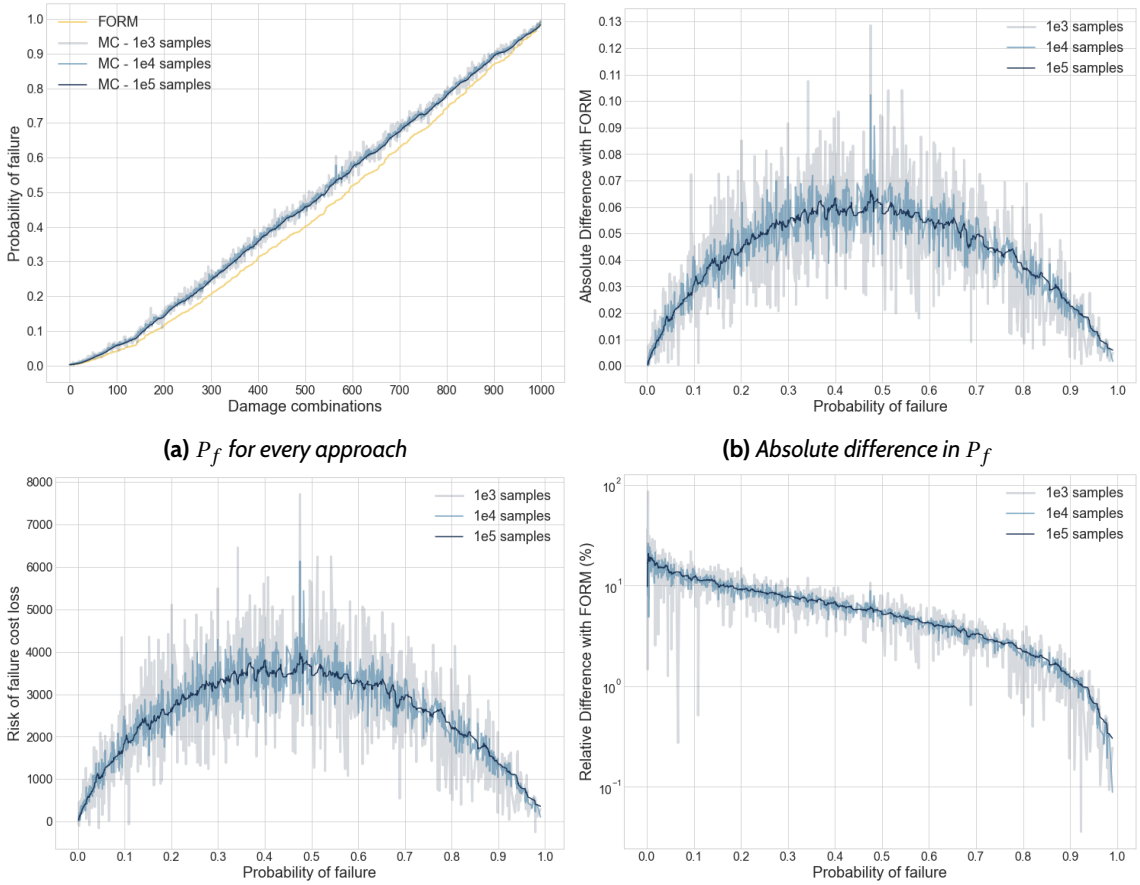
The step-by-step procedure followed to apply FORM is exhibited in Algorithm 16.

Algorithm 16: First Order Reliability Method (FORM) geometric interpretation

FORM *(Damage distributions $\underline{D}_{6x1}$, FE model, failure threshold u_F)*:
    Transform $\underline{D}$ into standard normal variables, $\underline{U}_{6x1}$
    $\mu(\underline{D}) \leftarrow$ **FE_linear_static_analysis**$(\underline{D})$ `// FE model's drift`
    $M(\underline{D}) = u_F - \mu(\underline{D}) = 0$ `// Limit State Surface (LSS)`
    Transform $M(\underline{D})$ to the normal space, $M(\underline{U})$
    $\beta \leftarrow$ min distance between LSS and the origin of $\underline{U}$ coord system
    $P_f \leftarrow \Phi(-\beta)$

This procedure is not the exact function applied by the proposed framework, but the parameter updating in the absence of the DRL aspect of the tool.

Even though FORM is significantly lighter from a computational time point of view, it relies on the linearization of the LSS to find the design point, U^*, which can lead to inaccuracies in highly non-linear problems. This is why, for the current application, which itself is not a linear problem, FORM's performance needs to be checked, through a comparison with a brute force MC sampling approach. This comparison, as displayed in Figure 43, is made for three different amounts of samples, and for a 1000 different damage combinations and P_f's.



(a) *P_f for every approach*

(b) *Absolute difference in P_f*

(c) *Absolute difference in C_{risk}*

(d) *Relative difference (%)*

Figure 43: *Comparison between FORM and MC*

In Figure 43a, the probability of failure P_f is plotted for different damage combinations, having applied all different approaches, meaning FORM and MC with 3 levels of samples. The results have been sorted and plotted in an ascending order of P_f for a clearer representation. It can be observed that FORM is consistently underestimating P_f, especially for values that are neither too big nor too small, i.e. approximately in the range of $0.1 - 0.9$. In the next two plots, i.e. Figures 43b and 43c, it is highlighted that the absolute difference between the two approaches is not significant, especially again for either too small or too big values of P_f. What is more, it is again confirmed that higher values of P_f are yielded with MC sampling, which results also in higher risk of failure costs as seen in Figure 43c. Lastly, the relative difference between the two methods, expressed as a percentage, is being plotted in logarithmic scale in Figure 43d.

The accuracy of FORM constitutes an important modelling aspect to be considered. Nevertheless, since it is capable of capturing small P_f's and it will be employed both for the benchmark solution as well as the proposed framework, it is adequate for the time being, leaving some space for future improvements.

5.2 Framework

Regarding the DRL aspect of the framework, not all of the algorithms examined in the current project are suitable for such an application. Owing to the multiple components, and the immense amount of possible action vectors, DDQN would not perform efficiently for this case. As already been stated in the existing literature, DDQN requires discrete action spaces, and the more the actions, the more difficult it is for the agent to arrive to optimal strategies. This is not the case with actor-critic algorithms, which based on the given state compute the probability distribution of the actions as an output, instead of the action-state value function. Thus, assuming that the actions of the the system's components are conditionally independent, the policy derived from an actor network, $\pi_\theta(\underline{a_t} \mid s_t)$, can be decomposed and expressed as the product of multiple policies which would refer to each component individually, $a_t^1, a_t^2, \ldots$ instead of the full action vector, $\underline{a_t}$.

$$\pi_\theta(\underline{a_t} \mid s_t) = \pi_\theta(a_t^1 \mid s_t) \cdot \pi_\theta(a_t^2 \mid s_t) \ldots \pi_\theta(a_t^6 \mid s_t) = \prod_{i=1}^{6} \pi_\theta(a_t^i \mid s_t) \tag{56}$$

or,

$$\log\big(\pi_\theta(\underline{a_t} \mid s_t)\big) = \sum_{i=1}^{6} \log\big(\pi_\theta(a_t^i \mid s_t)\big) \tag{57}$$

Therefore, the output layer of the actor network needs to have only $3 \times 6 = 18$ neurons, i.e. the probability of taking each action for each component. This means that every three output probabilities are summing up to one (1). A schematic representation of both the actor and the critic DNNs, for PPO [19] is displayed in Figure 44.

[19] Due to the poor performance of A2C in both the discrete and continuous versions of the toy problem, it is not going to be tested for the case study, even though it could have handle the fact of multiple components, for it being an actor-critic algorithm.

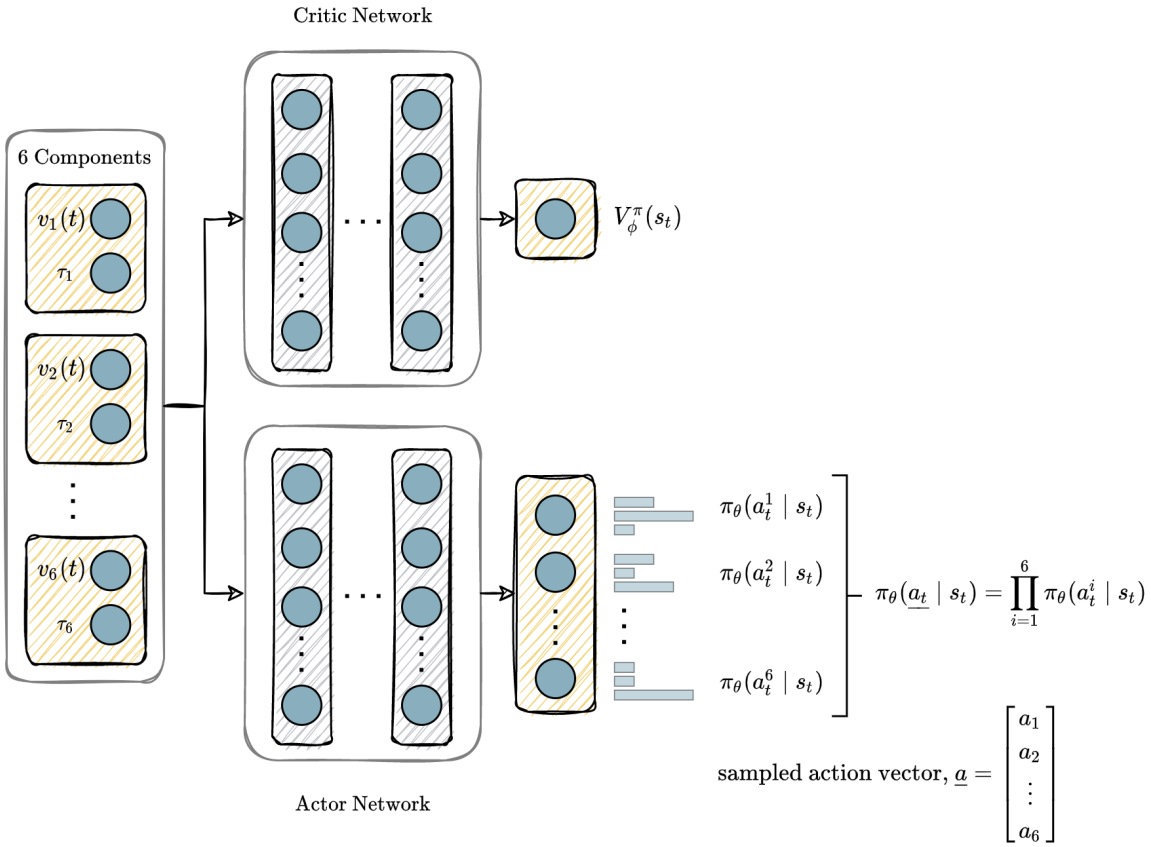


Figure 44: *PPO architecture - Centralized states and actions*

In the depicted architecture, instead of each component having its own agent, hence its own independent network, there is only one *centralized* actor with shared parameters θ for all components. As elaborated also in [43] with DCMAC, using such a network means that every agent is aware of all other agents' states, by getting as input the entire system state s_t, while being affected implicitly by their actions, too, through the common network weights θ.

Various alternatives regarding the network architecture are presented as part of the future work in Section 6.3.

Having elaborated on every aspect and sub-routine of the proposed framework, a summary of the complete procedure is illustrated in Figure 45.

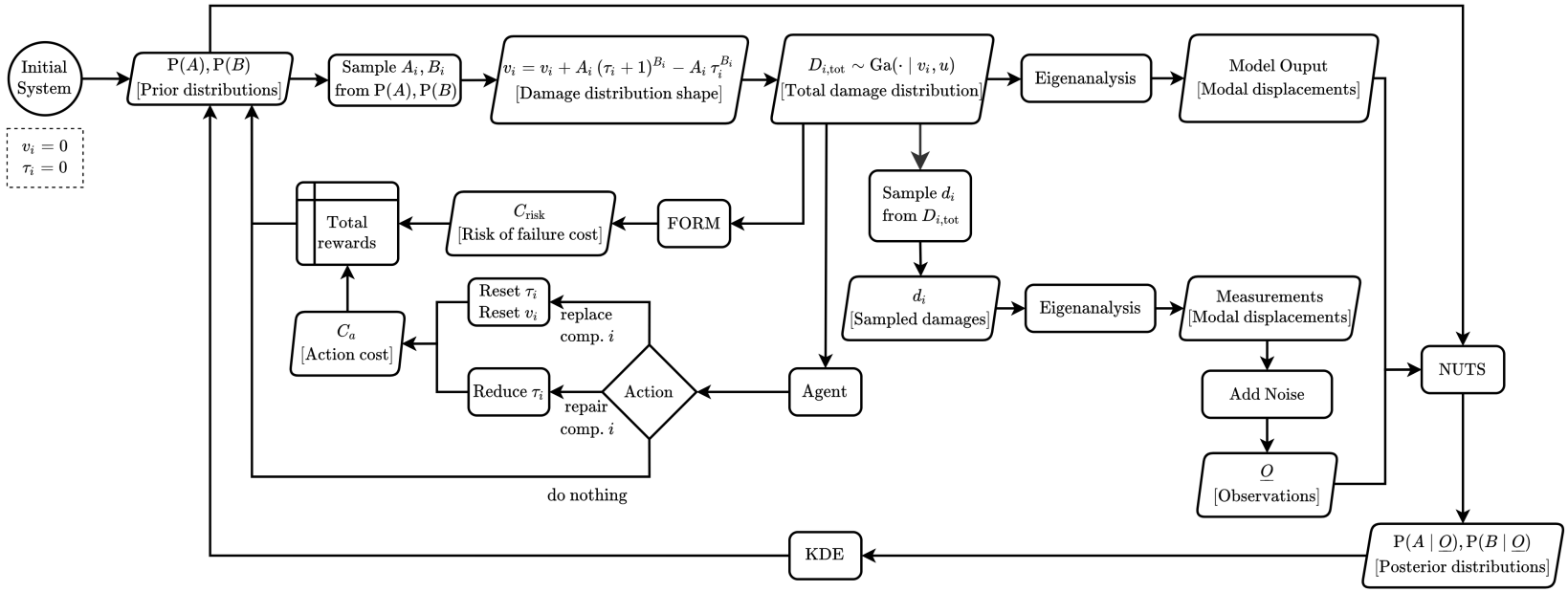


Figure 45: *Case Study complete flowchart*

Taking into account the DRL aspect of the problem, and the steps included in the training of the PPO agent, the developed tool is presented in a thorough and more formal way in Algorithm 17.

Algorithm 17: Proximal Policy Optimization (PPO) - Case Study

Initialize policy (actor) network weights θ
Initialize value function (critic) network weights ϕ
for $episode = 1$ **to** M **do**
    $s_t \leftarrow$ reset environment // initialize A, B, $\underline{\tau}_{6x1} \leftarrow \underline{\mathbb{O}}_{6x1}$, $\underline{\upsilon}_{6x1} \leftarrow \underline{\mathbb{O}}_{6x1}$
    for $t \leftarrow 1$ **to** T **do**
        $t \leftarrow t+1$, $\underline{\tau} \leftarrow \underline{\tau} + \underline{\mathbb{I}}$
        BMU for params A, B // procedure shown in Algorithm 15
        $\pi_\theta(\underline{a}_t \mid s_t) \leftarrow$ **Actor Net** (s_t) // $\pi_\theta(\underline{a}_t \mid s_t) = \pi_\theta(a_t^1 \mid s_t) \cdot \pi_\theta(a_t^2 \mid s_t) \ldots \pi_\theta(a_t^6 \mid s_t)$
        $V_\phi(s_t) \leftarrow$ **Critic Net** (s_t)
        $\underline{a}_t \leftarrow$ sample from $\pi_\theta(\underline{a}_t \mid s_t)$ // a_t^1 from $\underline{a}_t[0:3]$, a_t^2 from $\underline{a}_t[3:6] \ldots$, a_t^6 from $\underline{a}_t[15:18]$
        for $i \leftarrow 1$ **to** 6 **do**
            if a_t^i *is "replace"* **then**
                $\tau_i \leftarrow 0$
            else if a_t^i *is "repair"* **then**
                $\tau_i \leftarrow \max(\tau_i - 2, 0)$
        $P_f \leftarrow$ **FORM** $(\underline{\upsilon})$ // the shapes of the Gamma (damage) distributions are passed as an input
        $C_{a_t} \leftarrow numReplaces \times C_{\mathsf{R}} + numRepairs \times C_{\mathsf{M}}$
        $R(s_t, \underline{a}_t) \leftarrow C_{a_t} + P_f C_{\mathsf{F}}$
        Observe next state s_{t+1} // $s_{t+1} = \langle$shapes $\underline{\upsilon}$, deterioration rates $\underline{\tau}\rangle$
        Store tuple $\left(s_t, a_t, \pi_\theta(a_t \mid s_t), V_\phi(s_t), R(s_t, a_t)\right)$ in $\mathcal{D}_k$ // $s_t = \langle \underline{\upsilon}, \underline{\tau} \rangle$
        $s_t \leftarrow s_{t+1}$
        if $t = T$ **or** $n = N$ **then**
            if $t = T$ **then**
                $V_\phi(s_{t+1}) \leftarrow 0$
                $s_t \leftarrow$ reset environment // initialize A, B, $\underline{\tau} \leftarrow \underline{\mathbb{O}}$, $\underline{\upsilon} \leftarrow \underline{\mathbb{O}}$
            else
                $V_\phi(s_{t+1}) \leftarrow$ **Critic Net**(s_t)
            Returns $\delta_t \leftarrow R(s_t, \underline{a}_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
            Advantages $A_t \leftarrow \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}$
            Store δ_t, A_t in $\mathcal{D}_k$
    Train Agent $(\mathcal{D}_k)$

The indexing and slicing of vectors follows `Python` notation, i.e. `0` is the starting index and the upper bound is exclusive.

The steps needed to train the PPO agent for the case study, i.e. the function "*Train Agent*" are demonstrated in Algorithm 18.

Algorithm 18: Proximal Policy Optimization (PPO) agent training - Case study

Train Agent $(\mathcal{D}_k)$**:**

Update parameters ϕ, using the Critic cost function:

$$L^{\mathsf{VF}}(\phi) = \sum_{t=1}^{T} \left(V_\phi(s_t) - \delta_t \right)^2$$

Update parameters θ, using the Actor loss function:

$$L^{\mathsf{CLIP}}(\theta) = \sum_{t=1}^{T} \min \left(\frac{\pi_\theta\left(\underline{a}_t \mid s_t\right)}{\pi_{\theta_{\mathsf{old}}}\left(\underline{a}_t \mid s_t\right)} A_t\left(s_t, \underline{a}_t\right), \mathsf{clip}\left(\frac{\pi_\theta\left(\underline{a}_t \mid s_t\right)}{\pi_{\theta_{\mathsf{old}}}\left(\underline{a}_t \mid s_t\right)}, 1-\varepsilon, 1+\varepsilon \right) A_t \right)$$

via minibatch stochastic gradient ascent with Adam

5.3 Benchmarking

For the sake of comparison and evaluation of the proposed framework, a benchmark approach needs to be applied on the case study, setting the threshold (in terms of cost) that the PPO agent will try to surpass.

As mentioned also for the toy problem in Section 4.3, usually a heuristic approach is chosen, which indicates when it is more beneficial to perform a maintenance action based on a variety of metrics. Common options are the maximum damage allowed (per component), the maximum probability of failure of the structure as a whole, or even a time threshold which would specify every how many decision steps (e.g. years) a maintenance action should be performed.

Although, for the SDOF system, the control quantity would not lead to considerable differences, it is expected, that for a multi-component system, monitoring the deterioration of each component separately can lead in a more efficient maintenance strategy and life-cycle cost. Nevertheless, both the optimal damage threshold and the optimal maintenance time interval will be sought, and ultimately the yielded results of these two heuristic approaches, namely Condition-Based Maintenance (CBM) benchmark and Time-Based Maintenance (TBM) benchmark, will be compared with the ones of the proposed methodology.

Regarding the CBM benchmark, a fine grid of repair and replace thresholds is created, in order to check which combination would yield the minimum maintenance cost. In particular, increments of 0.05 are considered starting from 0 damage, up to 0.5. Due to the high stochasticity of the corrosive environment, an abundance of episodes was ran for each pair of values. The obtained thresholds as well as the resulting maintenance cost mean and standard deviation are displayed in Table 14, while a policy realization of such a heuristic approach is depicted in Figure 46.

Table 14: *CBM Benchmark maintenance thresholds and costs - Case Study*

Optimal Thresholds			
Repair	**Replace**	**Mean Cost**	**St. Dev.**
None	0.10	**117819.96**	25854.85

As displayed also in Table 14, according to the benchmark, it is more beneficial to let the components deteriorate and perform directly a replace action when 0.10 damage is reached, rather than perform a partial repair earlier.

A realization of a maintenance policy following the benchmark approach, i.e. heuristic damage thresholds, is displayed in Figure 46.

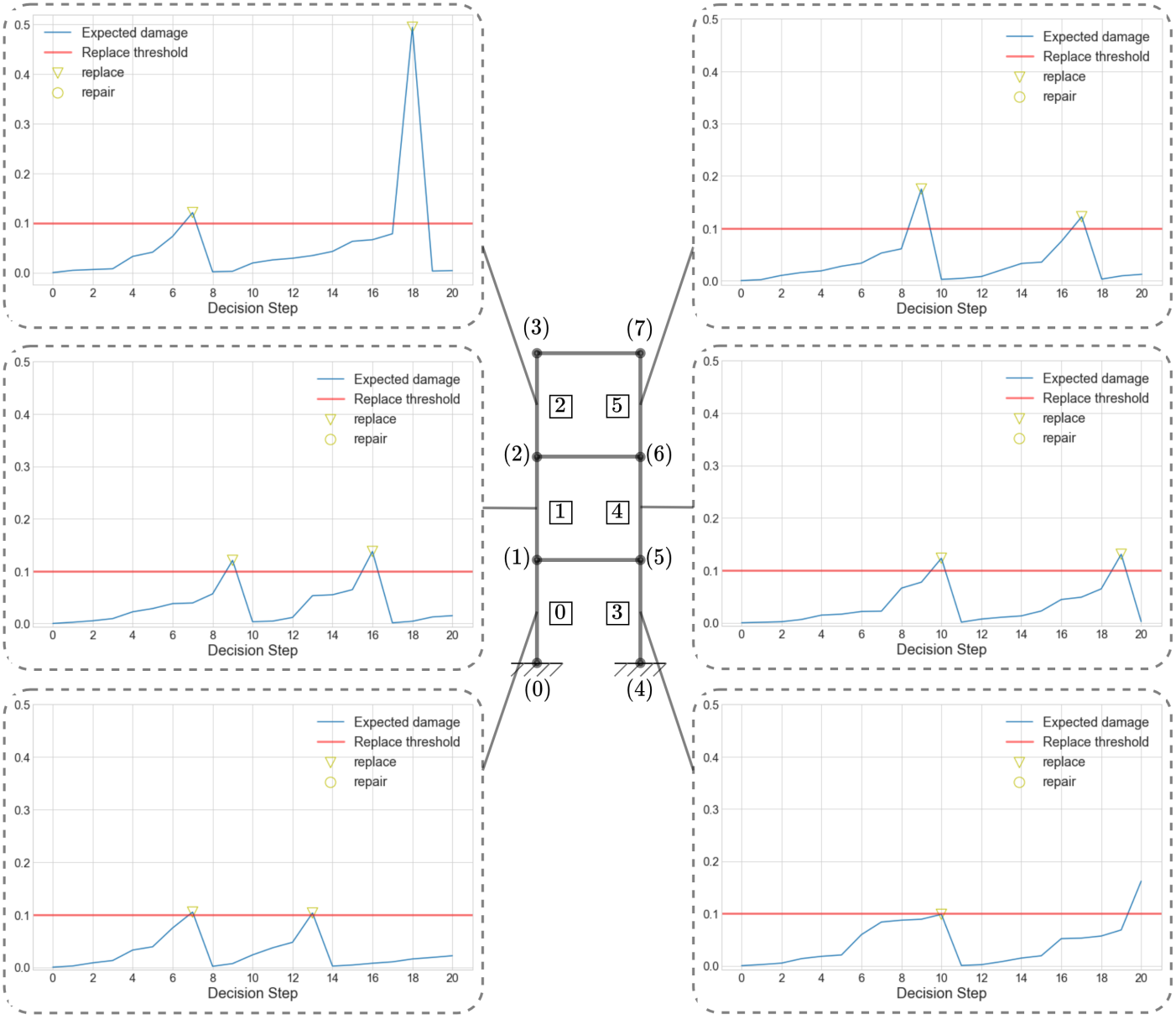


Figure 46: *Policy realization for all components - CBM Benchmark*

As far as the TBM benchmark is concerned, once more a plethora of threshold combination was examined, as well as a great amount of episodes due to the stochastic nature of the environment. The scenario of performing replace actions to all components periodically was considerably more beneficial compared to partial repairs. The different maintenance costs obtained for the different replace intervals are plotted in Figure 47.

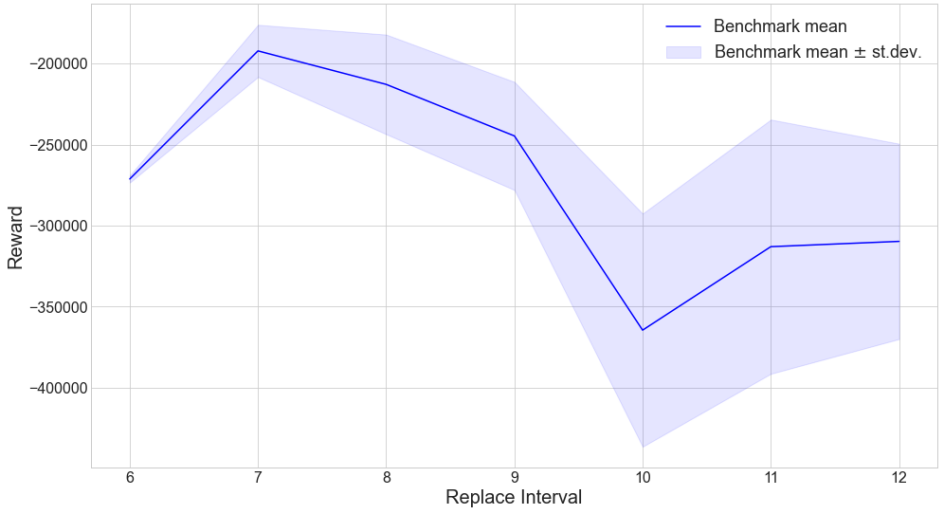


Figure 47: *TBM benchmark costs over replace intervals*

It is observed that the optimal maintenance strategy in a periodic fashion, would be to perform a total replacement of all components every 7 decision steps. Furthermore, it is interesting to note that by increasing the replace interval there is a significantly higher variance in the rewards (maintenance costs). This is something expected, since for a shorter maintenance interval, the deterioration of the system does not evolve as much, leading to a total cost that consists almost explicitly of the maintenance actions' cost, rather than the risk of failure one.

5.4 Results

Having elaborated on the individual sub-routines employed as well as the workflow of the complete framework, the proposed tool is applied to the 2D frame at hand. Before proceeding to the results and plots, the hyper-parameters that were used, are presented in Table 15.

Table 15: *PPO hyper-parameters - Case Study*

Hyper-parameter	**Value**
gamma	0.99
clip ratio	0.15
lambda	0.95
number of inner layers	2
size of inner layers	256
policy learning rate	$1.0\mathrm{E}-3$ to $2.0\mathrm{E}-5$
value function learning rate	$5.0\mathrm{E}-3$ to $5.0\mathrm{E}-5$

It should be mentioned that the learning rates that were used are not constant. To elaborate, at the beginning of the training higher values were used, i.e. $1.0\mathrm{E}-3$ and $5.0\mathrm{E}-3$ for the policy (actor network) and the value function (critic network) respectively, which serve for an initial exploration of the action and the solution space. Over the course of the training episodes, both the learning rates were refined, reaching $2.0\mathrm{E}-5$ and $5.0\mathrm{E}-5$, both to ensure a smoother training, and not to get stuck in sub-optimal solutions.

In Figure 48 the training of the agent is plotted for over 6000 episodes, along with the two benchmark thresholds (CBM and TBM).

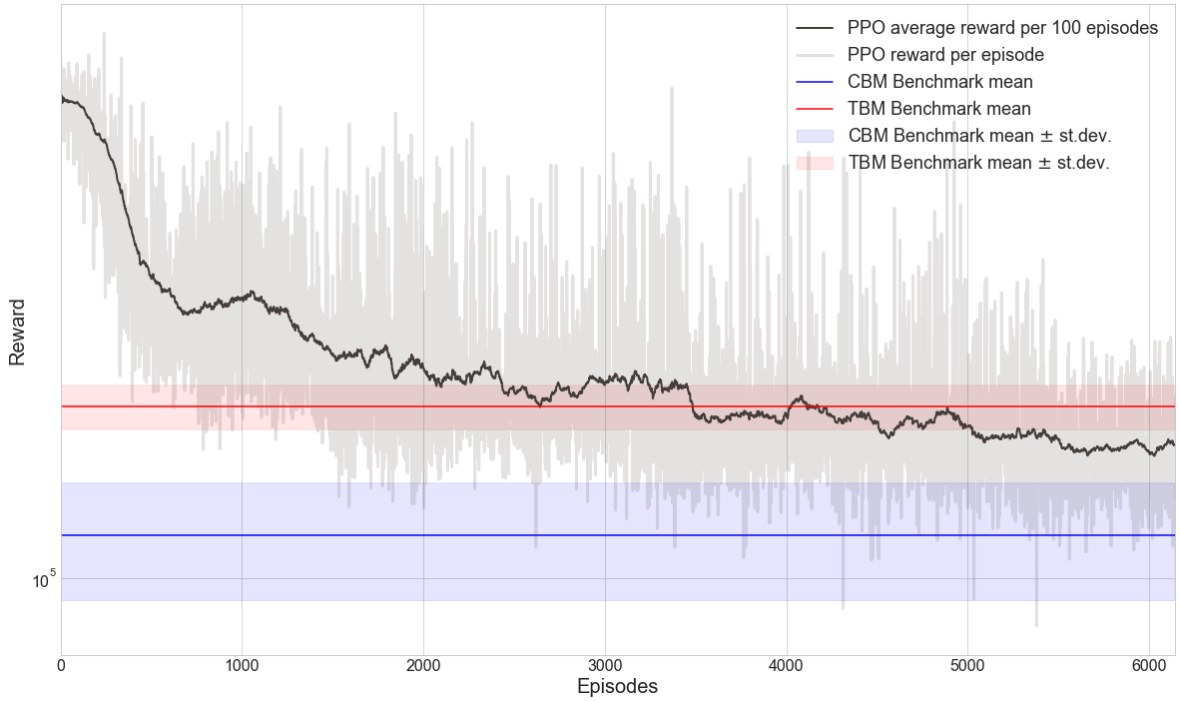


Figure 48: *PPO applied on case study*

It is apparent that the proposed framework could not manage to beat both benchmarks, specifically it surpassed TBM but not CBM. Most probably this is the case because of the dominance of the action costs, and the small contribution of the risk of failure one. As a reminder, the total cost per decision step is decomposed as follows:

$$R_t = C_t = C_{a_t} + C_{\text{risk}}$$

Additionally, it has been observed that the deterioration, as defined in this case study, is developing suddenly and rapidly. If there was a more gradual degradation, this would result in a probability of failure cost that would contribute significantly and during many decision steps to the total maintenance expenses. Undoubtedly, the sheer computational time needed for such a training did not allow for a proper experimentation with different hyper-parameters that would possibly perform better, reaching in the end to a better policy. Nevertheless, the agent seems to perform considerably better than the TBM benchmark, which is not a surprise, since periodic maintenance can not account for the localization of the damage, thus, many components, that their remaining capacity is sufficient, are forced to be maintained along with the rest of the structure.

A way of improving the decisions of the agent, having observed that the deterioration and the resulting probability of failure, do not grow that rapidly, especially during the first decision steps,

would be to include some hardcoded constraints. To be more precise, in this case, it was considered, that no maintenance action would be beneficial during the first 4 decision steps, and also no action should be made if the age (deterioration rate) of a component is less than 2. Once again many realizations were made in order to obtain a representative average maintenance cost for such a policy. The total results, both for the benchmarks and the DRL approaches, are summarized in Table 16. It should be mentioned that for all approaches, more than 100 episodes were ran.

Table 16: *Benchmark and DRL performance on the Case Study*

Maintenance Approach	Mean Cost	St. Dev. Cost	Relative Difference*
CBM Benchmark	117820.0	25854.9	100.00%
TBM Benchmark	192069.2	16121.0	163.02%
PPO	166148.2	27809.7	141.02%
Constrained PPO	133368.7	11180.2	113.20%

* The mean cost achieved by each approach is compared with the minimum of all, i.e. the CBM Benchmark one

The fact that a constrained PPO agent can arrive at lower maintenance costs, proves that a better tuning of the hyper-parameters can yield more efficient policies.

What is more, another interesting plot is the policy realization for the constrained agent. Of course, a single realization is not the most representative, but it still provides a qualitative representation of how the agent chooses its actions. Such a plot, is displayed in Figure 49, where the damage over the time horizon and the corresponding actions are plotted for all the components.

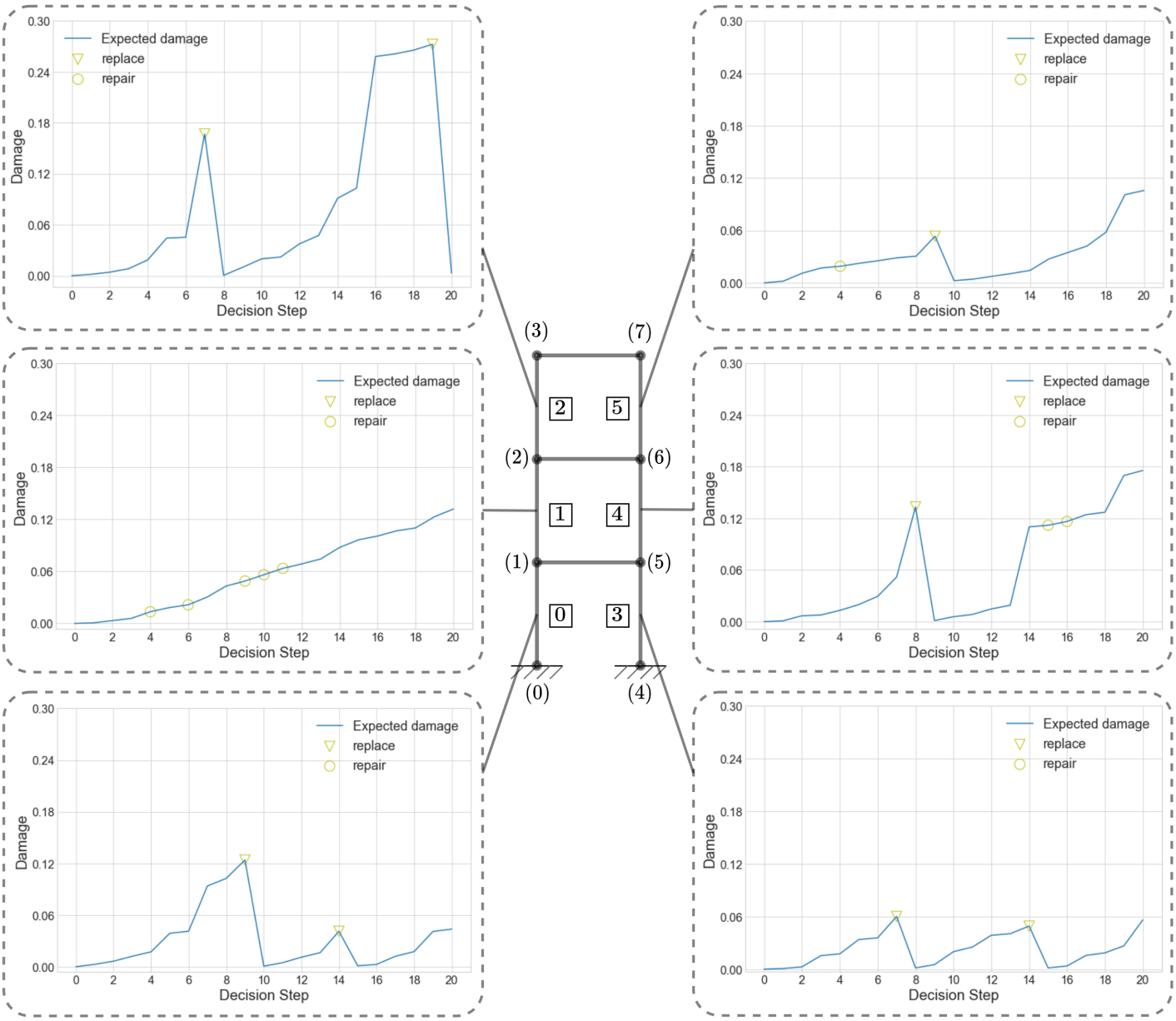


Figure 49: *Policy realization for all components (constrained)*

It is evident that there is not a specific pattern that the agent follows neither on a global scale nor on a component one. Undoubtedly, it does not allow the damage to reach extremely high values, since this would also affect significantly the horizontal displacement of the top storey, which is the control quantity for the structure's failure.

In Figures 50 and 51, policy realizations are plotted for different levels of training. In particular, there is a policy realization every 1200 episodes, to demonstrate the learning of the agent, and how from completely uninformed and random actions, it shifted to more reasonable and beneficial ones. To serve this purpose from a presentation point of view, a color-bar is included in the plots, to display the decrease of the total maintenance costs over the course of training episodes.

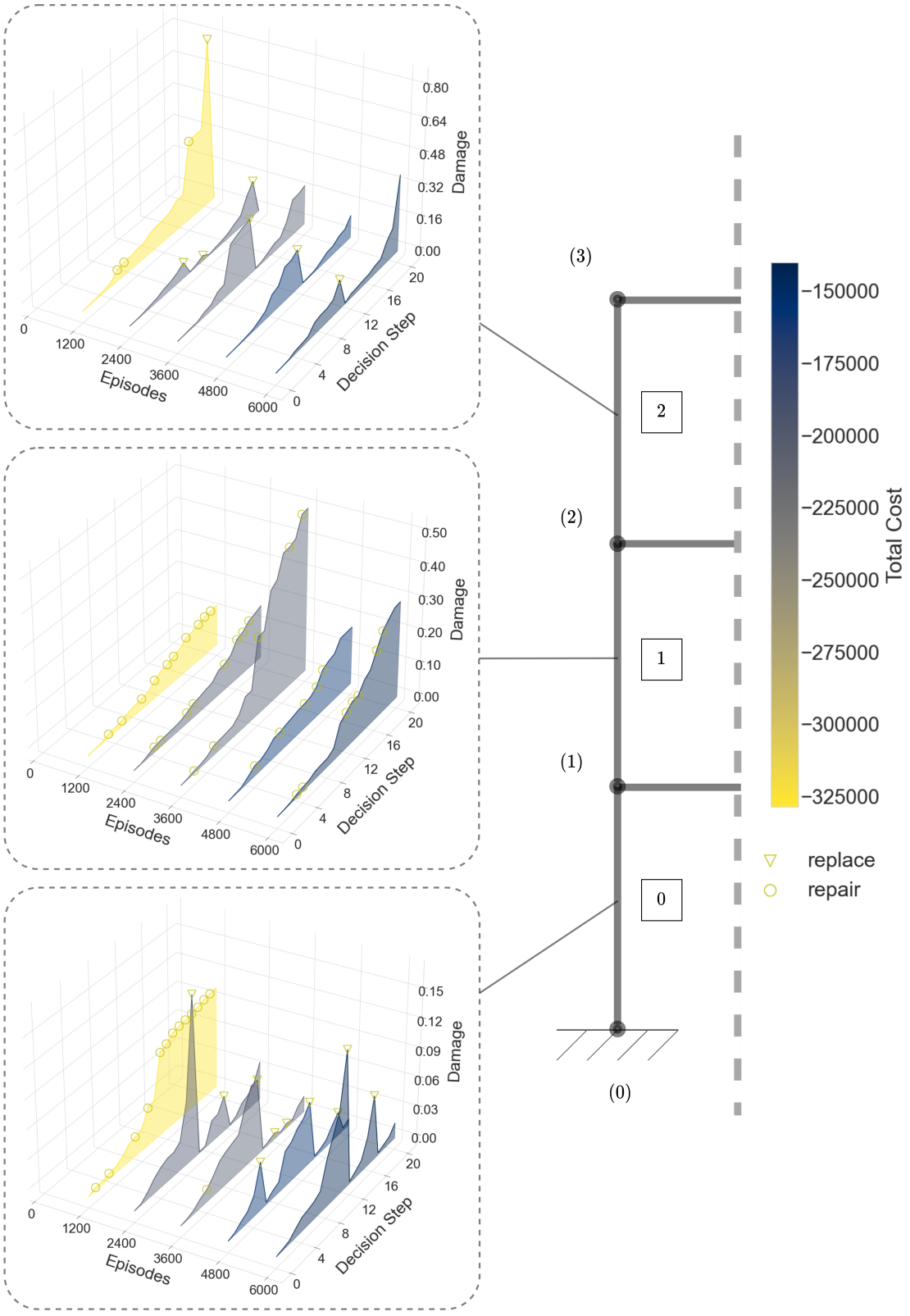


Figure 50: *Policy realization for all components for different training episodes - Left side*

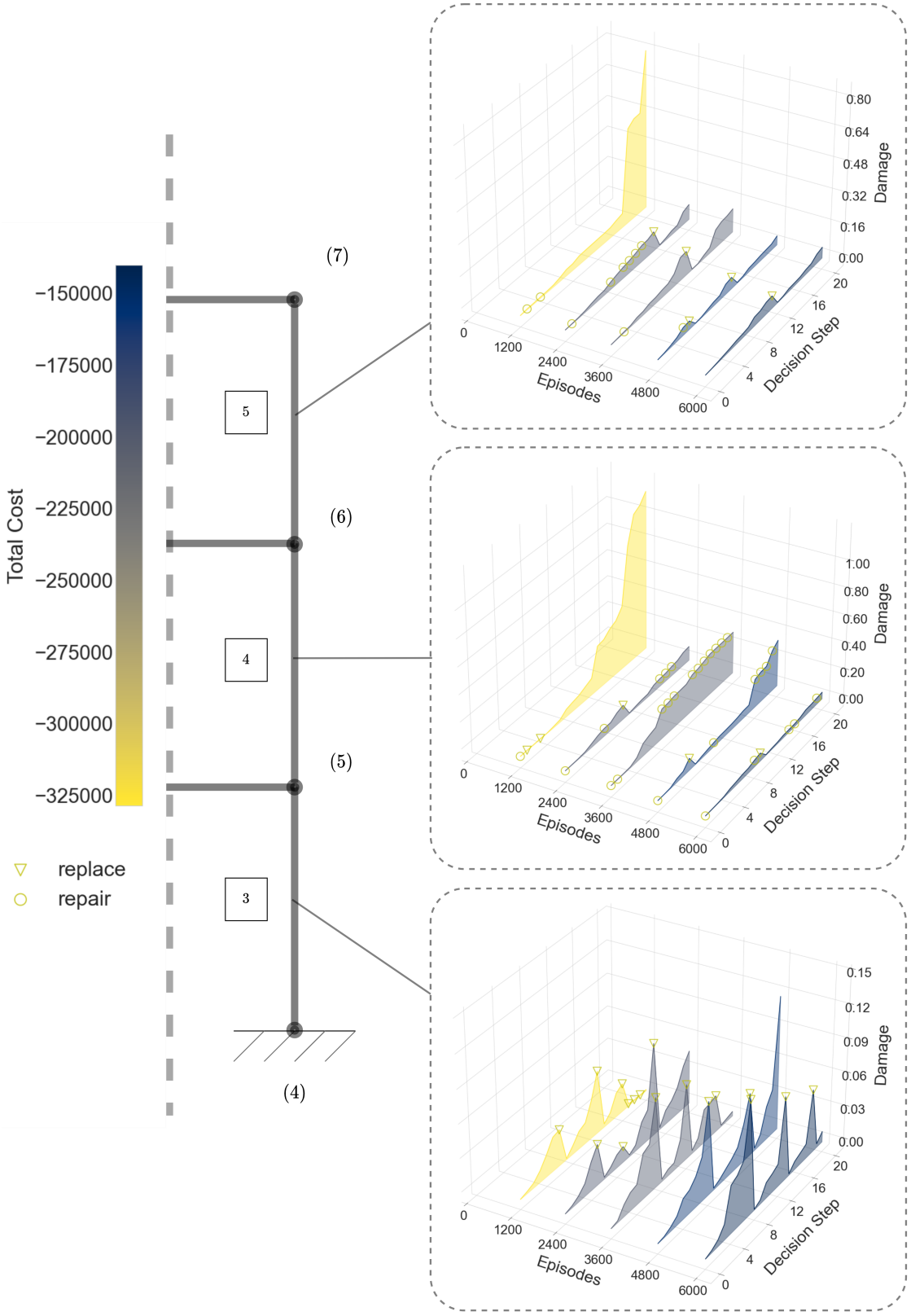


Figure 51: *Policy realization for all components for different training episodes - Right side*

It goes without saying, that the agent during the early learning stages was allowing the deterioration to grow significantly, something that changed over the course of more episodes, since higher damage values combined with the degradation of other components as well can lead to a greater probability of failure P_f. Another interesting observation is that, especially for the left side of the frame, the agent gets much more sensitive about the damage of the base storey, rather than the ones of the two above. This is a reasonable strategy, because the damage of the lowest column has a greater contribution to the global failure. Surprisingly, this is not the same case for the right side, where all columns are limited to small damage values, regardless from their location. As already explained, this could be attributed to the high stochasticity of the corrosive environment, which makes a single realization a non-representative measure.

5.5 Conclusions

Summarizing the chapter, some general comments will be made and conclusions will be drawn, regarding the case study.

Moving from a simplistic application such as the toy problem, to a more complicated case study undoubtedly made the optimal maintenance strategy harder to determine. The considered 2D frame is a multi-component system that poses greater challenges, with the more obvious one being the enhancement of the action and state spaces. The benchmark approach, and more specifically the CBM version of it, was more difficult to beat, and it was highlighted that a better tuning of the hyper-parameters is necessary. It is also possible that the initial problem setup needs to be reconsidered, and particularly the assumptions made about the costs/rewards during every decision step.

6 Discussion - Conclusion

6.1 Discussion of results

Having presented the proposed framework both in theory and in hands-on applications, it is evident that such an approach, i.e. coupling DRL with BMU to determine an optimal maintenance strategy, carries a plethora of advantages. In particular, the basic theory regarding the developed tool was described, moving to a simplified first application, where the framework's superiority is firstly highlighted, with a culmination of this thesis being the use of this framework for a more complicated and realistic case study.

As far as the toy problem is concerned, there were two DRL algorithms[20] that were tested, namely DDQN and PPO, which both performed better than the traditional maintenance approaches. A more thorough presentation and elaboration of the results is presented in Section 4.4, where apart from the learning process of the agent over the training episodes, a variety of important quantities, is plotted for a plethora of policy realizations. Of course, such an application is over-simplified and even though it fortifies the potential of the proposed framework, it is not directly applicable to real-life cases.

After the toy problem, the application of the proposed framework in a case study was presented. In particular, a 2D 3-storey frame was chosen, with its 6 columns being the deteriorating components, and the PPO algorithm being selected for the training of the agent. Unfortunately, the proposed methodology did not manage to beat the benchmark, and to be more precise, the CBM benchmark, which gave rise to plenty of discussion points, regarding what are the possible reasons for such a performance.

The main cause that possibly leads to the inferior performance of the developed tool might be the assumptions that were made about the failure of the structure and subsequently the cost related to the risk of failure. For the case study the failure was defined through an SLS check regarding the drift of the top storey, and the cost of failure was assumed to be equal to a total replacement of all the components. Even though such an assumption could be realistic, as explained also in Section 5.1.1, since in the case of a global collapse there might be some lightly damaged components that could be reused, the corresponding cost was proved to have a minor contribution to the total reward, making the cost of actions dominant. Therefore, it was possible to arrive at an optimal sequence of actions using a heuristic threshold-based approach that was able to locate these maintenance expenses in the most cost-efficient way along the structure's lifecycle.

Another interesting observation, that is worth being discussed, is the decisions that the agent takes for columns that belong to different storeys. Based on elementary structural mechanics, it is

[20] A third algorithm was tested, too, namely A2C, which unfortunately did not yield optimal maintenance strategies even for the simplest of the cases, this is why it was disregarded for the rest of the thesis.

apparent that the damage of the bottom columns contributes more to the possible failure of the structure. Since the failure is translated to a risk of failure cost, the agent is capable of arriving at a policy that would limit the damage of the more "important" (failure-wise) components, while on the other hand allowing the deterioration of the less crucial columns to reach higher values. This way of maintaining a multi-component system can not be achieved using heuristic threshold-based approaches; a fact that compliments the benefits of having a DRL agent as a decision-maker. In Figure 52 the deterioration of all components is plotted over the decision steps of a single policy realization. It is evident that the strictest maintenance actions take place for components 0 and 3 which correspond to the base columns.

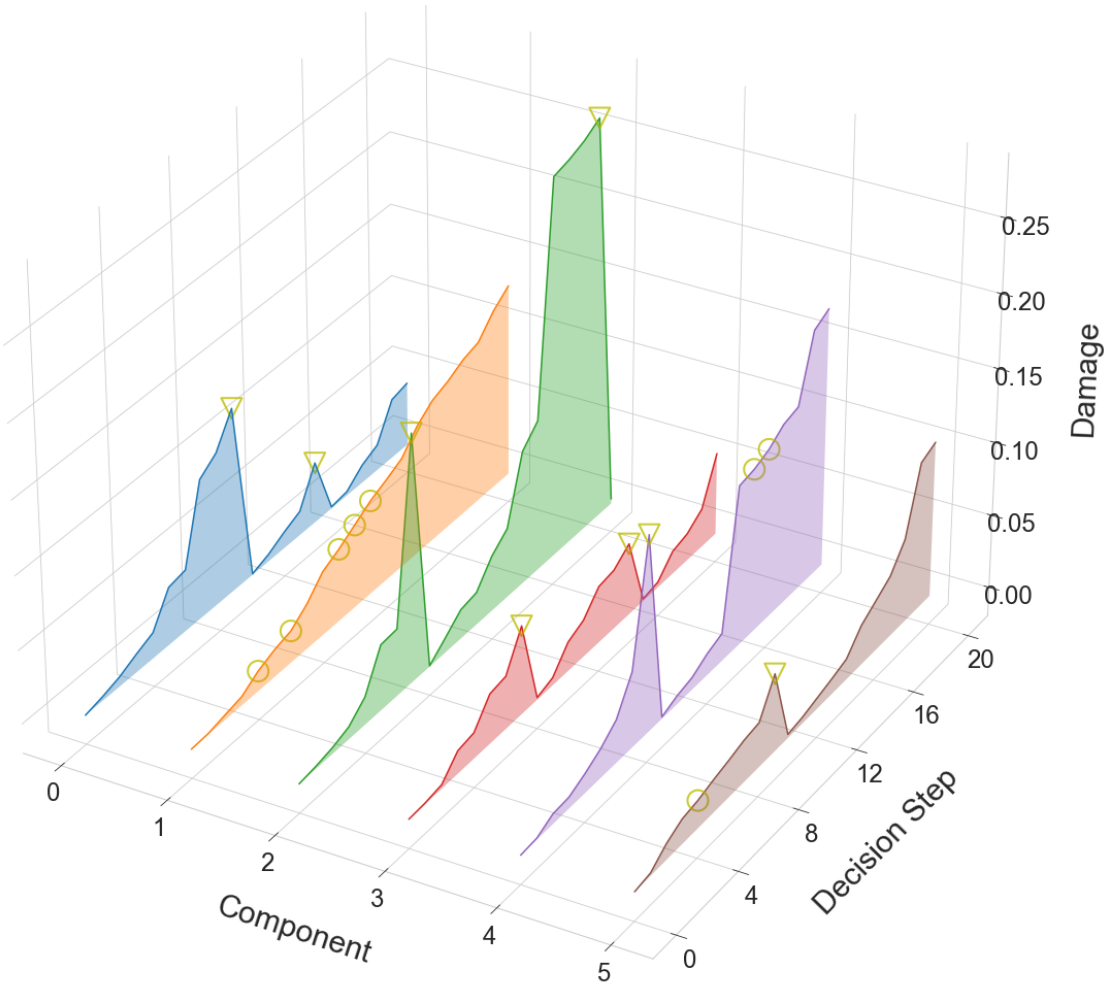


Figure 52: *Policy realization for all components (constrained) - 3D*

Last but not least, it should be mentioned that the proposed methodology is not strictly applicable only to structural engineering applications. The maintenance of any kind of multi-component engineering system, whose deterioration can be expressed through a stochastic process, can be dealt with using such an approach. Taking advantage of both the capabilities of the DRL agent for the sequential decision problem, as well as the more accurate modeling of the deteriorating environment that the continuous variable Bayesian inference provides, renders this framework a promising tool to tackle the problem of maintenance in a general sense.

6.2 Limitations

As in every research project, there is a trade-off between the modeling accuracy, i.e. the simplifications and assumptions made, and the corresponding computational time. Limiting the assumptions/simplifications for the proposed framework, had the expected outcome regarding the runtime. Therefore, the most important limitation of the developed tool is the high computational time needed for the training of the agent. It makes it significantly difficult to tune the hyper-parameters or do simple modifications to the system's dynamics, which would require the training of the agent from scratch. Nevertheless, it could be characterized as a disadvantage worth having, since such a tool can yield the optimal maintenance strategy for the whole lifetime of an engineering system, making the runtime seem less important on a relative scale. Of course, as displayed also in this thesis, the computational resources needed are proportional to the complexity of the considered system, as the toy problem both in its discrete and continuous version was much faster to solve in comparison with the more complicated case study. A possible solution for this limitation would be the further optimization of the Bayesian inference since it was the least time-efficient part of the algorithm. Additionally, the ever-increasing computational power closely connected with the rapid development of the technology could also help in this aspect in the future.

6.3 Future Development

The current research investigates the benefits and the potential of a workflow that integrates both DRL and BMU, aiming to determine the optimal sequence of maintenance decisions. Although the first results of such an approach, which were presented in this thesis, are a promising indicator of its capabilities, many possible additions and modifications can be incorporated into the proposed workflow and would be interesting to examine. These ideas for further development and future research are presented in this section.

Regarding the multi-component system that was examined as a case study, there are many possible alternatives as far as the actor network architecture is concerned. The actor network architecture used in this project was a *centralized* one, meaning that there was a single neural network for all the components and all the possible actions (this is explained more thoroughly in Section 5.2). A different approach would be to *decentralize* each component's network, in a similar fashion as for DDMAC in [44]. Even though the states remain *centralized* and s_t constitutes the input for every agent, there are as many networks and weights θ_i as the control points, i.e. the components of the system. This means, each agent chooses an action independently, but they are still aware of one another's condition, owing to the common input they are getting. The described architecture is illustrated in Figure 53.

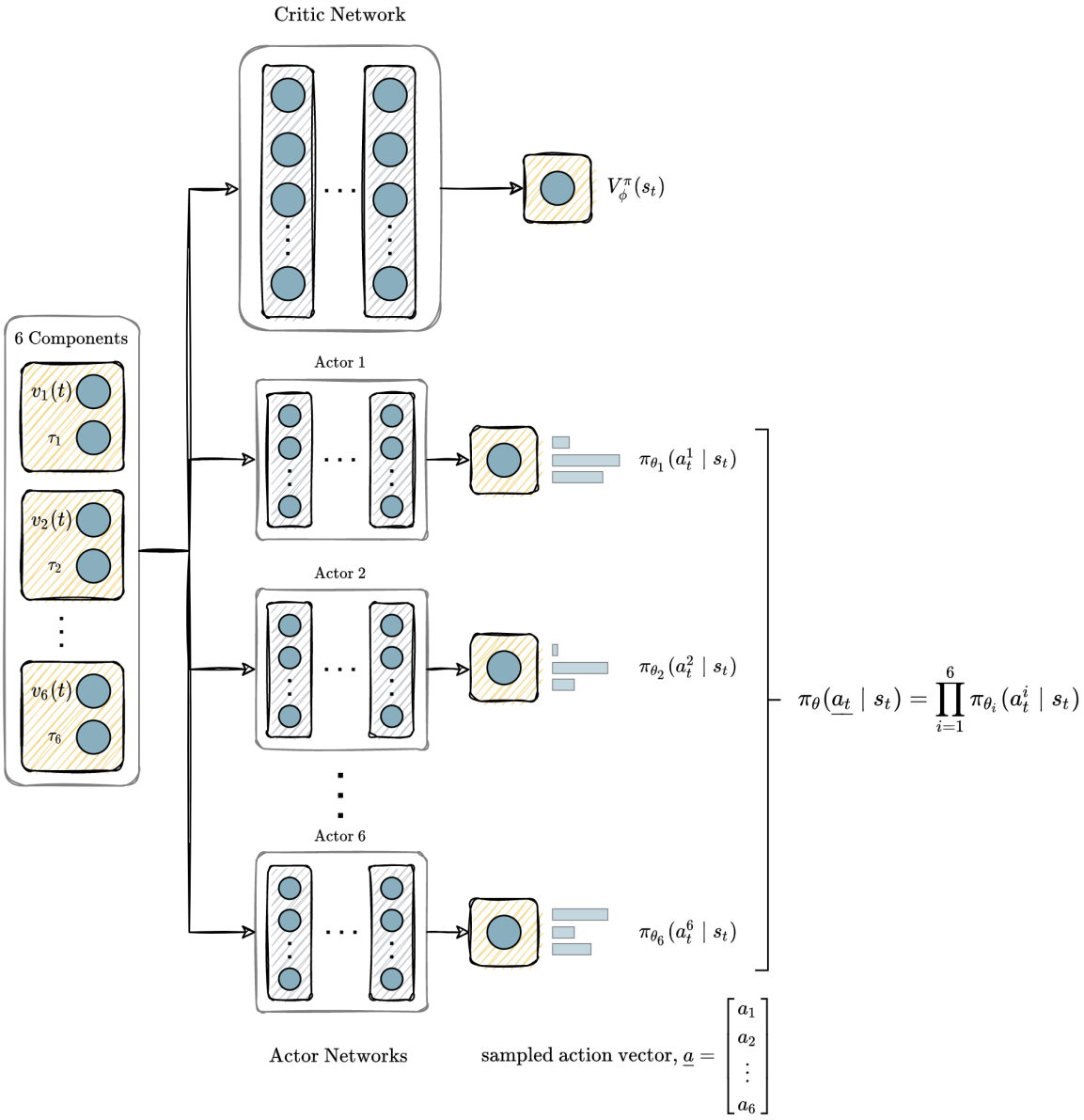


Figure 53: *PPO architecture - Centralized states and decentralized actions*

A variation of this architecture would be to group the elements that belong to the same floor. From an engineering point of view, it is a valid assumption that columns of the same floor would stochastically be described by a single neural network. Also for this option, the state s_t which contains the damage and the deterioration rate of all components, would be common for all 3 networks (*centralized* states). The output of these *floor* networks would still be a softmax, i.e. π_{θ_i}, from which two actions would be sampled (one for each column) during every decision step, as depicted in Figure 54. The validity of this proposal can be further backed up if in the case of a *decentralized* 6 sub-network architecture (Figure 53) the agent chooses eventually symmetrical actions, meaning that similar decisions are made for the columns that belong to the same floor.

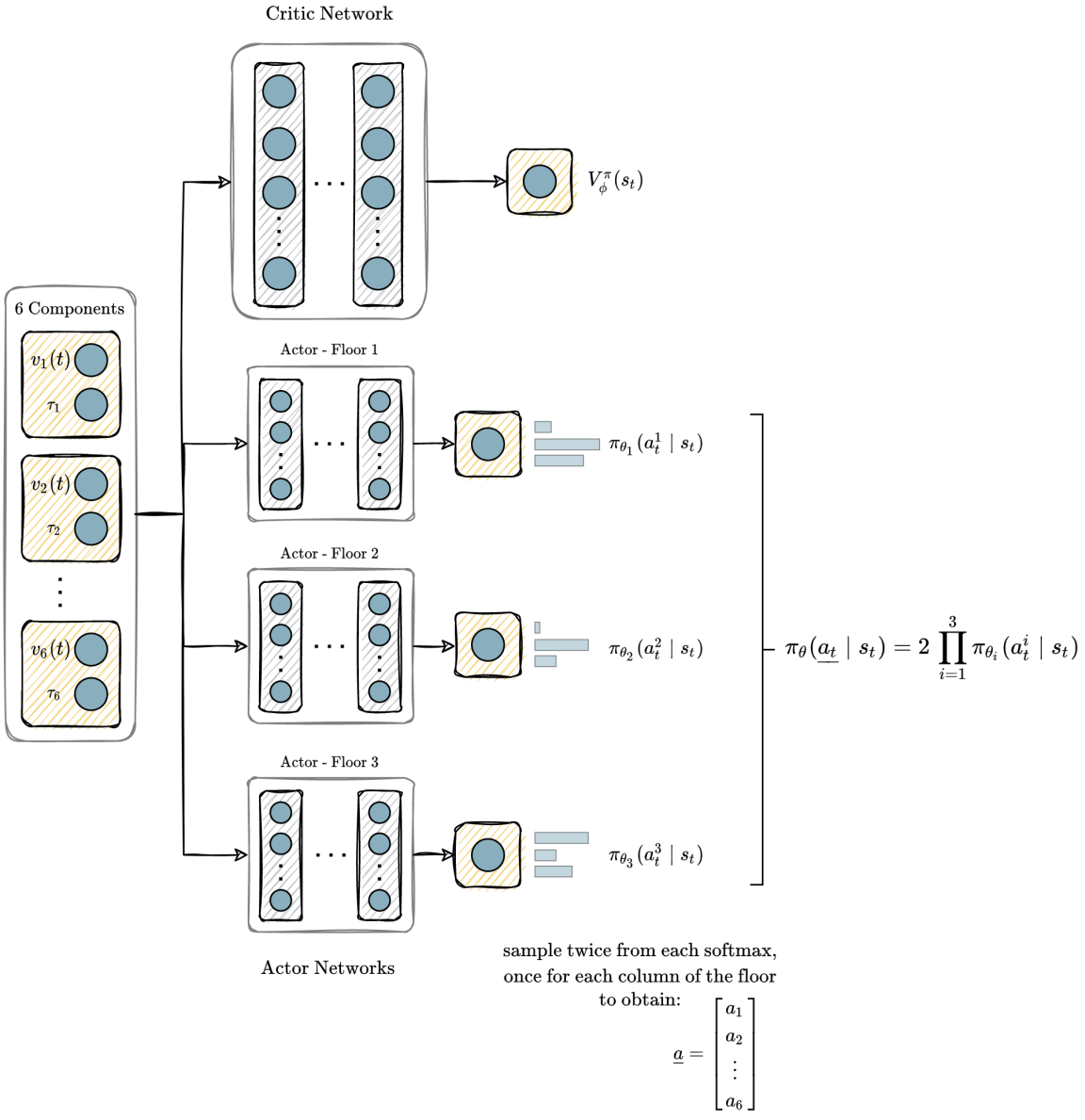


Figure 54: *PPO architecture - Centralized states and decentralized actions - Floor variation*

Lastly, another idea for an actor network architecture would be to consider in the inputs an ID for each component. There will be only one network and shared parameters θ, which would output a single policy π_θ with three probabilities which would refer to the component with the given ID. The state s_t containing information for the whole structure will remain as is in the input layer (*centralized* approach), and there will be six 6 forward propagations of the actor network to obtain the six 6 component policies. This idea is depicted in Figure 55.

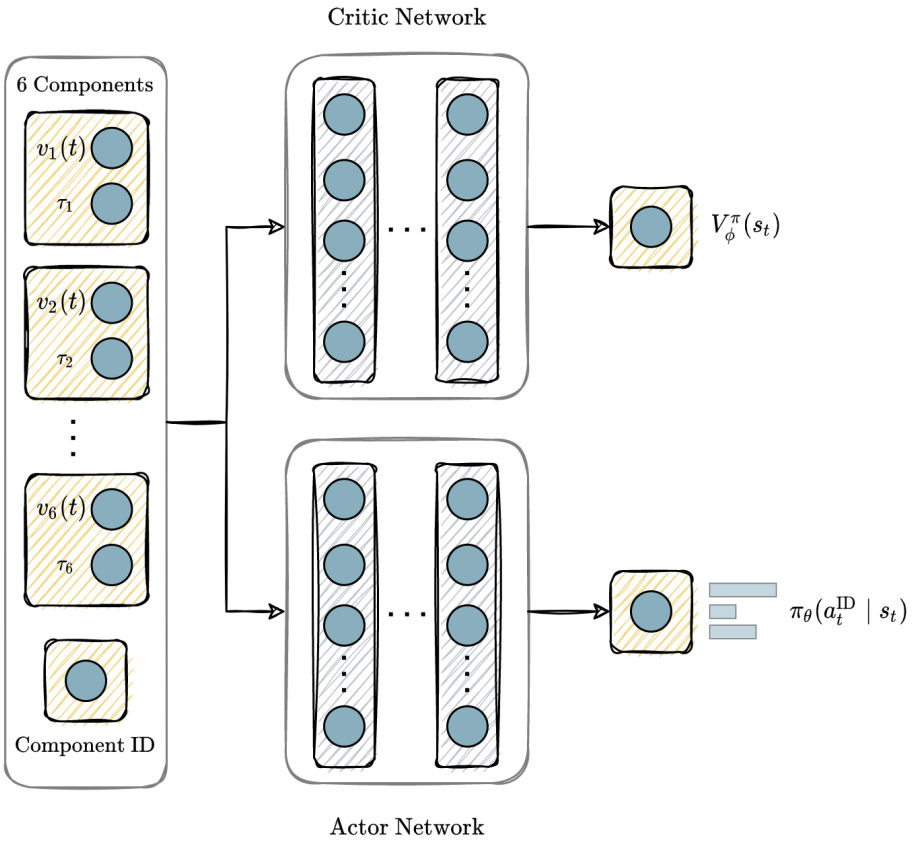


Figure 55: *PPO architecture - Component's ID*

An interesting idea for a future investigation is the inclusion of the stochastic parameters A and B, which are updated in every decision step, to the state s_t, hence to the inputs of the neural networks. This modification could lead to the agent making more confident decisions in the later decision steps when it has developed a greater knowledge about the deteriorating environment. Such a feature would also be an important advantage of the proposed methodology in comparison with the traditional approaches, which need to follow a strategy from start to end, and can not capture the decrease of the uncertainties through continuous model updates, subsequently the increase of the decision-maker's confidence.

What is more, a worthy addition to this framework would be to account for different/additional actions. One particularly interesting and realistic action scenario would be to reward the grouping of the components' maintenance. More specifically, this can highlight how probable would it be for the agent to choose a single component's action at a higher cost, instead of waiting for other ones to deteriorate further and perform maintenance to more members during the same decision step. This can be modeled using a fixed base cost, e.g. for the maintenance crew to reach the structure's venue, and add to that the cost of the maintenance actions.

On a more technical note, currently, PPO trains the agent using a rollout buffer, i.e. the stored samples are discarded after their use. Alternatively, to reuse the samples and tackle more efficiently the stochasticity of the problem, a replay buffer can be employed, that will keep the samples even after the training. During the training a weighted sampling can be applied, for the more recent samples (generated by more recent policies) to be more important compared to the old ones ([61], [43]).

Lastly, as already mentioned, the modeling choice of the failure and its corresponding cost, play an important role in the ability of the DRL agent to beat the benchmark. To verify this argument, it would be useful to perform a sensitivity analysis for various scenarios of risk of failure cost. Unfortunately, although such runs were initiated for this thesis as well, the computational time was restricting and the proper training of the agent along with hyper-parameter exploration was not possible.

A Appendices

A.1 Appendix 1 - Case Study

A.1.1 Load Case

To begin with, it should be mentioned that the calculations of the applied loads are made in an approximate fashion, since the case study is not an existing structure, meaning that any assumption could be justified as valid. This section aims to describe the step-by-step procedure of load calculation even for a real life application.

The elastic quasi-static forces per floor of the frame will be calculated according to the provisions of Eurocode 8 and the elastic response spectra. Using Equations (3.2) - (3.5) provided in Clause 3.2.2.2 of EN1998-1 [62], and the values included in Table 17, the elastic response spectrum can be drawn[21].

Table 17: *Values of the parameters describing the recommended Type I elastic response spectra [62]*

Ground type	S	$T_B(s)$	$T_C(s)$	$T_D(s)$
A	1.00	0.15	0.4	2.0
B	1.20	0.15	0.5	2.0
C	1.15	0.20	0.6	2.0
D	1.35	0.20	0.8	2.0
E	1.40	0.15	0.5	2.0

From the eigenanalysis of the frame, the eigenfrequencies are derived, with the fundamental one being $f_1 = 4.42\,\mathrm{Hz}$, meaning that the fundamental period is $T_1 = 0.226\,\mathrm{sec}$. Based on this period, and the drawn elastic response spectrum, the design ground acceleration is derived $S_{a,\mathrm{el}} = 6.769\,\mathrm{m/s^2}$ as illustrated in Figure 56.

[21] It is assumed that for the current application the ground type is C

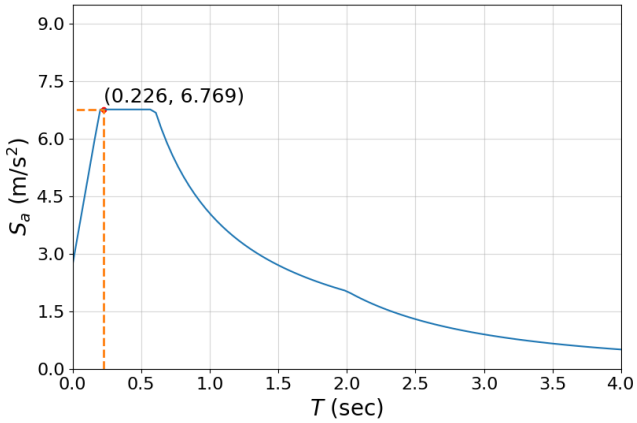


Figure 56: *Elastic response spectrum*

A simplified mass model for the frame is presented in Figure 57, where M represents the mass of the two columns and the beam of each floor. In particular, accounting for structural steel's mass density, $\rho = 7850\,\mathrm{kg/m^3}$ and the geometric dimensions of the members (cross-sectional area and length), M is calculated to be $1250\,\mathrm{kg}$. Following this assumption, that the mass per floor is concentrated, the elastic quasi-static forces, are calculated as follows:

$$F_{\mathrm{el},i} = S_{a,\mathrm{el}}\,\Gamma\, m_i\, \varphi_i \tag{58}$$

where,

$m_i =$ mass at floor i [kg]

$\varphi_i =$ first eigenvector value, corresponding to floor i [−]

$\Gamma =$ modal participation factor, $\dfrac{\sum \varphi_i \cdot m_i}{\sum \varphi_i^2 \cdot m_i}$ [−] (for a diagonal mass matrix M)

The first eigenvector is plotted also in Figure 57.

$$\varphi = \begin{Bmatrix} 0.33 \\ 0.66 \\ 1.00 \end{Bmatrix} \quad , \quad \underline{\underline{M}} = \begin{bmatrix} 1250 & 0 & 0 \\ 0 & 1250 & 0 \\ 0 & 0 & 1250 \end{bmatrix} \quad , \quad \Gamma = 1.288$$

The needed quantities to calculated the quasi-static forces are included in Table 18.

Table 18: *Elastic quasi-static forces based on EN1998-1 [62]*

Floor, i [-]	ϕ_i [-]	Floor mass [kg]	Γ [-]	$S_{a,\text{el}}$ [m/sec^2]	$F_{\text{el},i}$ [kN]
1	0.33	1250	1.288	6.769	3.6
2	0.66	1250	1.288	6.769	7.2
3	1.00	1250	1.288	6.769	10.8

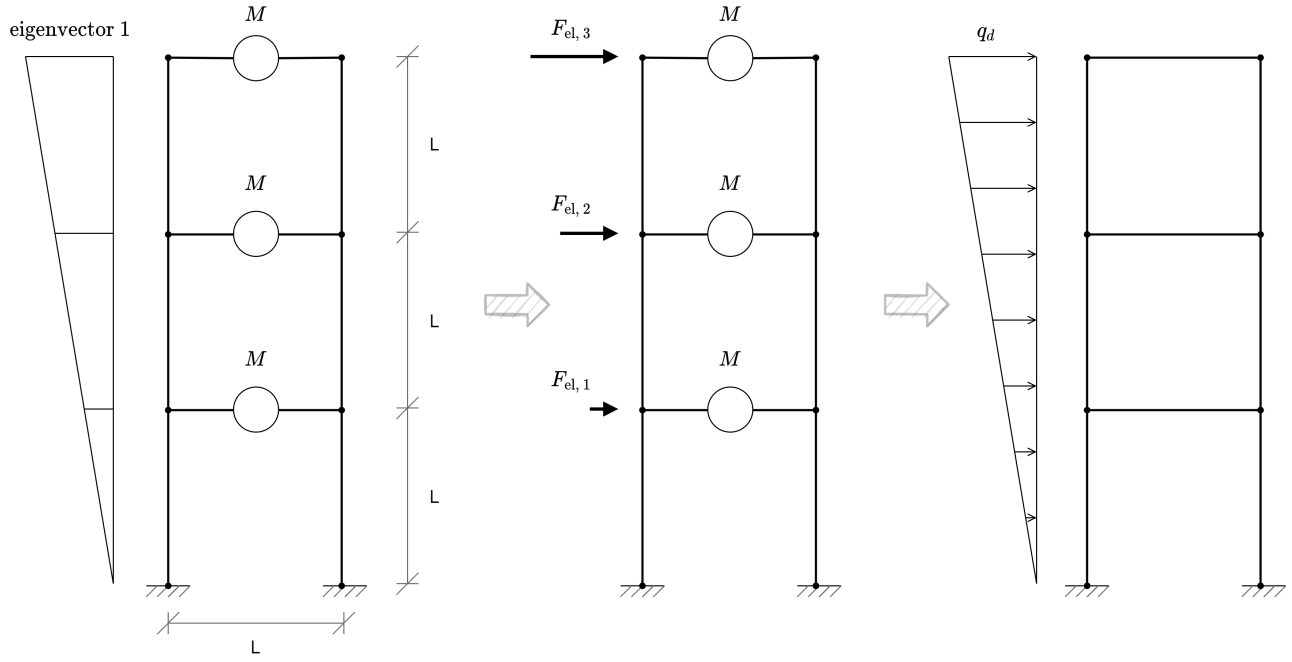


Figure 57: *Simplified model of the frame*

The superposition of these three concentrated loads will be now transformed into a triangular one, of value q_d at the top node, as depicted also in Figure 57.

$$\sum_{i=1}^{3} F_{\text{el},i} = \frac{1}{2} 3 L q_d$$
$$\Rightarrow q_d = 3.6\,\text{kN/m} \tag{59}$$

A.1.2 Deterioration

In the current project, the damage $d(\tau)$ is defined as the cross section loss, i.e. the ratio of the current cross-section area at deterioration rate τ, over the initial one.

$$d(\tau) = \frac{A(\tau)}{A_0}$$

However, from a more pragmatic point of view, the damage, which is usually a reduction in thickness does not affect in the same way the flexural and the bending stiffnesses. The correlation between these two reductions is elaborated in this section.

Denoting as c the corrosion penetration depth, and assuming that it is constant along the perimeter of the cross-section (Figure 38), the initial stiffnesses are:

$$A_0 = 2\, t_f\, b_f + t_w\, h_w \tag{60}$$

$$I_0 = 2\Big[\frac{1}{12}\, b_f\, t_f^3 + b_f\, t_f\, (\frac{h_w + t_f}{2})^2\Big] + \frac{1}{12}\, t_w\, h_w^3 \tag{61}$$

where t_f, b_f are the thickness and the width of the flange, and t_w, h_w are the thickness and the height of the web, as displayed in Figure 58.

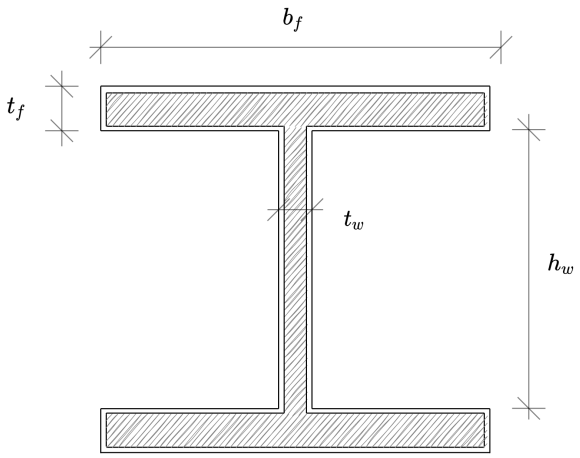


Figure 58: *I-beam cross-section*

The degraded cross-sectional area is:

$$A = 2\,(t_f - 2\,c)\,(b_f - 2\,c) + (t_w - 2\,c)\,(h_w + 2\,c) \tag{62}$$

Subsequently, it holds:

$$
\begin{aligned}
A_0\, d &= A \\
\Rightarrow \left[2\, t_f\, b_f + t_w\, h_w\right] d &= 2\,(t_f - 2\,c)\,(b_f - 2\,c) + (t_w - 2\,c)\,(h_w + 2\,c) \\
\Rightarrow 4\,c^2 + 2\,c\,(t_w - h_w - 2\,t_f - 2\,b_f) + (1-d)\left[2\,t_f\,b_f + t_w\,h_w\right] &= 0 \\
\Rightarrow c = \frac{-2\,(t_w - h_w - 2\,t_f - 2\,b_f) \pm \sqrt{\left(2\,(t_w - h_w - 2\,t_f - 2\,b_f)\right)^2 - 16\,(1-d)\left(2\,t_f\,b_f + t_w\,h_w\right)}}{8} &\qquad (63)
\end{aligned}
$$

Then based on this value of corrosion penetration depth, c, the degraded moment of inertia is:

$$
I = 2\left[\frac{1}{12}\,(b_f - 2\,c)\,(t_f - 2\,c)^3 + (b_f - 2\,c)\,(t_f - 2\,c)\left(\frac{h_w + t_f}{2}\right)^2 + \frac{1}{12}\,(t_w - 2\,c)\,(h_w + 2\,c)^3\right] \qquad (64)
$$

Concluding, using Equations 63, 62, 64, the degraded stiffnesses are calculated based on the damage of every decision step.

References

[1] *Proximal policy optimization*, Accessed: 24-Jan-2022. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/ppo.html.

[2] M. G. Stewart and D. V. Rosowsky, "Time-dependent reliability of deteriorating reinforced concrete bridge decks," *Structural safety*, vol. 20, no. 1, pp. 91–109, 1998.

[3] F. Akgül and D. M. Frangopol, "Lifetime performance analysis of existing steel girder bridge superstructures," *Journal of Structural Engineering*, vol. 130, no. 12, pp. 1875–1888, 2004.

[4] D. V. Val, M. G. Stewart, and R. E. Melchers, "Effect of reinforcement corrosion on reliability of highway bridges," *Engineering structures*, vol. 20, no. 11, pp. 1010–1019, 1998.

[5] T. Moan, "Reliability-based management of inspection, maintenance and repair of offshore structures," *Structure and Infrastructure Engineering*, vol. 1, no. 1, pp. 33–62, 2005.

[6] I. Lotsberg, G. Sigurdsson, A. Fjeldstad, and T. Moan, "Probabilistic methods for planning of inspection for fatigue cracks in offshore structures," *Marine Structures*, vol. 46, pp. 167–192, 2016.

[7] P. Wirsching, "Fatigue reliability in welded joints of offshore structures," *International Journal of Fatigue*, vol. 2, no. 2, pp. 77–83, 1980.

[8] P. Schaumann, S. Lochte-Holtgreven, and S. Steppeler, "Special fatigue aspects in support structures of offshore wind turbines," *Materialwissenschaft und Werkstofftechnik*, vol. 42, no. 12, pp. 1075–1081, 2011.

[9] W. Dong, T. Moan, and Z. Gao, "Fatigue reliability analysis of the jacket support structure for offshore wind turbine considering the effect of corrosion and inspection," *Reliability Engineering & System Safety*, vol. 106, pp. 11–27, 2012.

[10] B. Yeter, Y. Garbatov, and C. G. Soares, "Fatigue damage assessment of fixed offshore wind turbine tripod support structures," *Engineering Structures*, vol. 101, pp. 518–528, 2015.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[12] R. van Leeuwen, *Feastruct*, Accessed: 04-Apr-2022. [Online]. Available: https://github.com/robbievanleeuwen/feastruct.

[13] S. Geyer, A. Kamariotis, I. Papaioannou, L. Sardi, D. Straub, and F. Uribe, *First-order reliability method*, Accessed: 06-May-2022. [Online]. Available: https://www.cee.ed.tum.de/en/era/software/reliability/first-order-reliability-method/.

[14] K. G. Papakonstantinou and M. Shinozuka, "Planning structural inspection and maintenance policies via dynamic programming and markov processes. part i: Theory," *Reliability Engineering & System Safety*, vol. 130, pp. 202–213, 2014.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] J. F. Andersen, A. R. Andersen, M. Kulahci, and B. F. Nielsen, "A numerical study of markov decision process algorithms for multi-component replacement problems," *European Journal of Operational Research*, 2021.

[17] M. Compare, P. Marelli, P. Baraldi, and E. Zio, "A markov decision process framework for optimal operation of monitored multi-state systems," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 232, no. 6, pp. 677–689, 2018.

[18] R. Schöbi and E. N. Chatzi, "Maintenance planning using continuous-state partially observable markov decision processes and non-linear action models," *Structure and Infrastructure Engineering*, vol. 12, no. 8, pp. 977–994, 2016.

[19] R. B. Corotis, J. Hugh Ellis, and M. Jiang, "Modeling of risk-based inspection, maintenance and life-cycle cost with partially observable markov decision processes," *Structure and Infrastructure Engineering*, vol. 1, no. 1, pp. 75–84, 2005.

[20] K. T. Nguyen, P. Do, K. T. Huynh, C. Bérenguer, and A. Grall, "Joint optimization of monitoring quality and replacement decisions in condition-based maintenance," *Reliability Engineering & System Safety*, vol. 189, pp. 177–195, 2019.

[21] K. G. Papakonstantinou and M. Shinozuka, "Planning structural inspection and maintenance policies via dynamic programming and markov processes. part ii: Pomdp implementation," *Reliability Engineering & System Safety*, vol. 130, pp. 214–224, 2014.

[22] K. G. Papakonstantinou, C. P. Andriotis, and M. Shinozuka, "Pomdp and momdp solutions for structural life-cycle cost minimization under partial and mixed observability," *Structure and Infrastructure Engineering*, vol. 14, no. 7, pp. 869–882, 2018.

[23] C. P. Andriotis, K. G. Papakonstantinou, and E. N. Chatzi, "Value of structural health information in partially observable stochastic environments," *Structural Safety*, vol. 93, p. 102 072, 2021.

[24] P. Morato, K. Papakonstantinou, C. Andriotis, J. Nielsen, and P. Rigo, "Optimal inspection and maintenance planning for deteriorating structural components through dynamic bayesian networks and markov decision processes," *Structural Safety*, vol. 94, p. 102 140, 2022.

[25] P. G. Morato, J. S. Nielsen, A. Q. Mai, and P. Rigo, "Pomdp based maintenance optimization of offshore wind substructures including monitoring," 2019.

[26] P. Dallaire, C. Besse, S. Ross, and B. Chaib-draa, "Bayesian reinforcement learning in continuous pomdps with gaussian processes," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 2604–2609.

[27] A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte, "Parametric pomdps for planning in continuous state spaces," *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 887–897, 2006.

[28] E. Zhou, M. C. Fu, and S. I. Marcus, "Solving continuous-state pomdps via density projection," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1101–1116, 2010.

[29] P. L. Durango-Cohen, "Maintenance and repair decision making for infrastructure facilities without a deterioration model," *Journal of Infrastructure Systems*, vol. 10, no. 1, pp. 1–8, 2004.

[30] P. Zhang, X. Zhu, and M. Xie, "A model-based reinforcement learning approach for maintenance optimization of degrading systems in a large state space," *Computers & Industrial Engineering*, vol. 161, p. 107 622, 2021.

[31] M. Memarzadeh and M. Pozzi, "Model-free reinforcement learning with model-based safe exploration: Optimizing adaptive recovery process of infrastructure systems," *Structural Safety*, vol. 80, pp. 46–55, 2019.

[32] S. Wei, Y. Bao, and H. Li, "Optimal policy for structure maintenance: A deep reinforcement learning framework," *Structural Safety*, vol. 83, p. 101 906, 2020.

[33] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[34] L. Meng, R. Gorbet, and D. Kulić, "Memory-based deep reinforcement learning for pomdp," *arXiv preprint arXiv:2102.12344*, 2021.

[35] M. Egorov, "Deep reinforcement learning with pomdps," Tech. Rep.(Technical Report, Stanford University, 2015), Tech. Rep., 2015.

[36] R. Rocchetta, L. Bellani, M. Compare, E. Zio, and E. Patelli, "A reinforcement learning framework for optimal operation and maintenance of power grids," *Applied energy*, vol. 241, pp. 291–301, 2019.

[37] D. Y. Yang, "Adaptive risk-based life-cycle management for large-scale structures using deep reinforcement learning and surrogate modeling," *Journal of Engineering Mechanics*, vol. 148, no. 1, p. 04 021 126, 2022.

[38] N. Zhang and W. Si, "Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks," *Reliability Engineering & System Safety*, vol. 203, p. 107 094, 2020.

[39] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[41] L. Pinciroli, P. Baraldi, G. Ballabio, M. Compare, and E. Zio, "Optimization of the operation and maintenance of renewable energy systems by deep reinforcement learning," *Renewable Energy*, vol. 183, pp. 752–763, 2022.

[42] Y. Chen, Y. Liu, and T. Xiahou, "A deep reinforcement learning approach to dynamic loading strategy of repairable multistate systems," *IEEE Transactions on Reliability*, 2021.

[43] C. Andriotis and K. Papakonstantinou, "Managing engineering systems with large state and action spaces through deep reinforcement learning," *Reliability Engineering & System Safety*, vol. 191, p. 106 483, 2019.

[44] ——, "Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints," *Reliability Engineering & System Safety*, vol. 212, p. 107 551, 2021.

[45] Y. Zhou, B. Li, and T. R. Lin, "Maintenance optimisation of multicomponent systems using hierarchical coordinated reinforcement learning," *Reliability Engineering & System Safety*, vol. 217, p. 108 078, 2022.

[46] A. Kamariotis, E. Chatzi, and D. Straub, "Value of information from vibration-based structural health monitoring extracted via bayesian model updating," *Mechanical Systems and Signal Processing*, vol. 166, p. 108 465, 2022.

[47] E. Simoen, G. De Roeck, and G. Lombaert, "Dealing with uncertainty in model updating for damage assessment: A review," *Mechanical Systems and Signal Processing*, vol. 56, pp. 123–149, 2015.

[48] A. Lye, A. Cicirello, and E. Patelli, "Sampling methods for solving bayesian model updating problems: A tutorial," *Mechanical Systems and Signal Processing*, vol. 159, p. 107 760, 2021.

[49] M. D. Hoffman, A. Gelman, *et al.*, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[50] J. Van Noortwijk and M. Pandey, "A stochastic deterioration process for time-dependent reliability analysis," in *Reliability and optimization of structural systems*, CRC Press, 2020, pp. 259–265.

[51] J. M. van Noortwijk, "A survey of the application of gamma processes in maintenance," *Reliability Engineering & System Safety*, vol. 94, no. 1, pp. 2–21, 2009.

[52] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, e55, 2016.

[53] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[54] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical programming*, vol. 120, no. 1, pp. 221–259, 2009.

[55] M. C. O. Keizer, S. D. P. Flapper, and R. H. Teunter, "Condition-based maintenance policies for systems with multiple dependent components: A review," *European Journal of Operational Research*, vol. 261, no. 2, pp. 405–420, 2017.

[56] A. Grall, C. Bérenguer, and L. Dieulle, "A condition-based maintenance policy for stochastically deteriorating systems," *Reliability Engineering & System Safety*, vol. 76, no. 2, pp. 167–180, 2002.

[57] G. Barone and D. M. Frangopol, "Reliability, risk and lifetime distributions as performance indicators for life-cycle maintenance of deteriorating structures," *Reliability Engineering & System Safety*, vol. 123, pp. 21–37, 2014.

[58] Q. Li, C. Wang, and B. R. Ellingwood, "Time-dependent reliability of aging structures in the presence of non-stationary loads and degradation," *Structural Safety*, vol. 52, pp. 132–141, 2015.

[59] EN 1993-1-1 (2005). "Eurocode 3: Design of steel structures - Part 1-1: General rules and rules for buildings", The European Union Per Regulation 305/2011, Directive 98/34/EC, Directive 2004/18/EC.

[60] A. Cicirello and R. S. Langley, "Efficient parametric uncertainty analysis within the hybrid finite element/statistical energy analysis method," *Journal of Sound and Vibration*, vol. 333, no. 6, pp. 1698–1717, 2014.

[61] Z. Wang, V. Bapst, N. Heess, *et al.*, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[62] EN 1998-1 (2004). "Eurocode 8: Design of structures for earthquake resistance – Part 1: General rules, seismic actions and rules for buildings", The European Union Per Regulation 305/2011, Directive 98/34/EC, Directive 2004/18/EC.