Multi-objective Energy Optimization Of Locomotions For Underwater Snake Robot

Kenny Duy Luong

Master of Science in Cybernetics and Robotics
Submission date: July 2017
Supervisor: Kristin Ytterstad Pettersen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description:

Snake robots are a biologically inspired type of robot with a very high level of agility. They have the ability to move in terrain types that would be impossible to traverse with wheeled or legged robots. In 2003, NTNU and SINTEF started a research program on snake robots, which has developed several prototypes for both ground robots, such as the Mamba robot in the picture, as well as underwater robots. The research group has also developed mathematical models of both high and low complexity to describe the equations of motion of the snake robots, as well as algorithms for guidance and control of the robots.

This project will investigate how genetic algorithms can be used to evolve control system parameters for forward motion control of a snake robot. Genetic algorithms are global, population-based optimization algorithms inspired by the basic concepts of biological evolution and genetics, such as natural selection and inheritance. They are mainly used in optimization and search problems with complex, nonlinear objective functions, and can be applied to a wide range of problems in different areas due to the fact that they make few a-priori assumptions about the problem.

Based on the results from your project work fall 2016:

1. Review the existing literature on multiobjective optimization methods. Select a method from and compare it with NSGA II, using both methodological information from the review and data from multiple optimization runs.

2. Investigate the significance of the initial population on the optimization. In particular, evaluate the use of an individual from a single-objective optimization run.

3. Replace the sinusoidal motion pattern with a more general representation, for instance a Fourier series or a Taylor series, and optimize with respect to the series coefficients.

4. Give a theoretical analysis of the pareto fronts, the scaling function g(n) and the general motion pattern representation from point 3. For example, use curve fitting tools to derive analytical expressions for the various functions, and put the expressions into context.

The report shall be written in English and edited as a research report including Abstract, Introduction with motivation, literature survey, contributions of the project work, and the outline of the report. This is followed by the chapters describing the results of the project work, simulation results and corresponding discussion, and a conclusion including a proposal for further work. Source code should be provided on a CD with code listing enclosed in appendix. It is supposed that Department of Engineering Cybernetics, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work.

Oppgaven gitt: 13.02.2017 Besvarelsen leveres: 18.07.2017 Utført ved Institutt for teknisk kybernetikk Medveiledere ved FFI: Else-Line Malene Ruud, Martin Syre Wiig, Sondre Engebråten, Jonas Moen Medveileder ved NTNU: Eleni Kelasidi

Sammendrag

Mange bilogiske inspirerte Autonome undervannsfartøyer (AUVs) har blitt utviklet gjennom de siste tiårene. Denne oppgaven fokuserer på Autnonome undervannsfartøyer som er biologisk inspirert av slanger, kalt undervannsSlangeroboter. Et velkjent problem med disse undervannsfartøyene, er den langsikitge autonomien. For å oppnå langsikitg autonomi, bruk av energi effektive metoder er nødvending. En rekke studier har basert seg på enkelt-objektiv optimalisering med hensyn på energi effektiviteten på undervannsslangerobotene, men få har blitt gjort med fler-objektiv optimalisering. Denne oppgaven presenter multi-objektiv optimalisering med forskjellige bevegelser av undervannsslangeroboten. Fler-objektiv optimaliseringen som blir presentert, baserer seg på maksimeringen av forover hastigheten, samt minimaliseringen av effekt forbruket på slangen. For beregningen av den effektive slange bevegelsen, to fler-objektiv evolusjonær aloritmer er presentert, kalt Non-dominated Sort Genetic Algorithm II (NSGA-II), og Hypervolume Estimation Algorithm for Multi-objective Optimization (HypE), er brukt. En utfordring med undervannsslangeroboter, er dens evne for tilpasning av forskjellige bevegelses mønstrer. Med forskjellige begevlser, oppstår det nye søke rom for optimaliseringsproblemet. Vi presenterer en studie på to mest vansligste bevegelses mønstrene for undervannsslangeroboten: (i) Lateral bølging (ii) og ål-lignende bevegelse. Dessuten, vi introduserer også en undersøkelse på tre endret bevegelser for undervannsslangeroboten. Hensikten med disse beveglsesmønstrene er å la evolusjons algoritmen selv bestemme utfallet på bevegelsesmønsteret. Fra simulerings resultatene, viser det seg at en av bevegelsene approksimerer en tilnærmet bevelgse for lateral bølging. Denne bevegelsesmønsteret er baset på Fourier-rekker. De oppnådde simulerings resultatene er basert på optimalisering med optimale parametere i evolusjons algoritmen, fra en rekke optimaliseringsforsøk. Siden dette er fler-objektiv optimaliseringsproblemer, så blir resultatet i form av Pareto fronter. Disse Pareto frontene kan brukes til om handling mellom forover hastigheten og effekt forbruket på undervannsslangeroboten. I tillegg til optimaliseringsproblemene, introduserer vi multivariat analyse av simulerings resultatene ved bruk av prinsipial komponent analyse, for å finne likeheter mellom bevegelses mønstrene. Dessuten, ut i fra analysen, kan noen antagelser bli gjort for formen på de endrete bevelgses mønstrene. Basert resultatene fra prinsipial komponent analysene, intrdouserer vi regresjonsmodeller beregnet av minste kvadraters regresjons metode, for predikering av optimale bevegelses parametere med objektiv verdiene fra Pareto fronten.

Nøkkelord

undervannsslangeroboter, Energi effektivitet, Multi-Objective Evolutionary Algorithm, Non-dominated Sort Genetic Algorithm II, Hypervolume Estimation Algorithm for Multi-objective Optimization, Principal Component Analysis, Partial Least Square Regression, Multi-Objective Optimization Problem, Pareto fronter, Bevegelses generatorer.

Abstract

Biologically inspired Autonomous Underwater Vehicles (AUVs) have been developed in the recent decades. This thesis focuses on the AUVs that are biologically inspired by snakes, called Underwater Snake Robots (USRs). A well-known issue of the USRs, or any AUVs, is the long-term autonomy. To achieve this, energy efficient approaches are required. Many studies have considered single-objective optimization problems regarding the energy efficiency of the USR, but almost none with Multi-Objective Optimization Problems (MOPs). This thesis presents MOPs of different locomotions of the USR. The presented MOPs consider the energy efficient optimization of maximizing the forward velocity, while minimizing the power consumption of the USR. For computing the efficient motion patterns, two Multi-Objective Evolutionary Algorithms (MOEAs) called Non-dominated Sort Genetic Algorithm II (NSGA-II), and Hypervolume Estimation Algorithm for Multi-objective Optimization (HypE) are applied. A challenging topic of the USR, is their adaptability of different locomotions. Different locomotions of the USR give rise to different search spaces for optimization. We present simulation studies of the two most common snake locomotions: (i) lateral undulation and (ii) eel-like motion. Furthermore, we also present and investigate three altered motion pattern of the USR. The aim of the altered locomotions is to let the MOEAs generate efficient locomotions through evolutionary, which we do not know the gait of. From the simulation results, it turns out that one of the altered motion pattern approximates a motion similar to the lateral undulation. This motion pattern is generated based on Fourier series. The obtained simulation results are based on optimization with optimal Genetic Algorithm (GA) parameters, found by numerous presimulations of the MOPs. Since this is multi-objective optimization problems, the end results will be in the form of Pareto fronts. These Pareto fronts can be used as trade-offs for selecting the forward velocity and power consumption of the USR. Additional to the optimization results of the MOPs, we present multivariate analysis of the simulation results using Principal Component Analysis (PCA), for finding relationships between the motion patterns. Furthermore, through the analysis, some assumptions on the shape of the altered locomotion can be given. Based on the results from the PCA, we also present regression models computed by Partial Least Square Regression (PLSR) for predicting the optimal gait parameters using the objective values from the Pareto fronts.

Keywords

Underwater snake robot, Energy efficiency, Multi-Objective Evolutionary Algorithm, Non-dominated Sort Genetic Algorithm II, Hypervolume Estimation Algorithm for Multi-objective Optimization, Principal Component Analysis, Partial Least Square Regression, Multi-Objective Optimization Problem, Pareto fronts, Pattern generators.

Table of contents

List of Figures

List of Tables

List of Abbreviations

AUV	Autonomous Underwater Vehicle
GA	Genetic Algorithm
HypE	Hypervolume Estimation Algorithm for Multi-objective Optimization
MOEA	Multi-Objective Evolutionary Algorithm
MOP	Multi-Objective Optimization Problem
NIPALS	Nonlinear Iterative Partial Least Squares
NSGA	Non-dominated Sort Genetic Algorithm
NSGA-II	Non-dominated Sort Genetic Algorithm II
ODE	Ordinary Differential Equation
PC	Principal Component
PCA	Principal Component Analysis
PLSR	Partial Least Square Regression
PSO	Particle Swarm Optimization
RMSE	Root Mean Squared Error
SPEA2	Strength Pareto Evolutionary Algorithm 2
USR	Underwater Snake Robot

Chapter 1

Introduction

1.1 Motivation

In the past few decades, the developement of Autonomous Underwater Vehicle has been of great interest. A survey on design and control of some AUVs are given in [1]. There exist many different assignments accociated with AUVs. In [2], the authors mention some applications with AUVs such as science missions, search and survey. Further application examples are presented in [3], such as environmental monitoring, data collection, instrumentation, subsea site survey, drill support, hydrotesting and commisioning subsea field developments, etc. This thesis will focus on AUVs that are inspired by biological snakes. These AUVs are called Underwater Snake Robot. The USR presented in this thesis will be based on the model given in [4]. The advantage of using an USR is that they can navigate in difficult environments, and since the USR is basically a moving arm manipulator, it can perform several manipulation tasks such as picking and placing objects. Additional, the USR are module based models, which implies that arbitrary number of degrees of freedom of the robot can be achieved by adding or removing modules [5, 6]. Despite all the various advantages and applications of the AUVs, they share a common limitation, i.e., the energy storage and power management [1, 2, 7, 8]. This motivates us to investigate on energy efficient motions for long-term autonomy of the USR. One of the main challenges of USR is their adaptability of different motion patterns, and therefore, case study of finding most energy efficient gait parameters of different locomotions of the USR are presented in this thesis.

1.2 Related work

One of the first developed snake-like robot prototypes was done by Hirose [9]. In the paper, he describes different propulsion experiments, where among them was the common motion of snakes known as the *serpentine* locomotion (lateral undulation). In [5], the authors purpose a closed form of the dynamic equations for the USR, where it can be applied to modern model-based control schemes. Furthermore, the model also takes into account both linear and nonlinear drag forces, the added mass effect, the fluid moments and current effects. Additional to the proposed model, the authors also present a mathematical formulation of the two most common locomotion of the snakes, i.e., the lateral undulation and *anguilliform* (eel-like) motion. In [10], central pattern generator (CPG) is applied to find gait parameters for an amphibious snake-like robot. The considered locomotion of the snake robot where the serpentine and anguilliform for crawling and swimming, respectively. The locomotion is controlled by the CPG, which are neural networks producing coordinated patterns of rythmic acitivity without any rhythmic inputs from sensory feedback or from higher control centers [11]. The CPG is used for online generating the gait parameters amplitude, frequency and total wavelength. Note that, in the paper, optimization is not considered, just trajectory generation and how these gait parameters influence the locomotion speed of the snake robot.

In [12], GA and Particle Swarm Optimization (PSO) are applied for finding optimal swimming gaits in hyper-redundant mechanisms (HRMs), which are snake-like robots. In the paper, the optimization problem is presented as a single-objective problem, with only considering the power consumption. By having a constant velocity, GA and PSO were applied for finding the optimal swimming gait parameters that minimize the total energy over a given distance. The obtained optimal gait parameters favored an anguilliform locomotion when energy recovery was considered, while *carangiform* locomotion was favored when the energy recovery was disregarded. In [13], the authors present a multi-objective optimization scheme for optimizing the energy efficiency of the USR model presented in [4, 5]. The presented MOP has the goal of maximizing the average forward velocity and minimizing the corresponding average power consumption of the USR. In the paper, the multi-objective optimization scheme is constructed by using the weighted-sum method to combine the two objectives into a single criterion function. Then optimization with PSO was applied for different weighting factors, i.e., multiple single-objective optimization is done with different weighting factors. Furthermore, the obtained results in the paper were based on the lateral undulation and eel-like motion. In [14], the authors present multi-objective optmization problem of automatically design and optimized heterogeneous snake-like modular robot. The presented multi-objective goal is to maximize the modular robot forward moving behaviour and minimize the complexity of the snake-modular using MOEAs.

Related work on motion pattern generators is given in [15]. In this paper, the authors present a scheme for generating gait patterns of modular robots. The presented scheme uses the CPG and a gradient-free optimization algorithm referred to as Powell's method [16]. The gait pattern generator is based on the CPG model for producing coordinated oscilliations, and then optimization with Powell's method for obtaining fast locomotions. The optimization in the pattern generator considers only the forward velocity of the modular robot. Another literature on gait pattern generators are given in [17], where the authors use Truncated Fourier series (TFS) with GA for generating bipedal locomotion. The idea of this gait generator is to let GA optimize the gait parmaters that TFS produces. The TFS can be used as an approximation of arbitrary periodic functions, and with the combination of GA, an efficient locomotion may be obtained. In the paper, the presented optimization problem is given as a single-objective problem, where the objective function is based on the walk distance of the robot. In [18], the authors introduce a evolutionary method, called Interactively Constrained Neuro-Evolution (ICONE) for generating a walking behaviour for a physical humanoid robot. The aim of ICONE, is to bias the search space of the problem towards the desired structure. However, the restriction with ICONE , is that, the evolutionary method require domain knowledge and user interaction to restrict the search spaces.

This thesis will focus on the modeling of the USR presented in [5], and thus the MOP presented in [13] will be considered. Instead of converting the MOP into single-objective optimization problems, we use MOEAs to optimize the multi-objective function as it is, which is a vector function consist of the average power consumption and the average forward velocity. Furthermore, two of the presented simulation results will also be based on the two common snake locomotions; (i) lateral undulation and (ii) eel-like motion. The other simulation results are based on three altered motion patterns of the USR, where two of them are based on Fourier series. The purpose of these altered motions is to see if the MOEAs are able to generate different optimal snake locomotions, that consider the energy efficiency of the USR.

1.3 Brief introduction to MOEAs

The field of Genetic Algorithm (GA) was introduced way back in the 1960s, starting with John H. Holland [19, 20]. The idea Holland presented was helpful to many researchers, regarding how to solve complex, single-optimization problems, e.g., problems with a nonlinear objective function or search spaces that were hard to interpret. The concept of GAs, based on the natural evolution of biological organisms, which involves natural selection and reproduction, have proven themselves to be a robust and practical tool for complex search and optimization problems [19, 21, 22]. This follows from that the algorithms are stochastic and derivative-free, and thus make them also flexible and adaptable [23, 24]. Another property of GA is that it always consider a population of solutions and thus parallelism may be introduced to increase the computational speed [25].

In reality, most optimization problems have multi-objectives, and thus many multi-objective optimization methods using GAs, called Multi-objective Optimization Evolutionary Algorithms (MOEAs), have been developed in the past three decades. Professor David E. Goldberg had a huge influence on the development of the current existing MOEAs where he introduced non-dominated ranking and selection. He also suggested the use of a niching technique to spread out the solutions on the Pareto fronts [26, 27]. Thus the selection scheme in most MOEAs consist of; (i) mating selection and (ii) environmental selection. The two of the first groundbreaking, still popular and state-of-the-art elitist MOEAs are the NSGA-II and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [28, 29]. In [30], a comparison of the two mentioned MOEAs on the Multi-objective Environmental/Economic Dispatch shows that the results obtained by SPEA2 are better than NSGA-II regarding convergence and diversity but at the expense of computational time. Furthermore, a study on performance scaling on the two algorithms shows that both have poor performance when the number of objectives are more than three [31]. Another study regarding performance analysis of both algorithms in noisy environments given in [32], shows that NSGA-II has better chance to filter noise. However, in environments with low noise level, SPEA2 slightly beats NSGA-II in performance.

In this thesis, an optimization problem with two objectives is considered for the USR presented in [5]. Since the optimization problem only has two objectives, both NSGA-II and SPEA2 fit for the task as the optimization solver. However, in this thesis, NSGA-II is chosen due to its computational time is better than SPEA2. Furthermore, we will compare the NSGA-II with a more recent developed elitist MOEA called HypE [33]. Another recent developed MOEA to mention is the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) [34]. The concept of this optimizer is to decompose the multi-objective problem into smaller subproblems and then optimize each subproblem by using the information of its neighboring subproblems. However, due to the decomposing of the multi-objective problem, the evaluation of the objective values of each individual solution has to be iterative and thus with a complex multi-objective problem, the computational time can be slow without parallel computing. In this thesis, the dynamic model of the USR is complex, and thus parallel computing is recommended for evaluation of the whole population.

1.4 Contributions of this thesis

The first main contribution of this thesis is to investigate on multi-objective optimization methods for finding energy efficient locomotion of the snake robot presented in [4, 5]. The efficient motion pattern will consider both the average forward velocity and power consumption of the USR. The two mentioned objectives contradicts each other and thus, we present MOPs for different snake motion patterns. So far, the only known previous literature on investigation of energy efficient snake locomotion using multi-objective optimization is given in [13]. The obtained simulation results are based on the two common snake locomotion called lateral undulation and eel-like motion. In respect to this paper, we present two alternative optimization scheme for finding the optimal gait parameters of the two mentioned motion patterns. The two presented multi-objective optimization schemes are the MOEAs called NSGA-II and HypE. Furthermore, we present pres-imulations of finding optimal GA parameters used in the MOEAs for the actual simulation results of each motion pattern.

For the second main contribution of this thesis, we propose three altered locomotions of the USR. These altered motion patterns do not have any well defined formulations. One of the main assumptions of the altered motions, are that they are periodic motions. Furthermore, these locomotions are not inspired by either biological snakes or fishes. The first presented altered motion allow each joint of the USR to follow different signal references with varying amplitudes. While the last two presented motions are based on Fourier series. Previous literatures of using Fourier series for pattern generators are presented in [17, 35]. However, energy efficient multi-objective optimization of the generated motion patterns are not considered in the literatures.

The final contribution of this thesis is a multivariate analsyis of the simulation results of each presented motion pattern. The goal of this analysis is to find relationships between the motion patterns. Furthermore, based on the multivariate analysis of the simulation results, we present regression models that predict the optimal gait parameters of the motion patterns. The aim of the these regression models is to use them as a utility for obtaining optimal gait paramters based on the desired average forward velocity and power consumption of the USR.

1.5 Outline of this thesis

The outline of this thesis is as follows. **Chapter 2** presents the theoretical background used in this thesis. The chapter gives an introduction to the theoretical background of GA, MOP, MOEA, Fourier series and multivariate analysis. **Chapter 3** presents the dynamic model of the USR and its mathematical terms. Furthermore, the chapter presents the mathematical formulation of the lateral undulation and eel-like motion, and also the formulation of the average forward velocity and power consumption of the USR. In **Chapter 4**, the MOPs of each motion pattern is presented. The chapter also presents the implementation of NSGA-II and HypE. Furthermore, the simulation study of this thesis is also given in this chapter. The simulation results of this thesis will be presented in **Chapter 5**, where we introduce different simulation cases of each motion pattern. Finally, the conclusions and future work will be given in **Chapter 6**.

Chapter 2

Theoretical background

In this chapter, the theoretical background for MOEAs will be introduced. The first two Section 2.1 and 2.2, will be a brief introduction to natural selection and genetics followed by the interpretation of Genetic Algorithm. In section Section 2.1, some terms used in GA will be presented, and in Section 2.2 the concept of GA will be explained. The concept of GA will be the backbone for the two MOEAs called NSGA-II and HypE presented in Section 2.4 and 2.5, respectively. A brief introduction to Fourier series will also be given in Section 2.6. Finally, multivariate analysis tools used for interpreting data sets will be presented in Section 2.7.

2.1 A brief inntroduction to natural selection and genetics

According to Charles Darwin, biological species evolve based on the principle of the natural selection, i.e., the fittest individuals have the best chance of surviving and thus pass their genetic traits to the future generations [36]. To illustrate the natural selection as Darwin did, we consider a population of brown and red foxes and assume that in overall the brown foxes are the stronger individuals. Furthermore, assume a strict environment with few preys to feed on. Therefore, not all individuals get to reproduce themselves. By the principle of natural selection, the brown foxes have the highest probability of surviving and to pass their genetic traits as offsprings. These offsprings will, thus, inherit fit genetic traits from the previous generation and will possibly survive and forward these traits to the future generations. Over successive generations, eventually, the entire population will consist of only brown foxes. This inspired John H. Holland to create an algortithm called GA, which is based on a simplified version of the natural evolution of biological organisms [19]. In the next paragraph, a quick introduction to basic genetics and related terms used in GA is given.

In nature, a set of rules is encoded in the *genes* and connected into long strings called *chromosomes*. A chromosome is a DNA molecule and is a fraction of all the genetic material called *genome*. The genome is the set of all genes that define the traits of an individual, and each gene has an unique position on the chromosome called *locus*. Furthermore, each gene on the genome represents a distinct trait in biology called *allele*. At each locus, there exist a composition of two genes, one inherited from each parent. In genetics, we call this gene composition the *genotype* and the observable setting of the genotype as the *phenotype* [22, 37]. An illustration of the genetic structure is shown in Figure 2.1.

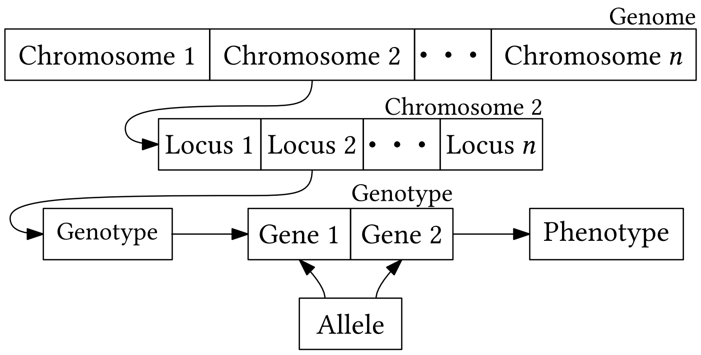


Figure 2.1: Structure of the genome

Example 2.1.1. Consider a genotype that represents the eye color. Furthermore, assume that the allele of the genes are either the color brown or blue. The genotype may therefore have four different settings, i.e, brown/brown, blue/blue, brown/blue and blue/brown. The observable setting (pheontype) of this genotype is either the eye color brown or blue. Note that even if we have four different settings of the genotype, only one of the two traits are observable. Further, assume that the allele associated with brown color dominates the one with blue color. This gives the genotype with the setting brown/blue or blue/brown a phenotype with brown color. Thus, the chances for the eye color to be brown and for it to be blue are 75% and 25%, respectively.

2.2 Genetic algorithm — single objective optimization

The natural evolution is the change of individuals over successive generations in terms to genetic change, mutation and natural selections. This gives the foundation for the concept of GA: start with an arbitrary population of individuals (in this thesis called solutions), and from there, over successive generations, a single fit individual is computed. Table 2.1 shows the comparison between GA and the genetic terms introduced in Section 2.1 [25, p. 20]. Note that there are no

Table 2.1: Comparison of natural evolution and GA

Natural Evolution	Genetic Algorithm
Genome	Chromosome
Chromosome	String
Locus	String positition
Gene	Trait
Allele	Trait value
Genotype	Setting
Phenotype	Decoded setting

difference between the genome and the chromosome in GA. In this section, the schemes called *mating selection* and *variation* will be introduced [38, p. 6], which in combination builds up the Genetic Algorithm. The Genetic Algorithm is commonly used to solve complex single objective optimization and search problems. This thesis focus on optimization problems of the form

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_{opt}(\boldsymbol{x}), \text{ for } \boldsymbol{x} \in \mathcal{X} \\ \text{subject to} & \begin{cases} \boldsymbol{c_i(x) = 0}, & i \in \mathcal{E} \\ \boldsymbol{c_i(x) \leq 0}, & i \in \mathcal{I} \end{cases}, \end{aligned} \tag{2.1}$$

where $\mathcal{X}$ is the decision space, $\boldsymbol{x}$ is the solution, $f_{opt}(\boldsymbol{x}) \in \mathbb{R}$ is the objective function, $\mathcal{E}$ is the set for all equality constraints and $\mathcal{I}$ is the set for all inequality constraints. Note that we can tranform a maximization into a minimization problem by taking $\max_{\boldsymbol{x}} f_{opt}(\boldsymbol{x}) = \min_{\boldsymbol{x}} - f_{opt}(\boldsymbol{x})$. Next, we define the search space of GA in terms of the problem given in (2.1) as

$$\mathcal{D} = \left\{ x \in \mathcal{X} | \begin{array}{c} \boldsymbol{c_i(x) = 0}, \ \forall i \in \mathcal{E} \\ \boldsymbol{c_i(x) \leq 0}, \ \forall i \in \mathcal{I} \end{array} \right\} \subset \mathcal{X} \tag{2.2}$$

which equals the feasible region of (2.1) and contains all multisets over $\mathcal{X}$. Every point in the search space represents a solution in GA, which can further be highlighted by a *fitness* value [25]. The fitness value determines how good a solution is with respect to the optimization problem. The goal in (2.1) is to find the *global optimum* [1] of the objective function subject to the given constraints. However, in most optimization cases, especially GA (a stochastic approach), often returns a *local optimum* [2] [39, Ch. 7]. In fact, it is not possible or very difficult to determine the global optimum. Thus, the algorithm returns an approximation of the true global optimum. Some local optimums can be avoided by selecting different starting positions. This may or may not assist the solutions to reach a better approximation.

2.2.1 The genetic representation

In GA, the chromosomes are the genetic representation of the solutions $\boldsymbol{x}$ in (2.1), made of genes that represent the elements in $\boldsymbol{x}$. Usually, fixed length binary strings are used to encode both chromosomes and genes. Fixed size of chromosomes makes certain genetic operations simpler to compute. To further explain the genetic representation, an example is given below.

Example 2.2.1. Consider an 8x8 grid shown in Figure 2.2. In this grid, a vehicle is located on the green field and can move up, down, left and right. The red field is the desired destination. The optimization problem here is to minimize the number of steps the vehicle takes from its starting position to its desired destination, and can be formulated as

$$\min_{\boldsymbol{x}} \quad f_{opt}(\boldsymbol{x}) = x_1 + x_2 + x_3 + x_4 \tag{2.3}$$

$$\text{subject to} \begin{cases} x_1 - 7 \leq \ 0 \\ x_2 - 7 \leq \ 0 \\ x_3 - 7 \leq \ 0 \\ x_4 - 7 \leq \ 0 \end{cases}, \tag{2.4}$$

[1]The global optimum $\boldsymbol{x}^*$ implies that $f_{opt}(\boldsymbol{x})$ is smallest in the whole search space $\mathcal{D}$.

[2]The local optimum $\boldsymbol{x}^*$ implies that $f_{opt}(\boldsymbol{x})$ is smallest in its neighbourhood $\mathcal{N}$.

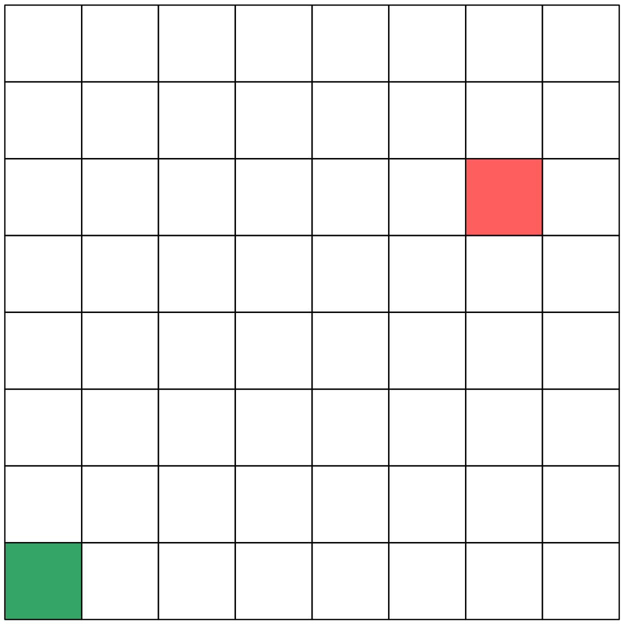


Figure 2.2: Grid example: green = vehicle and red = desired destination

where x_1 = up, x_2 = down, x_3 = left and x_4 = right. The number of steps is constrained to the maximum of seven steps in each direction. This gives us a search space $\mathcal{D}$ that covers the entire grid. Note that this is a simplified problem and thus the formulation of (2.3) includes no consideration for when the vehicle moves outside the grid.

To represent the solution $\boldsymbol{x}$ in the form of a chromosome, determine the traits that each gene represents. For this problem, it is intuitive to let the genes be x_1, x_2, x_3 and x_4, which are the directions of the vehicle. The binary string length of each gene should be long enough to cover the search space $\mathcal{D}$. In this case, a 3-bit long binary string ($2^3 - 1 = 7$) is sufficient. The representation of the chromosome has many different composition of the genes (genotype). Three possible compositions are $concat(x_{b,1}, x_{b,2}, x_{b,3}, x_{b,4})$, $concat(x_{b,3}, x_{b,2}, x_{b,4}, x_{b,1})$ and $concat(x_{b,4}, x_{b,1}, x_{b,2}, x_{b,3})$, where the subscript b denotes the binary string representation, and $concat(\cdot)$ concatenates the strings inside its argument. In this example, we let $concat(x_{b,1}, x_{b,4}, x_{b,2}, x_{b,3})$ be the encoded setting of the chromosome. The binary length of the chromosome is the binary length of each gene combined together and thus the length is $4 \cdot 3$-bit = 12-bit long. Note that all possible settings will give the same binary string length. The importance of a setting is to keep an order of the location of each gene on the chromosome (locus).

By inspecting the grid in Figure 2.2 and the constraints (2.4), the min and max value of (2.3) can be observed as 11 and 28, respectively. The fit solution, which is the min value, is to move 6 steps in the right and 5 steps in the up direction. Thus, the representation for the fit chromosome is therefore given as

$$101\ 110\ 000\ 000,$$

while the worst case solution is

$$111\ 111\ 111\ 111.$$

By using Table 2.1, we can summerize the results above in terms of genetics shown in Table 2.2.

Table 2.2: The results in terms of genetics

Natural Evolution	Genetic Algorithm
Chromosome	12-bit binary string
Gene	up, down, left, right
Allele	0-7
Genotype	$x_{b,1},\ x_{b,4},\ x_{b,2},\ x_{b,3}$
Phenotype	$\boldsymbol{x} = [x_1, x_2, x_3, x_4]^{\mathrm{T}}$

2.2.2 The mating selection and variation scheme

In GA, the natural evolution consists of the schemes called mating selection and variation, and in each generation, these schemes are performed on the population $\mathcal{P}$. There exist many different genetic operators that build up the natural evolution in GA [40]. However, this thesis will focus on the operators listed below.

1. Mating selection:
   - Binary tournament selection
2. Variation:
   - Multipoint crossover
   - Bit-flip mutation

2.2.2.1 The binary tournament selection

In this operator, two arbitrary selected solutions in $\mathcal{P}$ compete for the variation pool, and respect to the principle of the natural selection; the fittest solution wins the competition. The selection of the two solution in $\mathcal{P}$ is done randomly, which is important. If the selection was always the fittest solutions, the algorithm might neglect solutions with valuable genetic material that can aid the convergence to the global optimum. Furthermore, with only selecting the fittest solution, the algorithm will turn into a greedy optimization with increased chance of settling only on local optimums. The binary tournament selection is defined as

1. Select two random solutions in $\mathcal{P}$
2. Compare the two solutions and select the fittest one

The selected solution may be neglected in the next selection of parents to ensure that this solution will only be chosen once for variation in a generation.

2.2.2.2 The multipoint crossover

The crossover is an operation that combine the genetic material between two parents to produce offsprings. In [40], many different methods to do crossover are presented. However, this thesis will focus on the *multipoint crossover.* The multipoint crossover selects $2k$ arbitrary points on the chromosome, where $k = \{1, 2, 3, \ldots\}$, and then do crossing in the selected sections. Note that the

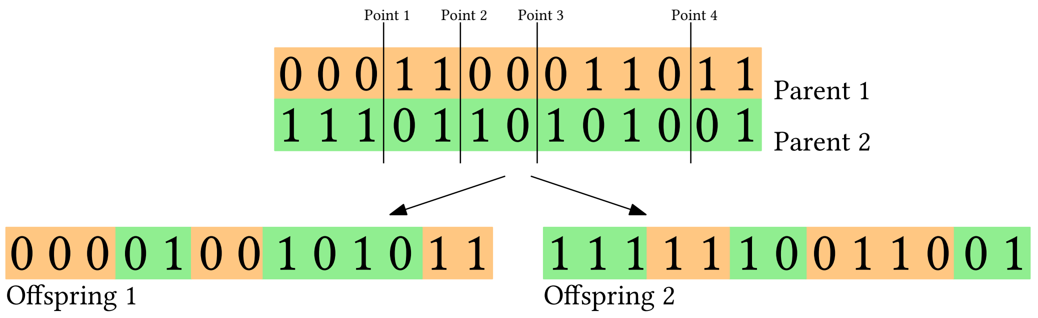


Figure 2.3: The multipoint crossover

operator selects arbitrary points, and so with fixed sized chromosomes, the process for crossing will be more intuitive. The multipoint crossover with $k = 2$ is illustrated in figure 2.3.

Observe that with $k = 1$, the method will be identical to two-point crossover in [40]. The crossover operators need a parameter called *the crossover rate*. This parameter is a probability parameter with float value from 0 to 1. The crossover rate indicates the percentage of the population $\mathcal{P}$ that underwent crossing in the previous generation and is usually set between 0.6 and 1.0 [41].

2.2.2.3 The bit-flip mutation

When two parents produce offsprings, there is a probability for the offsprings to mutate, which is an alteration of the genetic material. In [40], introduces different ways of mutating in GA. However, this thesis will focus on the mutation operator called the *bit-flip mutation*. The mutation operator selects a single arbitrary bit on the chromosome and flips it from $1 \rightarrow 0$ or $0 \rightarrow 1$. Figure 2.4 shows the illustration of the bit-flip mutation.

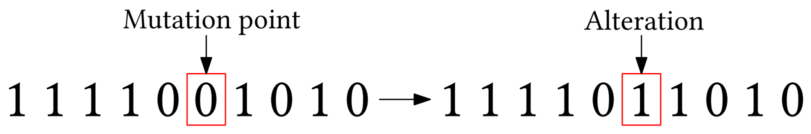


Figure 2.4: The bit-flip mutation

The operator includes a parameter called *the mutation rate*. It indicates the probability for a mutation to occur after a crossover. Furthermore, the mutation rate is usually set between 0.001 and 0.010 [42]. A way to interpret the mutation rate is to look at the parameter as a trade-off between *exploitation* and *exploration*. What this implies is that for low values of the mutation rate, we consider more exploitation, which means to use the currently best-known information for convergence to an optimum. As for exploration, we consider solutions in the search space $\mathcal{D}$ that have not been explored yet, that are solutions not in the current population $\mathcal{P}$.

When the algorithm has reached a fit solution $\boldsymbol{x}^*$, all solutions in $\mathcal{P}$ will have the same genetic material, which means that all the chromosomes are identical. At this point, further iterations in the algorithm will not yield any new results, since crossing between two identical parents will produce offsprings with the same genetic information. Thus, for locally $\boldsymbol{x}^*$, the mutation operator may help GA to escape its local optimum by applying perturbation in the form of genetic alteration. However, a mutation operator with high mutation rate will make the algorithm become a random search optimizer. The cause of this behavior is due to frequent genetic alteration acts as a random noise in the optimization.

2.2.3 The fitness function

The fitness function evaluates how good the solutions are in (2.1). If the objective function in (2.1) is sufficient as a fitness function, one could, therefore, use the function as it is for evaluation of the solutions. There are many ways of defining a fitness function. However, it is not always clear on what a good fitness function is. To further explain this, let us go back to example 2.2.1. The given objective function (2.3) is not sufficient for evaluation of $\boldsymbol{x}$. With this objective function, the fittest solution x^* is for the vehicle to stand still in its starting point, that is $x_1 = 0, x_2 = 0, x_3 = 0$, and $x_4 = 0$. As for the choice of the fitness function, there can be many variants, but a simple one can be a function that punishes the vehicle with a high value if it does not reach the desired destination. Another one is to add the euclidean distance between the vehicle and the desired destination as a punish value. This gives a way of dealing with the constraints given in (2.1), and is called the *penalty function.* With penalty functions, infeasible solutions $\boldsymbol{x} \notin \mathcal{D}$ can be punished. Thus, a fitness function can be denoted as

$$f_{fit}(\boldsymbol{x}) = f_{opt}(\boldsymbol{x}) + f_{penalty}(\boldsymbol{x}) \tag{2.5}$$

In [43], different penalty functions are presented and can be used in GA. In the following Section 2.4, a MOEA called NSGA-II is presented and has a fitness function in the form of a rank system.

2.2.4 The Genetic Algorithm framework

In this section we will introduce the framework of GA. The implementation of GA is illustrated in Figure 2.5. The algorithm starts with evaluating an initialized population $\mathcal{P}$. The initialized

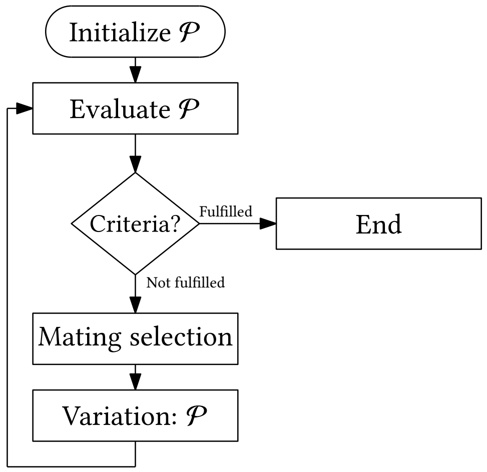


Figure 2.5: The framework of genetic algorithm.

population can be fully randomized, or it can include some fit solutions of the problem. After evaluating the solutions, the algorithm checks if the optimization criteria are fulfilled. If the criteria are satisfied, then stop the optimization. If not, the population will undergo a natural evolution, and a new generation is generated. The new population $\mathcal{P}$ will then be evaluated, and the process repeats itself. In GA, the optimization criteria can be the desired number of generations reached.

2.3 Multi-objective optimization

The outline for this section is to briefly introduce MOP and definitions that can later be used in solving such problems. In general, it is normal to have multiple objectives in an optimization problem. These objectives usually contradict each other, and therefore not possible to find a single solution. Wang et al. introduce a MOP for green supply chain network design, where the motivation is to reduce the CO_2 emission in companies [44]. The MOP is the trade-off between total cost and CO_2 emissions. The objectives contradict because a reduction of emissions will increase the cost.

In this thesis, the multi-objective problem is denoted in the same way as for a single objective problem in Section 2.2

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{f}_{opt}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots, f_m(\boldsymbol{x})] \\ \text{subject to} & \begin{cases} \boldsymbol{c}_i(\boldsymbol{x}) = 0, & i \in \mathcal{E} \\ \boldsymbol{c}_i(\boldsymbol{x}) \leq 0, & i \in \mathcal{I} \end{cases}, \end{aligned} \tag{2.6}$$

where $\boldsymbol{f}_{opt}(\boldsymbol{x}) \in \mathcal{Z} = \mathbb{R}^m$, $\mathcal{Z}$ is the objective space and m is the number of objective functions. In MOP, the concept of optimality is not a single solution, but rather a set of *non-dominated* solutions. The terms dominance and Pareto optimal are defined by

Definition 2.3.1. (Pareto dominance) Consider vector $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$. The vector $\boldsymbol{u}$ is said that to dominate $\boldsymbol{v}$ if and only if

1. All elements in $\boldsymbol{u}$ are lesser or equal $\boldsymbol{v}$, i.e.

$$u_i \leq v_i \forall i \in \{1, 2, \dots, n\} \quad \text{and} \tag{2.7}$$

2. At least one element in u is strict lesser than v, i.e.

$$\exists i \in \{1, 2, \dots, n\} : u_i < v_i \tag{2.8}$$

Definition 2.3.2. (Pareto optimal) A given solution $\boldsymbol{x} \in \mathcal{D}$ is said to be Pareto optimal in the domain $\mathcal{D}$ if and only if there exists no $\boldsymbol{y} \in \mathcal{D}$ such that $\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{y}) \in \mathbb{R}^n$ dominates $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^n$

From definition 2.3.2 [45], both the solution $\boldsymbol{x}$ and the vector $\boldsymbol{u}$ are called non-dominated. The terms local and global optimum introduced in Section 2.2 will now be in the form of Pareto local and global optimum front (set). The optimality of Pareto fronts is defined as

1. A set of solutions $\mathcal{P}$ is called Pareto approximation front, if for every fit solution $\boldsymbol{x}^* \in \mathcal{P}$ there exist a small neighbourhood $\mathcal{N}$, such that, no $\boldsymbol{x} \in \mathcal{N}$ dominates $\boldsymbol{x}^*$.

2. A set of solutions $\mathcal{P}$ is called Pareto optimal front, if there exists no solution $\boldsymbol{x} \in \mathcal{D}$, that dominates any member of $\boldsymbol{x}^* \in \mathcal{P}$, where $\mathcal{D}$ is called the search space.

In MOP, the diversity of $\mathcal{P}$ is important. Thus, we want to have the population $\mathcal{P}$ to cover as much of the search space $\mathcal{D}$, rather than around a single solution. As stated in [46], the goal in MOP is to try to achieve Pareto optimal, but also keep the diversity in $\mathcal{P}$. Furthermore, the term Pareto optimal and approximation is like global and local optimum in single-objective optimization problem, respectively [47]. Unlike GA, the MOEA computes an approximation that is hopefully close to the Pareto optimal front.

2.4 The NSGA-II Multi-Objective Evolutionary Algorithm

This section presents a MOEA called Non-dominated Sort Genetic Algorithm II. The NSGA-II includes a fast non-dominated sort algorithm that ranks the solutions in the population by its non-dominance given in section 2.3. As highlighted in [48], its predecessor Non-dominated Sort Genetic Algorithm (NSGA) had a high computational complexity for the sorting of non-dominated solutions, lack of elitism and the need for a sharing parameter to keep diversity in the population which needed tuning. This motivated the researchers in the paper to develop NSGA-II, introducing elitism to improve the convergence towards an approximation of the Pareto optimal front[48, 49]. The elitism is done by storing all non-dominated solutions in each generation. Furthermore, A crowding distance was also introduced, which estimates the density to keep the diversity in the population. The crowding distance is parameterless unlike the sharing parameter in NSGA.

2.4.1 A fast non-dominated sorting

The concept of NSGA-II is to divide the solutions in $\mathcal{D}$ into different pareto fronts $\mathcal{F}_i$ with the help of a non-dominated sorting. The solutions that are least dominated will be placed into the lower fronts starting from $\mathcal{F}_1$. Thus, for a given population $\mathcal{P}$ with N solutions, in worst case there exist a maximum of N pareto fronts. The case when one obtains N pareto fronts is when all the solutions dominate each other, and thus leads to only one solution existing in each front.

To find out whether a solution $\boldsymbol{x}_i$ is dominated, $\boldsymbol{x}_i$ is compared with every other solution $\boldsymbol{x}_j$ in $\mathcal{P}$. The fast non-dominated sorting algorithm calculates two entities for each solution in $\mathcal{P}$ [48]. The calculated entities are

1. n_i - the number of solutions which dominate the solution i
2. $\mathcal{S}_i$ - a set of solutions which solution i dominates

The lowest level Pareto front $\mathcal{F}_1$ will therefore consist of solutions that have $n_i = 0$, which states that there are no solutions in $\mathcal{P}$ that dominate solution $\boldsymbol{x}_i$. When the first front is calculated, take $\mathcal{F}_1$ and iterate through each solution $\boldsymbol{x}_j$ in its set $\mathcal{S}_i$ and decrease n_j by one. For each entity n_j that reaches 0, place solution $\boldsymbol{x}_j$ into a set $\mathcal{H}$. When all the solutions have been iterated through, let the set $\mathcal{H}$ be a new Pareto front $\mathcal{F}_2$ which is one level higher. For the following Pareto fronts, repeat the process with the new computed front. The non-dominated sorting is completed when $\mathcal{H} = \varnothing$.

Table 2.3: An example population

Solution	Objectives	
$\boldsymbol{x}_i$	f_1	f_2
$\boldsymbol{x}_1$	1	3
$\boldsymbol{x}_2$	7	2
$\boldsymbol{x}_3$	8	6
$\boldsymbol{x}_4$	9	3
$\boldsymbol{x}_5$	3	4
$\boldsymbol{x}_6$	4	4
$\boldsymbol{x}_7$	5	7
$\boldsymbol{x}_8$	9	2

Example 2.4.1. Consider a population with objective values given in Table 2.3. Observe that each solution has two corrsepsonding objective functions f_1 and f_2. With the notation given in Definition 2.3.2, we write $\boldsymbol{u} = f_{opt}(\boldsymbol{x}) = [f_1, f_2]$, and $\boldsymbol{x} \in (\mathcal{P} \subset \mathcal{D})$, where $\mathcal{P} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_8\}$, such that $\boldsymbol{u}_1 = f(\boldsymbol{x}_1) = [1, 3]$ and $\boldsymbol{u}_2 = f(\boldsymbol{x}_2) = [7, 2]$. To compute the non-dominated sort by hand, we first calculate the two entities n_i and $\mathcal{S}_i$ for each $\boldsymbol{x} \in \mathcal{P}$. By comparing $\boldsymbol{x}_1$ with $\boldsymbol{x}_2$, we see that the first element in $\boldsymbol{u}_1$ dominates $\boldsymbol{u}_2$ but the second element in $\boldsymbol{u}_2$ dominates $\boldsymbol{u}_1$, and hence they are non-dominated with each other. For the comparison between $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$, $\boldsymbol{u}_1$ dominates $\boldsymbol{u}_3$ and we place $\boldsymbol{x}_3$ into $\mathcal{S}_1$ and increment the entity n_3 by one. Table 2.4 shows the two entities for each $\boldsymbol{x}$, after comparing the solutions.

Table 2.4: The computed entities

i	$\mathcal{S}_i$	n_i
1	$\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7\}$	0
2	$\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_8\}$	0
3	$\varnothing$	4
4	$\varnothing$	3
5	$\{\boldsymbol{x}_3, \boldsymbol{x}_6, \boldsymbol{x}_7\}$	1
6	$\{\boldsymbol{x}_3, \boldsymbol{x}_7\}$	2
7	$\varnothing$	3
8	$\boldsymbol{x}_4$	1

Another way to compute the two entities n_i and $\mathcal{S}_i$ is to place the objective values in the euclidean space $\mathbb{R}^2$ and compare the position of each solution. Note that this method is viable since the number of objectives are lesser than three. The location of the solutions in $\mathbb{R}^2$ are shown in Figure 2.6. By taking an arbitrary solution $\boldsymbol{x}_i$, all other solutions $\boldsymbol{x}_j$ are dominated by $\boldsymbol{x}_i$, if their location is in the first quadrant of $\boldsymbol{x}_i$. For instance, let $i = 1$, then by investigating Figure 2.6, solution $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_3$, $\boldsymbol{x}_4$, $\boldsymbol{x}_5$, $\boldsymbol{x}_6$ and $\boldsymbol{x}_7$. Observe that the solution $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ dominates no other solutions and thus $\mathcal{S}_3 = \varnothing$ and $\mathcal{S}_4 = \varnothing$.

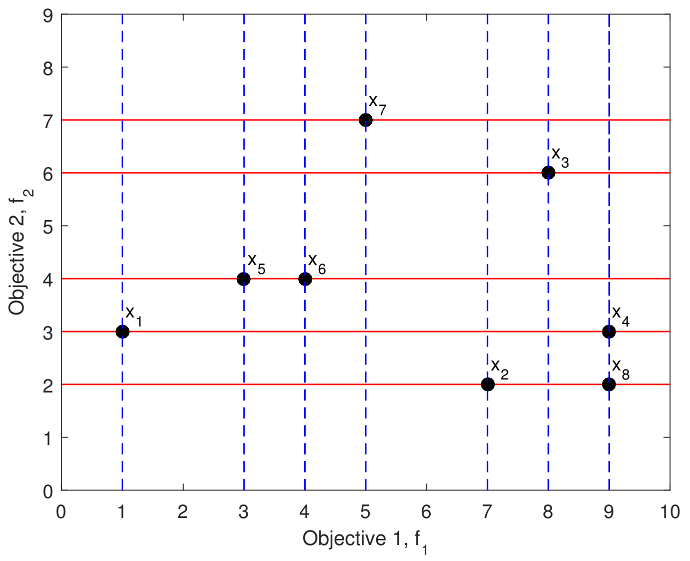


Figure 2.6: The example population in the $\mathbb{R}^2$ space

From Table 2.4, taking solutions with $n_i = 0$, the initial Pareto front $\mathcal{F}_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$. Furthermore, all $\boldsymbol{x}_i \in \mathcal{F}_1$ have a *non-domination rank* equal one, that is $\boldsymbol{x}_{i,rank} = 1$. Note that the non-domination rank is based on the Pareto front level. To obtain the next Pareto front $\mathcal{F}_2$, start decrementing n_j of each solution dominated by $\mathcal{F}_1$, i.e., $\boldsymbol{x}_j \in \mathcal{S}_1$ and $\boldsymbol{x}_j \in \mathcal{S}_2$, then place each solution $\boldsymbol{x}_j$ with $n_j = 0$ into $\mathcal{F}_2$. The following Pareto fronts are computed by repeating the process above with the new computed Pareto front. Table 2.5 shows the calculation of each Pareto front.

2.4.2 The crowding distance

The measurement called *the crowding distance* keeps the diversity in $\mathcal{P}$. This measurement is computed by taking the geometric position of one solution $\boldsymbol{x}_i$ and compare its distance between two closest solutions $\boldsymbol{x}_{i+1}$ and $\boldsymbol{x}_{i-1}$. The crowding distance can be interpreted as a cuboid enclosing the solution $\boldsymbol{x}_i$ and is illustrated in Figure 2.7 [48].

The way to compute the crowding distance is proposed in [50], and is listed below

1. Sort the objective values in ascending order
2. Assign the first and last solution in the sorted set with infinity crowding distance
3. For the remaining solutions, assign the crowding distance given by

$$d_j = d_j + \frac{f_{i,j+1} - f_{i,j-1}}{max(f_i) - min(f_i)}, \tag{2.9}$$

where d_j is the crowding distance of solution $\boldsymbol{x}_j$, and $f_{i,j}$ is the objective value f_i of $\boldsymbol{x}_j$.

Table 2.5: Steps for computing the Pareto fronts. $\mathcal{F}_1$ is the initial front, and following Pareto fronts are computed by decrementing n_j of each solution in $\mathcal{S}_i$. The new Pareto fronts are made of solutions with $n_j = 0$.

i	$\mathcal{S}_i$	n_3	n_4	n_5	n_6	n_7	n_8
	Initial Pareto front $\mathcal{F}_1 = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$						
1	$\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7\}$	3	2	0	1	2	
2	$\{\boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_8\}$	2	1				0
	New Pareto front $\mathcal{F}_2 = \{\boldsymbol{x}_5, \boldsymbol{x}_8\}$						
5	$\{\boldsymbol{x}_3, \boldsymbol{x}_6, \boldsymbol{x}_7\}$	1			0	1	
8	$\{\boldsymbol{x}_4\}$		0				
	New Pareto front $\mathcal{F}_3 = \{\boldsymbol{x}_4, \boldsymbol{x}_6\}$						
4	$\varnothing$						
6	$\{\boldsymbol{x}_3, \boldsymbol{x}_7\}$	0				0	
	Final Pareto front $\mathcal{F}_4 = \{\boldsymbol{x}_3, \boldsymbol{x}_7\}$						

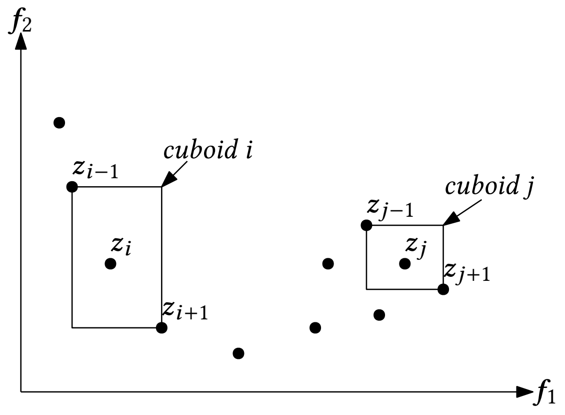


Figure 2.7: Illustrate the crowding distance in an objective space with $m = 2$ objectives. Note that $\boldsymbol{z}_i = f_{opt}(\boldsymbol{x}_i)$.

Example 2.4.2. Let us go back to example 2.4.1. To compute the crowding distance for this population, start sorting the objective values of f_1 in the ascending order. Table 2.6 shows the sorted list. The crowding distances in the table are computed by (2.9). To take an example, the crowding distance of $\boldsymbol{x}_6$ can be written as

$$d_5 = 0 + \frac{5-3}{9-1} = \frac{2}{8} = 0.250. \tag{2.10}$$

When all the crowding distances have been computed with respect to f_1, repeat the process for f_2. The computed crowding distances with f_2 are shown in Table 2.7. For the visualization of the solutions position, see Figure 2.6. The solution $\boldsymbol{x}_3$ has least solutions surrounding it, and thus it has the highest crowding distance. The crowding distance is not infinity since $\boldsymbol{x}_3$ is not a boundary point, which by inspection are $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_7$ and $\boldsymbol{x}_8$.

Table 2.6: The objective f_1 values in the ascending order

$\boldsymbol{x}_i$	f_1	Crowding distance: d_j
$\boldsymbol{x}_1$	1	inf
$\boldsymbol{x}_5$	3	0.375
$\boldsymbol{x}_6$	4	0.250
$\boldsymbol{x}_7$	5	0.375
$\boldsymbol{x}_2$	7	0.375
$\boldsymbol{x}_3$	8	0.250
$\boldsymbol{x}_4$	9	0.125
$\boldsymbol{x}_8$	9	inf

Table 2.7: The objective f_2 values in the ascending order

$\boldsymbol{x}_i$	f_2	Crowding distance: d_j
$\boldsymbol{x}_2$	2	inf
$\boldsymbol{x}_8$	2	inf
$\boldsymbol{x}_1$	3	inf
$\boldsymbol{x}_4$	3	0.375
$\boldsymbol{x}_5$	4	0.575
$\boldsymbol{x}_6$	4	0.650
$\boldsymbol{x}_3$	6	0.850
$\boldsymbol{x}_7$	7	inf

2.4.3 The environmental selection

To ensure diversity in the population, an operator ($\geq_n$) called "the crowded comparison" is introduced [48]. This operator is used to spread out solutions in the Pareto front uniformly. The operator is defined as, if $\boldsymbol{x}_i \geq_n \boldsymbol{x}_j$ then

1. The non-domination rank $\boldsymbol{x}_{i,rank} < \boldsymbol{x}_{j,rank}$, or

2. The non-domination rank $\boldsymbol{x}_{i,rank} = \boldsymbol{x}_{j,rank}$ and the crowding distance $d_i \geq d_j$

Note that the non-domination rank is based on the level of the Pareto front, and therefore solutions in the lower levels are considered better. The crowding operator ($\geq_n$) selects the solution with the highest crowding distance for spreading out the Pareto front. This type of selection scheme is called the *environmental selection*. The reason for its name is that the selection takes into account the location of $\boldsymbol{x}$ in the Pareto front.

2.4.4 The NSGA-II framework

Similar to the GA framework introduced in Section 2.2.4, the NSGA-II framework starts with initializing a population of solutions. However, the difference with NSGA-II is the non-dominated sorting and the environmental selection. Furthermore, it also works for multi-objective problems. Note that in NSGA-II, the fitness function is the non-domination rank and the crowding distance of x. Thus, the mating selection introduced in Section 2.2.2, consider solutions with the best rank (solutions in the lower level Pareto fronts). Furthermore, with the crowding distance, the selection further distinguish between fit solutions, e.g., for solutions with same rank, consider the one with the best crowding distance. The NSGA-II algorithm given in [50, p. 186] is illustrated in Figure 2.8.

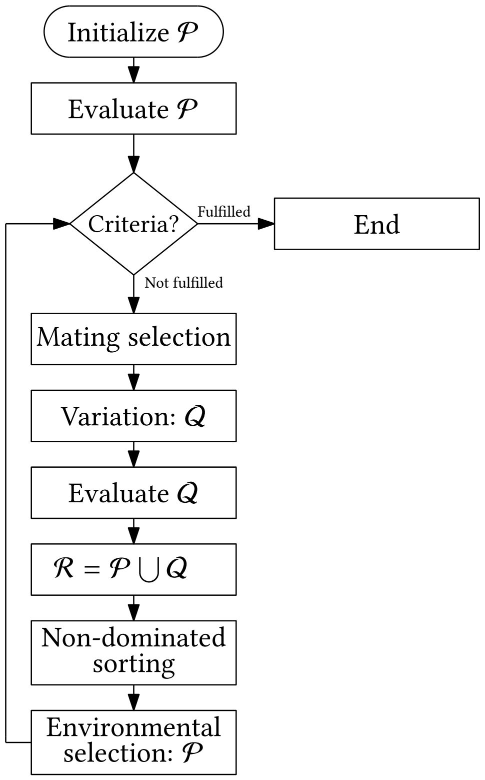


Figure 2.8: The framework of NSGA-II

In NSGA-II, the number of solutions in $\mathcal{Q}$ equals the number of solution in $\mathcal{P}$, i.e., $|\mathcal{Q}| = |\mathcal{P}| = N$, where $|\cdot|$ denotes the cardinality of a set. The new set $\mathcal{Q}$ will later be concatenated with $\mathcal{P}$ to create a set $\mathcal{R}$ with $2N$ solutions. After concatinating the populations, a non-dominated sorting is performed to get the Pareto fronts. By combining the two sets, the algorithm will take both new and old solutions in consideration of generating the new $\mathcal{P}$. The steps for creating $\mathcal{P}$ in the environmental selection are listed below.

1. Let $\mathcal{P} = \varnothing$
2. Select the lowest level Pareto front $\mathcal{F}_i$ (starts with $\mathcal{F}_1$)
3. Assign crowding distance to the solutions.

4. Sort the crowding distance in the descending order.
5. If the number of solutions in $\mathcal{F}_i \bigcup \mathcal{P}$ exceeds the maximum population number N, select solutions with the highest crowding distance and place them inside $\mathcal{P}$ until it has N solutions.
6. Else, place all solutions in the Pareto front into $\mathcal{P}$ and go back to step 2. and redo the process with the next Pareto front level.

With the non-dominated sorting and the environmental selection, the overall complexity of the algorithm is $O(MN^2)$, where M is the number of objectives and N is the population size [51].

2.5 The HypE Multi-Objective Evolutionary Algorithm

This section briefly introduce a MOEA called HypE presented in [33]. Unlike NSGA-II that ranks the solutions with non-dominated sorting, the HypE assign ranks through a measure called the *hypervolume indicator.* The advantage of using hypervolumes is that the measure is known to be the only quality indicator that is fully sensitive to Pareto dominance [33]. This imply that, if a Pareto approximation front $\mathcal{F}_a$ dominates another front $\mathcal{F}_b$, the hypervolume indicator of $\mathcal{F}_a$ is strictly better than $\mathcal{F}_b$. Furthermore, if $\mathcal{F}_a$ achieves the highest possible indicator value, the measure guarantees that $\mathcal{F}_a$ contains all Pareto optimal sets of a given problem [52]. With the properties of the hypervolumes, the HypE algortihm works well for $m > 3$ MOP. However, the worst-case runtime of computing the *exact hypervolme indicator* is $O(N^{m-1})$, which is exponential in the number of objective functions. In [33], an estimation of the hypervolume indicator is introduced for MOP with $m > 3$, based on Monte Carlo simulation. However, this thesis focus on MOP with $m = 2$ and thus the exact hypervolume indicator is considered.

2.5.1 Independent Pareto set dominance

In this section, the notation of Pareto dominance given in Definition 2.3.1 will further be expanded to yield between different sets. The notations are the same as the preference relation ($\preccurlyeq$) given in [33, 52].

Definition 2.5.1. Let $\mathcal{D}$ be the search space, and $\mathcal{F}_a, \mathcal{F}_b \in \mathcal{D}$ be the Pareto approximation sets. Then the set preference relation $\preccurlyeq$ on $\mathcal{D}$ is defined as

$$\mathcal{F}_a \preccurlyeq \mathcal{F}_b :\Leftrightarrow \forall \boldsymbol{y} \in \mathcal{F}_b \; \exists \boldsymbol{x} \in \mathcal{F}_a : \boldsymbol{x} \preceq \boldsymbol{y}, \tag{2.11}$$

where

$$\boldsymbol{x} \preceq \boldsymbol{y} :\Leftrightarrow f_{opt}(\boldsymbol{x}) \leq f_{opt}(\boldsymbol{y}) \text{ for } \boldsymbol{x} \in \mathcal{F}_a \text{ and } \boldsymbol{y} \in \mathcal{F}_b \tag{2.12}$$

Remark 2.5.1. In [33, 52], a term called *weakly dominated* is introduced. This term imply that if $\boldsymbol{u}$ weakly dominates $\boldsymbol{v}$, then $\boldsymbol{u}$ dominates $\boldsymbol{v}$ regardless of which set they belong too. For instance, let $\mathcal{F} \subset \mathcal{D}$, $\boldsymbol{x} \in \mathcal{X}$ and that $\mathcal{F}$ weakly dominates $\boldsymbol{x}$, then with respect to Definition 2.5.1, it is written as $\mathcal{F} \preccurlyeq \{\boldsymbol{x}\}$. Another example, a solution $\boldsymbol{x} \in \mathcal{X}$ weakly dominates objective $\boldsymbol{z} \in \mathcal{Z}$ imply that $f_{opt}(\boldsymbol{x})$ dominates $\boldsymbol{z}$.

2.5.2 Fitness assigning with hypervolume indicator

Bader and Zitzler [33] present a strategy for assigning fitness values that takes into account the entire objective space $\mathcal{Z}$ weakly dominated by a population $\mathcal{P}$, i.e., $\mathcal{P} \preccurlyeq \mathcal{Z} :\Leftrightarrow \forall \boldsymbol{z} \in \mathcal{Z}\ \exists \boldsymbol{x} \in \mathcal{P} : \boldsymbol{x} \preceq \boldsymbol{z} :\Leftrightarrow f_{opt}(\boldsymbol{x}) \leq \boldsymbol{z}$. The fitness values will then be used in the mating selection to distinguish between fit solutions. This section will focus on the defintions and theorems of the the hypervolume indicator based on lebesgue measure [33, Def. III.1].

Definition 2.5.2. (Hypervolume indicator) Let $\mathcal{F} \in \mathcal{D}$ be a Pareto approximation front and $R \subset \mathcal{Z}$ be a reference set of mutually non-dominating objective vectors. Then the hypervolume indicator I_H can be defined as

$$I_H(\mathcal{F}, R) := \lambda(H(\mathcal{F}, R)), \tag{2.13}$$

where

$$H(\mathcal{F}, R) := \{\boldsymbol{z} \in \mathcal{Z} |\ \exists \boldsymbol{x} \in \mathcal{F}, \exists \boldsymbol{r} \in R : \boldsymbol{f}(\boldsymbol{x}) \leq \boldsymbol{z} \leq \boldsymbol{r}\} \tag{2.14}$$

and λ is the Lebesgue measure with

$$\lambda(H(\mathcal{F}, R)) = \int_{\mathbb{R}^m} \mathbf{1}_{H(\mathcal{F},R)}(\boldsymbol{z}) d\boldsymbol{z} \tag{2.15}$$

and $\mathbf{1}_{H(\mathcal{F},R)}$ being the characteristic function of $H(\mathcal{F}, R)$.

Remark 2.5.2. The set $H(\mathcal{F}, R)$ is a subset of $\mathcal{Z}$ and denotes the set of objective vectors enclosed by $f_{opt}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \mathcal{F}$ and R. Furthermore, the set can be further split into partitions $H(S, \mathcal{F}, R)$, that is

$$H(S, A, R) := \Big[\bigcap_{\boldsymbol{s} \in S} H(\{\boldsymbol{s}\}, R)\Big] \Big\backslash \Big[\bigcup_{\boldsymbol{x} \in \mathcal{F} \backslash S} H(\{\boldsymbol{x}\}, R)\Big] \subseteq \mathcal{Z} \tag{2.16}$$

Example 2.5.1. Consider a Pareto approximation $\mathcal{F} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$, $S_1 = \{\boldsymbol{x}_3, \boldsymbol{x}_4\}$, $S_2 = \{\boldsymbol{x}_2\}$ and the reference set $R = \{r\}$. An illustration of the objective space $H(\mathcal{F}, R) = \{\boldsymbol{z} \in \mathcal{Z} |\ \exists \boldsymbol{x} \in \mathcal{F}, \exists \boldsymbol{r} \in R : f_{opt}(\boldsymbol{x}) \leq \boldsymbol{z} \leq \boldsymbol{r}\}$ is given in Figure 2.9.

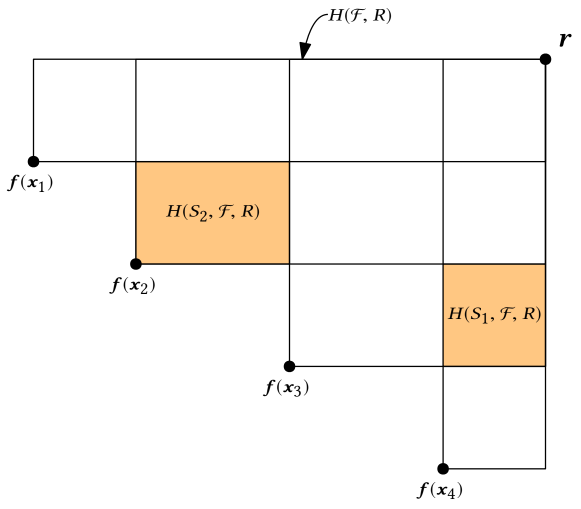

$H(\mathcal{F}, R)$
$\boldsymbol{r}$
$\boldsymbol{f}(\boldsymbol{x}_1)$
$H(S_2, \mathcal{F}, R)$
$\boldsymbol{f}(\boldsymbol{x}_2)$
$H(S_1, \mathcal{F}, R)$
$\boldsymbol{f}(\boldsymbol{x}_3)$
$\boldsymbol{f}(\boldsymbol{x}_4)$

Figure 2.9: Objective space of a Pareto approximation set

The subset $H(S_1, \mathcal{F}, R)$ and $H(S_2, \mathcal{F}, R)$ is defined as

$$\begin{aligned} H(S_1, \mathcal{F}, R) &:= \Big[\bigcap_{s \in S_1} H(\{\boldsymbol{s}\}, R)\Big] \Big\backslash \Big[\bigcup_{\boldsymbol{x} \in \mathcal{F} \setminus S_1} H(\{\boldsymbol{x}\}, R)\Big] \\ &= \Big[H(\{\boldsymbol{x}_3\}, R) \bigcap H(\{\boldsymbol{x}_4\}, R)\Big] \Big\backslash \Big[H(\{\boldsymbol{x}_1\}, R) \bigcup H(\{\boldsymbol{x}_2\}, R)\Big] \end{aligned} \tag{2.17}$$

$$\begin{aligned} H(S_2, \mathcal{F}, R) &:= \Big[\bigcap_{s \in S_2} H(\{\boldsymbol{s}\}, R)\Big] \Big\backslash \Big[\bigcup_{\boldsymbol{x} \in \mathcal{F} \setminus S_2} H(\{\boldsymbol{x}\}, R)\Big] \\ &= \Big[H(\{\boldsymbol{x}_2\}, R)\Big] \Big\backslash \Big[H(\{\boldsymbol{x}_1\}, R) \bigcup H(\{\boldsymbol{x}_3\}, R) \bigcup H(\{\boldsymbol{x}_4\}, R)\Big] \end{aligned} \tag{2.18}$$

and is illustrated as the filled color sections in Figure 2.9. Observe that,

$$\dot{\bigcup_{S \subseteq \mathcal{F}}} H(S, \mathcal{F}, R) = H(\mathcal{F}, R) \tag{2.19}$$

Remark 2.5.3. It is infeasible to determine all distinct $H(S, \mathcal{F}, R)$ due to combinatorial explosion [33, p. 3s]. Thus, the splitting in (2.16) will be extended to a more compact and defined as

$$H_i(\boldsymbol{x}, \mathcal{F}, R) := \bigcup_{\substack{S \subseteq \mathcal{F} \\ \boldsymbol{x} \in S \\ |S| = i}} H(S, \mathcal{F}, R) \tag{2.20}$$

Example 2.5.2. Consider the objective space of the Pareto approximation $\mathcal{F}$ and the reference R given in Example 2.5.1. Then the compact splitting of $H_i(\boldsymbol{x}_2, \mathcal{F}, R)$, for $i = \{1, 2, 3, 4\}$ is defined as

$$H_1(\boldsymbol{x}_2, \mathcal{F}, r) = H(\{\boldsymbol{x}_2\}, \mathcal{F}, R) \tag{2.21}$$

$$H_2(\boldsymbol{x}_2, \mathcal{F}, r) = H(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}, \mathcal{F}, R) + H(\{\boldsymbol{x}_2, \boldsymbol{x}_3\}, \mathcal{F}, R) \tag{2.22}$$

$$H_3(\boldsymbol{x}_2, \mathcal{F}, r) = H(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}, \mathcal{F}, R) + H(\{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}, \mathcal{F}, R) \tag{2.23}$$

$$H_4(\boldsymbol{x}_2, \mathcal{F}, r) = H(\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}, \mathcal{F}, R) \tag{2.24}$$

Figure Figure 2.10 shows an illustration of the splitting.

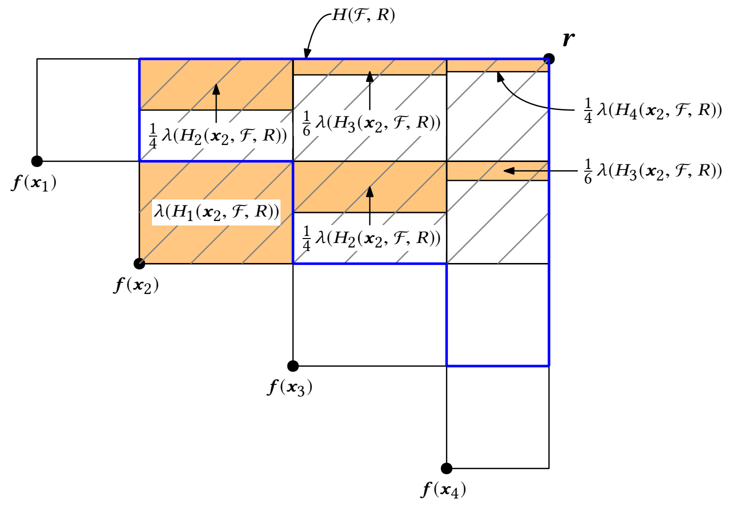
With the definitions and the remarks given above, a scheme can be defined for assigning hypervolume indicator to each solution $\boldsymbol{x}$ in a population $\mathcal{P}$ [33, Def. III.2].

Definition 2.5.3. Let $\mathcal{F} \in \mathcal{D}$ and $R \subset \mathcal{Z}$. Then the function I_h with

$$I_h(\boldsymbol{x}, \mathcal{F}, R) := \sum_{i=1}^{|\mathcal{F}|} \frac{1}{i} \lambda(H_i(\boldsymbol{x}, \mathcal{F}, R)) \tag{2.25}$$

gives for each solution $\boldsymbol{x} \in \mathcal{F}$ the hypervolume that can be assigned to $\boldsymbol{x}$ with regard to the overall hypervolume $I_H(\mathcal{F}, R)$.

Figure 2.10 illustrates the hypervolume indicator of a solution. The following theorem shows that the overall hypervolume $I_H(\mathcal{F}, R)$ is distributed among the distinct solutions [33, Th. III.3].



$H(\mathcal{F}, R)$
$\boldsymbol{r}$
$\frac{1}{4}\lambda(H_4(\boldsymbol{x}_2, \mathcal{F}, R))$
$\frac{1}{6}\lambda(H_3(\boldsymbol{x}_2, \mathcal{F}, R))$
$\frac{1}{4}\lambda(H_2(\boldsymbol{x}_2, \mathcal{F}, R))$
$\frac{1}{6}\lambda(H_3(\boldsymbol{x}_2, \mathcal{F}, R))$
$\lambda(H_1(\boldsymbol{x}_2, \mathcal{F}, R))$
$\frac{1}{4}\lambda(H_2(\boldsymbol{x}_2, \mathcal{F}, R))$
$f(\boldsymbol{x}_1)$
$f(\boldsymbol{x}_2)$
$f(\boldsymbol{x}_3)$
$f(\boldsymbol{x}_4)$

Figure 2.10: The objective space is based on Example 2.5.1 and shows a) the partitions of $H_i(\boldsymbol{x}_2, \mathcal{F}, R)$, for $i = \{1, 2, 3, 4\}$ (shaded area), and b) the assignment of the hypervolume indicator $I_h(\boldsymbol{x}_2, \mathcal{F}, R)$ (color filled area). The area inside the blue lines shows the fixed parts of the hypervolume (not affected if any arbitrary solution $\boldsymbol{x} \in \mathcal{P}$ is removed from $\mathcal{P}$)

Theorem 2.5.4. Let $\mathcal{F} \in \mathcal{D}$ and $R \in \mathcal{Z}$. Then it holds

$$I_H(\mathcal{F}, R) = \sum_{\boldsymbol{x} \in \mathcal{F}} I_h(\boldsymbol{x}, \mathcal{F}, R) \tag{2.26}$$

By using Definition 2.5.3, each $\boldsymbol{x} \in \mathcal{P}$ is assigned with a scalar value that determine their ranks in the population. The scalar value can then be used in the mating selection to determine fit solutions for variation. In [33, Def. III.5], the environmental selection is formulated in terms of the subset selection problem and is defined as.

Definition 2.5.5. (Environmental Selection) Let $\mathcal{F} \in \mathcal{D}$, $R \subset \mathcal{Z}$, and $k \in \{0, 1, \dots, |\mathcal{F}|\}$. The hypervolume subset selection problem (HSSP) is defined as the problem of finding a subset $\mathcal{P} \subseteq \mathcal{F}$ with $|\mathcal{P}| = |\mathcal{F}| - k$ such that the overall hypervolume loss is minimum, i.e,

$$I_H(\mathcal{P}, R) = \max_{\substack{\mathcal{F}' \subseteq \mathcal{F} \\ |\mathcal{F}'| = |\mathcal{F}| - k}} I_H(\mathcal{F}', R) \tag{2.27}$$

From Definition 2.5.5, the parameter k is the number of solutions that will be removed from $\mathcal{F}$. In HypE, the environmental selection resembles the one given in NSGA-II, i.e., both algorithms consider a sorted non-dominated population $\mathcal{R} = \mathcal{P} \bigcup \mathcal{Q}$, which contain $2N$ solutions. The algorithm selects the lowest level pareto fronts $\mathcal{F}_i$ until it exceeds the population size N (ignore higher level fronts). This can be done, since dominated solutions do not affect the overall hypervolume. Figure 2.11 illustates the dominated solutions in the objective space. The k solutions that exceeds N will then be removed based on the environmental selection. The two approaches of the environmental selection can be[33, p. 5]:

1. *Iterative*: Assign hypervolume indicator to the population k times; each time, remove the worst solution (lowest indicator value).

2. *One shot*: Assign hypervolume indicator to the population once and then remove k worst solutions.

In this thesis, we will focus on the iterative approach. There is a drawback of using the scheme

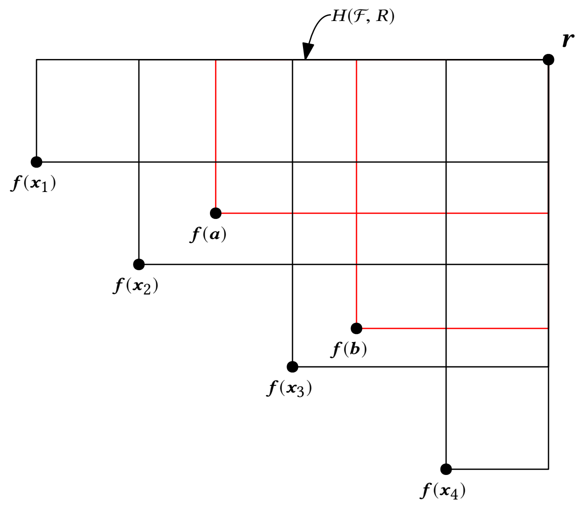


Figure 2.11: An illustration to show that removing the dominated soltuions $\boldsymbol{a}$ and $\boldsymbol{b}$ will not affect the overall hypervolume

given in Definition 2.5.3, and that is the indicator also consider the fixed parts of the hypervolumes. The fixed parts are the portions of the hypervolumes that will not be affected if an arbitrary or multiple solutions are removed from the population. Figure 2.10 illustrates the disadvantage of Definition 2.5.3. In [33, Def. III.6], a new extended scheme is presented to avoid this drawback and is defined as.

Definition 2.5.6. Let $\mathcal{F} \in \mathcal{D}$, $R \subset \mathcal{Z}$, and $k \in \{0, 1, \cdots, |\mathcal{F}|\}$. Then the function

$$I_h^k(\boldsymbol{x}, \mathcal{F}, R) := \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \left[\sum_{\substack{T \subseteq S \\ \boldsymbol{x} \in T}} \frac{1}{|T|} \lambda(H(T, \mathcal{F}, R)) \right] \tag{2.28}$$

where $\mathcal{S} = \{S \subseteq \mathcal{F} \mid \boldsymbol{x} \in S \wedge |S| = k\}$ contains all subsets of $\mathcal{F}$ that include $\boldsymbol{x}$ and have cardinality k gives for each solution $\boldsymbol{x} \in \mathcal{F}$ the expected hypervolume loss that can be attributed to $\boldsymbol{x}$ when $\boldsymbol{x}$ and $k-1$ uniformly randomly chosen solutions from $\mathcal{F}$ are removed from $\mathcal{F}$.

Remark 2.5.4. Observe that $I_h^1(\boldsymbol{x}, \mathcal{F}, R) = \lambda(H_1(\boldsymbol{x}, \mathcal{F}, R))$ and $I_h^{|\mathcal{F}|}(\boldsymbol{x}, \mathcal{F}, R) = I_h(\boldsymbol{x}, \mathcal{F}, R)$. Thus, Definition 2.5.6 can be viewed as a generalization of Definition 2.5.3. With $k = 1$, it is sufficient to assign $I_h^1(\boldsymbol{x}, \mathcal{F}, R) = \lambda(H_1(\boldsymbol{x}, \mathcal{F}, R))$ to the solutions, and from Figure 2.10, $\boldsymbol{x}_1$ and $\boldsymbol{x}_4$ have the lowest indicator value (based on the area size).

In the following theorem, a method for calculating $I_h^k(\boldsymbol{x}, \mathcal{F}, R)$ without averaging over all subsets $S \in \mathcal{S}$ is introduced [33, Th. III.7].

Theorem 2.5.7. Let $\mathcal{F} \in \mathcal{D}$, $R \subset \mathcal{Z}$, and $k \in \{0, 1, \ldots, |\mathcal{F}|\}$. Then it holds

$$I_h^k(\boldsymbol{x}, \mathcal{F}, R) = \sum_{i=1}^{k} \frac{\alpha_i}{i} \lambda(H_i(\boldsymbol{x}, \mathcal{F}, R)), \tag{2.29}$$

where

$$\alpha_i := \prod_{j=1}^{i-1} \frac{k - j}{|\mathcal{F}| - j} \tag{2.30}$$

Example 2.5.3. Let $\mathcal{P} = \boldsymbol{x}_1, \boldsymbol{x}_2$, $R = \{\boldsymbol{r}\}$ and $k = |\mathcal{P}| = 2$ where,

$$\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3] \tag{2.31}$$

$$\boldsymbol{f}(\boldsymbol{x}_2) = [3, 1] \tag{2.32}$$

$$\boldsymbol{r} = [4, 5] \tag{2.33}$$

To compute the exact hypervolume indicator, we use the Algorithm 1 and 2 given in [33, Algorithm 1-2]. Thus, by calling *doSlicing*$(\mathcal{F} = \bigcup_{\boldsymbol{x}_1 \in \mathcal{P}} \{(\boldsymbol{x}_1, 0)\}, R = \{\boldsymbol{r}\}, k = 2, n = 2, V = 1, \boldsymbol{z} = [\infty, \infty])$ in Algorithm 1, the algorithm 2 starts a recursion method for computing the exact hypervolume indicator for each $\boldsymbol{x} \in \mathcal{P}$. The calculation of the exact hypervolume indicator is given in Section A.1. Figure 2.12 shows an overview of the recursion steps.

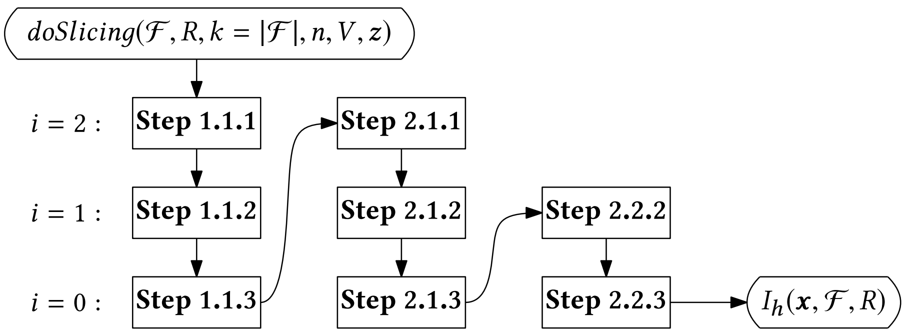


Figure 2.12: An overview of the recursion steps given in Section A.1

The exact hypervolume indicator for the population $\mathcal{P}$ with reference R is computed as

$$I_H(\mathcal{P}, R) = \sum_{\boldsymbol{x} \in \mathcal{P}} I_h(\boldsymbol{x}, \mathcal{P}, R) = I_h(\boldsymbol{x}_1, \mathcal{P}, R) + I_h(\boldsymbol{x}_2, \mathcal{P}, R) = 5 + 3 = 8 \tag{2.34}$$

2.5.3 The framework of HypE

The framework of HypE is similar to the NSGA-II given in Section 2.4.4. The only difference is that HypE include fitness through hypervolumes. Figure 2.13 shows the framework of HypE. With the exact hypervolume indicator has the fitness function, the worst-case complexity of the algorithm is $\mathcal{O}(N^D + DN\log(N))$, where D is the number of objectives and N the number of solutions.

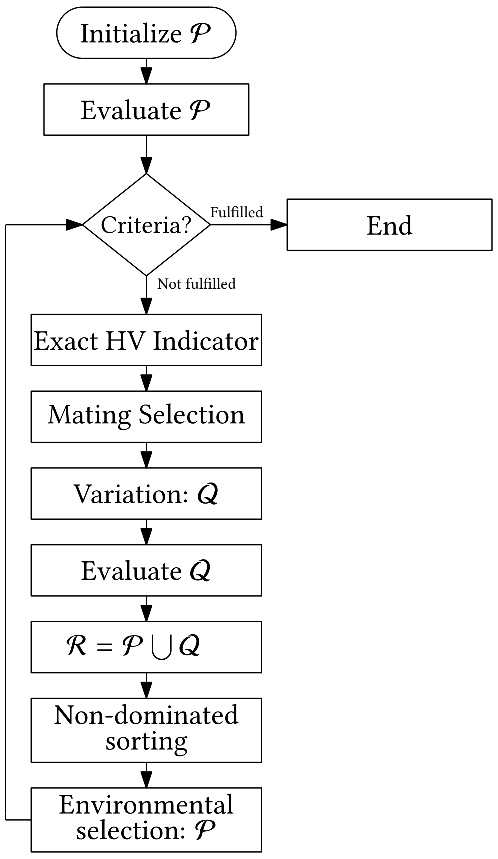


Figure 2.13: The framework of HypE

2.6 Fourier series

2.6.1 2π-Periodic functions

Let f be a 2π-periodic function. Then it is known that $\{\cos(nt), \sin(t)\}$ for $n \in \mathbb{N}$ form a complete orthogonal system over $[-\pi, \pi]$, and by using the the method for a *generalized Fourier series* with $f_1(x) = \cos(x)$ and $f_2(x) = \sin(x)$ given in [53], the Fourier series for any f over $[-\pi, \pi]$ is given by [54]

$$f(t) = a_0 + \sum_{n\in\mathbb{N}} \left(a_n\cos(nt) + b_n\sin(nt)\right), \quad \text{for} \ \ n \in \mathbb{N} \tag{2.35}$$

where

$$a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)dt \tag{2.36}$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(nt)dt, \quad \text{for} \ \ n \in \mathbb{N} \tag{2.37}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(nt)dt, \quad \text{for} \ \ n \in \mathbb{N} \tag{2.38}$$

2.6.2 A convergence result

In [55, pp. 584-585], the convergence of the Fourier series is given as

Theorem 2.6.1. Suppose that f is 2π-periodic and piecewise continous on $[-\pi, \pi]$. Then its Fourier series converges

a) to the value $f(t)$ where f is continous, and
b) to the average of the right and left hand limits of f where f is discontinous

2.6.3 $2L$-Periodic Functions

With a suitable change of variables, we can expand the fourier series given in (2.35) to be applied on any function f with an arbitrary $2L$ period [55]. It is known that $f(t + 2\pi) = f(t)$ if f is 2π-periodic, then $f(t + 2L) = f(t)$ if f is $2L$-periodic. Let c be a scaling parameter such that $f(c(t + 2\pi)) = f(ct + 2\pi c) = f(ct)$. Then with $c = \frac{L}{\pi}$, we get

$$f\left(\frac{Lt}{\pi} + 2L\right) = f\left(\frac{Lt}{\pi}\right) = a_0 + \sum_{n\in\mathbb{N}} \left(a_n\cos(nt) + b_n\sin(nt)\right) \tag{2.39}$$

where

$$a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f\left(\frac{Lt}{\pi}\right)dt \tag{2.40}$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\left(\frac{Lt}{\pi}\right)\cos(nt)dt, \quad \text{for} \ \ n \in \mathbb{N} \tag{2.41}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\left(\frac{Lt}{\pi}\right)\sin(nt)dt, \quad \text{for} \ \ n \in \mathbb{N} \tag{2.42}$$

Now with change of variables $t \mapsto \frac{L}{\pi}t$, the fourier series for an arbitrary $2L$-periodic function f will be given as

$$f(t) = a_0 + \sum_{n\in\mathbb{N}} \left(a_n \cos\left(\frac{n\pi t}{L}\right) + b_n \sin\left(\frac{n\pi t}{L}\right)\right) \tag{2.43}$$

where

$$a_0 = \frac{1}{2L}\int_{-L}^{L} f(t)dt \tag{2.44}$$

$$a_n = \frac{1}{L}\int_{-L}^{L} f(t)\cos\left(\frac{n\pi t}{L}\right)dt, \quad \text{for } n \in \mathbb{N} \tag{2.45}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(t)\sin\left(\frac{n\pi t}{L}\right)dt, \quad \text{for } n \in \mathbb{N} \tag{2.46}$$

2.7 Multivariate Analysis

This section presents two multivariate anlysis called PCA and PLSR. The first method called PCA takes the original variables and projects them onto smaller number of Principal Component (PC) (latent variables). Each PC explains a portion of the total information of the original data. Furthermore, the first PC explains most of the information and decreases in the following PCs. The PLSR models both $\boldsymbol{X}$ and $\boldsymbol{Y}$ data to find the latent variables in $\boldsymbol{X}$ that will best predict the latent variables in $\boldsymbol{Y}$. These latent variables are similar to the PCs in PCA, but are referred as *factors*.

2.7.1 Software

In this thesis, we consider the software called *The Unscrambler X 10.3* for the multivariate analysis tool. The software was originally developed by Harald Martens and then taken over by CAMO Software[56, 57]. The Unscrambler X 10.3 includes many tools for preprocessing and multivariate analysis of data sets.

2.7.2 Multivariate analysis terms and symbols

The mathematical terms and symbols used in Multivariate analysis are given in Table 2.8 are based on the reference manual in [58].

2.7.3 Bilinear subspace model

Both PCA and PLSR use the bilinear subspace models to describe the structures in a multivariate data set [56, 59]. The bilinear subspace is a linear combination of the latent variables, which span a subspace in the variable space. The bilinear model of $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ is given as

$$\hat{\boldsymbol{X}} = \boldsymbol{T}\boldsymbol{P}^{\mathrm{T}} = \sum_{i=1}^{a} \boldsymbol{t}_i \boldsymbol{p}_i^{\mathrm{T}} \in \mathbb{R}^{n\times m} \tag{2.47}$$

Table 2.8: Terms and symbols used in multivariate analysis

Symbol	Description	Dimension
n	Number of objects	$n = \{1, 2, \dots\}$
m	Number of $\boldsymbol{X}$ variables	$m = \{1, 2, \dots\}$
k	Number of $\boldsymbol{Y}$ variables	$k = \{1, 2, \dots\}$
a	Number of PC	$a = \{1, 2, \dots\}$
$\mathbf{1}$	Vector of ones	$\mathbf{1} \in \mathbb{R}^n$
$\bar{\boldsymbol{x}}$	Column means of $\boldsymbol{X}$	$\bar{\boldsymbol{x}} \in \mathbb{R}^m$
$\bar{\boldsymbol{y}}$	Column means of $\boldsymbol{Y}$	$\bar{\boldsymbol{y}} \in \mathbb{R}^k$
$\boldsymbol{X}, \boldsymbol{Y}$	Data sets	$\boldsymbol{X} \in \mathbb{R}^{n \times m}, \boldsymbol{Y} \in \mathbb{R}^{n \times k}$
$\boldsymbol{T}$	Scores of X	$\boldsymbol{T} \in \mathbb{R}^{n \times a}$
$\boldsymbol{U}$	Scores of Y	$\boldsymbol{U} \in \mathbb{R}^{n \times a}$
$\boldsymbol{P}$	Loadings of $\boldsymbol{X}$	$\boldsymbol{P} \in \mathbb{R}^{m \times a}$
$\boldsymbol{Q}$	Loadings of $\boldsymbol{Y}$	$\boldsymbol{Q} \in \mathbb{R}^{m \times a}$
$\boldsymbol{E}$	Resudials of $\boldsymbol{X}$	$\boldsymbol{E} \in \mathbb{R}^{n \times m}$
$\boldsymbol{F}$	Resudials of $\boldsymbol{Y}$	$\boldsymbol{F} \in \mathbb{R}^{n \times k}$
$\boldsymbol{B}$	Regression coefficients	$b \in \mathbb{R}^k$

where $\boldsymbol{t_i}\boldsymbol{p}_i^{\mathrm{T}}$ is the latent variable i (in PCA called Principal Component). The resudial (error) model that the bilinear subspace does not consider is given as

$$\boldsymbol{E} = \boldsymbol{X} - \hat{\boldsymbol{X}} \tag{2.48}$$

The matrix $\boldsymbol{T}$ and $\boldsymbol{P}$ are called the *scores* and *loadings*, respectively. Each matrix reprents different information of $\boldsymbol{X}$;

1. The scores $\boldsymbol{t_i} \in [\boldsymbol{t}_1, \dots, \boldsymbol{t}_a]$ visualize the connections among the objects in $\boldsymbol{X}$. They represent the projection of the objects onto the loading vectors $\boldsymbol{p}_i \in [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_a]$. Furthermore, the scores can be viewed as an approximation of the objects regarding the latent variable.

2. The loadings $\boldsymbol{p_i} \in [\boldsymbol{p}_1, \dots, \boldsymbol{p}_a]$ visualize the connections among the variables in $\boldsymbol{X}$. Furthermore, the loadings can be interpreted as the mapping between the variable space and the subpace in (2.47) [59, pp. 40-42].

2.7.4 Principal Component Analysis — PCA

In PCA, the goal is to find the maximum variance of the variables in $\boldsymbol{X}$ and represent the information in PCs. Variables with little variation will be seen as noise since variables with insufficient variations have no meaningful information related to it. The PCs are arranged in order of decreasing explained variance and are orthogonal to each other. The bilinear model of the PCA model is given as

$$\boldsymbol{X} = \mathbf{1}\bar{\boldsymbol{x}}^{\mathrm{T}} + \sum_{i=1}^{a} \boldsymbol{t}_i\boldsymbol{p}_i^{\mathrm{T}} + \boldsymbol{E} = \mathbf{1}\bar{\boldsymbol{x}}^{\mathrm{T}} + \boldsymbol{T}\boldsymbol{P}^{\mathrm{T}} + \boldsymbol{E} \tag{2.49}$$

where $\bar{\boldsymbol{x}}$ is the column means of the data set $\boldsymbol{X}$. Figure 2.14 illustrates the PCA model (2.49) . The

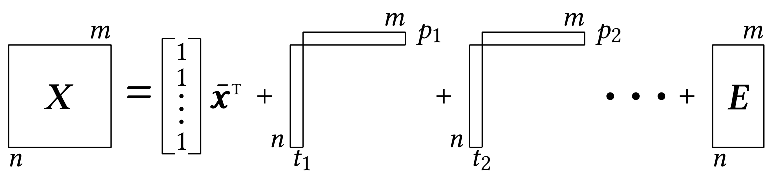


Figure 2.14: An illustration of the PCA model

residual $\boldsymbol{E}$ can be neglected when interpreting the results in the PCA model. This is okay to do, if the residual error is small. The matrix $\boldsymbol{P}$ and $\boldsymbol{T}$ are defined as

$$\boldsymbol{P} = eig(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}) \in \mathbb{R}^{m\times a} \tag{2.50}$$

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{P} \in \mathbb{R}^{n\times a} \tag{2.51}$$

where the properties of the loadings and scores are listed below [56]

- The score vectors $\boldsymbol{t}_i \in [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_a]$ are orthogonal to each other:

$$\boldsymbol{t}_i\boldsymbol{t}_2 = 0, \text{ for all} i \neq j \tag{2.52}$$

- The loading vectors $\boldsymbol{p}_i \in [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_a]$ are orthogonal to each other:

$$\boldsymbol{p}_i\boldsymbol{p}_2 = 0, \text{ for } i \neq j \tag{2.53}$$

- The loading vectors $\boldsymbol{p}_i \in [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_a]$ have unit length:

$$||\boldsymbol{p}_i|| = 1, \text{ for all } i = \{1, \ldots, a\}, \tag{2.54}$$

  where $||\cdot||$ denotes the euclidean norm.

With PCA, we may reduce the dimensionality of the data set, i.e., some variables may not have significant effect on $\boldsymbol{X}$ and can be ignored. This reduced dimensionality will have a new coordinate system made of PCs and is illustrated in Figure 2.15. A full explaination of all the mathematical

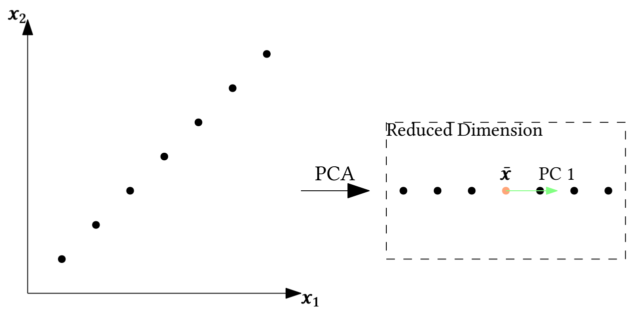


Figure 2.15: Mapping from the original coordinates to Principal Components

formulations of the PCA is not given, but instead, we will focus on how to use PCA as a tool for graphical interpretation of $\boldsymbol{X}$. The Unscrambler X 10.3 includes PCA and can be used to interpret the data $\boldsymbol{X}$ in a graphical manner. It includes *cross-validation*, which is an operation to avoid or reduce *overfitting*, i.e., fitting to noisy variables. This thesis will focus on the *full cross-validtion*, where it leaves out one object in $\boldsymbol{X}$ to create a model and then validate the model to the left out object. This is done n times to validate each object in $\boldsymbol{X}$ [56]. The full cross-validtion is illustrated in Figure 2.16. For computing the PCA models in Unscrambler X 10.3, the algorithm Nonlinear Iterative Partial Least Squares (NIPALS) is used [59]. Furthermore, we consider mean center data, no rotation and all weights to be equal one. The configurations used for running PCA is given in Section A.2.

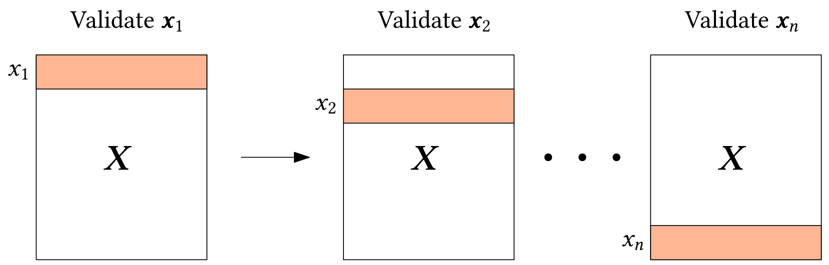

Validate $\boldsymbol{x}_1$
Validate $\boldsymbol{x}_2$
Validate $\boldsymbol{x}_n$
x_1
x_2
x_n
$\boldsymbol{X}$
$\boldsymbol{X}$
$\boldsymbol{X}$

Figure 2.16: Full cross-validation

2.7.4.1 Graphical interpreation — Scores and Loadings

Figure 2.14 shows the results of a PCA model in Unscrambler X 10.3. The figure shows a 2-D plot of the scores and the correlation plot of the loadings. The x and y axes are the principal components. The scores show structures, differences and similarities, and are connections between the objects in $\boldsymbol{X}$. In the scores plot, two different clusters can be seen: (i) the left cluster that goes vertically, and (ii) the right cluster that goes horizontally. This can be interpreted has there exists two different subsets of $\boldsymbol{X}$ that share similarities. The objects that are close along the same PCA are similar. Another important property on how to read the scores plot is the position of each object: objects that are further away from the origin along the same PC have higher or lower variable values than the mean value in terms of the loadings position. Thus, scores cannot be interpreted without loadings, and the same argument for the loadings. In Figure 2.14, the loadings are plottet in a correlation plot. Loadings that are on the same PC are highly correlated. Furthermore, loadings close to each other are positively correlated, while loadings with opposite signs are negatively correlated. Two loadings that are on different PCs are uncorrelated. From Figure 2.14, the two loadings on the right are positively correlated to each other, but are negatively correlated with the loading on the left (on PC 1). Furthermore, all three loadings are uncorrolated with the loading on the top (on PC 2). The steps for reading the loadings plot:

1. Look for high loadings (close to +1 or -1) and discard the small loadings (close to origin). The high loadings are meaningful and interpretable, while small loadings are badly valuated by a particular PC.
2. Interpret the correlations of each loading.

The benefit of PCA is to interpret both scores and loadings at the same time. Objects with high scores and loadings have variable values that are higher or lower than the average value. That is,

1. positive scores and loadings: higher than average

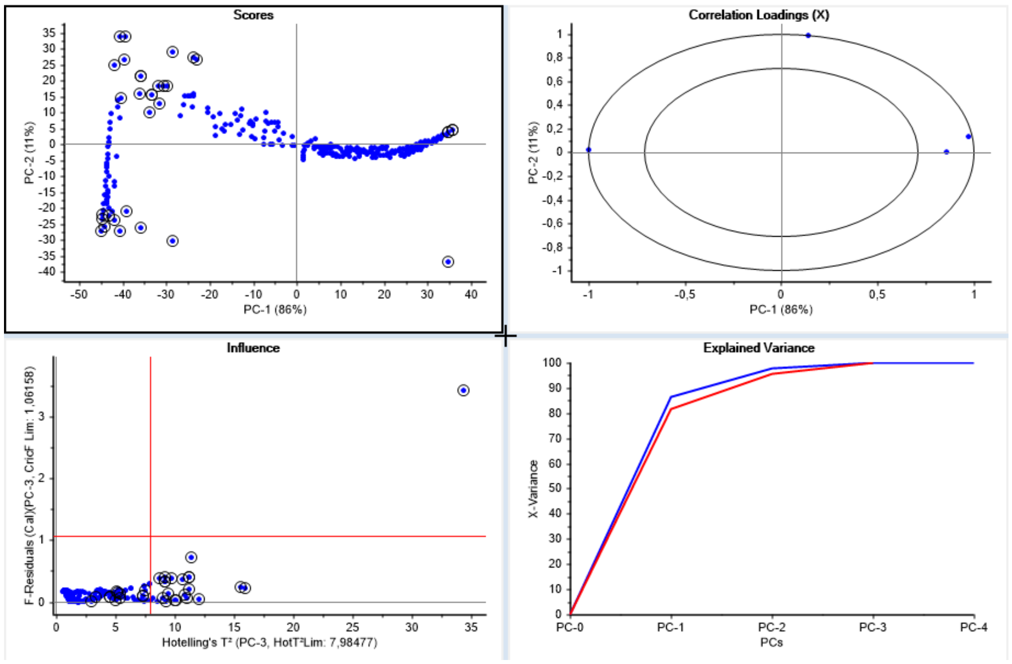


Figure 2.17: A PCA model in Unscrambler X 10.3

2. postive scores and negative loadings: lower than average
3. negative scores and loadings: higher than average
4. negative scores and positive loadings: lower than average

The larger the score value is, higher influence on the corresponding variable (either positive or negative). Furthermore, the larger the loading of a variable, quicker increase of that variable as the score increases.

2.7.4.2 Graphical interpretation — Influence

The noisy part of a PCA model is called *outliers*. These outliers are objects that do not fit well or influence the model too much. The outliers may cause one or more PCs to only focus on them even if they are unimportant. In The Unscrambler X 10.3, there various way to determine the outliers:

- Look at the scores plot and see the objects that are far from the others or chaotic. In Figure 2.14, the object that is far bottom right is most likely an outlier.
- Look at the influence plot and study the objects that might influence the model too much. There are three cases in the influence plot that can determine the outliers.
  1. Objects that lies in the fourth quadrant are fit but can influence the model.
  2. Objects that lies in the second quadrant do not fit the PCA model, but they do not influence the model too much.
  3. Objects that lies in the first quadrant do not fit the PCA model and can influence it (these are the worst kind of outliers).

From the influence plot in Figure 2.14, most of the objects are located in the third quadrant. These fit well to the PCA model.

2.7.4.3 Graphical interpretation — Explained variance

The plot that shows the explained variance in Figure 2.14 describes the explained variance in each PC. From the plot, the first PC explains 86% of the variance, while PC 2 explains 11%. Furthermore, the optimal number of PCs is three to describe PCA model. Even if the optimal number of PCs is three, the first two PCs already explain 97% of the variance. Thus, the first two PCs is sufficient for this PCA model.

2.7.5 Partial Least Squares Regression — PLSR

The difference between PCA and PLSR is that PCA maximizes the variance in a particular set $\boldsymbol{X}$, while PLSR maximizes the covaraiance between two different set $\boldsymbol{X}$ and $\boldsymbol{Y}$. Usually, PLSR is used for prediction of $\boldsymbol{Y}$ data (response) from $\boldsymbol{X}$ data (predictor). PLSR creates a model for both $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that the latent variables in $\boldsymbol{X}$ best predict the latent variables in $\boldsymbol{Y}$. In context of PLSR, the latent variables will be referred to as factors. In [60], the mathematical description of the PLSR model is given as

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^{\mathrm{T}} + \boldsymbol{E} \tag{2.55}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^{\mathrm{T}} + \boldsymbol{F} \tag{2.56}$$

where T is defined as

$$\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W}^{*} \tag{2.57}$$

$$\boldsymbol{W}^{*} = \boldsymbol{W}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W})^{-1} \tag{2.58}$$

The (2.56) is computed by the $\boldsymbol{U}$-scores, $\boldsymbol{Q}$-loadings and the residual $\boldsymbol{F}$ based on $\boldsymbol{Y}$ (a PCA model of $\boldsymbol{Y}$). The $\boldsymbol{W}$ is the weight matrix that maps the $\boldsymbol{X}$ to $\boldsymbol{T}$. They express how $\boldsymbol{T}$ are computed from $\boldsymbol{X}$ to obtain orthogonal decomposition. In PLSR, the $\boldsymbol{X}$ data are used as predictors of Y, and thus the prediction $\hat{\boldsymbol{Y}}$ is given as

$$\hat{\boldsymbol{Y}} = \boldsymbol{T}\boldsymbol{Q}^{\mathrm{T}} + \boldsymbol{F} = (\boldsymbol{X}\boldsymbol{W}^{*})\boldsymbol{Q}^{\mathrm{T}} + \boldsymbol{F} = \boldsymbol{X}\boldsymbol{B} + \boldsymbol{F} \tag{2.59}$$

where $\boldsymbol{B} = \boldsymbol{W}^{*}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{W}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{W})^{-1}\boldsymbol{Q}^{\mathrm{T}}$ is the regression coefficients, called the Beta coeffictients. The residual $\boldsymbol{E}$ and $\boldsymbol{F}$ are negligible, even if they are shown in the equations. This can be done, since the residuals are usually small. Figure 2.18 shows an illustration of the mathematical description of PLSR. The scores and loadings in PLSR are interpreted in the same way as in PCA. With the only difference that PLSR computes two different score and loading matrices. Furthermore, the link between $\boldsymbol{T}$ and $\boldsymbol{U}$ is a summary of the relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$. The $\boldsymbol{T}$-scores is the mapping from the objects in $\boldsymbol{X}$, which is always available. The $\boldsymbol{U}$-scores are available when $\boldsymbol{Y}$ is available, which is predicted by $\boldsymbol{X}$. Thus, $\boldsymbol{T}$ is the structure in $\boldsymbol{X}$ that best predict $\boldsymbol{Y}$ and $\boldsymbol{U}$ is the structure in $\boldsymbol{Y}$ that is explained by $\boldsymbol{X}$. Figure 2.19 shows a PLSR model overview in the Unscrambler X 10.3.

The PLSR overview shown in Figure 2.17, three of the plots describe the same informations as in the PCA overview. The new plot called *Predicted vs. Reference* show the prediction of the variables and it shows the prediction of the selected variable. The red points in the plot are the $\boldsymbol{Y}$-variables that are computed when building the PLSR model, i.e., the Y-variables were available.

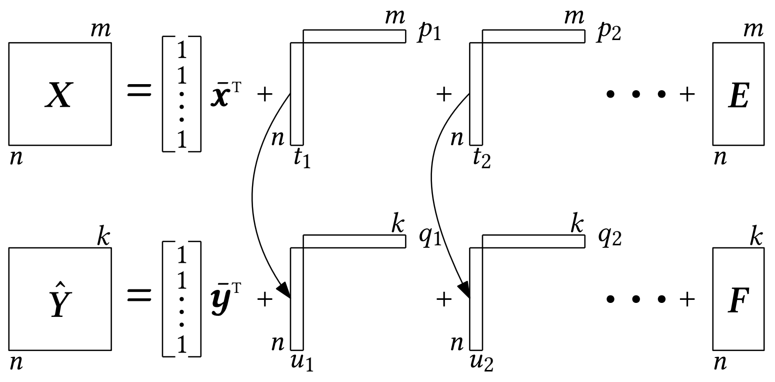
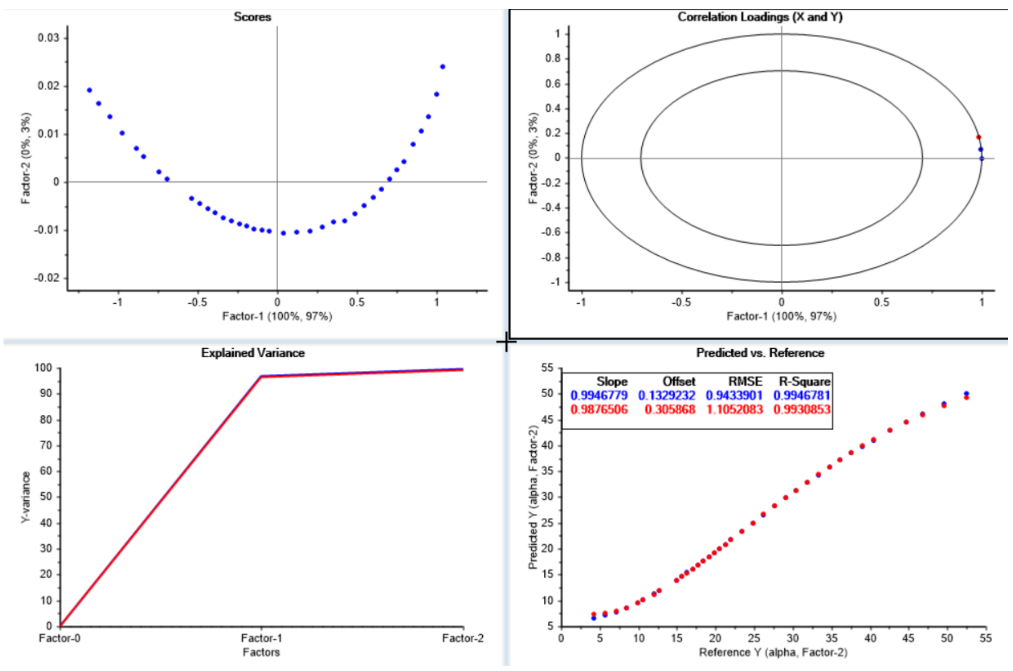


Figure 2.18: An illustration of the PLSR model



Slope	Offset	RMSE	R-Square
0.9946779	0.1329232	0.9433901	0.9946781
0.9876506	0.305868	1.1052083	0.9930853

Figure 2.19: A PLSR model in Unscrambler X 10.3

The blue points are the predicted Y-variables that are not included in the PLSR model. The most important element of the plot is the error measure RMSE. It gives the error of Y in absolute value and it depends on the user to determine how high the RMSE can be. The lower the RMSE, the better is the prediction.

Chapter 3

Dynamic model of the underwater snake robot

This section will briefly present some fundamental properties of the underwater snake robot, like the forward velocity, average power consumption, kinematics and dynamics. Furthermore, we want to define the mathematical terms that will be used in the optimization for the upcoming sections. The underwater snake robot in this thesis is based on the dynamic model presented in [5, 8, 13]. The physical parameters for the dynamic model are chosen to be equal to the ones given in [4, 13]. In addition to the model, we also use the low-level PD-controller presented in [13].

3.1 Mathematical terms and symbols

The mathematical terms of the underwater snake robot are given in [5], which are summerized in Table 3.1. Furthermore, some properties of the underwater snake robot are listed as:

- Consists of n rigid links of equal length $2l$
- The links are interconnected by $n - 1$ joints
- The links have the same mass m, and a moment of inertia $\frac{1}{3}mL^2$
- The center of mass (CM) is located at the center of each link, and the mass is uniformly distributed.
- The total mass of the snake is, nm

In addition to the mathematical terms given in Table 3.1, the following vectors and matrices given in [5], are used in formulating the kinematics and hydrodynamics of the underwater snake robot.

$$\boldsymbol{A} = \overbrace{\begin{bmatrix} 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}}^{\text{Addition matrix}}, \quad \boldsymbol{D} = \overbrace{\begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix}}^{\text{Difference matrix}} \tag{3.1}$$

Table 3.1: Mathematical terms

Symbol	Description	Vectors
n	The number of links	
l	The half length of a link	
m	Mass of each link	
J	Moment of inertia of each link	
θ_i	Angle between link i and the global x axis	$\boldsymbol{\theta} \in \mathbb{R}^n$
ϕ_i	Angle of joint i	$\boldsymbol{\phi} \in \mathbb{R}^{n-1}$
(x_i, y_i)	Global coordinates of the CM of link i	$\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^n$
(p_x, p_y)	Global coordinates of the CM of the robot	$\boldsymbol{p}_{CM} \in \mathbb{R}^2$
u_i	Actuator torque of the joint between link i and link $i+1$	$\boldsymbol{u} \in \mathbb{R}^{n-1}$
u_{i-1}	Actuator torque of the joint between link i and link $i-1$	$\boldsymbol{u} \in \mathbb{R}^{n-1}$
$(f_{x,i}, f_{y,i})$	Fluid force on link i	$\boldsymbol{f}_x, \boldsymbol{f}_y \in \mathbb{R}^n$
τ_i	Fluid torque on link i	$\boldsymbol{\tau} \in \mathbb{R}^n$
$(h_{x,i}, h_{y,i})$	Joint constraint force on link i from link $i+1$	$\boldsymbol{h}_x, \boldsymbol{h}_y \in \mathbb{R}^{n-1}$
$-(h_{x,i}, h_{y,i})$	Joint constraint force on link i from link $i-1$	$\boldsymbol{h}_x, \boldsymbol{h}_y \in \mathbb{R}^{n-1}$

where $\boldsymbol{A}, \boldsymbol{D} \in \mathbb{R}^{(n-1)\times n}$. Furthermore,

$$\begin{aligned}
\boldsymbol{e} &= \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^n, & \mathbf{E} &= \begin{bmatrix} \boldsymbol{e} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{n\times 1} & \boldsymbol{e} \end{bmatrix} \in \mathbb{R}^{2n\times 2} \\
\sin\boldsymbol{\theta} &= \begin{bmatrix} \sin\theta_1 & \dots & \sin\theta_n \end{bmatrix}^T \in \mathbb{R}^n, & \mathbf{S}_\theta &= \operatorname{diag}(\sin\boldsymbol{\theta}) \in \mathbb{R}^{n\times n} \\
\cos\boldsymbol{\theta} &= \begin{bmatrix} \cos\theta_1 & \dots & \cos\theta_n \end{bmatrix}^T \in \mathbb{R}^n, & \mathbf{C}_\theta &= \operatorname{diag}(\cos\theta) \in \mathbb{R}^{n\times n} \\
\operatorname{sgn}\boldsymbol{\theta} &= \begin{bmatrix} \operatorname{sgn}\theta_1 & \dots & \operatorname{sgn}\theta_n \end{bmatrix}^T \in \mathbb{R}^n, & \dot{\boldsymbol{\theta}}^2 &= \begin{bmatrix} \dot{\theta}_1^2 & \dots & \dot{\theta}_n^2 \end{bmatrix}^T \in \mathbb{R}^n
\end{aligned} \tag{3.2}$$

$$\begin{aligned}
&\mathbf{J} = j\mathbf{I}_n, \qquad \mathbf{L} = l\mathbf{I}_n, \qquad \mathbf{M} = m\mathbf{I}_n \\
&\mathbf{K} = \mathbf{A}^T(\mathbf{D}\mathbf{D}^T)^{-1}, \quad \mathbf{H} = \left(\mathbf{I}_n - \frac{1}{n}\mathbf{e}\mathbf{e}^T\right)^{-1}\mathbf{K}^T, \quad \mathbf{V} = \mathbf{A}^T(\mathbf{D}\mathbf{D}^T)^{-1}\mathbf{A}
\end{aligned} \tag{3.3}$$

where $\boldsymbol{e}$ is the summation vector and $\mathbf{D}\mathbf{D}^T$ is assumed to be invertible.

3.2 The kinematics of the underwater snake robot

In [5], Figure 3.1 and 3.2 illustrate the kinematics of the underwater snake robot, where vectors in the global and local coordinate system are denoted with the superscript *global* and *link,i*, respectively. Furthermore, the dynamic model is assumed to move in a virtual horizontal and flat plane, and fully immersed in water. This gives the underwater snake robot a total of $n+2$ degrees of freedom (n links in the $x-y$ plane) [5].

From Figure 3.1, the *link angle* $\theta_i(t)$ in each link $i \in \{1, \dots, n\}$ is defined as the angle between the global x axis and link i, where counterclockwise is defined as the positive direction. The joint angle of joint $i \in \{1, \dots, n-1\}$ is denoted by $\phi_i(t)$ and defined as

$$\phi_i(t) = \theta_i(t) - \theta_{i-1}(t) \tag{3.4}$$

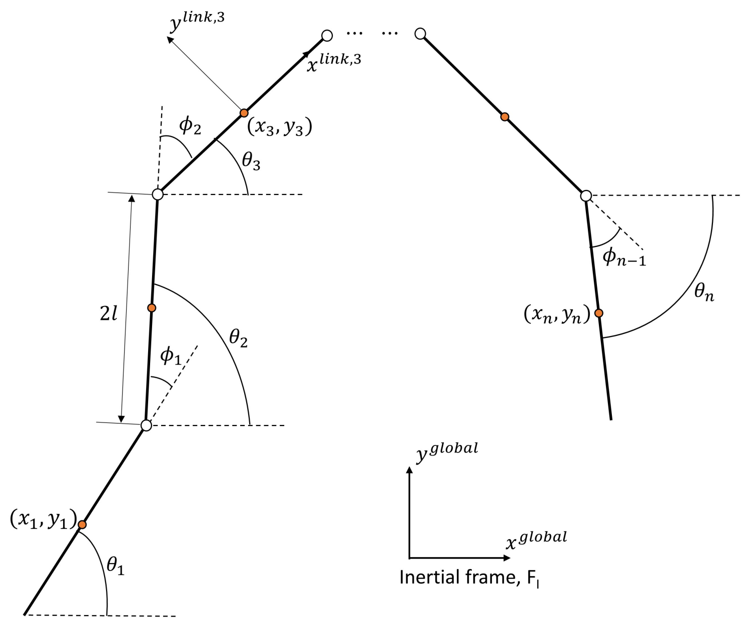


Figure 3.1: Kinematics of the underwater snake robot

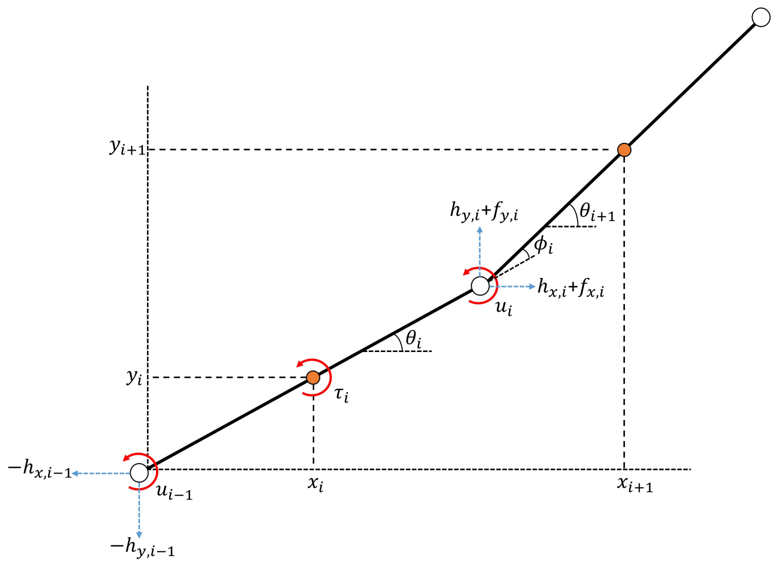


Figure 3.2: Forces and torques acting on each link

Furthermore, the angles and the joint angles are assembled in the vectors $\boldsymbol{\theta}(t) = [\theta_1(t), \ldots, \theta_n(t)]^T \in \mathbb{R}^n$ and $\boldsymbol{\phi}(t) = [\phi_1(t), \ldots, \phi_{n-1}(t)]^T \in \mathbb{R}^{n-1}$, respectively. In [5], the heading $\bar{\theta}(t) \in \mathbb{R}$ of the underwater snake robot is defined as the average of the link angles

$$\bar{\theta}(t) = \frac{1}{n}\sum_{i=1}^{n} \theta_i(t), \tag{3.5}$$

which also can be interpreted as the orientation. The global frame position $P_{CM} \in \mathbb{R}^2$ of the center of mass of the underwater snake robot is given by

$$\mathbf{P}_{CM}(t) = \begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{nm}\sum_{i=1}^{n} mx_i(t) \\ \frac{1}{nm}\sum_{i=1}^{n} my_i(t) \end{bmatrix} = \frac{1}{n}\begin{bmatrix} \mathbf{e}^T\mathbf{X} \\ \mathbf{e}^T\mathbf{Y} \end{bmatrix}, \tag{3.6}$$

where $(x_i(t), y_i(t))$ are the global frame coordinates of the CM of link i, $\mathbf{X}(t) = [x_1(t), \dots, x_n(t)]^T \in \mathbb{R}^n$ and $\mathbf{Y}(t) = [y_1(t), \dots, y_n(t)]^T \in \mathbb{R}^n$. Furthermore, the links constrained by the joints can be expressed as

$$\mathbf{DX}(t) + l\mathbf{A}\cos\theta(t) = 0 \Leftrightarrow \mathbf{DX}(t) = -l\mathbf{A}\cos\theta(t) \tag{3.7}$$

$$\mathbf{DY}(t) + l\mathbf{A}\sin\theta(t) = 0 \Leftrightarrow \mathbf{DY}(t) = -l\mathbf{A}\sin\theta(t). \tag{3.8}$$

A more detailed derivation of the kinematics can be found in [5].

3.3 Hydrodynamic model

In this section, we will briefly present the dynamics of the underwater snake robot. The formulation of the dynamics for the underwater snake robot is quite complicated. In [5], it is shown that the global frame fluid forces on the links are given as

$$\mathbf{f}(t) = \begin{bmatrix} \mathbf{f}_\mathrm{x}(t) \\ \mathbf{f}_\mathrm{y}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_{\mathrm{A_x}}(t) \\ \mathbf{f}_{\mathrm{A_y}}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{f}^{\mathrm{I}}_{\mathrm{D_x}}(t) \\ \mathbf{f}^{\mathrm{I}}_{\mathrm{D_y}}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{f}^{\mathrm{II}}_{\mathrm{D_x}}(t) \\ \mathbf{f}^{\mathrm{II}}_{\mathrm{D_y}}(t) \end{bmatrix}, \tag{3.9}$$

where $\mathbf{f}_{\mathrm{A_x}}(t)$ and $\mathbf{f}_{\mathrm{A_y}}(t)$ represent the effects from added mass forces. Furthermore, the vectors $\mathbf{f}^{\mathrm{I}}_{\mathrm{D_x}}$, $\mathbf{f}^{\mathrm{I}}_{\mathrm{D_y}}$ and $\mathbf{f}^{\mathrm{II}}_{\mathrm{D_x}}(t)$, $\mathbf{f}^{\mathrm{II}}_{\mathrm{D_y}}(t)$ represent the effects from the linear and nonlinear drag forces, respectively. The fluid torques on all the links in matrix form are given by

$$\boldsymbol{\tau}(t) = -\mathbf{\Lambda}_1\ddot{\theta}(t) - \mathbf{\Lambda}_2\dot{\theta}(t) - \mathbf{\Lambda}_3\dot{\theta}(t)|\dot{\theta}(t)|, \tag{3.10}$$

where $\mathbf{\Lambda}_1 = \lambda_1\mathbf{I_n}$, $\mathbf{\Lambda}_2 = \lambda_2\mathbf{I_n}$ and $\mathbf{\Lambda}_3 = \lambda_3\mathbf{I_n}$. For a more detailed derivation of the dynamics for the underwater snake robot, see [5]. Observe the dot notation in the equtaion. This simply implies that $\dot{\boldsymbol{\theta}}(t)$ and $\ddot{\boldsymbol{\theta}}(t)$ is the first and second derivative with respect to time t, respectively.

3.4 Equations of motion

In [5], it can be shown that the links of the underwater snake robot are constrained by the joints according to Figure 3.2, and the force balance equations for all links may be expressed in matrix form as

$$m\ddot{\mathbf{X}}(t) = \mathbf{D}^T\mathbf{f}_\mathrm{x}(t) + \mathbf{f}_\mathrm{x}(t), \quad m\ddot{\mathbf{Y}}(t) = \mathbf{D}^T\mathbf{h}_\mathrm{y}(t) + \mathbf{f}_\mathrm{y}(t). \tag{3.11}$$

Furthermore, we can compute the acceleration of CM by differentiating equation (3.6) twice with respect to time, inserting equation (3.11) and cancel the constraint forces $\mathbf{h_x}$ and $\mathbf{h_y}$ when the link accelerations are summed. This gives the equation for acceleration of the CM as

$$\ddot{\mathbf{P}}_{CM}(t) = \begin{bmatrix} \ddot{p}_x(t) \\ \ddot{p}_y(t) \end{bmatrix} = \frac{1}{n}\begin{bmatrix} \mathbf{e}^T\ddot{\mathbf{X}}(t) \\ \mathbf{e}^T\ddot{\mathbf{Y}}(t) \end{bmatrix} = \frac{1}{nm}\begin{bmatrix} \mathbf{e}^T & \mathbf{0}_{1\times n} \\ \mathbf{0}_{1\times n} & \mathbf{e}^T \end{bmatrix}\mathbf{f}(t) = \frac{1}{nm}\mathbf{E}^T\mathbf{f}(t). \tag{3.12}$$

In [5], the torque balance equations for all links of the underwater snake robot in matrix form is given by

$$\mathbf{J}\ddot{\theta}(t) = \mathbf{D}^T\mathbf{u}(t) - l\mathbf{S}_\theta\mathbf{A}^T\mathbf{h}_x(t) + l\mathbf{C}_\theta\mathbf{A}^T\mathbf{h}_y(t) + \boldsymbol{\tau}(t). \tag{3.13}$$

By inserting (3.11) into (3.13) and replacing $\ddot{\mathbf{X}}$, $\ddot{\mathbf{Y}}$ in (3.11) by the double derivatives with respect to time in (3.7), we get:

$$\mathbf{M}_\theta\ddot{\theta}(t) + \mathbf{W}_\theta\dot{\theta}^2(t) + \mathbf{V}_\theta\dot{\theta}(t) + \mathbf{\Lambda}_3\dot{\theta}(t)|\dot{\theta}(t)| - l\mathbf{S}_\theta\mathbf{K}\mathbf{f}_{\mathrm{D_x}}(t) + l\mathbf{C}_\theta\mathbf{K}\mathbf{f}_{\mathrm{D_y}}(t) = \mathbf{D}^T\mathbf{u}(t), \tag{3.14}$$

where $\mathbf{f}_{\mathrm{D_x}}(t) = \mathbf{f}^{\mathrm{I}}_{\mathrm{D_x}}(t) + \mathbf{f}^{\mathrm{II}}_{\mathrm{D_x}}(t)$ and $\mathbf{f}_{\mathrm{D_y}}(t) = \mathbf{f}^{\mathrm{I}}_{\mathrm{D_y}}(t) + \mathbf{f}^{\mathrm{II}}_{\mathrm{D_y}}(t)$ are the drag forces in x and y directions, respectively, and $\mathbf{u}(t) \in \mathbb{R}^{n-1}$ the joint control input. The complete equations of motion for the underwater snake robot are given by (3.12) and (3.14). A more detailed derivation of the equations of motion for the underwater snake robot can be found in [5].

3.5 Low-level joint controller for the underwater snake robot

In this thesis, the lateral and eel-like motion pattern for the snake robot is based on the sinusoidal reference signal proposed in [5],

$$\phi_i^*(t) = \alpha g(i,n)\sin(\omega t + (i-1)\delta) + \gamma, \quad i \in \{1, \ldots, n-1\}, \tag{3.15}$$

where n is the total number of joints, α the amplitude, ω the frequency, δ the phase shift between the joints and ϕ_i is the difference in joint angles given by (3.4). The scaling function $g(i,n)$ for the amplitude of joint i is used to create different motion patterns for the snake robot. In this thesis, the motion patterns called lateral undulation and eel-like motions are being focused on. To achieve the lateral undulation, which is a sinusoidal motion with constant amplitude α, the scaling function needs to be set to

$$g(i,n) = 1. \tag{3.16}$$

The eel-like motion pattern is a motion that starts with a small amplitude in the head of the snake robot and then gradually increases towards the tail. To achieve this motion, the scaling function g is set to

$$g(i,n) = \frac{n-i}{n+1}. \tag{3.17}$$

The torque signal u_i in each joint is calculated as proposed in [5], with a PD controller to compute the joints actuator torques from the joints reference angles given by

$$u_i(t) = k_p(\phi_i^*(t) - \phi_i(t)) + k_d(\dot{\phi}_i^*(t) - \dot{\phi}_i(t)), \quad i \in \{1, \ldots, n-1\} \tag{3.18}$$

where $kp > 0$ and $kd > 0$ are constant control gains.

3.6 The forward velocity and the average power consumption

For an underwater snake robot, the propulsion is generated by the motion of the joints and its interaction with the surrounding fluid. This implies that the actuator torque (4.9) input to the joints convert into a combination of joint motion and the energy that is dissipated by the fluid. Furthermore, we assume that the underwater snake robot has perfect joints. Thus, the total amount of energy of the dynamic system generated by the controller (4.9) is a combination of the sum of kinetic energy and the energy that dissipates to the surrounding fluid [8]. In physics, the change of energy with respect to time (work) is given by the integral

$$\Delta E = \int_{t_1}^{t_2} \hat{\tau}\hat{\omega}\, dt, \tag{3.19}$$

where $\hat{\tau}$ is the torque and $\hat{\omega}$ is the angular velocity. Thus, the total energy consumption for the propulsion of the dynamic model proposed in [13, 8], is given by

$$E_s = E_{kinetic} + E_{fluid} = \int_0^T \left(\sum_{i=1}^{n-1} u_i(t)\dot{\phi}_i(t) \right) dt, \tag{3.20}$$

where T is the time of a complete swimming cycle of the dynamic model, $u_i(t)$ is the actuation torque of joint i given by the PD controller (4.9) and $\dot{\phi}_i(t)$ the angular velocity of joint i defined as

$$\dot{\phi}(t) = \dot{\theta}_i(t) - \dot{\theta}_{i-1}(t). \tag{3.21}$$

For a complete swimming cycle T, the average power consumption of the dynamic model is computed by

$$P_{avg} = \frac{1}{T} \int_0^T \left(\sum_{i=1}^{n-1} u_i(t)\dot{\phi}_i(t) \right) dt, \tag{3.22}$$

while the forward velocity is defined as

$$\bar{v} = \frac{\sqrt{(p_x(T) - p_x(0))^2 + (p_y(T) - p_y(0))^2}}{T}, \tag{3.23}$$

[13, 8]. This forward velocity is computed by taking the traveling distance of the CM given by (3.6) at the initial and the final points, and then divide the distance with the time of a complete swimming time cycle T. Note that the actuator torque $u_i(t)$, the joint angle $\phi_i(t)$ and angular velocity $\dot{\phi}(t)$ of joint i are time-dependent, but for our convenience, we will disregard the notation of t, such that we simply write u_i, ϕ_i and $\dot{\phi}_i$ in the upcoming sections.

Chapter 4

Methodology

In this chapter, a formulation of the multi-objective optimization problem for the dynamic model is given. In addition to the formulation of the MOP, different motion patterns of the snake robot are also introduced. The two most common motion patterns called the *lateral undulation* and *eel-like* motion are inspired by the motion seen in nature, while the other motion patterns are altered. The purpose of the altered motion patterns is to let the MOEAs search for an efficient motion. The shape of the altered motion patterns is difficult to conclude since the formulation of the patterns is not concrete. However, all the motion patterns are assumed to be periodic. The goal is to examine the advantages and disadvantages of each motion pattern. In this thesis, the altered motion are called the *modified*, *Fourier series* and *multi-Fourier* motion pattern. The first altered motion pattern (modified) take the basis of the lateral undulation and eel-like motion, while the two latter motions are based on the Fourier series given in Section 2.6. This chapter also presents the steps for setting up the NSGA-II and HypE for solving the MOP, i.e., selecting the chromosome representation, genotype, phenotype, etc. The arrangement of this chapter is given as follows. Section 4.1 and 4.2 present the MOP of the dynamic model and the constraint handling, respectively. The encoding and decoding of the genetic representation of the solution $\boldsymbol{x}$ = *gait parmaters* are introduced in Section 4.3. A brief introduction of the implementation of the MOEA and the simulator is given in Section 4.5. Finally, a simulation study, which gives an overview of the simulations in this thesis is presented in Section 4.6.

4.1 The multi-objective optimization problem

This section presents the multi-objective optimization problem for each motion pattern. The only differences between each MOP are the constraints on the external parameters called the *gait parameters*. In the context of GA, the gait paramaters represent the genes in MOEAs. The common internal constraints of the MOP for all the motion patterns is given as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_{opt}(\boldsymbol{x}) \\ \text{subject to} & \begin{cases} \left|\phi_i^*\right| \leq \phi_i^{max} \\ \left|\dot{\phi}_i^*\right| \leq \dot{\phi}_i^{max} \\ |u_i| \leq u_i^{max} \end{cases}, \end{aligned} \tag{4.1}$$

where the physical constraints ϕ_i^{max} and $\dot{\phi}_i^{max}$ are for the joints and u_i^{max} for the servo motors [13]. The goal in this MOP is to minimize and maximize the average power consumption (3.22) and the forward velocity (3.23) of the underwater snake robot, respectively. Thus, the multi-objective function $\boldsymbol{f}_{opt}$ is defined as

$$\boldsymbol{f}_{opt}(\boldsymbol{x}) = [P_{avg}, -\bar{v}]. \tag{4.2}$$

4.1.1 Lateral undulation and Eel-like motion

In [13], the multi-objective optimization problem for the underwater snake robot with lateral undulation and eel-like motion is defined as

$$\begin{aligned} &\min_{\alpha,\omega,\delta} \quad \boldsymbol{f}_{opt} \\ &\text{subject to} \begin{cases} 0 \le \alpha \le \alpha^{max} \\ 0 \le \omega \le \omega^{max} \\ 0 \le \delta \le \delta^{max} \end{cases}, \end{aligned} \tag{4.3}$$

where the parameter α, ω and δ are the gait parameters given in (3.15). Furthermore, the given gait parameters are not time-dependent.

4.1.2 Altered motion pattern 1 — Modified motion

In this section, we expand the optimization problem for the USR with lateral undulation motion given in (4.3) to include varying amplitudes in the joints, i.e., no common amplitude α in each joint. This is done by adding the vector $\boldsymbol{g} = [g_1, g_2, \dots, g_{n-1}] = [\alpha_1, \alpha_2, \dots, \alpha_{n-1}]$ to be a decision variable in the formulation of the MOP. Thus, by including $\boldsymbol{g}$, the optimization problem will be defined as

$$\begin{aligned} &\min_{\omega,\delta,\boldsymbol{g}} \quad \boldsymbol{f}_{opt} \\ &\text{subject to} \begin{cases} 0 \le \omega \le \omega^{max} \\ 0 \le \delta \le \delta^{max} \\ 0 \le g_i \le \alpha^{max} \end{cases}. \end{aligned} \tag{4.4}$$

For this optimization problem, the sinusoidal reference signal will therefore be given as

$$\phi_i^*(t) = g_i \sin(\omega t + (i-1)\delta) + \gamma, \quad i \in \{1, ..., n-1\}. \tag{4.5}$$

Note that with this MOP, the complexity in the search space $\mathcal{D}$ has significantly increased compared to the one in (4.3), i.e., the total number of decision variables will now be equal $n+1$ variables, where n is the number of links. Whereas the number of decision variables in (4.3) is only 3.

4.1.3 Altered motion pattern 2 — Fourier series motion

Instead of using the sinusoidal reference signal given in (3.15) as the motion pattern, we want to use a more general reference signal, e.g., in the form of Fourier series. There are some assumptions to be made beforehand, e.g., the reference signal is a $2L$-periodic and C^2 smooth function. With these assumptions and Theorem 2.6.1, its Fourier series converges to $\phi^*(t)$, $\dot{\phi}^*(t)$ and $\ddot{\phi}^*(t)$. Since $\phi^*(t)$ is $2L$-periodic function, its Fourier series is given as

$$\hat{\phi}_i^*(t) = a_0 + \sum_{k\in\mathbb{N}} \left(a_k \cos\left(\frac{k\pi t}{L}\right) + b_k \sin\left(\frac{k\pi t}{L}\right)\right) \tag{4.6}$$

Furthermore, the underwater snake robot proposed in [5], has n-links and thus we further assume that the reference signal (4.6) is equally phase shifted on all the joints such that

$$\hat{\phi}_i^*(t) = a_0 + \sum_{k\in\mathbb{N}} \left(a_k \cos\left(\frac{k\pi t}{L} + (i-1)\delta\right) + b_k \sin\left(\frac{k\pi t}{L} + (i-1)\delta\right)\right), \quad i \in \{1, \dots, n-1\}. \tag{4.7}$$

Addition to the phase shift δ, we can set $a_0 = 0$ to remove the offset of the reference signal in (4.7),

$$\hat{\phi}_i^*(t) = \sum_{k\in\mathbb{N}} \left(a_k \cos\left(\frac{k\pi t}{L}\right) + b_k \sin\left(\frac{k\pi t}{L}\right)\right), \quad i \in \{1, \dots, n-1\}. \tag{4.8}$$

By setting $a_0 = 0$, we ensure that $\int_{-L}^{L} \phi^*(t)dt = 0$. The torque signal u_i in each joint will be the same as the other motion patterns, but now with the Fourier series (4.7) as the reference signal. This gives us

$$u_i(t) = k_p(\hat{\phi}_i^*(t) - \phi_i(t)) + k_d(\dot{\hat{\phi}}_i^*(t) - \dot{\phi}_i(t)), \quad i \in \{1, \dots, n-1\}, \tag{4.9}$$

where

$$\dot{\hat{\phi}}_i^*(t) = \sum_{k\in\mathbb{N}} \frac{k\pi}{L} \left(-a_n \sin\left(\frac{k\pi t}{L} + (i-1)\delta\right) + b_n \cos\left(\frac{k\pi t}{L} + (i-1)\delta\right)\right) \tag{4.10}$$

$$\ddot{\hat{\phi}}_i^*(t) = \sum_{k\in\mathbb{N}} -\left(\frac{k\pi}{L}\right)^2 \left(a_n \cos\left(\frac{k\pi t}{L} + (i-1)\delta\right) + b_n \sin\left(\frac{k\pi t}{L} + (i-1)\delta\right)\right) \tag{4.11}$$

With this motion pattern, the formulation of the MOP is defined as

$$\begin{aligned} \min_{\omega,\delta,\boldsymbol{a},\boldsymbol{b}} \quad & f_{opt} \\ \text{subject to} & \begin{cases} 0 \le \omega \le \omega^{max} \\ 0 \le \delta \le \delta^{max} \\ 0 \le g_i \le \alpha^{max} \end{cases}. \end{aligned} \tag{4.12}$$

With this motion pattern, the complexity in the search space $\mathcal{D}$ is based on how many Fourier coefficients we inlcude in the reference signal, i.e., the total number of decision variables in (4.12) is $2 + 2k$, where k is the number of coefficients.

4.1.4 Altered motion pattern 3 — Multi-Fourier series motion

This motion pattern is based on the idea of the modified motion, i.e., we assign unique Fourier series to each joint. Thus, with this motion pattern, each joint have their unique reference signal they follow. The reference signal for this motion pattern is given as

$$\hat{\phi}_i^*(t) = a_0 + \sum_{k\in\mathbb{N}} \left(a_{k,i} \cos\left(\frac{k\pi t}{L} + (i-1)\delta\right) + b_{k,i} \sin\left(\frac{k\pi t}{L} + (i-1)\delta\right)\right), \ i \in \{1, \dots, n-1\}. \tag{4.13}$$

The MOP with this motion pattern is given as

$$\min_{\omega,\delta,\boldsymbol{a},\boldsymbol{b}} \quad f_{opt}$$
$$\text{subject to} \begin{cases} 0 \leq \omega \leq \omega^{max} \\ 0 \leq \delta \leq \delta^{max} \\ |a_{k,i}| \leq \alpha^{max} \\ |b_{k,i}| \leq \alpha^{max} \end{cases} . \tag{4.14}$$

The total number of decision variables with this motion pattern is $2 + 2k(n-1)$, where k and n is the number of Fourier coefficients and links, respectively.

4.2 Constraint handling

In this thesis, the constraint handling used in the MOEA is a simple scheme that punishes the infeasible solutions with an arbitrary fitness value [27, 61]. In NSGA-II, the fitness is in the form of ranks, where the lowest rank are the better solutions. Therefore, in the constraint handling scheme, we assign a higher rank to the infeasible solutions. This is done by adding an arbitrary value between $3-7$ with its current rank. Thus, with this method, we will not completely shut off the infeasible solutions so that they may still be selected. However, in the environmental selection, the feasible solutions will have the highest priority, e.g., first consider the feasible solutions and then the infeasible solutions. The scheme for creating the population $\mathcal{P}$ with the constraint handling is illustrated in Figure 4.1. The same concept applies to HypE, but instead of a rank

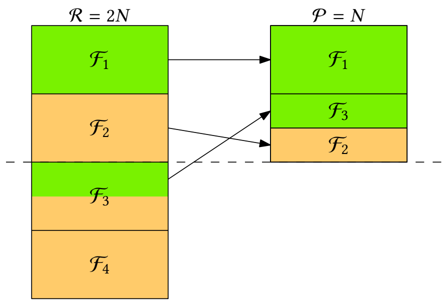


Figure 4.1: Constraint handling in MOEAs. The green and orange sections represent the feasible and infeasible solutions, respectively.

system, the solutions are assigned with a hypervolume indicator, where the fit solutions are the solutions with highest hypervolume indicator. Thus, instead of adding, we subtract an arbitrary value of the hypervolume indicator of each infeasible solution.

4.3 Genetic representation

In this section, the genetic representation of each gait parameters presented in Section 4.1 will be introduced. This involves the encoding and decoding of the chromosome representation.

4.3.1 Lateral undulation, Eel-like and Modified motion

The gait parameters involved with this motion patterns are α, ω and δ. Table 4.1 shows the genetic representation of each gait parameters. The upper bound value of the gait paramters given in

Table 4.1: Genetic representation for lateral undulation and eel-like motion

Gait Paramter	Bit String	Allele	Phenotype
α	7-bit	$[0, 127]$	$[0°, \alpha^{max}]$
ω	7-bit	$[0, 127]$	$[0°, \omega^{max}]$
δ	7-bit	$[0, 127]$	$[0°, \delta^{max}]$

Table 4.1 has been taken into account for selecting the bit string length of each gene. Note that the maximum allele value for ω does not reach its maximum $\omega^{max} = 210°$ Nm. This is acceptable since small frequency differences is not so critical, and the same argument goes for α and δ. To interpret this more, we will look at the decoding scheme for α, ω and δ. The decoding of the gait parameters in lateral undulation and eel-like motion are computed by dividing its current allele value with its maximum value and then multiply with its boundary value. This gives us,

$$\alpha = \frac{\alpha_{int}}{\alpha_{int}^{max}}\alpha^{max}, \qquad \omega = \frac{\omega_{int}}{\omega_{int}^{max}}\omega^{max}, \qquad \delta = \frac{\delta_{int}}{\delta_{int}^{max}}\delta^{max} \tag{4.15}$$

where the subscript *int* represents the integer value of a specific gene. Consider $\alpha_{int} = 64$, then the decoding of the α gene is given as $\alpha = \frac{64}{127} \cdot 90 = 45.35°$ Nm. For the modified motion pattern, the only difference is that each joint has a unique reference signal similar to the lateral undulation that they follow. Thus, instead of a single α paramter, we have g_i parameters. Table 4.2 shows the genetic representation of the paramters. Furthermore, the decoding of each g_i parameter is the

Table 4.2: Genetic representation for the modified motion

Gait Paramter	Bit String	Allele	Phenotype
g_i	7-bit	$[0, 127]$	$[0°, \alpha^{max}]$

same as the α paramter, i.e.,

$$g_i = \frac{g_i}{g_i^{max}}\alpha^{max} \tag{4.16}$$

For the sake of simplicity, the chromosome representation of the motion patterns is selected in the same order as they were presented in this thesis. Figure 4.2 and 4.3 show the chromosome representation of lateral undulation, eel-like and modified motion. From the figures, the chrom-

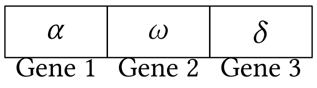


Figure 4.2: Chromsome representation of the lateral undulation and eel-like motion patterns

some length for lateral undulation and eel-like motion is $3 \cdot 7$-bits $= 21$-bits, while for the modified motion pattern is $(n + 1) \cdot 7$-bits.

ω	δ	g_i	$\cdots$	g_{n-1}
Gene 1	Gene 2	Gene 3		Gene n+1

Figure 4.3: Chromsome representation of the modified motion pattern

4.3.2 Fourier and multi-Fourier series motion

For the Fourier series motion patterns, we replace α with Fourier coefficients. Table 4.3 shows the genetic representation of the Fourier coefficients. In this motion pattern, we want to include

Table 4.3: Genetic representation for the Fourier series motion

Gait Paramter	Bit String	Allele	Phenotype
a_0	7-bit	$[0, 127]$	$[-\alpha^{max}, \alpha^{max}]$
a_n	7-bit	$[0, 127]$	$[-\alpha^{max}, \alpha^{max}]$
b_n	7-bit	$[0, 127]$	$[-\alpha^{max}, \alpha^{max}]$

negative values to the Fourier coefficients and thus the decoding scheme for the coefficients is given as

$$a_0 = \begin{cases} 0, \text{ if } \frac{a_{0,int}^{max}}{2} \leq a_{0,int} \leq \frac{a_{0,int}^{max}}{2} + 1 \\ \left(\frac{2a_{0,int}}{a_{0,int}^{max}} - 1\right)\alpha^{max}, \text{ else} \end{cases} \tag{4.17}$$

$$a_n = \begin{cases} 0, \text{ if } \frac{a_{n,int}^{max}}{2} \leq a_{n,int} \leq \frac{a_{n,int}^{max}}{2} + 1 \\ \left(\frac{2a_{n,int}}{a_{n,int}^{max}} - 1\right)\alpha^{max}, \text{ else} \end{cases}, \quad n \in \mathbb{N} \tag{4.18}$$

$$b_n = \begin{cases} 0, \text{ if } \frac{b_{n,int}^{max}}{2} \leq b_{n,int} \leq \frac{b_{n,int}^{max}}{2} + 1 \\ \left(\frac{2b_{n,int}}{b_{n,int}^{max}} - 1\right)\alpha^{max}, \text{ else} \end{cases}, \quad n \in \mathbb{N} \tag{4.19}$$

Note that the decoding scheme is uniformly, i.e., the value steps from zero to negative and postive values is the same. For example, let $a_{0,int} = 62$ and $a_{1,int} = 65$ then using the decoding scheme gives us $a_0 = \left(\frac{2\cdot62}{127} - 1\right)\cdot 90^\circ = -\left(\frac{270}{127}\right)^\circ$ and $a_1 = \left(\frac{2\cdot65}{127} - 1\right)\cdot 90^\circ = \left(\frac{270}{127}\right)^\circ$. For the chromsome representation, we will disregard a_0 (no offset) and consider the motion pattern (4.8). The chromsome representation of the Fourier and multi-Fourier series motion is shown in Figure 4.5 and 4.5. The

ω	δ	a_1	$\cdots$	a_k	b_1	$\cdots$	b_k
Gene 1	Gene 2	Gene 3		Gene k+2			Gene 2k+2

Figure 4.4: Chromsome representation of the Fourier series motion patterns

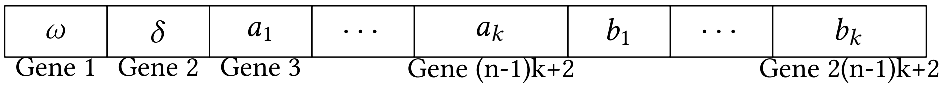


Figure 4.5: Chromsome representation of the multi-Fourier series motion patterns

total number of genes really depends on the number of Fourier coefficients we select. For example, let $k = 1$ then the total number of genes is 4 and 20 for Fourier and multi-Fourier series motion, respectively. Note that the number of genes increases significantly as the number of Fourier coefficients increases. Since all the genes have the same bit string length, the chromosome length for the Fourier and multi-Fourier series motion are therefore $(2k+2)\cdot 7$-bits and $(2(n-1)k+2)\cdot 7$-bits, respectively. The parameter L given in (4.7), is selected to be $L = \frac{k\pi}{\omega}$, where k is the number of Fourier coefficients. This is to limit the maximum frequency, i.e., $L^{min} = \frac{k\pi}{\omega^{max}} \leq L \leq k\pi = L^{max}$.

4.4 Simulation paramters

In this thesis, the simulation paramters such as the physical, hydrodynamic and fluid paramters are based on the values given in [5, 13, 4]. Furthermore, the gains for the low-level joint actuation controller (4.9) are selected identical to the values given in [13]. An overview of the model paramters are given in Table 4.4.

Table 4.4: Simulation paramters

Symbol	Value	Unit	Description
n	10		Number of links
l	0.18	[m]	Length of each link
m	0.8	[kg]	Mass of each link
a	0.055	[m]	Elliptical section (major)
b	0.05	[m]	Elliptical section (minor)
ρ	1000	[kg/m^3]	Fluid density
C_f	0.03		Drag coefficient in x
C_D	2		Drag coefficient in y
C_A	1		Added mass coefficient
C_M	1		Added inertia coefficient
k_p	20		P-gain
k_d	5		D-gain

4.5 Implementation

In section section, a brief introduction of the implementation for the MOEAs is presented. The implementation is done in *Python 3.5* with the help of the python libraries called *Numpy 1.13.0* and *DEAP 1.1.0* [62, 63, 64]. The Numpy library includes useful math operations and a powerful

N-dimensional array object, and DEAP includes a distributions of Evolutionary Algorithms, such as NSGA-II, SPEA2, PSO, etc. However, the distributed evolutionary algorithms from DEAP are not used in this thesis. All the algorithms are implemented from scratch. The DEAP library is used for storing the population and fitness values. Both the NSGA-II and HypE have the same main interface, which is given in Listing A.1. The only difference between those two is the environmental selection. The implementation of the environmental selection for both NSGA-II and HypE is shown in Listing A.2 and A.3, respectively. Furthermore, the number of crossover points in the multi-point crossover is selected to be equal three. An illustration of the optmization flow is given in Figure 4.6. In HypE, there is a need for a reference $\boldsymbol{R}$. This reference is chosen to be the

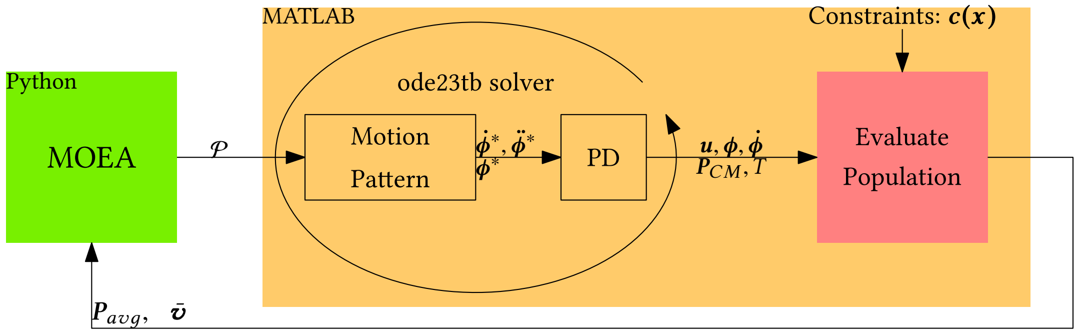

MATLAB
Constraints: $\boldsymbol{c}(\boldsymbol{x})$
Python
MOEA
ode23tb solver
$\mathcal{P}$
Motion Pattern
$\dot{\boldsymbol{\phi}}^*, \ddot{\boldsymbol{\phi}}^*$
$\boldsymbol{\phi}^*$
PD
$\boldsymbol{u}, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}$
$\boldsymbol{p}_{CM}, T$
Evaluate Population
$P_{avg}, \bar{\boldsymbol{v}}$

Figure 4.6: Illustration of the optimization loop.

boundary points of the simluations in the $\mathbb{R}^2$ space. These boundary points were found by running some test simulation and are given as $\boldsymbol{R} = \{[100, -2], [0, 0.05]\}$, where the first and second element equal $y_1 = P_{avg}$ and $y_2 = -\bar{v}$, respectively. Furthermore, the exact hypervolume indicator in HypE is computed by the MATLAB codes provided in [65]. The simulation model is implemented in *MATLAB 2017a* and provided from supervisor Krsitin Y. Pettersen and co-supervisor Eleni kelasidi [66]. Some modification has been made in the simulator, i.e., including the MATLAB function *parfor* in the *Parallel Computing Toolbox*. This function helps the computation time of evaluating the population; solving the Ordinary Differential Equations (ODEs) in parallel. The total number of solutions that can be evaluated in parallel depends on the number of physical cores the computer have. With two cores, one can simulate two solutions at the same time. This is doable, since in Python, we decode all the solutions beforehand and then sending them to MATLAB for evaluation. The computer specifications used for running the simulations in this thesis is a *Intel® Core™ i7-6700 Processor* with 32 *DDR4 RAM*. This processor has four cores and thus the simulation time is four times faster with the function parfor. The solver used for computing the ODEs was the *ode23tb solver* with a *realtive* and *absolute* error tolerance of 10^{-3}. Furthermore, the simulation total swimming time cycle T was set to 20 seconds.

4.6 Simulation study

In this section, we will present five different scenarios for simulations and interpretations:

- *Scenario 1:* Find optimal GA paramters.
- *Scenario 2:* Simulations with optimal GA paramters.
- *Scenario 3:* Using multivariate analysis for interpretation.

The first part consist of pre-simulations to find optimal or suboptimal GA parameters of each motion pattern, i.e., the population size N, crossover rate and mutation rate. These parameters will then further be used for the actual simulation results, were we will do a more thorough interpretation of the results. The tuning of the GA paramters is done by running consecutive simulations with varying parameters. Table 4.5 shows the different cases of varying the crossover and mutation rate. The role of the population size can be seen as a trade-off between diversity and

Table 4.5: Cases for tuning the GA paramters

Case	Crossover rate	Mutation rate
1	0.6	0.050
2	0.6	0.100
3	0.6	0.150
4	0.7	0.050
5	0.7	0.100
6	0.7	0.150
7	0.8	0.050
8	0.8	0.100
9	0.8	0.150

computation time. This implies that large initial population (expands more of the searching space) can help MOEAs to escape local optimums but at the cost of slower runtime [67, 68]. The choice of the population size is not trivial, one could do error and trial of the population sizes. If we base on the Shannon-Nyquist sampling theorem, which imply that converting a signal into numeric sequence, one should have a sample-rate of at least two [69]. Thus, in our case the population size N would be,

$$N = (sample\_rate \cdot 2^{n\_genes})^2 \tag{4.20}$$

However, with this scheme, the population size will be tremendously large, e.g., with the modified motion and a sample-rate of two, the population size would be $N = (2 \cdot 2^{11})^2 = 37748736$. Instead, we will consider an alternative scheme, which is given as

$$N = (sample\_rate \cdot n\_genes)^2. \tag{4.21}$$

With (4.21) and a sample-rate of two, the population size for the modified motion will be $N = (2 \cdot 11)^2 = 484$, which is significantly lesser. Table 4.6 shows the population sizes for tuning of each motion pattern. The tuning of GA parameters are computed with NSGA-II. After finding the optimal paramters, simulations are done for both NSGA-II and HypE.

Table 4.6: The population sizes for tuning the GA parmeters

Motion pattern	Number of genes	Population size
Lateral undulation & Eel-like	3	$N = \{100, 200, 300\}$
Modified	11	$N = \{200, 500, 800\}$
Fourier series ($k = 1$)	4	$N = \{100, 200, 300\}$
Fourier series ($k = 3$)	8	$N = \{100, 300, 500\}$
Multi-Fourier series	20	$N = \{200, 500, 800\}$

Chapter 5

Simulation results

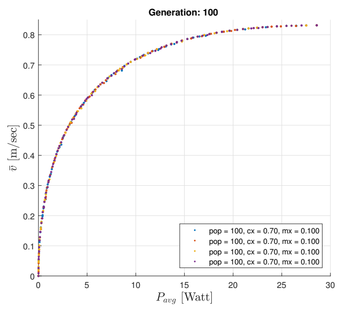
This chapter presents the simulations results. The arrangement of this chapter is given as follows. Section 5.1 presents the results of finding the optimal GA parameters for the actual simulations. In this section, the hypervolumes are used for comparison between the simulation cases given in Section 4.6. Section 5.2 inspects the motion patterns of the actual simulations, using PCA and PLSR. Furthermore, a regression model is constructed for each motion pattern.

5.1 Optimal GA paramters

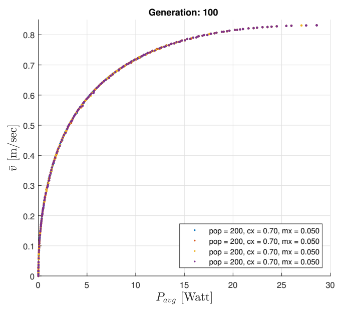
In this section, we will find the optimal GA paramters for the actual simulations in Section 5.2. To find the optimal GA paramters, we have to get a decent set of simulation data. Due to the long computation time, we have set the number of samples of each motion pattern to be four. Thus, with four samples, the total number of simulation runs for each motion pattern in Table 4.6 is $4 \cdot 9 \cdot 3 = 108$. Except for the Fourier series motion with $k = 3$ and multi-Fourier series. These motion patterns are only simulated with one sample. This is because of the time scope of this thesis. To ensure that the solutions converge, we set the total number of generations to be equal 100. This will give us a total number of 10800 generations to simulate for each motion pattern. The total number of data for comparison will therefore be overwhelmingly large. To give an example, the Figures A.3-A.5 show all the simluation results for the Fourier serires ($k = 3$) motion pattern. For simplicity, not all the simulation figures are included, only the valuable data based on the hypervolumes and how the soultions spread out the Pareto front are selected. Since we do not know the optimal Pareto front, the hypervolume indicator is a good way for comparing the simulation results.

5.1.1 Lateral undulation and Eel-like motion

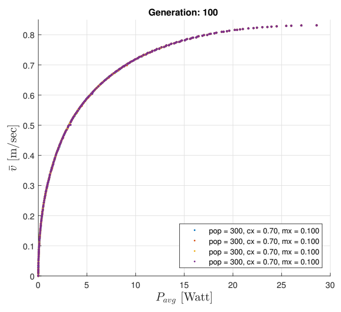
The hypervolume indicators for the lateral undulation given in Figure A.6, is an average of the four simulation samples. The figure shows that all the simulation results converge, i.e., the hypervolume indicator stops increasing. From Figure A.6, we select the cases that give the best average hypervolume indicator. The selected cases are shown in Figure 5.1, and these were based on the average hypervolume indicator given in Figure A.6d and A.6e. Observe that the Pareto front of all the samples do not have any gaps, i.e., the solutions cover the entire front and are uniformly spread out. In Figure 5.1a, the solutions have more space between them and is, therefore, less



(a) $cx = 0.70$, $mx = 0.100$, $N = 100$



(b) $cx = 0.70$, $mx = 0.050$, $N = 200$



(c) $cx = 0.70$, $mx = 0.100$, $N = 300$

Figure 5.1: Lateral undulation: The three cases with the best average hypervolume indicator.

dense than the fronts in Figure 5.1b and Figure 5.1c. The cause for this effect, is that the population size N for this case is smaller. With the computer specification given in Section 4.5, the total simulation time for one generation was approximated:

- 25 seconds for $N = 100$
- 57 seconds for $N = 200$
- 1 minute and 33 seconds for $N = 300$

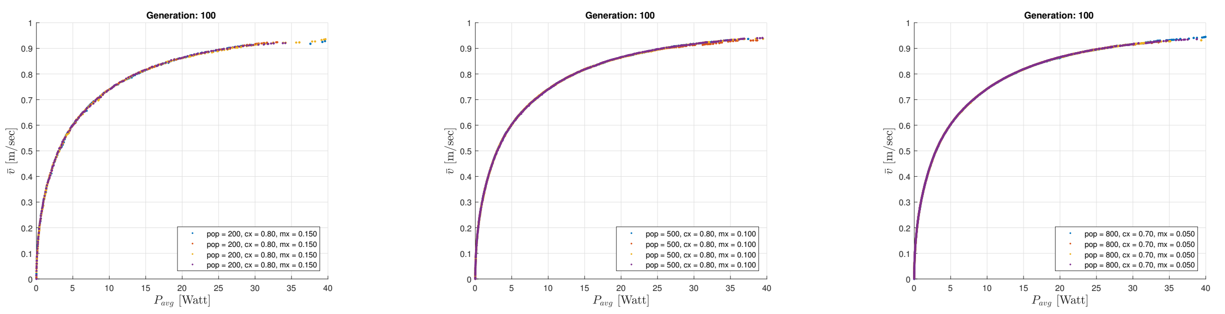
Thus, by considering the average hypervolume indicator, the Pareto fronts and the computation time, we select the optimal GA paramters for the lateral undulation and eel-like motion to be $N = 200$, $cx = 0.70$ and $mx = 0.050$.

5.1.2 Modified motion

The same approach for selecting the optimal GA parameters in the previous section will be applied for the modified motion. Figure A.7 shows the average hypervolume indicator for the cases with the modified motion. Observe that the hypervolume indicators in each subfigure given in Figure A.7 deviate more with each other compared to Figure A.6. This implies that the search space $\mathcal{D}$ in this case is not as simple as the search space of the lateral undulation. That is not so unexpected since the modified motion includes more decision variables. Figure 5.2 shows the three cases with the best average hypervolume indicator, respectively. Observe also that the solutions cover most of the Pareto front. Further observation of the Pareto fronts shows that in Figure 5.2a, some solutions are missing at the end. While the Pareto fronts in Figure 5.2b and 5.2c resemble each other. The total simulation time for one generation with the modified motion was about:

- 54 seconds for $N = 200$
- 2 minutes and 23 seconds for $N = 500$
- 4 minutes and 4 seconds for $N = 800$

Note that most of the hypervolumes for the case with $N = 200$ converge to a lower value than $N = 500$ and $N = 800$, which means that with $N = 200$, the solutions either cover less of the



(a) $cx = 0.80$, $mx = 0.150$, $N = 200$

(b) $cx = 0.80$, $mx = 0.100$, $N = 500$

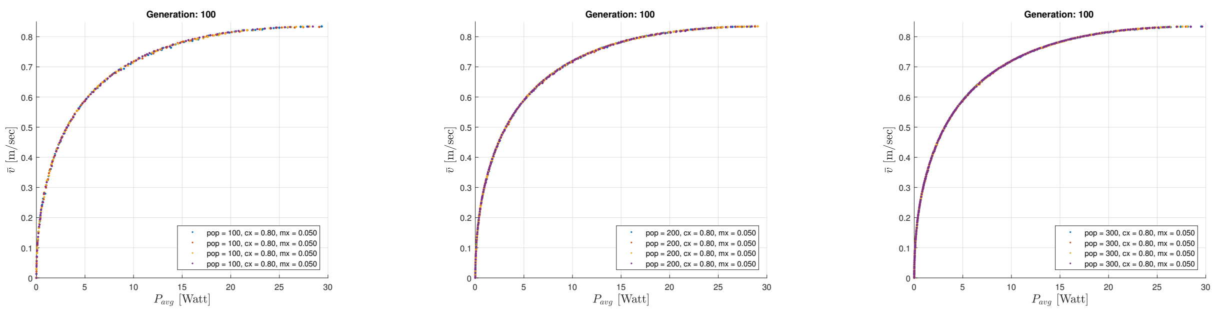
(c) $cx = 0.70$, $mx = 0.050$, $N = 800$

Figure 5.2: Modified: The three cases with the best average hypervolume indicator.

Pareto fronts or converge to a local optimum worse than the others. Thus, by considering the computation time, the Pareto fronts and the hypervolumes, the GA paramters for the modified motion are selected to be $N = 500$, $cx = 0.80$ and $mx = 0.100$.

5.1.3 Fourier series motion

In this section, the optimal GA paramters will be decided for the Fourier series motion pattern. Figure A.8 shows the average hypervolume indicator for the cases with $k = 1$. Observe that the hypervolume indicators in the subfigures resemble the indicators in Figure A.6. Figure 5.3 shows the samples based on the hypervolume indicator given in Figure A.8g. These hypervolumes



(a) $cx = 0.80$, $mx = 0.050$, $N = 100$

(b) $cx = 0.80$, $mx = 0.050$, $N = 200$

(c) $cx = 0.80$, $mx = 0.050$, $N = 300$

Figure 5.3: Fourier series ($k = 1$): The three cases with the best average hypervolume indicator.

compared to the other subfigures have the smallest deviation between them and also show that the samples have good convergence. Observe that the subfigures in Figure 5.3 are similar to each other. Furthermore, since the hypervolume indicator end up in the same value and have the same structure, the selection of the GA parameters for this motion pattern is not too crucial. The simulation time for one generation with this motion pattern was about:

- 40 seconds for $N = 100$

- 1 minutes and 25 seconds for $N = 200$
- 2 minutes and 10 seconds for $N = 300$

With the same arguments as in the previous sections, the selected optimal GA parameters for the Fourier series with $k = 1$ motion are $N = 200$, $cx = 0.80$ and $mx = 0.050$.

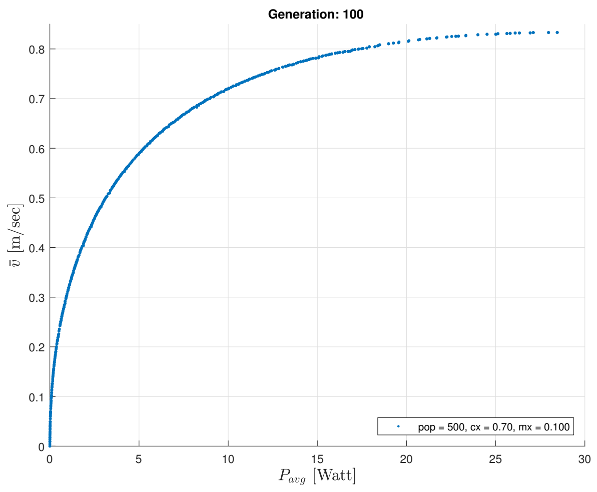


Figure 5.4: Fourier series ($k = 3$): The sample with $N = 500$ and highest hypervolume indicator.

For the Fourier series with $k = 3$ coefficients, the total number of decision variables increases to 8. Furthermore, with more Fourier coefficients, the approximation of an optimal reference ϕ_i^* will be better. However, with more coefficients, the search space $\mathcal{D}$ becomes more complicated. This can be seen in Figure A.9. The hypervolume indicators of this case is not as smooth as the hypervolumes in Figure A.8. Many of the samples converge to a local optimum with poor Pareto fronts, shown in Figure A.3-A.5. By observations, almost all the samples in Figure A.5 give a sufficient Pareto front, compared to the samples in Figure A.3 and A.4. This shows that the quality of the Pareto fronts for this case are greatly affected by the population size N. With a larger population size, the solutions cover more of the search space $\mathcal{D}$, which may help the MOEAs to avoid the local optimums. Thus, the selected optimal GA parameters for this motion pattern are based on the highest hypervolume indicator with $N = 500$, which gives $cx = 0.70$ and $mx = 0.100$. Furthermore, with these GA parameters, the approximated simulation time for one generation was about 5 minutes and 42 seconds. Note the Pareto front shown in Figure 5.4 resembles the fronts in Figure 5.3. This may or may not indicate that there are almost no difference between $k = 1$ and $k = 3$ coefficients. Further investigation of this motion pattern will be given in the upcoming sections.

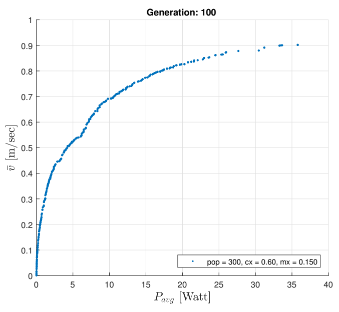
5.1.4 Multi-Fourier series motion

Figure A.10 shows the hypervolumes of the multi-Fourier series motion. The observations of the subfigures show that not all the samples have converged. This indicate that simulation with a total number of generation equal 100 is not sufficient for this motion pattern. Figure 5.5 shows the best selected samples. In Figure 5.5a the Pareto front is not as smooth as the fronts in Figure 5.5b and

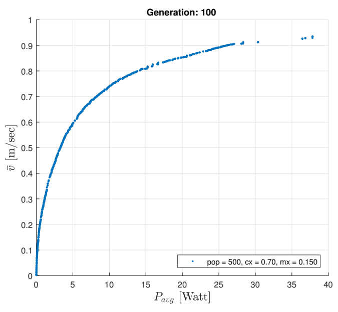
5.5c. The Pareto fronts in Figure 5.5b and 5.5c are quite similar to each other. However, the front in Figure 5.5b is more extended and have a gap between the solutions at the end of the front. This gap shows that the solutions may not have converged to their optimal values yet or that the search space $\mathcal{D}$ is too complexed for the MOEAs to evenly spread all the solutions on the whole Pareto front. The simulation time for one generation with this motion pattern was about:

- 4 minutes and 30 seconds for $N = 300$
- 6 minutes and 48 seconds for $N = 500$
- 9 minutes and 6 seconds for $N = 800$

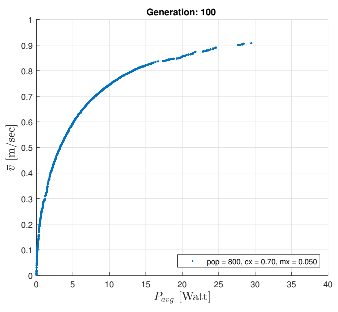
Observe the long the simulation runtime for this motion pattern. The cause for this long simulation time is because to the number of decision variables (20 variables), which may bring the USR model to operate in the infeasible search space. Furthermore, some of the samples with $N = 300$ had really poor performance. Based on the simulation time and Pareto fronts in Figure 5.5, the optimal GA paramters for this motion pattern is selected to be $N = 500$, $cx = 0.70$ and $mx = 0.150$.



(a) $cx = 0.80$, $mx = 0.050$, $N = 100$



(b) $cx = 0.80$, $mx = 0.050$, $N = 200$



(c) $cx = 0.80$, $mx = 0.050$, $N = 300$

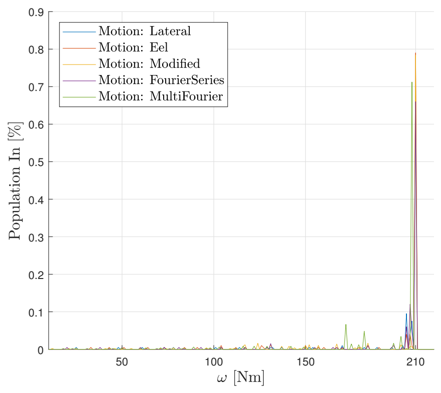
Figure 5.5: Multi-Fourier series: The three cases with the best average hypervolume indicator.

5.2 Simulations with optimal GA paramters

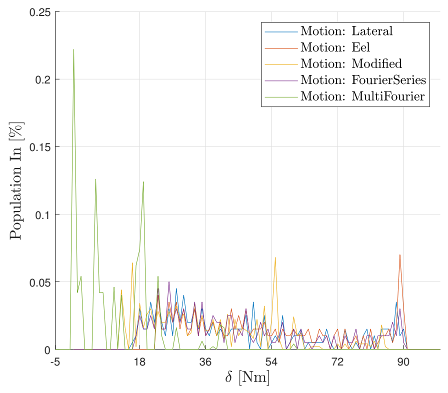
This section presents the simulation results with the optimal GA paramters. The number of generations for the modified and the multi-Fourier motion have been increased to 200. This is to ensure that the population properly converges. For the lateral undulation, eel-like and Fourier series motion, the number of generations will stay the same, which is 100. This section is organized as follows: (i) An investigation of the common gait paramters ω and λ of each motion pattern. (ii) Comparison between NSGA-II and HypE, using both the hypervolume indicators and Pareto fronts. (iii) An investigation of the Pareto fronts for each motion pattern. (iv) Theoretical anlysis of each motion pattern using the multivariate analysis introduced in Section 2.7.

5.2.1 Distribution of the gait paramters ω and δ

In this section, we inspect the distibution of ω and δ in the population of each motion pattern. The aim of this investigation is to check if it is possible to disregard either ω or δ, to reduce the number of decision variables. The distribution of ω and δ in the population is shown in Figure 5.6. From



(a) Distribution of ω.



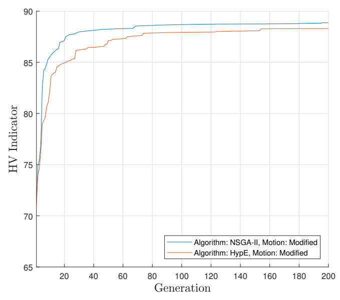
(b) Distribution of δ.

Figure 5.6: Distribution of ω and δ in the population $\mathcal{P}$.

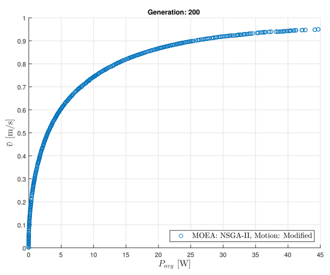
Figure 5.6a, almost all the solutions have $\omega = \omega^{max}$. This may indicate that the optimal frequency is ω^{max}. One may therefore try and disregard ω by replacing it with ω^{max}. The structure of the phase shift δ, is not as trivial as the frequency ω. Further investigation of this gait parameter is done with multivariate analysis using PCA.

5.2.2 Comparison between NSGA-II and HypE

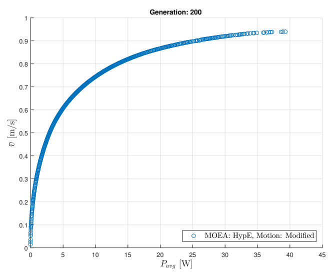
The hypervolume indicator of the Pareto front for a particular motion is used for comparison between NSGA-II and HypE. The modified motion is considered for comparing the MOEAs. This



(a) Modified motion: Hypervolumes.



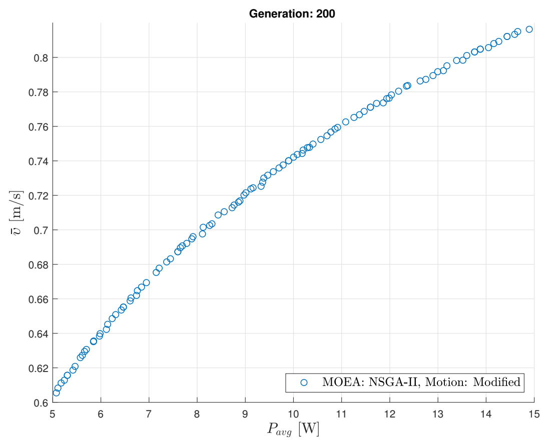
(b) Modified motion: NSGA-II.



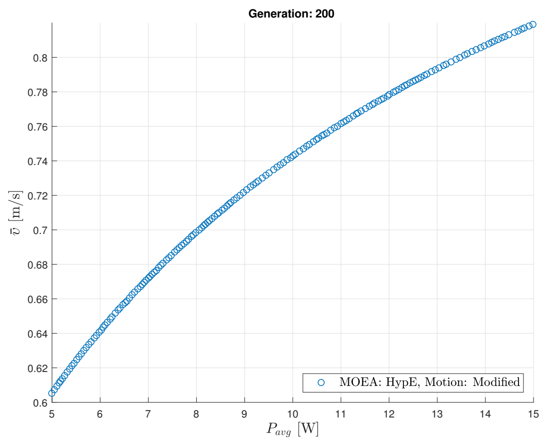
(c) Modified motion: HypE.

Figure 5.7: Comparison between NSGA-II and HypE with modified motion.

is due to the total number of decision variables the modified motion has. Figure 5.7 shows the hypervolume indicator of the Pareto fronts of both NSGA-II and HypE given in Figure 5.7b and



(a) Modified motion: NSGA-II.



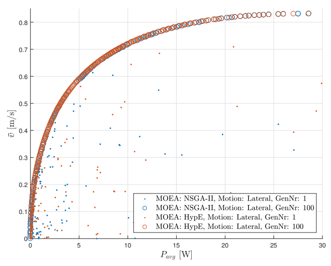
(b) Modified motion: HypE.

Figure 5.8: Comparison of the Pareto front for modified motion between NSGA-II and HypE. The HypE algorithm spreads the solutions more uniformly on the Pareto front compared to NSGA-II.

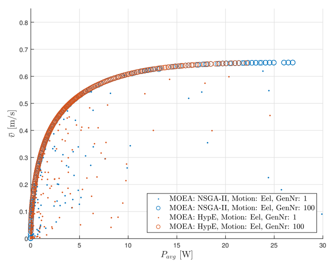
5.7c, respectively. Observe that the Pareto front in Figure 5.7b extends a little bit furter on the x-axis. This extension affects the hypervolume indicator shown in Figure 5.7a. As a consequence, the hypervolume indicator of the Pareto front computed by NSGA-II is larger than with HypE. By observing the hypervolumes, both MOEAs seems to converge at the same rate. For further comparison between the MOEAs, a smaller section of the Pareto front is considered. Figure 5.8 shows the Pareto front of NSGA-II and HypE between $5 \leq P_{avg} \leq 15$. By comparing Figure 5.8a with 5.8b, we see that the Pareto front computed by HypE has a more uniformly spread out solutions than NSGA-II. The cause for this effect, is that HypE considers the hypervolume indicaotor for spreading out the solutions.

5.2.3 Pareto fronts of the motion patterns

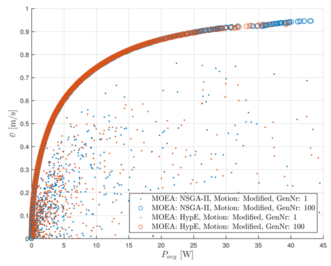
The Pareto fronts of the first and last generation for each motion pattern are shown in Figure 5.9. The subfigures show that the population $\mathcal{P}$ is arbitrary initialized in both NSGA-II and HypE and ends up as the Pareto front in the last generation. Note that there exist no gaps in the Pareto fronts, which is a good indication. This shows that $\mathcal{D}$ of all the motion patterns is not too complexed. Furthermore, observe that Figure 5.9a, 5.9d and 5.9e resembles each other and Figure 5.9c with 5.9f. The only Pareto front that is unique or different is the eel-like motion shown Figure 5.9b. From the observations of the distribution plot of ω shown in Figure 5.6, resimulations of the motion patterns with $\omega = \omega^{max}$ are done. Figure 5.10 shows the Pareto fronts of the resimulated motion patterns. These Pareto fronts resemble the fronts shown in Figure 5.9, with the exception in Figure 5.10c and 5.10f, where solutions with low $\bar{v}$ and P_{avg} in the modified motion are missing. In the upcoming sections, further interpretations of the motion patterns will be presented.



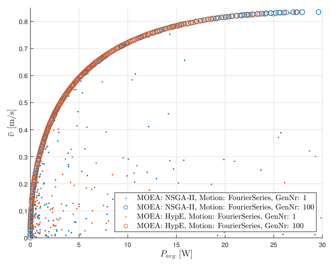
(a) Lateral undulation



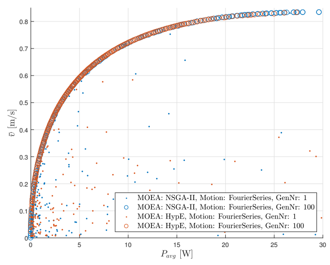
(b) Eel-like



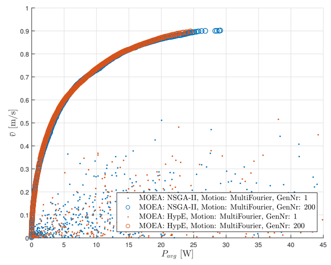
(c) Modified



(d) Fourier series ($k = 1$)

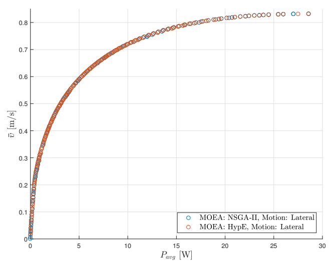


(e) Fourier series($k = 3$)

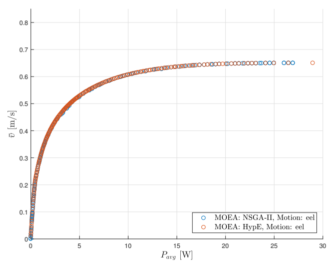


(f) Multi-Fourier series

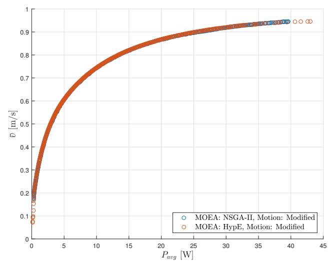
Figure 5.9: Initial and final Pareto fronts of each motion pattern.



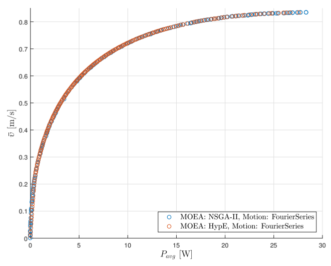
(a) Lateral undulation



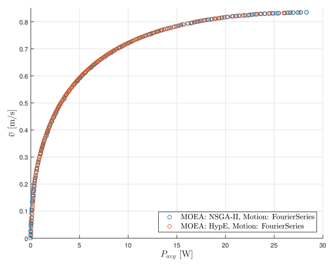
(b) Eel-like



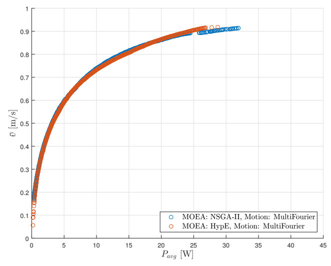
(c) Modified



(d) Fourier series ($k = 1$)



(e) Fourier series ($k = 3$)



(f) Multi-Fourier series

Figure 5.10: Final pareto fronts of each motion pattern without ω.

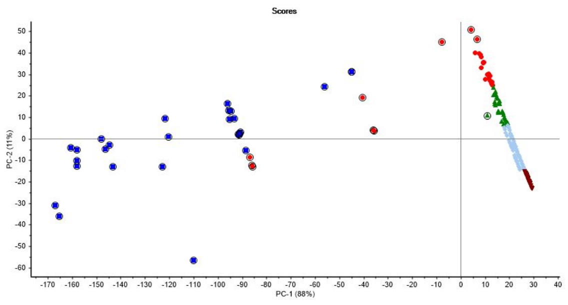
5.2.4 Multivariate analysis of The motion patterns

This section presents the interpretation of the motion patterns using multivariate analysis. The interpretation of the lateral undulation will be more detailed than the others. However, the interpretation procedure is the same for all the motion patterns.

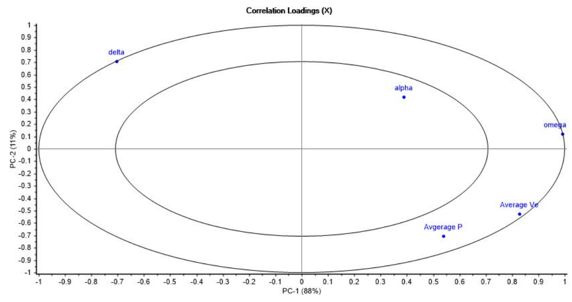
5.2.4.1 Lateral undulation

Figure 5.11 shows the PCA overview for the lateral undulation computed by NSGA-II. The samples in the scores plot shown in Figure 5.11a, are the solutions $\boldsymbol{x} \in \mathcal{P}$. In this plot, the samples are highlighted with different colors. The colors repersent the average velocity of each sample, such as:

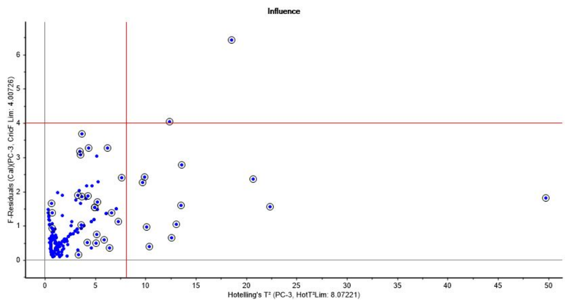
- $0.0 \leq \bar{v} < 0.2$ m/s: blue color
- $0.2 \leq \bar{v} < 0.4$ m/s: red color
- $0.4 \leq \bar{v} < 0.6$ m/s: green color
- $0.6 \leq \bar{v} < 0.8$ m/s: light blue color
- $0.8 \leq \bar{v} < 1.0$ m/s: brown color



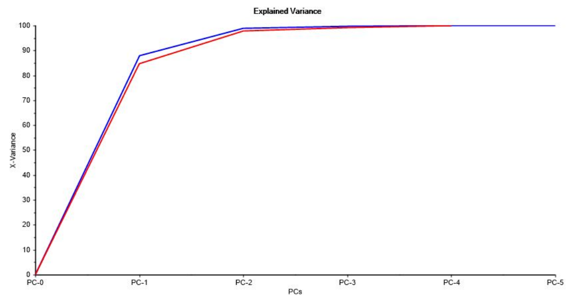
(a) Scores



(b) Loadings



(c) Influence

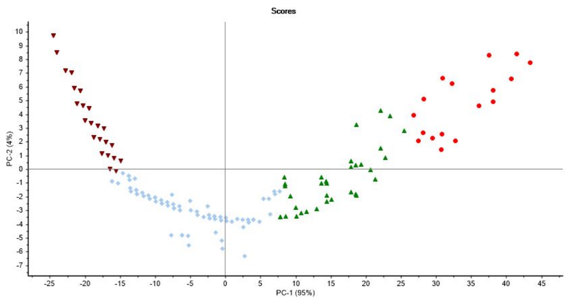


(d) Explained Variance

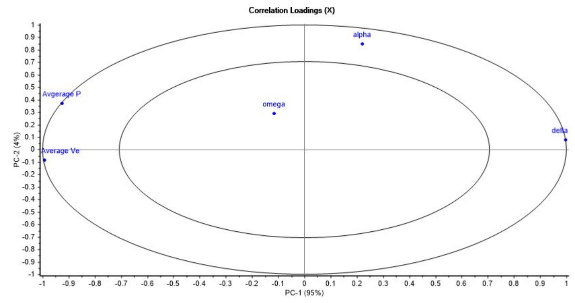
Figure 5.11: Lateral undulation: PCA overview

By observing both the figures Figure 5.11a and Figure 5.11c, some outliers can be identified, i.e., the marked samples in the figure. The loadings plot given in Figure 5.11b, shows that samples with high frequency ω also have high P_{avg} and $\bar{v}$. However, this model is hard to interpret because of the outliers, which influence the PCA model significantly. Note that the outliers are the samples with low P_{avg} and $\bar{v}$, i.e., the blue samples which have average velocities between 0 and 0.2 m/s.

Observe also that from Figure 5.11d, the optimal number of PCs for this PCA model is equal to two. To get a better PCA model, we recalculate the model without the outliers. The new PCA model is shown in Figure 5.12. The scores plot in Figure 5.12a of this PCA model shows a more

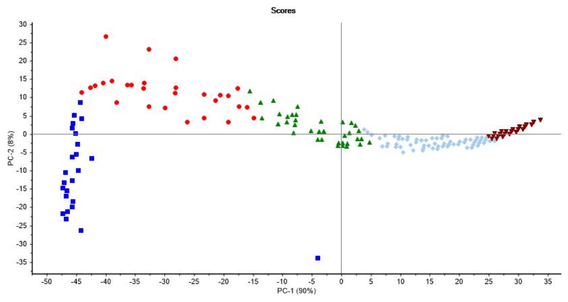


(a) Scores.

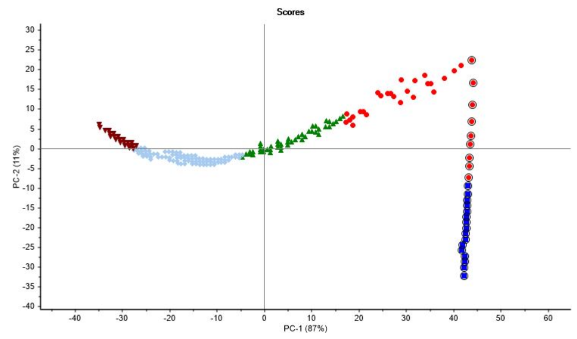


(b) Loadings

Figure 5.12: Lateral undulation: Scores and Loadings



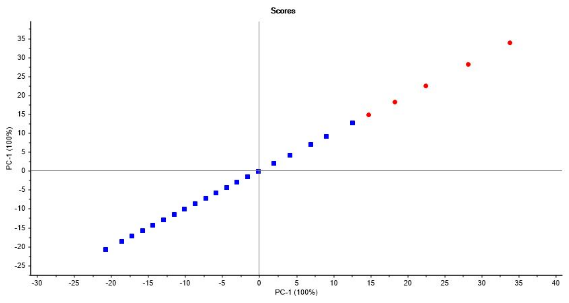
(a) Lateral undulation: Scores with NSGA-II.



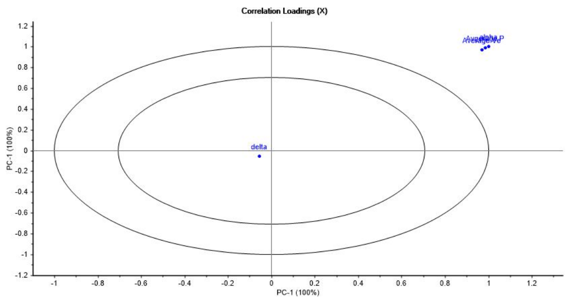
(b) Lateral undulation: Scores with HypE.

Figure 5.13: Lateral undulation: Scores plot with NSGA-II and HypE.

visible structure of the solutions with less outliers, and it explains 99% variance with only two PCs. Furthermore, the loadings plot shown in Figure 5.12b have also changed. In the loadings plot, the variable ω has moved closer to origo. This may indicate that the variable ω is badly valuated and has insignifcant variance to not be interpretable in the PCA model. The variable ω acts as noise to the PCA model, i.e., the variable has insignificant variance. Thus, with this observation, the following PCA models are based on the Pareto fronts given in Figure 5.10. The Figure 5.13a and 5.13b show the scores plot for the lateral undulation without ω computed by NSGA-II and HypE, respectively. The plots can be seen as almost reflection of each other about the y-axis. However, this is not important. What we are concerned about is the strucure of the scores. In Figure 5.13b, the scores are less spread out and more dense than the scores in Figure 5.13a. Furthermore, some distinct outliers can be detected in Figure 5.13a, while almost none outliers can easily be seen in Figure 5.13b. The cause for this difference in the scores structure is because HypE spreads out the solutions more uniformly on the Pareto front than NSGA-II does. With this observation, the following PCA models in this section and further sections will be based on the solutions computed by HypE. In Figure 5.13b, two different clusters can be seen: cluster 1, the marked samples (rightmost vertical samples), and cluster 2, the unmarked samples. These two clusters

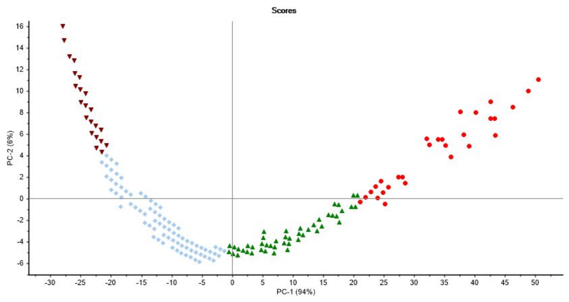


(a) Lateral undulation; Cluster 1 scores.

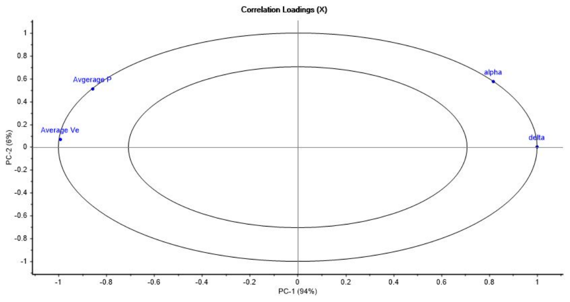


(b) Lateral undulation: Cluster 2 loadings.

Figure 5.14: Lateral undulation: The PCA scores and loadings of cluster 1.

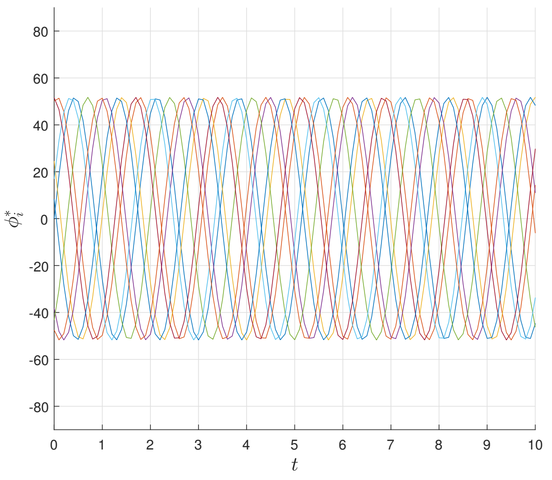


(a) Lateral undulation: Cluster 2 scores.

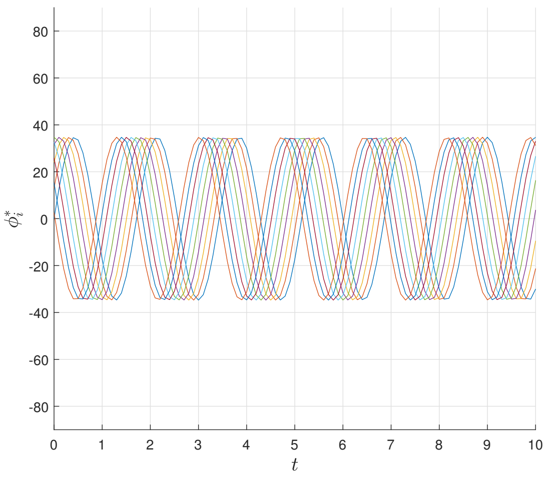


(b) Lateral undulation: Cluster 2 loadings.

Figure 5.15: Lateral undulation: PCA scores and loadings of cluster 2.



(a) Low velocity, $\bar{v} \approx 0.3$ m/s



(b) High velocity, $\bar{v} \approx 0.8$ m/s

Figure 5.16: Lateral undulation: reference ϕ_i^* with low and high $\bar{v}$.

will be interpreted in seperate PCA models. Figure 5.14 shows the scores and loadings plot for the cluster 1. In this PCA model, only one PC is sufficient to explain all the variance of the data set. Observe that in cluster 1, the $\bar{v}$, P_{avg} and α are highly positively correlated to each other. That is, if α increases, so do $\bar{v}$ and P_{avg}. This can also be seen in the raw data of cluster 1 shown in Table A.1. For the PCA model with cluster 2, two PCs are needed for explaining 99% of the variance. In the loadings plot shown in Figure 5.15, the variable δ is positively correlated with α and negatively correlated with P_{avg} and $\bar{v}$. Note that δ and $\bar{v}$ are located on the same PC, which is PC-1. Thus, as δ decreases, $\bar{v}$ increases. It is more difficult to interpret the other two variables P_{avg} and α since we must consider both PC-1 and PC-2. To interpret these variables, both the scores and loadings plot must be examined together. The structure of the samples resemble a convex curve and as we move on this curve from right to left, values of α decreases. But as soon as we pass the y-axis, the curve starts moving towards postive y-values. This has an oppisite effect on the α-values, i.e., the effect of moving along the negative x-axis is reduced by the effect of moving along the postive y-axis. For the variable P_{avg}, its value increases from moving right to left on the convex curve, but it increases more significantly after passing the y-axis.

The reference ϕ_i^* shown in Figure 5.16, where each curve represents one joint, are the joint references for low and high $\bar{v}$ of the lateral undulation. In Figure 5.16a, we see that ϕ_i^* has low amplitudes between [-20,20]. Furthermore, note also that the phase shift δ is large in the subfigure. For high velocity shown in Figure 5.16b, the amplitudes and the phase shift of ϕ_i^* has increased and decreased, respectively. Note that these results correspond to the observations and assumptions from interpreting the PCA models. After interpreting the PCA models, a PLSR model for both cluster 1 and 2 is constructed. The purpose of the PLSR models is to see if the gait parameters α and δ can be predicted using the objectives P_{avg} and $\bar{v}$. Thus, the predictors and responses in the PLSR model will be the objectives and gait paramters, respectively. The RMSE values after constructing the PLSR models are shown in Table 5.1. From the RMSE table, the values of Factor

Table 5.1: Lateral undulation: The RMSE of cluster 1 and 2

	Cluster 1		Cluster 2	
Factor	α	δ	α	δ
0	15.342	0.240	6.403	17.926
1	3.994	0.256	5.815	8.942
2	1.114	0.263	1.814	1.906

2 is the one we are concerned about. The RMSE values of this factor are lesser than 5, which is good. This shows that the differences between the observed and predicted values are lesser than 5. The Beta coefficients of the PLSR models are computed as

$$\boldsymbol{B}_1 = \begin{bmatrix} 6.924 & 89.686 \\ 260.512 & -7.216 \\ -35.954 & 1.602 \end{bmatrix} \qquad \boldsymbol{B}_2 = \begin{bmatrix} 69.356 & 106.988 \\ 1.149 & 0.286 \\ -6.915 & -11.378 \end{bmatrix} \tag{5.1}$$

With the Beta coefficients, the regression model can be written as

$$\hat{Y}_1 = \boldsymbol{X}\boldsymbol{B}_1 \tag{5.2}$$

$$\hat{Y}_2 = \boldsymbol{X}\boldsymbol{B}_2 \tag{5.3}$$

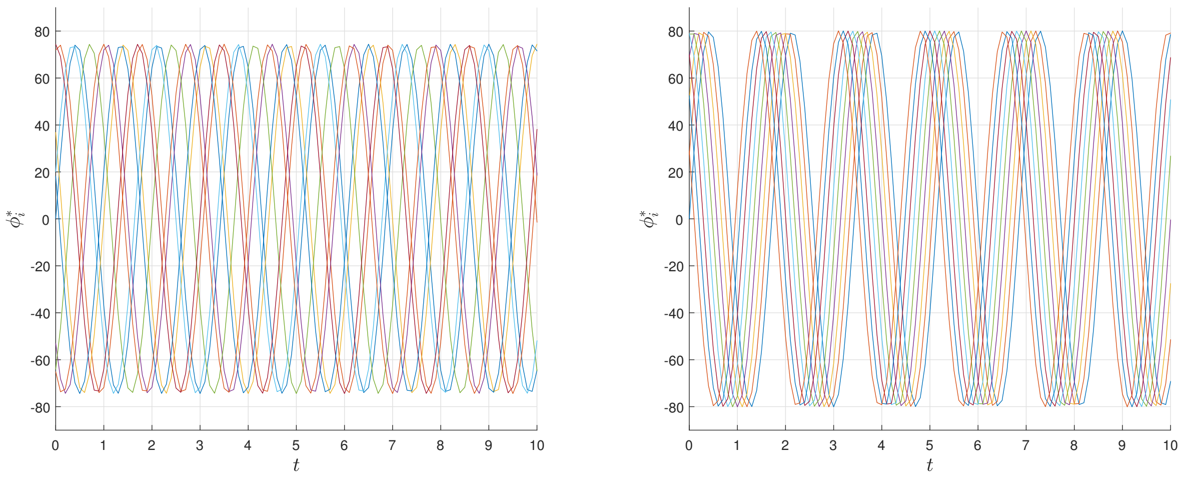
where $\boldsymbol{X} = \left[\mathbf{1}, \boldsymbol{P_{avg}}, \bar{\boldsymbol{v}}\right]$ are the predictors and $\hat{Y} = [\boldsymbol{\alpha}, \boldsymbol{\delta}]$ are the responses. By inspecting the raw data of cluster 1 given in Section A.6, the maximum $\bar{v}$ is close to 0.2 m/s. Thus, we can combine the two regression models by letting

$$\hat{\boldsymbol{y}} = \begin{cases} \boldsymbol{x}\boldsymbol{B}_1, & \text{if } \bar{v} \leq 0.2 \text{ m/s} \\ \boldsymbol{x}\boldsymbol{B}_2, & \text{if } \bar{v} > 0.2 \text{ m/s} \end{cases} \tag{5.4}$$

Since the regression model takes both P_{avg} and $\bar{v}$ as arguments for predicting the gait parameters, the values for both objectives must therefore be available. As a consequence, a curve fitting function $\bar{v} = f(P_{avg})$ is created for mapping between P_{avg} and $\bar{v}$ based on the Pareto fronts given in Figure 5.10. The curve function is created by using *polyfit* of degree 16 in MATLAB and is used in the regression model. Therefore, by selecting an arbitrary P_{avg}, we can predict the gait paramters. For example, consider that $P_{avg} = 6$ W with lateral undulation, then the corresponding gait paramters are given as $\hat{y}(P_{avg}) = [33.101, 37.707] = [\alpha, \delta]$. Some predicted gait paramters with $P_{avg} = \{0, 0.2, \ldots, 1, 2, 4, \ldots, P_{max} \approx 28\}$ W is given in Table A.2. Further interpretation of the prediction models will be given in Section 5.2.5.

5.2.4.2 Eel-like motion

Same as the lateral undulation, the PCA model for the eel-like motion has two different score clusters. The scores and loadings plot of the PCA model for the two clusters are shown in Figure A.11. All the subfigures resembles the figures given in Figure 5.14 and 5.15. Thus, the arguments given for the lateral undulation in Section 5.2.4.1 will also apply for this motion pattern. The only difference, is that, with eel-like motion, no samples with $\bar{v} > 0.8$ exist. Figure 5.17 shows that the shape of ϕ_i^* resembles the lateral undulation in Figure 5.16. The only difference is the amplitude α, which is larger in both low and high $\bar{v}$. However, the amplitudes of the eel-like motion has a scaling function given in (3.17). This scaling function will affect the amplitude of ϕ_i^* to decrease as we move closer to the head of the USR.

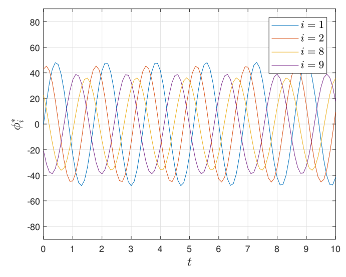


(a) Low velocity, $\bar{v} \approx 0.3$ m/s **(b)** High velocity, $\bar{v} \approx 0.8$ m/S

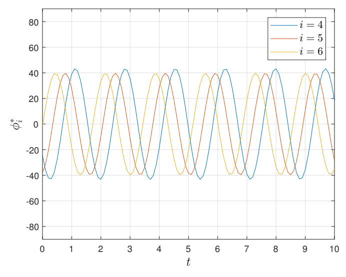
Figure 5.17: Eel-like motion: Reference ϕ_i^* with low and high $\bar{v}$.

5.2.4.3 Modified motion

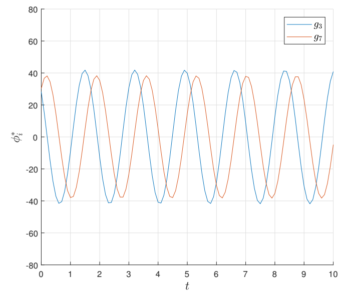
In this section, the interpretation of the modified motion pattern is presented. Figure A.12 shows the PCA scores and loadings of the model. In the loadings plot shown in Figure A.12b, the variable



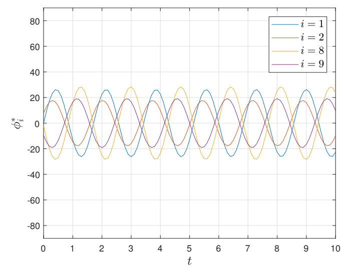
(a) Low velocity, $\bar{v} \approx 0.3$ m/s: ϕ_i^*, $i \in \{1, 2, 8, 9\}$



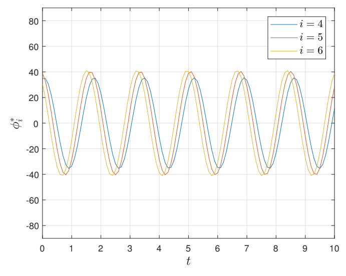
(b) Low velocity, $\bar{v} \approx 0.3$ m/s: ϕ_i^*, $i \in \{4, 5, 6\}$



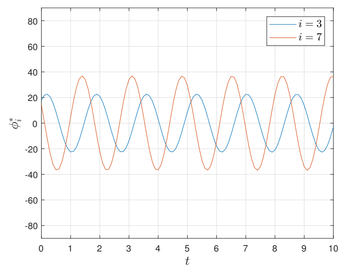
(c) Low velocity, $\bar{v} \approx 0.3$ m/s: ϕ_i^*, $i \in \{3, 7\}$



(d) High velocity, $\bar{v} \approx 0.8$ m/s: ϕ_i^*, $i \in \{1, 2, 8, 9\}$



(e) High velocity, $\bar{v} = 0.8$ m/s: ϕ_i^*, $i \in \{4, 5, 6\}$



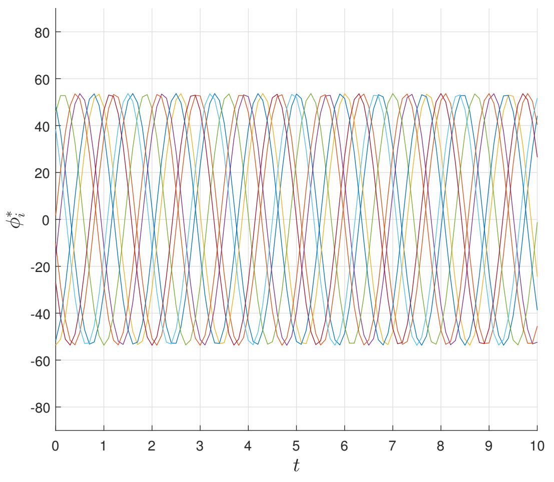
(f) High velocity, $\bar{v} = 0.8$ m/s: ϕ_i^*, $i \in \{3, 7\}$

Figure 5.18: Modified motion: Reference ϕ_i^* with low and high $\bar{v}$.

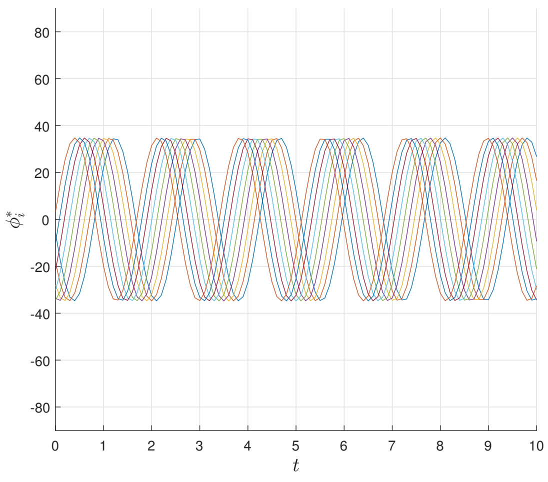
δ, g_1, g_2, and g_9 are close to each other. This imply that the variables are positively correlated with each other. Furthermore, observe that the varables g_5 and g_6 are also positively correlated with each other. These variables will therefore have a strong positive linear relationship between them. By observing both the scores and loadings plot, some assumption of the modified motion can be made. That is, high phase shift δ and amplitudes of the tail (g_1 and g_2) and head (g_9) joints cause low $\bar{v}$ ($\bar{v} \approx 0.2$). For high $\bar{v}$ ($\bar{v} \approx 0.8$), the oppisite situation, i.e., lower δ and amplitudes in the head and tail joints. Note that the scores resemble a convex curve and thus the values of g_2, g_3 and g_8 decreases faster than g_9, when moving on this curve from right to left. Since it is a convex curve, the values of g_4, g_5 and g_6 do not change that much on the right side of the y-axis. However, moving from right to left on this curve in the left side of the y-axis, will increase the values of these variables. Furthermore, note that g_3 is most affected by PC-2, which imply that this variable increases and decreases most when moving along the y-axis. Thus, high $\bar{v}$ with this motion pattern can be seen as high and low amplitudes in the middle and end joints of the USR, respectively. To strengthen these observations, plots of ϕ_i^* with both low and high $\bar{v}$ are given in Figure 5.18. In Figure 5.18a-5.18c, the amplitude of ϕ_i^* are somewhat close to each other. Thus the modified motion with low $\bar{v}$ resembles the lateral undulation. For high $\bar{v}$, the amplitudes in Figure 5.18d are much lower than in Figure 5.18e, which corresponds to the observations made from the PCA model. Note that only the high and low $\bar{v}$ were investigated, the motion pattern may vary in between $0.2 < \bar{v} < 0.8$. It is hard to precisely tell how this pattern looks like, because of all the varying amplitudes of each joint.

5.2.4.4 Fourier series motion

The interpretation of the Fourier series motion is more complicated than the other motion patterns. Figure A.13 shows the scores and loadings of two sample clusters for Fourier series with $k = 1$. The samples in Figure A.13a show that the $\bar{v}$ is not dependent on δ, and that $\bar{v} \approx 0.2$ m/s have larger and lower a_1 and b_1 values, respectively. Note in the PCA model, just two PCs explain 100% of the variance. For the PCA model of cluster 2, 98% of the variance is explained with two



(a) Low velocity, $\bar{v} \approx 0.30$ m/s

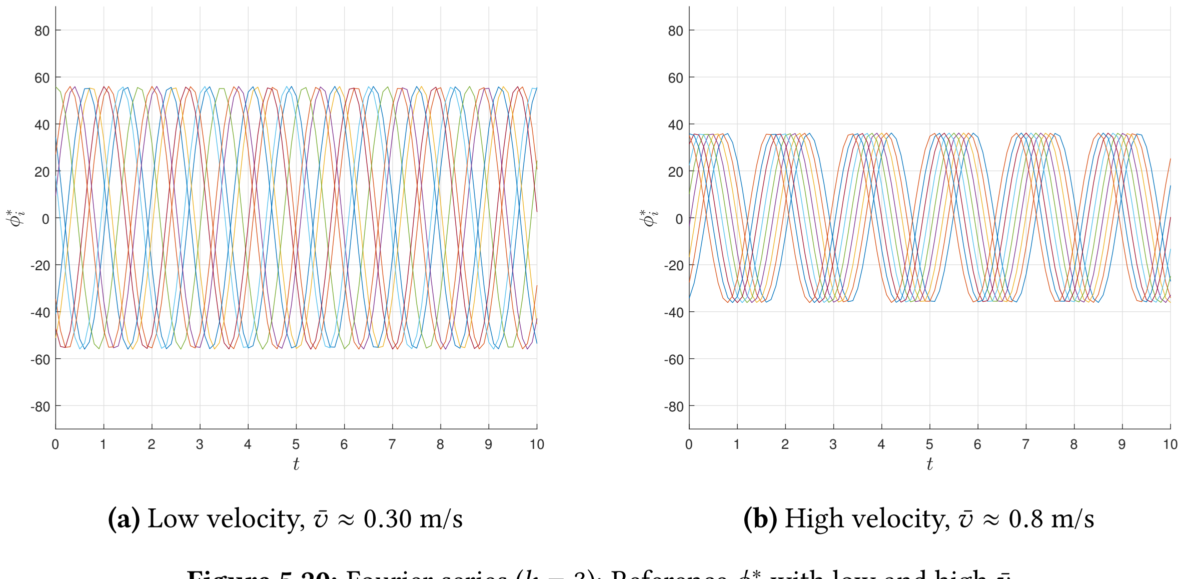


(b) High velocity, $\bar{v} \approx 0.80$ m/s

Figure 5.19: Fourier series ($k = 1$): Reference ϕ_i^* with low and high $\bar{v}$.

PCs. By observing the scores plot in Figure A.13c, no simple structures of the scores can be seen. However, a similarity with the motion patterns in the previous sections can be observed, i.e., $\bar{v}$ increases as δ decreases. Note that in the Figure A.13d, both a_1 and b_1 are located on the right side of the y-axis. Unlike in Figure A.13b, where a_1 and b_1 are located on the opposite side of the y-axis. Observe also that the variable a_1 (furthest on the right side and closest to PC-1) is most affected from moving along the x-axis. Furhtermore, by following the light blue scores from right to left to the brown colors, the value of b_1 might not decrease as much as a_1 does. Because b_1 is located furhter away from PC-1. The shape of the reference ϕ_i^* for this motion pattern is hard to take any assumptions. Therefore, plotting the referencce ϕ_i^* with low and high $\bar{v}$ shown in Figure 5.19, will make it simpler to investigate this motion pattern. The subfigures resemble ϕ_i^* in Figure 5.16. This indicate that the Fourier series with $k = 1$ approximates a reference ϕ_i^* similar to the lateral undulation. This may be expected, since with $k = 1$, the addition between one cosine and sine will generate a new sine function with a different amplitude and phase shift.

In the next simulation result, we will investigate the Fourier series motion with $k = 3$ coefficients. The PCA overview of this motion pattern is shown in Figure A.14. Some preprocessing of the raw data have been done for removing noise and outliers. The selected samples after the preprocessing are shown in Figure A.14a, where the grey samples are insignificant (noise) in the analysis. After seperating the outliers, two PCA models were conttructed: one for right and left cluster. We isolate the two clusters by calling the right cluster for cluster 1, and the left for cluster 2. The PCA scores of cluster 1 in Figure A.15a, show that the average velocity of the samples lie in $0.2 < \bar{v} < 0.8$ m/s. Furthermore, the loadings plot Figure A.15b shows that P_{avg} and $\bar{v}$ are negatively corrolated with δ. In the loadingds plot, the variables a_2 and b_1 are close to the origin, which implies that these variable are insignificant and not interpretable. The two Fourier coefficients that have most influence on the PCA model are the a_3 and b_3 . These two varables are associated with the last cosine and sine terms of the Fourier series, which is given as $\sin\left(\frac{k\pi t}{L} + (i-1)\delta\right) = \sin(\omega^{max} t + (i-1)\delta)$. The PCA model of cluster 2 shown in Figure A.16, only the fourier coefficient b_3 is shown in the loadings plot. This indicate that, for samples with $\bar{v} > 0.6$ m/s, only the variance of b_3 and δ affect the values of P_{avg} and $\bar{v}$. Thus, the solutions in this cluster are fully descibed by the last sine term of the Fourier series. In fact, the solutions in cluster 1 are also most affected by the last sine term of the Fourier series. This can be observed by investigating the raw data of the clusters. A section of the raw data is shown in Table A.6 and A.7. Observe from the raw data that the sign of a_3 are different between cluster 1 and 2. Furthermore, almost all the Fourier coefficients wiht low frequencies are close to zero. This indicate that low frequencies of the cosine and sine terms are insignificant and act as noise in PCA analysis. The reference ϕ_i^* of this motion pattern shown in Figure 5.20, resembles the reference given in Figure 5.19. This motion pattern may also be seen as an approximation of a motion pattern similar to lateral undulation.



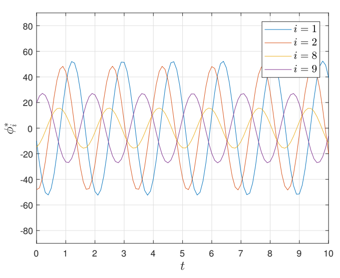
(a) Low velocity, $\bar{v} \approx 0.30$ m/s

(b) High velocity, $\bar{v} \approx 0.8$ m/s

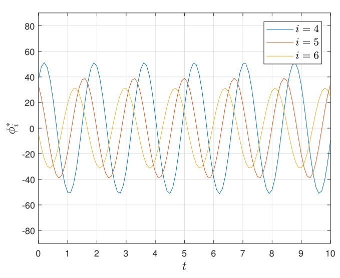
Figure 5.20: Fourier series ($k = 3$): Reference ϕ_i^* with low and high $\bar{v}$.

5.2.4.5 Multi-Fourier series motion

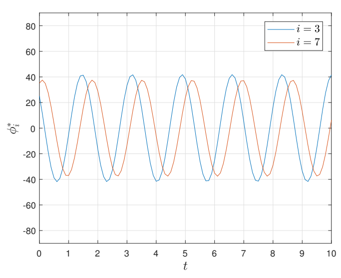
The multivariate analysis of the multi-Fourier series is the most difficult among the motion patterns. It has a total of 19 decision variables. The scores and loadings of the PCA model for this motion pattern is given in Figure A.17. In the scores plot Figure A.17a, three different clusters can be seen. Even splitting up the three clusters into different PCA models did not simplify the interpretation of the data set. Thus, the investigation of this motion pattern will be done by comparing the three different clusters in the same PCA model. By observing the loadings plot and considering the PC-1, the variables b_5, a_7 and a_8 are negatively corrolated with the varables b_1, b_2, b_7, b_8 and b_9. Considering these variables and the leftmost cluster shown in the scores plot. The solutions with $0.2 < \bar{v} < 0.6$ may have ϕ_i^* with low amplitude at the end joints, i.e., the head and tail. Thus the right cluster in the scores plot close to PC-1, may have the opposite effect, meaning high amplitude at the end joints. Furhter investigation of the PCA model, shows that decreasing δ, increases the variable P_{avg} and $\bar{v}$. This is the same as for the previous motion patterns presented in this section. The plot of reference ϕ_i^* given in Figure 5.21, shows that the observations of the PCA model given above were not precisely correct. Figure 5.21a shows that for low $\bar{v}$, the amplitude at the head joints are small, which was assumed from the PCA model. However, the ampliute at the tail joints are large. From the subfigures 5.21a-5.21c, the motion pattern with low $\bar{v}$ looks similar to the eel-like motion, i.e., low and high amplitude at the head and tail joints, respectively. For high velocity $\bar{v}$ the motion pattern resembles the modified motion pattern given in Figure 5.18, with the exception of ϕ_5^*.



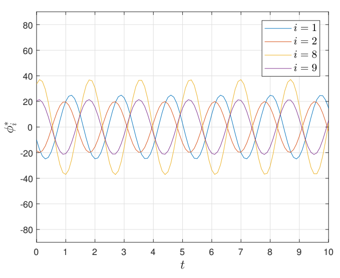
(a) Low velocity, $\bar{v} \approx 0.30$ m/s: ϕ_i^*, $i \in \{1, 2, 8, 9\}$



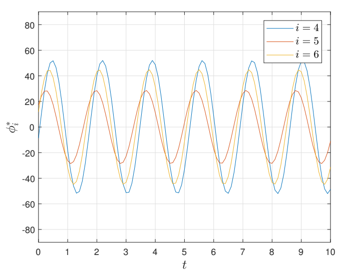
(b) Low velocity, $\bar{v} \approx 0.30$ m/s: ϕ_i^*, $i \in \{4, 5, 6\}$



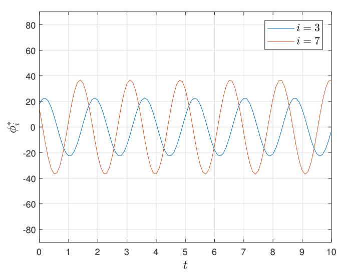
(c) Low velocity, $\bar{v} \approx 0.30$ m/s: ϕ_i^*, $i \in \{3, 7\}$



(d) High velocity, $\bar{v} \approx 0.80$ m/s: ϕ_i^*, $i \in \{1, 2, 8, 9\}$



(e) High velocity, $\bar{v} \approx 0.80$ m/s: ϕ_i^*, $i \in \{4, 5, 6\}$

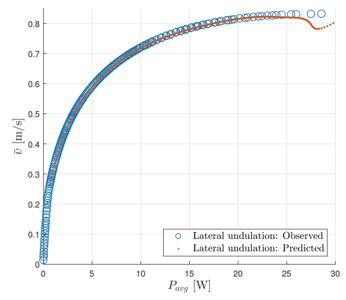


(f) High velocity, $\bar{v} \approx 0.80$ m/s: ϕ_i^*, $i \in \{3, 7\}$

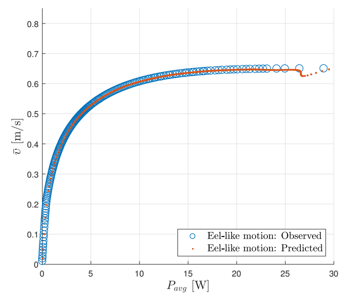
Figure 5.21: Multi-Fourier series: Reference ϕ_i^* with low and high $\bar{v}$.

5.2.5 Regression models for predicting the optimal gait paramters

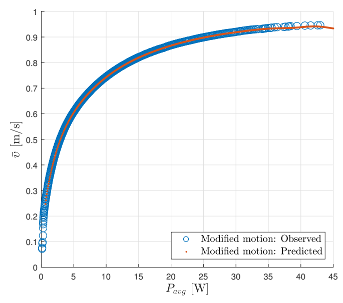
This section presents the regression models of each motion pattern, based on the PLSR models constructed from the PCA models given in the previous sections. The goal with the regression models is to see if it is possible to predict the gait paramters using the objective values. If so, then the models can be used as a utility for selecting optimal gait paramters based on the objective values. The procedure of constructing the prediction models is presented in Section 5.2.4.1, i.e., using the Beta coefficients given in (5.1) and Section A.6 computed by the PLSR models of each motion pattern. Figure 5.22 shows the regression models, where the observed solutions are the Pareto front solutions given in Figure 5.10. The average power used for predicting the gait paramters of all the motion patterns is given as $P_{avg} = \{0, 0.1, 0.2, \ldots, P_{max}\}$. The interpretation



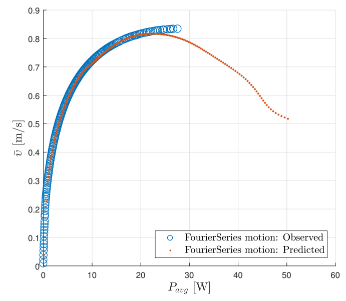
(a) Lateral undulation



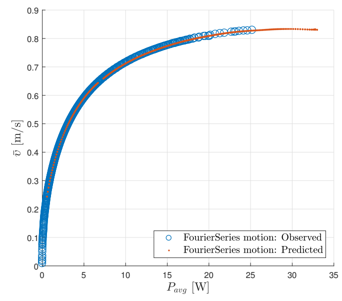
(b) Eel-like motion



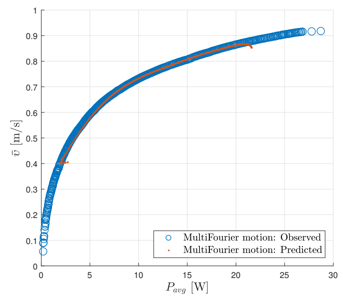
(c) Modified motion



(d) Fourier series ($k = 1$)



(e) Fourier series ($k = 3$)



(f) Multi-Fourier series

Figure 5.22: Prediction models of all the motion patterns.

of the prediction models is listed as follows:

1. Lateral undulation (Figure 5.22a): All the predicted objective values are located on the Pareto front, and the range of P_{avg} for the predicted values corresponds to the observed values. However, the model has problems of predicting values for $P_{avg} > 23$ W.

2. Eel-like motion (Figure 5.22b): All the predicted objective values are located on the Pareto front, with a range of P_{avg} for the predicted solutions matching the ovserved solutions. However, the model has problems of predicting values for $P_{avg} > 25$ W.

3. Modified motion (Figure 5.22c): All the predicted objective values are located on the Pareto front, and the range of P_{avg} for the predicted solutions corresponds to the observed solutions. However, the model has difficulty of predicting solutions for low P_{avg} and $P_{avg} > 42$ W.

4. Fourier series motion with $k = 1$ (Figure 5.22d): The range of P_{avg} for the predicted solutions does not correspond to the observed solution. It has a maximum of $P_{avg} \approx 50$. Furthermore,

the predicted solutions diverge from the Pareto front for $P_{avg} > 22$ W. Thus, the regression model for this motion pattern is poor.

5. Fourier series motion with $k = 3$ (Figure 5.22e): The predicted solutions are alligned with the Pareto front, and the range of P_{avg} for the predicted solutions extends a little bit past the observed soltuions. A gap can be observed on the curve of the predicted solutions. The observed gap is a consequence of the noise in the data set presented in Section 5.2.4.4.

6. Multi-Fourier series motion (Figure 5.22f): The prediction of the gait paramters of the multi-fourier series is not sufficient. The predicted solutions are gathered in the middle of the Pareto front, and diverge at the ends of the prediction curve.

Note that proper data sets will produce good regression models, which is the case for the lateral undulation, eel-like and modified motion. These motion patterns had a noticable structure (convex curve) of the samples in the scores plot. The scores of the Fourier series with $k = 3$ had somewhat good structures for the samples with high $\bar{v}$. From these observations, the regression models for lateral undulation, eel-like and modified motion predict the gait parameters with a sufficient degree. Hence, these regression models can be used as a utility for obtaining the optimal gait parameters with arbitrary P_{avg}. Note that all the prediction models have difficulty of predicting the gait parmaters for low P_{avg}. The cause of this effect, is that low P_{avg} can be seen as the noisy part of the motion pattern.

5.3 Summary

This section summerize the simulation results given in the previous sections. In Section 5.1 we found the optimal GA paramters of the MOPs of each motion pattern, and the collection of the optimal paramters of each motion pattern is given in Table 5.2. The distribution of gait paramter ω and δ in Section 5.2.1 shows that the optimal value of ω was ω^{max}. With only the distribution plot, no possible assumption could be made for δ. In Section 5.2.2, comparison betewen NSGA-II and HypE shows that both methods were sufficient as optimization method for the MOPs. However, HypE spreads out the solutions more uniformly on the Pareto fronts. The Pareto fronts of the actual simulations in Section 5.2.3, show that the solutions are evenly spread out with no gaps, in both NSGA-II and HypE. This section also shows that the Pareto fronts were still sufficient, even after disregarding ω, i.e., setting $\omega = \omega^{max}$. Furthermore, by comparing the Pareto fronts of each motion shows that the most energy efficient motions are the modified and multi-Fourier series. For example, consider $P_{avg} = 25$W, the modified motion has $\bar{v} \approx 0.9$ m/s, while lateral undulation and eel-like motion have $\bar{v} \approx 0.84$ and $\bar{v} \approx 0.65$ m/s, respectively. In Section 5.2.4, theoretical analysis of each motion was done using multivaraite analysis. The simulation study showed that all the motion patterns had a common structure of δ, i.e., decreasing δ would increase P_{avg} and $\bar{v}$, and vice versa. Furthermore, all the outliers observed in the PCA models were solutions with low P_{avg} and $\bar{v}$. This indicate that, the solutions with low $\bar{v}$ are hard to obtain. Further investigation of the simulation study shows that the obtained motion from Fourier series turns out to be an approximation of a motion pattern similar to lateral undulation, where the last sine term had the highest influence on the motion. Furthermore, the Fourier series with $k = 1$ coefficients was sufficient for approximating of the lateral undulation. In Section 5.2.5, regression models of the simulation results were presented. The goal of the regression models was to see if it was possible to predict optimal gait parameters using the objective values. The presented models showed to be feasible for prediction of the lateral undulation, eel-like and modified motion.

Table 5.2: The optimal GA paramters of each motion.

Motion	Population size	Crossover rate	Mutation rate
Lateral undulation	200	0.70	0.05
Eel-like motion	200	0.70	0.05
Modified motion	500	0.80	0.10
Fourier series ($k = 1$)	200	0.80	0.05
Fourier series ($k = 3$)	500	0.70	0.10
Multi-Fourier series	500	0.70	0.15

Chapter 6

Conclusions

In Chapter 4, we presented MOPs of five different motion patterns for an USR, where the first two motions are the common snake locomotions lateral undulation and eel-like motion. The last three are altered motions called modified, Fourier and multi-Fourier series. The objectives of the the presented MOPs are the energy efficiency of the USR presented in Chapter 3, where the multi-objective of the MOPs is given as min $\boldsymbol{f}_{opt} = [P_{avg}, -\bar{v}]$. For the optimization of these MOPs, we presented two multi-objective optimization methods called NSGA-II and HypE. One of the challenges with the MOEAs is finding the optimal GA parameters, such as, the population size, crossover rate and mutation rate. These optimal GA parameters were obtained by running multiple simulations and then comparing the best simulation results. The obtainedned simulation results show that the search space $\mathcal{D}$ of the MOPs was proper and thus the selction of the GA parameters was trivial, except for the Fourier series with $k = 3$ coefficients and multi-Fourier motion pattern. These motion patterns had a more difficult search space and was greatly dependent on the population size. However, larger population size leads to longer simulation runtime. That is, modified motion with $N = 800$ and the computer specification given in Section 4.5. The total runtime of 3600 generations is approximately 10 days.

The presented simulation results in Section 5.2 investigate each motion pattern after obtaining the optimal GA parameters in Section 5.1. The obtained Pareto fronts showed that both NSGA-II and HypE were sufficient as the optimization scheme. However, HypE had more uniformly spread out solutions on the Pareto front, which is desired. Furthermore, the obtained Pareto fronts showed that the modified motion and multi-Fourier motion were the most energy efficient motion patterns regarding the achieved forward velocity and power consumption of the USR.

The purpose of the simulation results is to find similarities between each motion pattern. The obtained results show that $\omega = \omega^{max}$ was the optimal gait parameter for all the motion patterns. However, this caused the modified motion to miss solutions with low $\bar{v}$. It turns out that solutions with low $\bar{v}$ are hard to interpret, which was shown by using PCA presented in Section 5.2.4. Further investigation based on PCA showed that the motion patterns shared the same structure of the gait parameter δ, i.e., decreasing δ caused P_{avg} and $\bar{v}$ to increase. The presented PCA models were based on the solutions computed by HypE, because of its soltuions spreading compared to NSGA-II. This made the investigation of each motion more apparent due to the evenly spread out solutions. The comparison of each PCA showed that the obtained motion pattern from the Fourier series had a similar locomotion as the lateral undulation. While the multi-Fourier series had a similar pattern to eel-like and modified motion for low and high $\bar{v}$, respectively. The presented analysis also showed that the last sine term of the Fourier series had the most influence of the obtained motion pattern, and hence it approximates a sinusoidal function similar to lateral undulation. Furthermore, some of the gait pattern of the modified motion were exposed through the multivariate analysis: (i) similar to lateral undulation for $\bar{v} \approx 0.3$ m/s, and (ii) high and low amplitudes of the joint references for the middle and end joints, respectively, for $\bar{v} \approx 0.8$ m/s.

Finally, we presented prediction models for predicting the optimal gait paramters using the objective values. The models were sufficient for predicting the gait parameters for the lateral undulation, eel-like and modified motion. With the obtained Pareto fronts and the prediction models, the optimal gait parameters could be predicted through a trade-offs between the power consumption P_{avg} and the forward velocity $\bar{v}$.

6.1 Future work

This thesis presented and investigated on two different MOEA schemes for finding efficient gait parameters of five different locomotions of the USR. The proposed optimization problem considers both the forward velocity and the power consumption. It would be interesting to further add more objectives into the optimization problem, such as minimizing the joint acceleration to prevent wear and tear, or minimizing the number of joints while preserving the energy efficiency of the USR. The problem that arises with more objectives than three, is that NSGA-II might not longer be a feasible optimization solver. The presented simulation results given in Chapter 5, showed that through multivariate analysis, all the five motion patterns had a similar structure of the gait paramter δ, i.e., the value of δ decreased as the forward velocity and power consumption increased of the USR. Hence, in the future, the gait paramter δ can be disregarded from the optimization problem, by having constant values of δ in each solution in the population. The only thing to consider is to uniformly decrease δ among the solutions. This will reduce the total number of decision variables of each motion pattern by one. For the motion pattern based on Fourier series presented in Chapter 4, the simulation results showed that the sine terms had the most influence on the locomotion of the USR, and therefore, one might disregard the cosine terms for reducing the number of decision variables.

One possible future extension of this thesis, is to investigate on different locomotion generator of the USR. Some related work on different locomotion generator were presented in Chapter 1, such as, CPG and ICONE. By obtaining domain knowledge of the USR, one can try restricting the search spaces using ICONE. Further extension is to investigate other regression analysis for constructing the prediction models presented in Section 5.2.5.

Appendix A

A.1 The recursion steps in Example 2.5.3

The computation of each recursion step in Example 2.5.3 listed below.

- **Step 1.1.1**: Arguments,

$$i = 2,\ V = 1,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 0)\},\ z = [\infty, \infty] \tag{A.1}$$

  The set UP, UR and U,

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3],\ \boldsymbol{f}(\boldsymbol{x}_2) = [3, 1]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\}, \quad U = UP \cup UR \tag{A.2}$$

  The compuation of U' and V':

$$u^* = \min_{\boldsymbol{u} \in U} u_i = 1 \Rightarrow z_i = 1 \tag{A.3}$$
$$U' = \{\boldsymbol{u} \in U | u_i > u^*\} = \{[1, 3], [4, 5]\} \tag{A.4}$$
$$V' = V \cdot (\min_{\boldsymbol{u}' \in U'} u'_i - u^*) = 1 \cdot (3 - 1) = 2 \tag{A.5}$$

  Next recursion step, $\mathcal{F}' = doSlicing(\mathcal{F}', R, k, i - 1, V', \boldsymbol{z} = [\infty, 1])$
- **Step 1.1.2**: Arguments,

$$i = 1,\ V = 2,\ \mathcal{F}' = \{(\boldsymbol{x}_1, v = 0), (\boldsymbol{x}_2, 0)\},\ z = [\infty, 1] \tag{A.6}$$

  The set UP, UR and U:

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_2) = [3, 1]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\}, \quad U = UP \cup UR \tag{A.7}$$

  The compuation of U' and V',

$$u^* = \min_{\boldsymbol{u} \in U} u_i = 3 \Rightarrow z_i = 3 \tag{A.8}$$
$$U' = \{\boldsymbol{u} \in U | u_i > u^*\} = \{[4, 5]\} \tag{A.9}$$
$$V' = V \cdot (\min_{\boldsymbol{u}' \in U'} u'_i - u^*) = 2 \cdot (4 - 3) = 2 \tag{A.10}$$

  Next recursion step, $\mathcal{F}' = doSlicing(\mathcal{F}', R, k, i = 1, V', \boldsymbol{z} = [3, 1])$

- **Step 1.1.3**: Arguments,

$$i = 0,\ V = 2,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 0)\},\ z = [3, 1] \tag{A.11}$$

  The set UP, UR:

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_2) = [3, 1]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\} \tag{A.12}$$

  Update hypervolumes,

$$\alpha = \prod_{j=1}^{|UP|-1} \frac{k-j}{|\mathcal{F}'| - j} = \frac{2-1}{2-1} = 1 \tag{A.13}$$

$$v_1 = v_1, \quad v_2 = v_2 + \frac{\alpha}{|UP|} \cdot V = 0 + 2 = 2 \tag{A.14}$$

  Return, $\mathcal{F}' = (\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 2)$
- **Step 2.1.1**: Arguments,

$$i = 2,\ V = 1,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 2)\},\ z = [\infty, \infty] \tag{A.15}$$

  The set U,

$$U = \{\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3], r = [4, 5]\} \tag{A.16}$$

  The compuation of U' and V':

$$u^* = \min_{\boldsymbol{u} \in U} u_i = 3 \Rightarrow z_i = 3 \tag{A.17}$$

$$U' = \{\boldsymbol{u} \in U | u_i > u^*\} = \{[4, 5]\} \tag{A.18}$$

$$V' = V \cdot (\min_{\boldsymbol{u}' \in U'} u'_i - u^*) = 1 \cdot (5 - 3) = 2 \tag{A.19}$$

  Next recursion step, $\mathcal{F}' = doSlicing(\mathcal{F}', R, k, i-1, V', \boldsymbol{z} = [\infty, 3])$
- **Step 2.1.2**: Arguments,

$$i = 1,\ V = 2,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 2)\},\ z = [\infty, 3] \tag{A.20}$$

  The set UP, UR and U,

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3],\ \boldsymbol{f}(\boldsymbol{x}_2) = [3, 1]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\}, \quad U = UP \cup UR \tag{A.21}$$

  The compuation of U' and V':

$$u^* = \min_{\boldsymbol{u} \in U} u_i = 1 \Rightarrow z_i = 1 \tag{A.22}$$

$$U' = \{\boldsymbol{u} \in U | u_i > u^*\} = \{[1, 3], [4, 5]\} \tag{A.23}$$

$$V' = V \cdot (\min_{\boldsymbol{u}' \in U'} u'_i - u^*) = 2 \cdot (3 - 1) = 4 \tag{A.24}$$

  Next recursion step, $\mathcal{F}' = doSlicing(\mathcal{F}', R, k, i-1, V', \boldsymbol{z} = [1, 3])$

- **Step 2.1.3**: Arguments,

$$i = 0,\ V = 4,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 0), (\boldsymbol{x}_2, 2)\},\ z = [1, 3] \tag{A.25}$$

The set UP, UR:

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\} \tag{A.26}$$

Update hypervolumes,

$$\alpha = \prod_{j=1}^{|UP|-1} \frac{k - j}{|\mathcal{F}'| - j} = \frac{2 - 1}{2 - 1} = 1 \tag{A.27}$$

$$v_1 = v_1 + \frac{\alpha}{|UP|} \cdot V = 0 + 4 = 4, \quad v_2 = v_2 \tag{A.28}$$

Return, $\mathcal{F}' = (\boldsymbol{x}_1, 4), (\boldsymbol{x}_2, 2)$

- **Step 2.2.2**: Arguments,

$$i = 1,\ V = 2,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 4), (\boldsymbol{x}_2, 2)\},\ z = [\infty, 3] \tag{A.29}$$

The set U,

$$U = \{\boldsymbol{f}(\boldsymbol{x}_1) = [3, 1], \boldsymbol{r} = [4, 5]\} \tag{A.30}$$

The compuation of U' and V':

$$u^* = \min_{\boldsymbol{u} \in U} u_i = 3 \Rightarrow z_i = 3 \tag{A.31}$$

$$U' = \{\boldsymbol{u} \in U | u_i > u^*\} = \{[4, 5]\} \tag{A.32}$$

$$V' = V \cdot (\min_{\boldsymbol{u}' \in U'} u'_i - u^*) = 2 \cdot (4 - 3) = 2 \tag{A.33}$$

Next recursion step, $\mathcal{F}' = doSlicing(\mathcal{F}', R, k, i - 1, V', \boldsymbol{z} = [3, 3])$

- **Step 2.2.3**: Arguments,

$$i = 0,\ V = 2,\ \mathcal{F}' = \{(\boldsymbol{x}_1, 4), (\boldsymbol{x}_2, 2)\},\ z = [3, 3] \tag{A.34}$$

The set UP, UR:

$$UP = \{\boldsymbol{f}(\boldsymbol{x}_1) = [1, 3], \boldsymbol{f}(\boldsymbol{x}_2) = [3, 1]\}, \quad UR = \{\boldsymbol{r} = [4, 5]\} \tag{A.35}$$

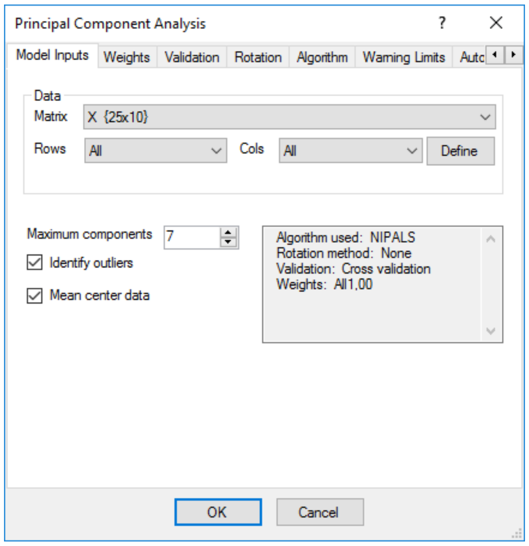
Update hypervolumes,

$$\alpha = \prod_{j=1}^{|UP|-1} \frac{k - j}{|\mathcal{F}'| - j} = \frac{2 - 1}{2 - 1} = 1 \tag{A.36}$$

$$v_1 = v_1 + \frac{\alpha}{|UP|} \cdot V = 4 + \frac{1}{2} \cdot 2 = 5, \quad v_2 = v_1 + \frac{\alpha}{|UP|} \cdot V = 2 + \frac{1}{2} \cdot 2 = 3 \tag{A.37}$$

Return, $\mathcal{F}' = (\boldsymbol{x}_1, 5), (\boldsymbol{x}_2, 3)$

A.2 Configurations in Unscrambler X 10.3


Principal Component Analysis

Model Inputs | Weights | Validation | Rotation | Algorithm | Warning Limits | Autc

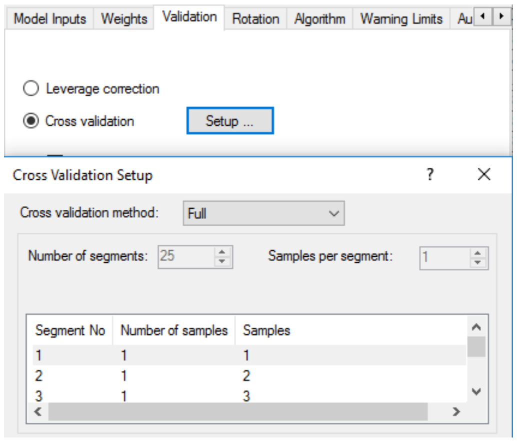
Data
Matrix: X {25x10}
Rows: All
Cols: All
Define

Maximum components: 7
☑ Identify outliers
☑ Mean center data

Algorithm used: NIPALS
Rotation method: None
Validation: Cross validation
Weights: All1,00

OK | Cancel

Figure A.1: PCA – Configurations


Model Inputs | Weights | Validation | Rotation | Algorithm | Warning Limits | Au

○ Leverage correction
● Cross validation
Setup ...

Cross Validation Setup

Cross validation method: Full

Number of segments: 25
Samples per segment: 1

Segment No	Number of samples	Samples
1	1	1
2	1	2
3	1	3

Figure A.2: PCA – Full cross-validation

A.3 Python Codes

Listing A.1: The main interface of NSGA-II and HypE

```python
for gen in range(n_gen, cg.n_generations+1):
    t1 = time.perf_counter()
    print("
        ==================================================================")
    print('Generation: %d' % gen)
    print('Evaluate population')

    # Mating Selection & Breeding
    offspring = moea.mating_selection(population, chrom_repr, toolbox, eng,
        loggerObj.cx, loggerObj.mx)

    # Evaluate Offsprings
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    sim_time = eval_objective_values(invalid_ind, eng, motion, with_omega)

    # Environmental Selection
    population = moea.environmental_selection(population + offspring, eng)

    # If Violations, Add Bad Fitness
    assign_violation_ranks(population)

    # Print Execution Time To Screen
    elapsed_time, elapsed_computational_time, estimated_time =
        compute_elapsed_time(t1, sim_time, gen, elapsed_time,
        elapsed_computational_time, estimated_time)

    # Log Data To History
    obj = [population[i].fitness.values for i in range(len(population))]
    logData.save_history(loggerObj, population, obj)

    # Plot multi-objective values
    plot_objectives(obj, eng, gen, method_name, motion)

    # Store Result Data
    if redundancy_check:
        store_data_with_redundancy_check(gen, loggerObj, cwd, result_dir)
    else:
        store_data(gen, loggerObj, cwd, result_dir)
print("====================== End Of Optimization Run ======================")
```

Listing A.2: Assign crowding distance and create a population with lowest rank and largest crowding distance

```python
  def create_parent_population(fronts):
      """ Return P with lowest rank and largest crowding distance """
      pop_size  = cg.pop_size
      P         = []
      container = []
      n_fronts  = len(fronts)

```

```python
    # Move non-violated and violated fronts into different containers
    for i in range(n_fronts):
        temp_container  = [[],[]]
        for j in range(len(fronts[i])):
            # Place feasible solutions in container 0
            if fronts[i][j].violations[0] == np.double(0):
                temp_container[0]    += [fronts[i][j]]
            # Place infeasible solutions in container 1
            else:
                temp_container[1]    += [fronts[i][j]]
        container += [temp_container]

    # Assign Crowding Distance. Consider Feasible Solutions First!
    for i in range(2):
        k = np.uint32(0)
        while k < n_fronts and len(container[k][i]) != 0 and len(P) <
            pop_size:
            crowd_front = crowding_distance(container[k][i])
            dist = [crowd_front[q].crowd_dist for q in range(len(crowd_front
                ))]
            j = np.argsort(dist)
            k += 1
            for r in j[::-1]:
                crowd_front[r].rank = k
                P += [crowd_front[r]]
                if len(P) >= pop_size:
                    break
    return P
```

Listing A.3: Assign Hypervolume indicator and create a population with highest values

```python
def create_parent_population(pop, eng):
    """ Retur P with higest HV indicator """
    P   = []
    N   = cg.pop_size
    r   = cg.reference

    # Non-domination Sorting, pop = parents + offsprings
    non_dominated_fronts = non_dominated_sort(pop)

    # Select the first non-dominted fronts
    for P1 in non_dominated_fronts:
        if len(P) + len(P1) > N:
            break
        P += P1

    # The number of overflow of solutions
    k = len(P) + len(P1) - N

    # Remove Solutions With Lowest Hypervolume (HV) Indicator
    while k > 0:
        # Calulcate HV indicator
        objectives = [P1[i].fitness.values for i in range(len(P1))]
        f = eng.hypeIndicatorExact(np.array(objectives).T.tolist(), r, k)

        # Remove the solution with lowest HV indicator
        i = np.argmin(f)
        del P1[i]
        k -= 1

    # Return parent population with size N
    return P + P1
```

A.4 Pareto fronts of the cases given in Section 5.1

A.4.1 Fourier Series with $k = 3$ coefficients

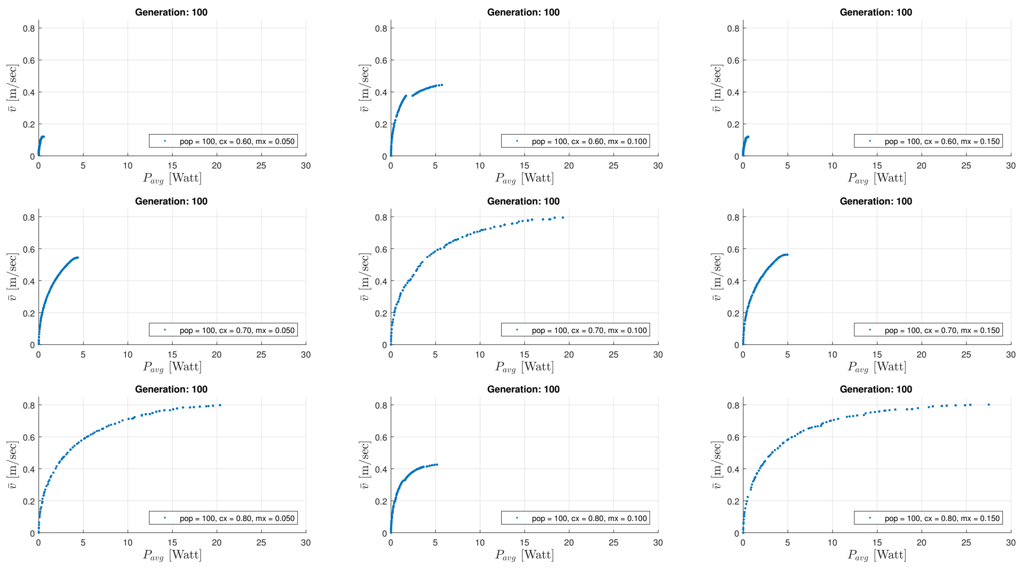


Figure A.3: Fourier series ($k = 3$): Pareto fronts for the case with $N = 100$ solutions

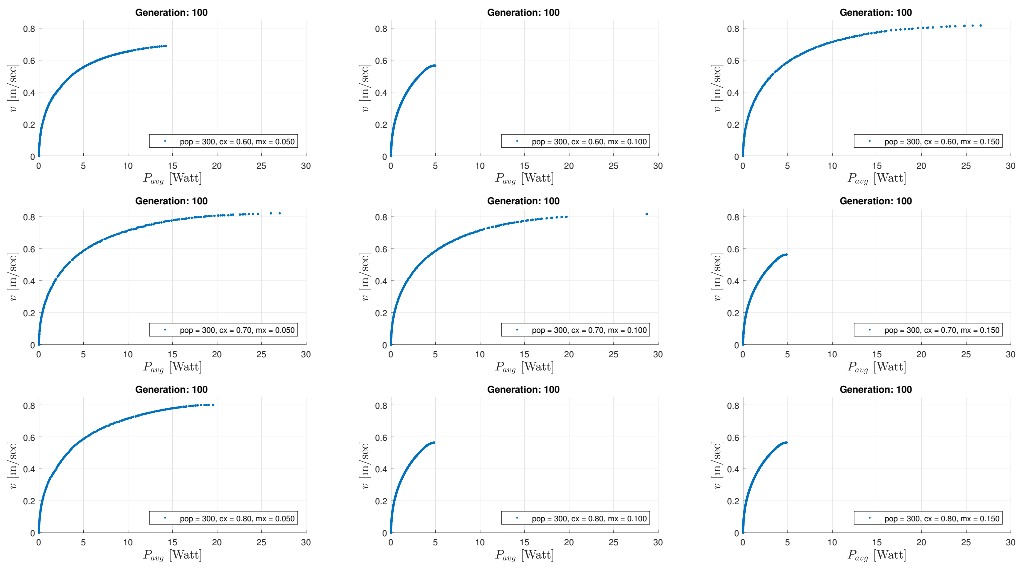


Figure A.4: Fourier series ($k = 3$): Pareto fronts for the case with $N = 300$ solutions

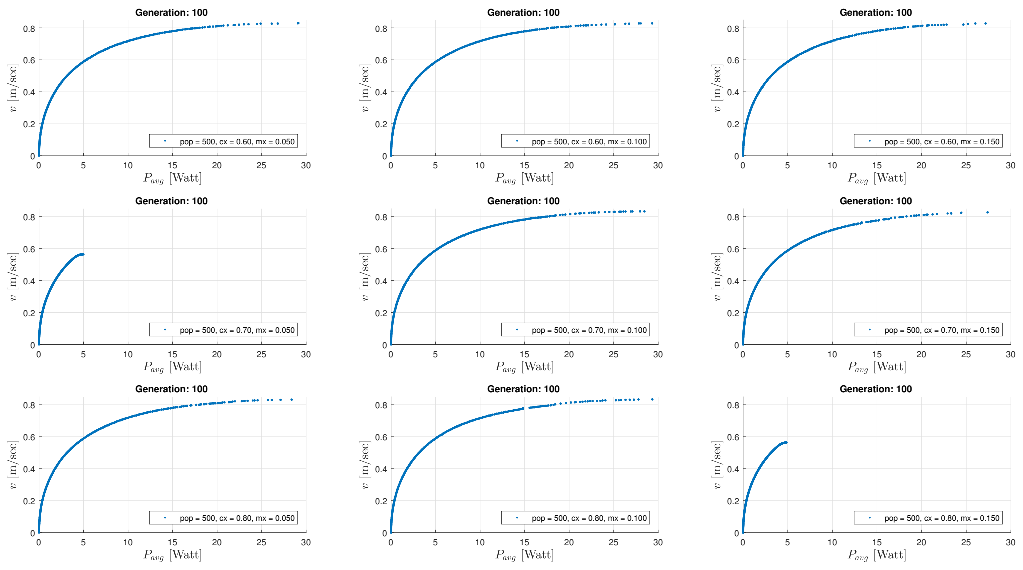
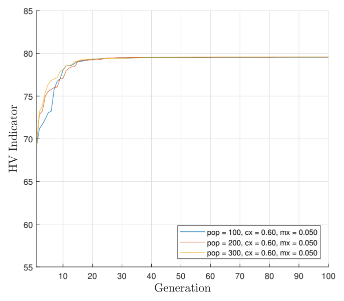


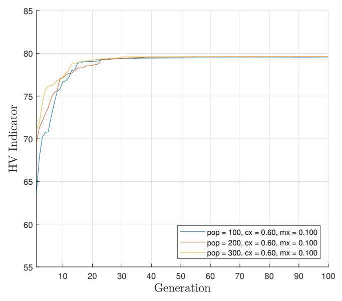
Figure A.5: Fourier series ($k = 3$): Pareto fronts for the case with $N = 500$ solutions

A.5 Hypervolumes of the cases given in Section 5.1

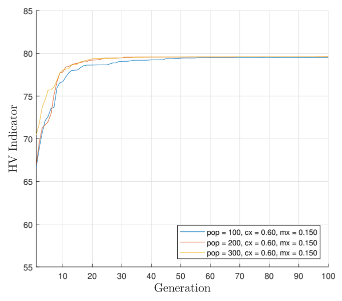
A.5.1 Lateral Undulation



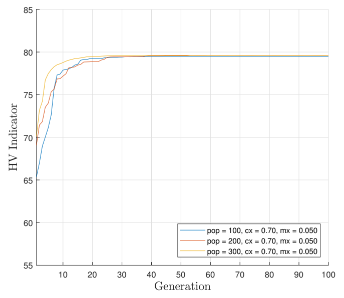
(a) $cx = 0.60$, $mx = 0.050$, $N = \{100, 200, 300\}$



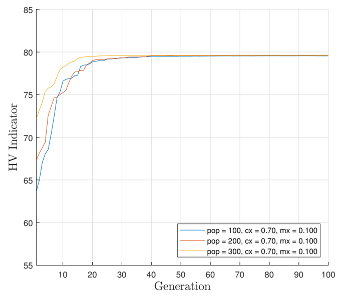
(b) $cx = 0.60$, $mx = 0.100$, $N = \{100, 200, 300\}$



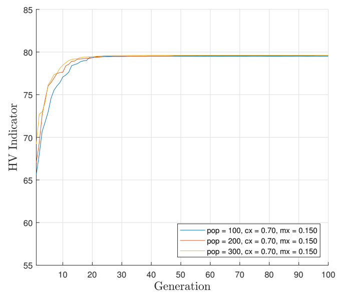
(c) $cx = 0.60$, $mx = 0.150$, $N = \{100, 200, 300\}$



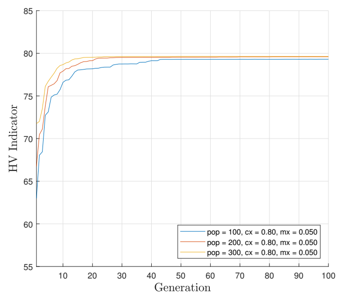
(d) $cx = 0.70$, $mx = 0.050$, $N = \{100, 200, 300\}$



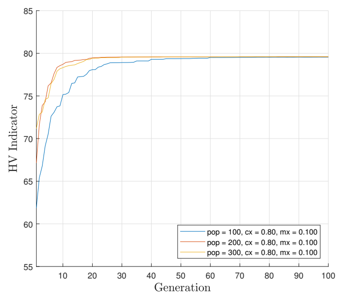
(e) $cx = 0.70$, $mx = 0.100$, $N = \{100, 200, 300\}$



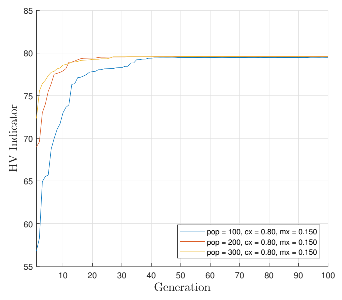
(f) $cx = 0.70$, $mx = 0.150$, $N = \{100, 200, 300\}$



(g) $cx = 0.80$, $mx = 0.050$, $N = \{100, 200, 300\}$



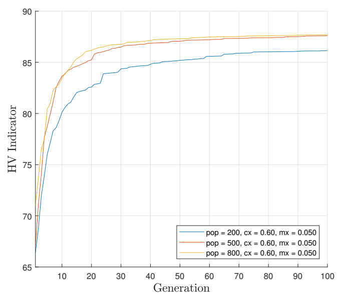
(h) $cx = 0.80$, $mx = 0.100$, $N = \{100, 200, 300\}$



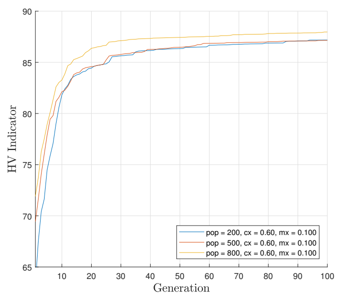
(i) cx = 0.80, mx = 0.150, $N = \{100, 200, 300\}$

Figure A.6: Lateral undulation: The average hypervolume indicator, where *cx* and *mx* is the crossover and mutation rate, respectively.

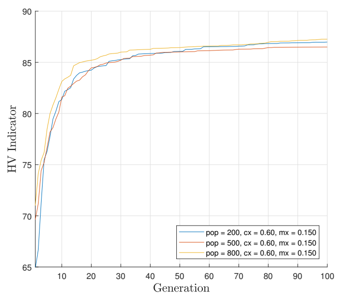
A.5.2 Modified Motion



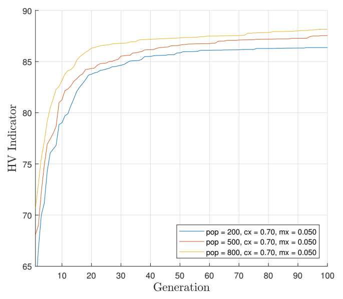
(a) $cx = 0.60$, $mx = 0.050$, $N = \{200, 500, 800\}$



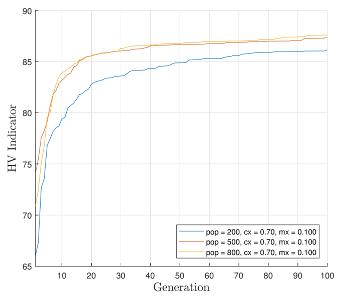
(b) $cx = 0.60$, $mx = 0.100$, $N = \{200, 500, 800\}$



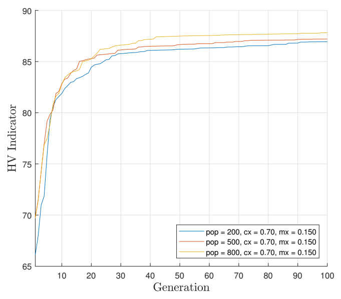
(c) $cx = 0.60$, $mx = 0.150$, $N = \{200, 500, 800\}$



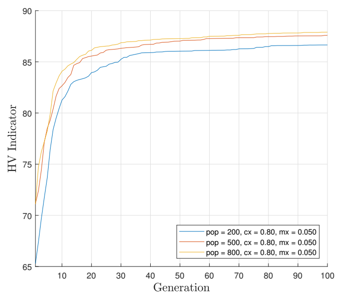
(d) $cx = 0.70$, $mx = 0.050$, $N = \{200, 500, 800\}$



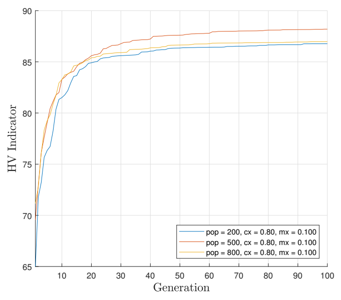
(e) $cx = 0.70$, $mx = 0.100$, $N = \{200, 500, 800\}$



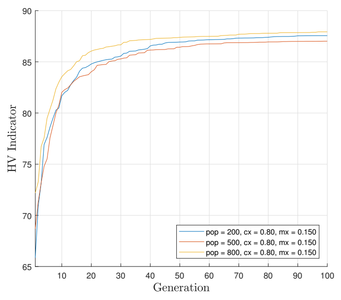
(f) $cx = 0.70$, $mx = 0.150$, $N = \{200, 500, 800\}$



(g) $cx = 0.80$, $mx = 0.050$, $N = \{200, 500, 800\}$



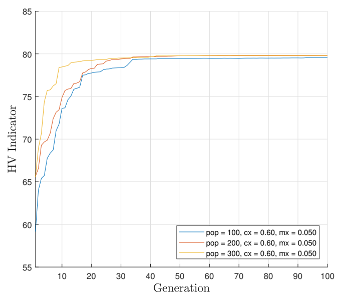
(h) $cx = 0.80$, $mx = 0.100$, $N = \{200, 500, 800\}$



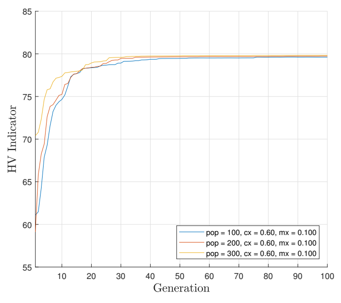
(i) cx = 0.80, mx = 0.150, $N = \{200, 500, 800\}$

Figure A.7: Modified: The average hypervolume indicator, where *cx* and *mx* is the crossover and mutation rate, respectively.

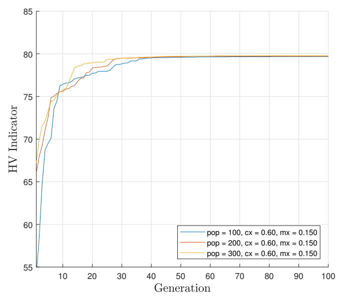
A.5.3 Fourier Series Motion with $k = 1$ Fourier coefficients



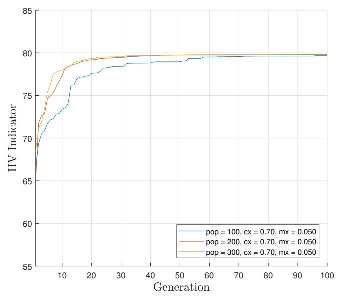
(a) $cx = 0.60$, $mx = 0.050$, $N = \{100, 200, 300\}$



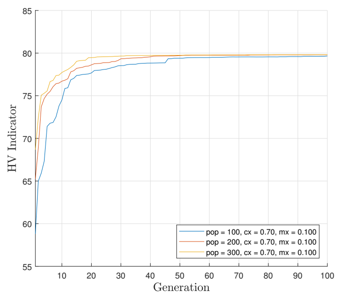
(b) $cx = 0.60$, $mx = 0.100$, $N = \{100, 200, 300\}$



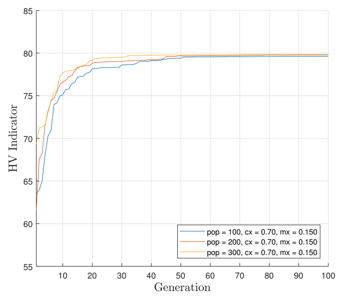
(c) $cx = 0.60$, $mx = 0.150$, $N = \{100, 200, 300\}$



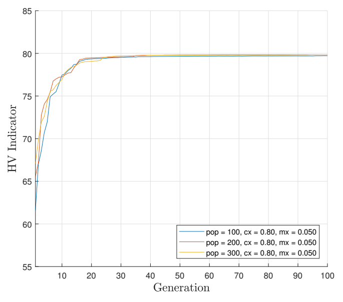
(d) $cx = 0.70$, $mx = 0.050$, $N = \{100, 200, 300\}$



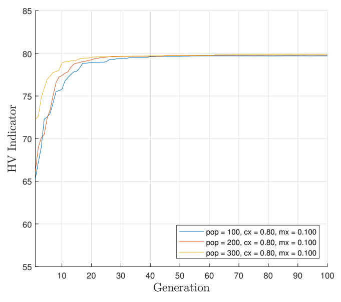
(e) $cx = 0.70$, $mx = 0.100$, $N = \{100, 200, 300\}$



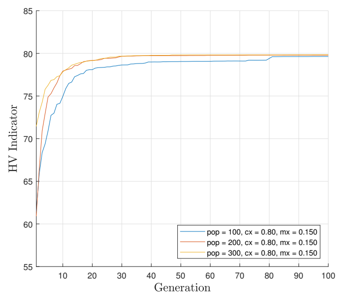
(f) $cx = 0.70$, $mx = 0.150$, $N = \{100, 200, 300\}$



(g) $cx = 0.80$, $mx = 0.050$, $N = \{100, 200, 300\}$



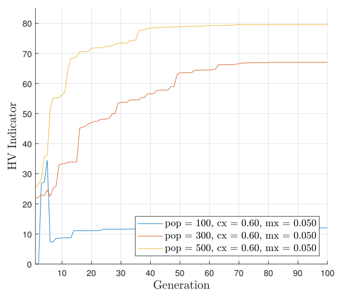
(h) $cx = 0.80$, $mx = 0.100$, $N = \{100, 200, 300\}$



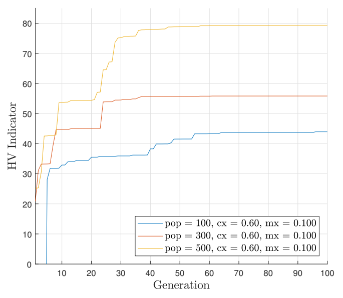
(i) cx = 0.80, mx = 0.150, $N = \{100, 200, 300\}$

Figure A.8: Fourier series ($k = 1$): The average hypervolume indicator, where cx and mx is the crossover and mutation rate, respectively.

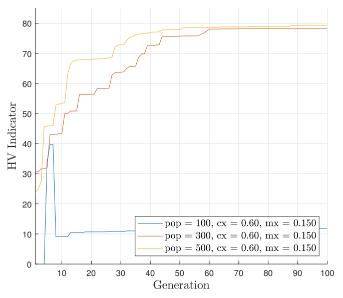
A.5.4 Fourier Series Motion with $k = 3$ Fourier coefficients



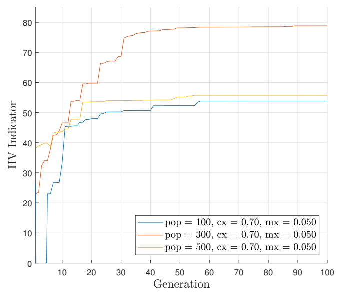
(a) $cx = 0.60$, $mx = 0.050$, $N = \{100, 300, 500\}$



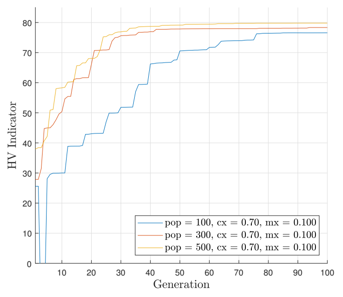
(b) $cx = 0.60$, $mx = 0.100$, $N = \{100, 300, 500\}$



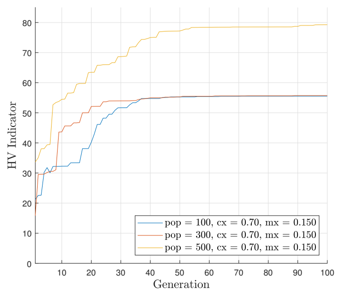
(c) $cx = 0.60$, $mx = 0.150$, $N = \{100, 300, 500\}$



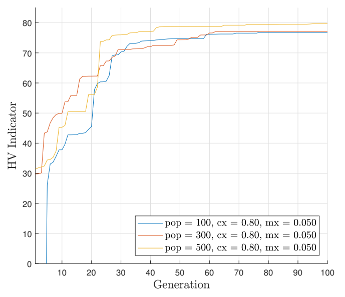
(d) $cx = 0.70$, $mx = 0.050$, $N = \{100, 300, 500\}$



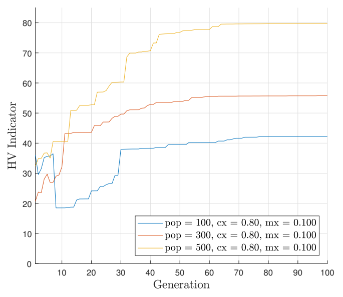
(e) $cx = 0.70$, $mx = 0.100$, $N = \{100, 300, 500\}$



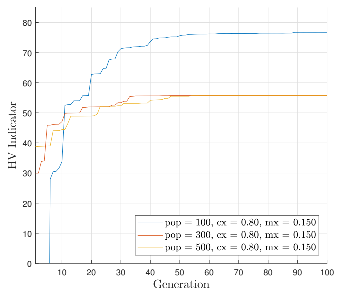
(f) $cx = 0.70$, $mx = 0.150$, $N = \{100, 300, 500\}$



(g) $cx = 0.80$, $mx = 0.050$, $N = \{100, 300, 500\}$



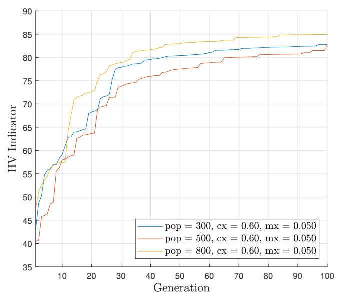
(h) $cx = 0.80$, $mx = 0.100$, $N = \{100, 300, 500\}$



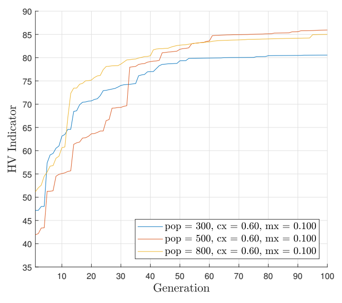
(i) $\mathrm{cx} = 0.80$, $\mathrm{mx} = 0.150$, $N = \{100, 300, 500\}$

Figure A.9: Fourier series ($k = 3$): The average hypervolume indicator, where *cx* and *mx* is the crossover and mutation rate, respectively.

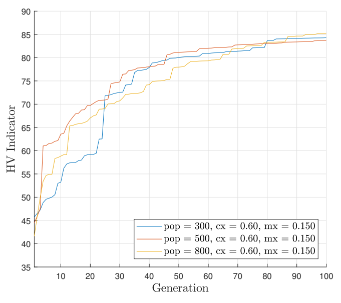
A.5.5 Multi-Fourier Series Motion



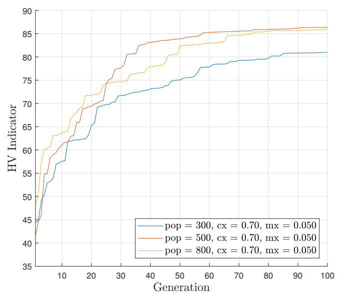
(a) $cx = 0.60$, $mx = 0.050$, $N = \{300, 500, 800\}$



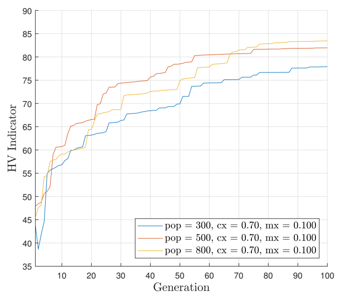
(b) $cx = 0.60$, $mx = 0.100$, $N = \{300, 500, 800\}$



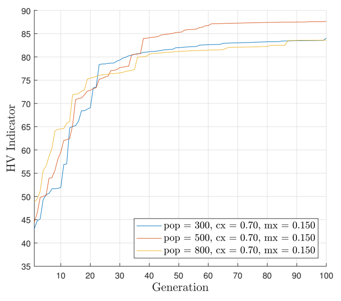
(c) $cx = 0.60$, $mx = 0.150$, $N = \{300, 500, 800\}$



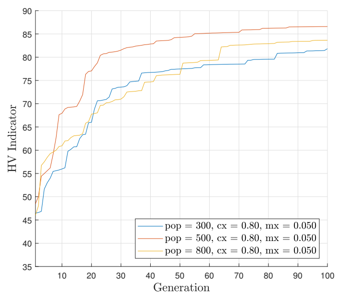
(d) $cx = 0.70$, $mx = 0.050$, $N = \{300, 500, 800\}$



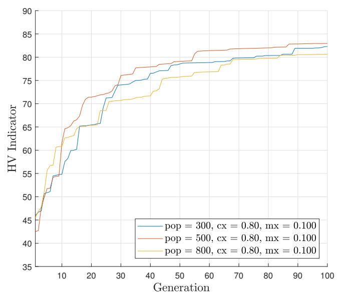
(e) $cx = 0.70$, $mx = 0.100$, $N = \{300, 500, 800\}$



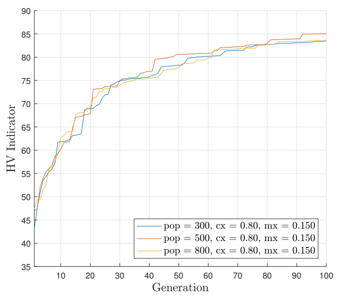
(f) $cx = 0.70$, $mx = 0.150$, $N = \{300, 500, 800\}$



(g) $cx = 0.80$, $mx = 0.050$, $N = \{300, 500, 800\}$



(h) $cx = 0.80$, $mx = 0.100$, $N = \{300, 500, 800\}$



(i) $\mathrm{cx} = 0.80$, $\mathrm{mx} = 0.150$, $N = \{300, 500, 800\}$

Figure A.10: Fourier series: The average hypervolume indicator, where cx and mx is the crossover and mutation rate, respectively.

A.6 PCA and PLSR Results

A.6.1 Lateral undulation

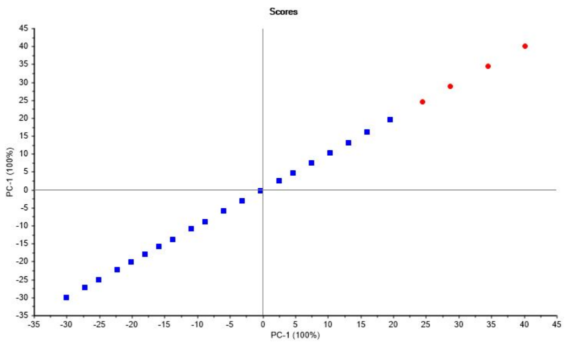
Table A.1: The raw data of cluster 1 given in Figure 5.14

Sample	α [Nm]	δ [Nm]	P_{avg} [W]	$\bar{v}$ [m/s]
1	5.6693	90.0000	0.0215	0.0131
2	7.7953	90.0000	0.0396	0.0237
3	9.2126	90.0000	0.0540	0.0320
4	10.6299	90.0000	0.0698	0.0410
5	12.0472	89.2913	0.0899	0.0516
6	13.4646	89.2913	0.1084	0.0616
7	14.8819	90.0000	0.1235	0.0703
8	16.2992	90.0000	0.1425	0.0802
9	17.7165	90.0000	0.1616	0.0901
10	19.1339	90.0000	0.1809	0.0997
11	20.5512	90.0000	0.1998	0.1091
12	21.9685	90.0000	0.2186	0.1182
13	23.3858	90.0000	0.2369	0.1269
14	24.8031	90.0000	0.2548	0.1351
15	26.2205	90.0000	0.2720	0.1431
16	28.3465	90.0000	0.2971	0.1541
17	30.4724	90.0000	0.3197	0.1644
18	33.3071	90.0000	0.3486	0.1765
19	35.4331	90.0000	0.3686	0.1846
20	38.9764	90.0000	0.3986	0.1963
21	41.1024	90.0000	0.4148	0.2023
22	44.6457	90.0000	0.4383	0.2110
23	48.8976	90.0000	0.4624	0.2192
24	54.5669	90.0000	0.4892	0.2268
25	60.2362	89.2913	0.5254	0.2345

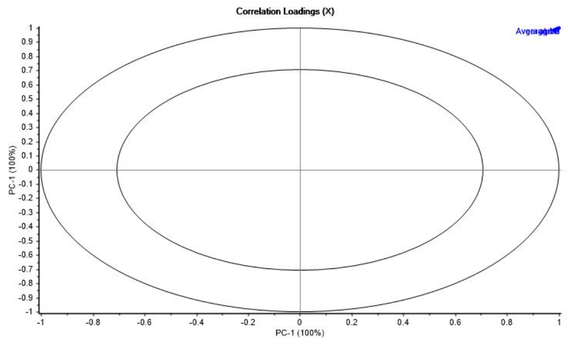
Table A.2: The predicted gait paramters computed by the prediction model presented in Section 5.2.4.

P_{avg} [W]	α [Nm]	δ [Nm]
0.000	9.634	89.565
0.200	18.414	90.053
0.400	41.858	89.887
0.600	52.984	79.087
0.800	50.554	74.766
1.000	48.752	71.482
2.000	42.815	60.109
4.000	36.161	45.951
6.000	33.101	37.707
8.000	31.605	32.037
10.000	31.149	28.079
12.000	31.251	25.037
14.000	32.028	23.108
16.000	32.985	21.473
18.000	34.378	20.557
20.000	36.001	20.020
22.000	37.721	19.642
24.000	39.774	19.812
26.000	41.935	20.159
28.000	42.616	18.070

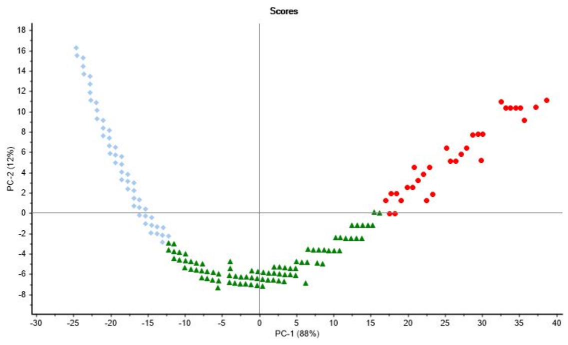
A.6.2 Eel-like motion



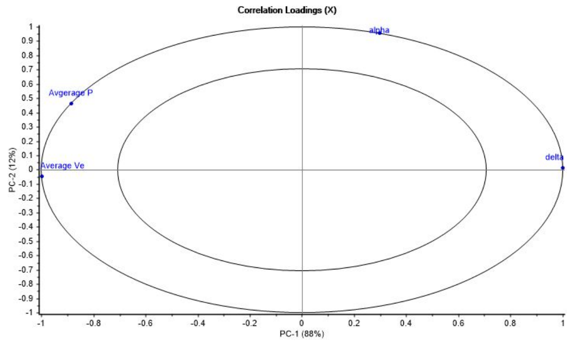
(a) PCA scores of cluster 1 for eel-like motion.



(b) PCA loadings of cluster 1 for eel-like motion.



(c) PCA scores of cluster 2 for eel-like motion.



(d) PCA loadings of cluster 2 for eel-like motion.

Figure A.11: Eel-like motion: PCA scores and loadings.

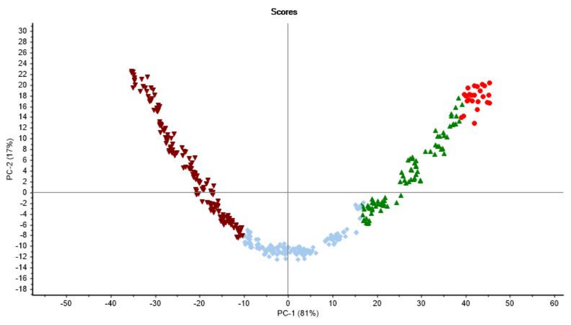
Table A.3: Eel-like: The RMSE values of the PLSR model for cluster 1 and 2

	Cluster 1	Cluster 2	
Factor	α	α	δ
0	20.660	5.976	15.800
1	4.163	5.954	7.329
2	1.616	1.229	0.541

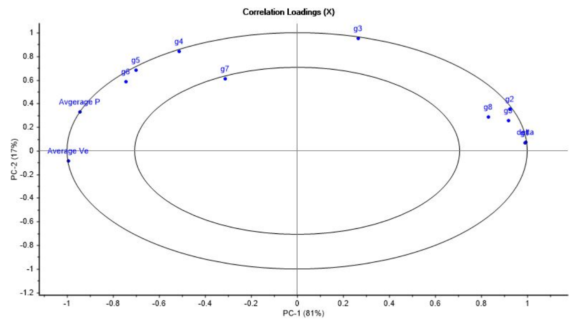
Beta coeffictions of the PLSR model:

$$\boldsymbol{B}_1 = \begin{bmatrix} 9.336 & 90.000 \\ 273.122 & 0.000 \\ -31.234 & 0.000 \end{bmatrix} \qquad \boldsymbol{B}_2 = \begin{bmatrix} 98.960 & 115.778 \\ 1.759 & -0.184 \\ -9.947 & -13.702 \end{bmatrix}. \tag{A.38}$$

A.6.3 Modified motion



(a) PCA scores.



(b) PCA loadings.

Figure A.12: Modified motion: PCA scores and loadings.

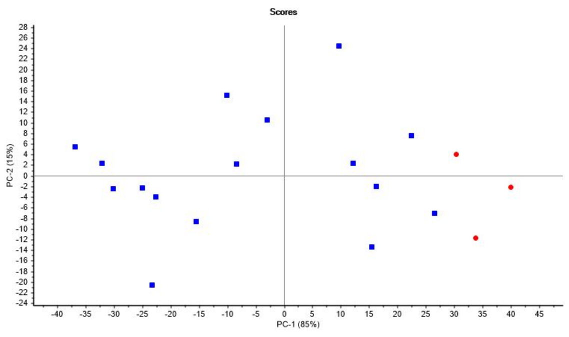
Table A.4: Modified: The RMSE of the PLSR model

Factor	δ	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9
0	13.815	9.917	8.871	5.897	6.350	5.769	3.843	1.600	2.417	4.752
1	5.440	3.954	5.738	5.911	4.211	2.790	1.751	1.400	1.739	2.941
2	0.699	1.473	1.436	1.322	1.290	1.550	1.259	1.164	1.175	1.449

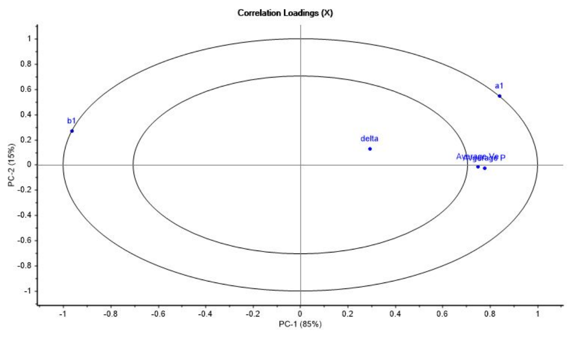
Beta coeffictions of the PLSR model:

$$\boldsymbol{B} = \begin{bmatrix} 91.885 & 69.957 & 75.560 & 69.679 & 58.325 & 50.229 & 46.110 & 42.624 & 43.145 & 46.915 \\ -0.059 & -0.091 & 0.659 & 1.517 & 1.576 & 1.175 & 0.705 & 0.291 & 0.140 & 0.233 \\ -8.128 & -5.544 & -8.228 & -8.361 & -5.715 & -3.261 & -1.697 & -1.118 & -1.908 & -3.806 \end{bmatrix}. \tag{A.39}$$

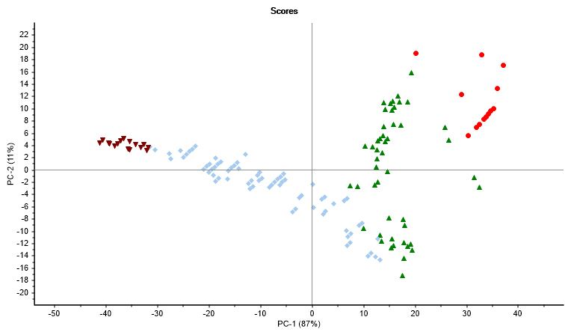
A.6.4 Fourier Series Motion With $k = 1$ Fourier coefficients



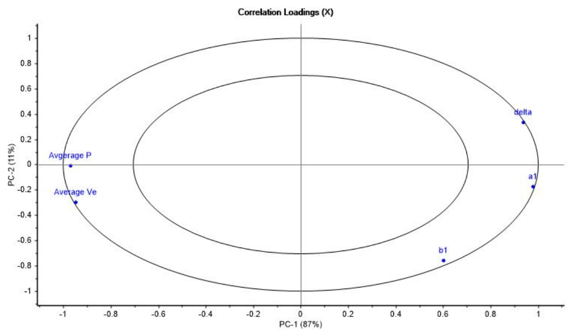
(a) PCA scores of cluster 1 for Fourier series.



(b) PCA loadings of cluster 1 for Fourier series.



(c) PCA scores of cluster 2 for Fourier series.



(d) PCA loadings of cluster 2 for Fourier series.

Figure A.13: Fourier series motion: PCA scores and loadings.

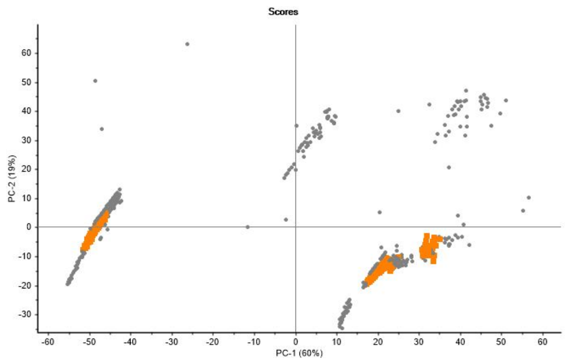
Table A.5: Eel-like: The RMSE values of the PLSR model for cluster 1 and 2

	Cluster 1			Cluster 2		
Factor	δ	a_1	b_1	δ	a_1	b_1
0	13.897	15.051	7.031	0.273	16.558	22.075
1	5.789	4.554	5.924	0.232	13.411	15.864
2	0.194	13.417	13.159	0.906	4.539	5.417

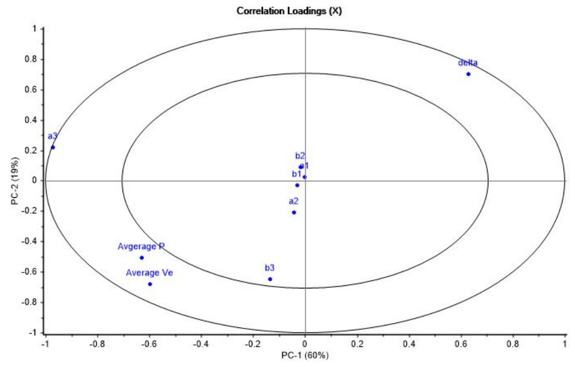
Beta coeffictions of the PLSR model:

$$\boldsymbol{B}_1 = \begin{bmatrix} 89.332 & -1.932 & 2.247 \\ -11.472 & 363.496 & -833.116 \\ 2.685 & -62.033 & 153.240 \end{bmatrix} \qquad \boldsymbol{B}_2 = \begin{bmatrix} 101.269 & 36.826 & -39.511 \\ 0.211 & -2.603 & -1.624 \\ -10.385 & 0.530 & 4.165 \end{bmatrix}. \tag{A.40}$$

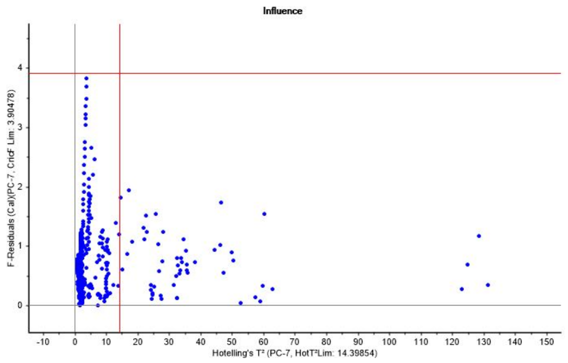
A.6.5 Fourier Series Motion With $k = 3$ Fourier coefficients



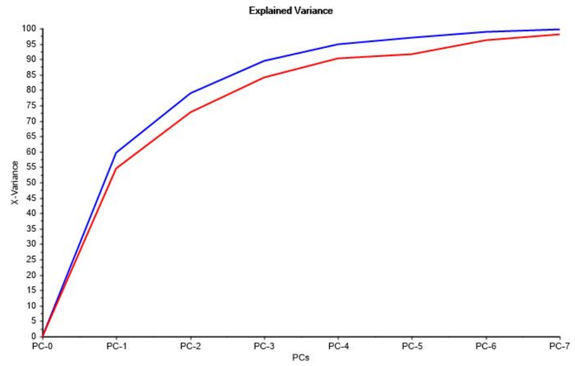
(a) Scores



(b) Loadings

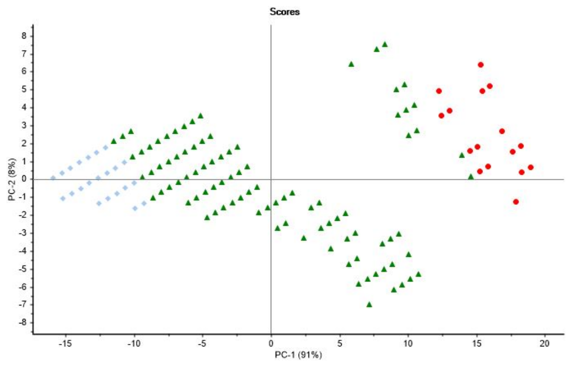


(c) Influence

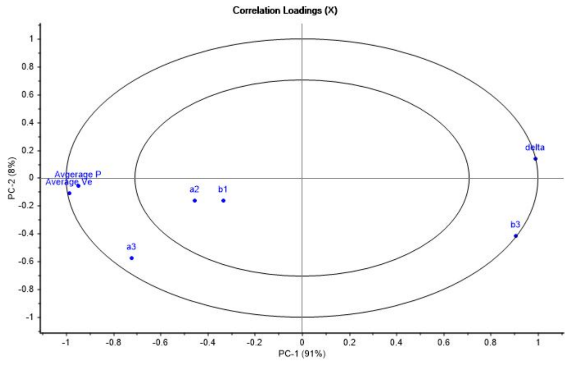


(d) Explained Variance

Figure A.14: Fourier series ($k = 3$): PCA overview

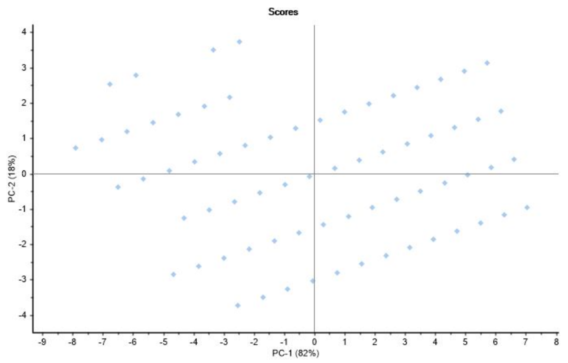


(a) PCA scores.

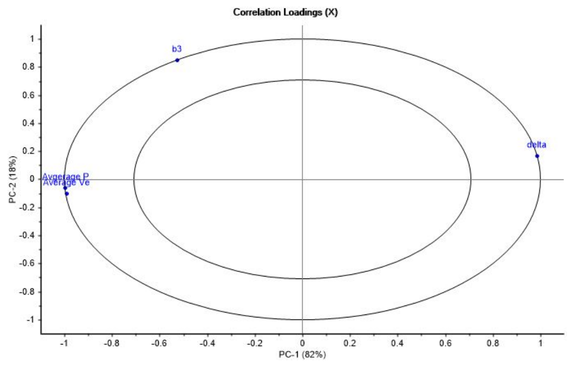


(b) PCA loadings.

Figure A.15: Modified motion: PCA scores and loadings of cluster 1.



(a) PCA scores.



(b) PCA loadings.

Figure A.16: Modified motion: PCA scores and loadings of cluster 2.

Table A.6: Fourier series ($k = 3$): some raw data of cluster 1

Sample	δ	a_1	a_2	a_3	b_1	b_2	b_3	P_{avg}	$\bar{v}$
150	60.236	0.000	-1.429	-40.000	-1.429	0.000	18.571	1.790	0.400
151	60.236	0.000	0.000	-40.000	0.000	0.000	20.000	1.808	0.402
155	58.819	0.000	0.000	-40.000	0.000	0.000	10.000	1.871	0.408
156	58.819	0.000	0.000	-40.000	0.000	0.000	12.857	1.891	0.410
157	58.819	0.000	0.000	-40.000	0.000	0.000	14.286	1.903	0.411
160	58.110	0.000	0.000	-40.000	0.000	0.000	10.000	1.949	0.415
161	58.110	0.000	0.000	-40.000	0.000	0.000	12.857	1.970	0.417
162	58.110	0.000	0.000	-40.000	0.000	0.000	14.286	1.983	0.418
163	58.110	0.000	0.000	-40.000	0.000	0.000	15.714	1.996	0.419
167	57.402	0.000	0.000	-40.000	0.000	0.000	14.286	2.067	0.425
168	57.402	0.000	0.000	-40.000	0.000	0.000	15.714	2.081	0.426
169	55.984	0.000	0.000	-34.286	0.000	0.000	18.571	2.103	0.428

Table A.7: Fourier series ($k = 3$): some raw data of cluster 2

Sample	δ	a_1	a_2	a_3	b_1	b_2	b_3	P_{avg}	$\bar{v}$
435	25.512	0.000	0.000	34.286	0.000	0.000	10.000	13.340	0.760
436	24.803	0.000	0.000	34.286	0.000	0.000	7.143	13.410	0.761
439	24.803	0.000	0.000	34.286	0.000	0.000	8.571	13.616	0.763
440	24.094	0.000	0.000	34.286	0.000	0.000	5.714	13.745	0.764
441	24.803	0.000	0.000	34.286	0.000	0.000	10.000	13.866	0.766
444	24.094	0.000	0.000	34.286	0.000	0.000	8.571	14.130	0.768
445	24.803	0.000	0.000	34.286	0.000	0.000	11.429	14.162	0.769
447	24.094	0.000	0.000	34.286	0.000	0.000	10.000	14.394	0.771
450	23.386	0.000	0.000	34.286	0.000	0.000	8.571	14.643	0.773
451	24.094	0.000	0.000	34.286	0.000	0.000	11.429	14.704	0.774
453	23.386	0.000	0.000	34.286	0.000	0.000	10.000	14.919	0.776
457	22.677	0.000	0.000	34.286	0.000	0.000	10.000	15.441	0.780

Table A.8: Fourier series ($k = 3$): The RMSE values of the PLSR model for cluster 1 and 2

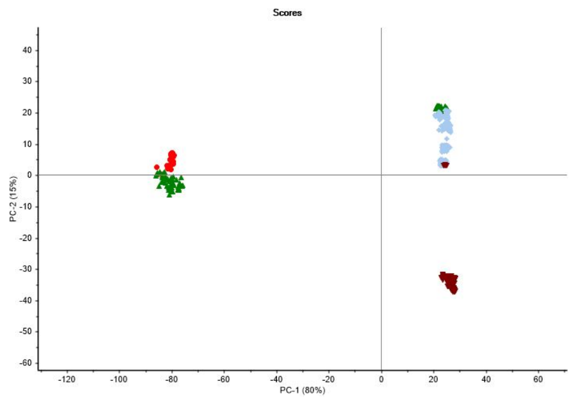
	Cluster 1					Cluster 2	
Factor	δ	a_2	a_3	b_1	b_3	δ	b_3
0	8.087	0.365	2.367	0.277	5.527	3.233	2.116
1	2.033	0.345	1.829	0.270	3.126	3.233	2.116
2	0.376	0.316	1.220	0.260	3.053	0.291	1.833

Beta coefficients of the PLSR model:

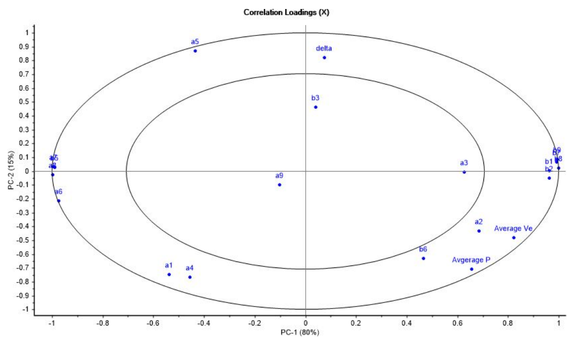
$$\boldsymbol{B}_1 = \begin{bmatrix} 111.703 & 0.000 & -3.340 & -66.465 & -2.008 & 0.000 & 40.593 \\ 2.344 & 0.000 & -0.396 & -3.524 & -0.243 & 0.000 & 0.069 \\ -13.989 & 0.000 & 0.915 & 8.557 & 0.554 & 0.000 & -5.694 \end{bmatrix} \tag{A.41}$$

$$\boldsymbol{B}_2 = \begin{bmatrix} 82.093 & 0.000 & 0.000 & 34.286 & 0.000 & 0.000 & 70.696 \\ -0.480 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 2.240 \\ -6.648 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -12.142 \end{bmatrix} \tag{A.42}$$

A.6.6 Multi-Fourier series



(a) PCA scores.



(b) PCA loadings.

Figure A.17: Multi-Fourier motion: PCA scores and loadings.

References

[1] Junku Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24, 2000.

[2] Elgar Desa, R Madhan, and P Maurya. Potential of autonomous underwater vehicles as new generation ocean data platforms. Indian Academy of Sciences, 2006.

[3] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles*. Butterworth-Heinemann, 2013.

[4] P. Liljebäck, Ø Stavdahl, K. Y. Pettersen, and J. T. Gravdahl. Mamba - a waterproof snake robot with tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 294–301, Sept 2014.

[5] E. Kelasidi, K. Y. Pettersen, J. T. Gravdahl, and P. Liljebäck. Modeling of underwater snake robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4540–4547, May 2014.

[6] W. Khalil, G. Gallot, and F. Boyer. Dynamic modeling and simulation of a 3-d serial eel-like robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1259–1268, Nov 2007.

[7] Thomas B Curtin, James G Bellingham, Josko Catipovic, and Doug Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94, 1993.

[8] E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl. Energy efficiency of underwater snake robot locomotion. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 1124–1131, June 2015.

[9] S. Hirose and M. Mori. Biologically inspired snake-like robots. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 1–7, Aug 2004.

[10] Alessandro Crespi and Auke Jan Ijspeert. Amphibot ii: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th international conference on climbing and walking robots (CLAWAR 2006)*, number BIOROB-CONF-2006-001, pages 19–27, 2006.

[11] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks*, 21(4):642–653, 2008.

[12] AJ Wiens and M Nahon. Optimally efficient swimming in hyper-redundant mechanisms: control, design, and energy recovery. *Bioinspiration & biomimetics*, 7(4):046016, 2012.

[13] E. Kelasidi, M. Jesmani, K. Y. Pettersen, and J. T. Gravdahl. Multi-objective optimization for efficient motion of underwater snake robots. *Artificial Life and Robotics*, 21(4):411–422, 2016.

[14] Wei Shun Chee, Jason Teo, and Kota Kinabalu. Empirically comparing three multi-objective optimization approaches for the automated evolution of snake-like modular robots. In *Proceedings of the international conference on artificial intelligence and pattern recognition (AIPR), Malaysia*, pages 175–183, 2014.

[15] Alexander Sproewitz, Rico Moeckel, Jérôme Maye, and Auke Jan Ijspeert. Learning to move in modular robots using central pattern generators and online optimization. *The International Journal of Robotics Research*, 27(3-4):423–443, 2008.

[16] Michael JD Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978.

[17] N. Shafii, M. H. Seyed Javadi, and B. Kimiaghalam. A truncated fourier series with genetic algorithm for the control of biped locomotion. In *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1781–1785, July 2009.

[18] Christian W Rempis and Frank Pasemann. An interactively constrained neuro-evolution approach for behavior control of complex robots. *Variants of Evolutionary Algorithms for Real-World Applications*, 87:305–341, 2012.

[19] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–72, 1992.

[20] Kenneth De Jong. Genetic algorithms: a 30 year perspective. *Perspectives on Adaptation in Natural and Artificial Systems*, 11, 2005.

[21] John J Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128, 1986.

[22] SN Sivanandam and SN Deepa. *Introduction to genetic algorithms.* Springer Science & Business Media, 2007.

[23] Thomas Back, Ulrich Hammel, and H-P Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[24] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[25] S.N. Sivanandam and S.N. Deepa. *Genetic Algorithms*, pages 15–37. Springer Berlin Heidelberg, 2008.

[26] Ajith Abraham and Lakhmi Jain. Evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, pages 1–6. Springer, 2005.

[27] David E. Goldberg and John H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3(2):95–99, 1988.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[29] Eckart Zitzler, Marco Laumanns, Lothar Thiele, et al. Spea2: Improving the strength pareto evolutionary algorithm, 2001.

[30] RTF Ah King, K Deb, and HCS Rughooputh. Comparison of nsga-ii and spea2 on the multiobjective environmental/economic dispatch problem. *University of Mauritius Research Journal*, 16(1):485–511, 2010.

[31] Christian von Lücken, Benjamín Barán, and Carlos Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58(3):707–756, 2014.

[32] Lam T Bui, Daryl Essam, Hussein A Abbass, and David Green. Performance analysis of evolutionary multi-objective optimization methods in noisy environments. In *Proceedings of the 8th Asia Pacific symposium on intelligent and evolutionary systems*, pages 29–39, 2004.

[33] Johannes Bader and Eckart Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation*, 19(1):45–76, 2011.

[34] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[35] N. Shafii, A. Khorsandian, A. Abdolmaleki, and B. Jozi. An optimized gait generator based on fourier series towards fast and robust biped locomotion involving arms swing. In *2009 IEEE International Conference on Automation and Logistics*, pages 2018–2023, Aug 2009.

[36] Charles Darwin. On the origin of species by means of natural selection. 1859. *London: Murray Google Scholar*, 1968.

[37] Mark Ridley. Evolution (3rd edn), 2004.

[38] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A tutorial on evolutionary multiobjective optimization. *Metaheuristics for multiobjective optimisation*, pages 3–37, 2004.

[39] James E Gentle, Wolfgang Karl Härdle, and Yuichi Mori. *Handbook of computational statistics: concepts and methods.* Springer Science & Business Media, 2012.

[40] S.N. Sivanandam and S.N. Deepa. *Terminologies and Operators of GA*, pages 39–81. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[41] AJ Umbarkar and PD Sheth. Crossover operators in genetic algorithms: a review. *ICTACT Journal on Soft Computing*, 6(1):1083–1092, 2015.

[42] Thomas Back. Optimal mutation rates in genetic search. In *Proceedings of the fifth international conference on genetic algorithms*, pages 2–8. Morgan Kaufmann, San Mateo, CA, 1993.

[43] Özgür Yeniay. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and Computational Applications*, 10(1):45–56, 2005.

[44] Fan Wang, Xiaofan Lai, and Ning Shi. A multi-objective optimization for green supply chain network design. *Decision Support Systems*, 51(2):262–269, 2011.

[45] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(1):26–37, Jan 1998.

[46] S. Favuzza, M.G. Ippolito, and E. Riva Sanseverino. Crowded comparison operators for constraints handling in nsga-ii for optimal design of the compensation system in electrical distribution networks. *Advanced Engineering Informatics*, 20(2):201 – 211, 2006. Engineering Informatics for Eco-Design.

[47] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

[48] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[49] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, June 2000.

[50] Aravind Seshadri. A fast elitist multiobjective genetic algorithm: Nsga-ii. *MATLAB Central*, 182, 2006.

[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[52] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. *The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration*, pages 862–876. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[53] Eric W. Weisstein. Generalized fourier series. MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GeneralizedFourierSeries.html, [Online; accessed 19-April-2017].

[54] Eric W. Weisstein. Fourier series. MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/FourierSeries.html, [Online; accessed 19-April-2017].

[55] Charles Henry Edwards and David E.. Penney. *Differential Equations and Boundary Value Problems: Computing and Modeling*, pages 572–589. Prentice Hall, 5th edition, 2014.

[56] Harald Martens and Tormod Naes. *Multivariate calibration*. John Wiley & Sons, 1992.

[57] CAMO Software. *The Unscrambler® X 10.3*. 2014. http://www.camo.com/, [Online; accessed 5-May-2017].

[58] CAMO Software. *The Unscrambler Appendices: Method References*. 2017. http://www.camo.com/TheUnscrambler/Appendices/, [Online; accessed 30-May-2017].

[59] Kim H Esbensen, Dominique Guyot, Frank Westad, and Lars P Houmoller. *Multivariate data analysis: in practice: an introduction to multivariate data analysis and experimental design*. Multivariate Data Analysis, 2002.

[60] Svante Wold, Michael Sjöström, and Lennart Eriksson. Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

[61] Carlos M Fonseca and Peter J Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(1):26–37, 1998.

[62] Python Sofware Foundation. *Python Language Referencem version 3.5*. Available at http://www.python.org, 2017.

[63] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[64] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[65] J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, November 2008. http://www.tik.ee.ethz.ch/sop/download/supplementary/hype/, [Online; accessed 20-April-2017].

[66] MATLAB. *version 9.2 (R2017a)*. The MathWorks Inc., Natick, Massachusetts, 2017.

[67] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436:54–70, 2012.

[68] Tobias Storch. On the choice of the population size. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 748–760. Springer, 2004.

[69] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, Jan 1949.