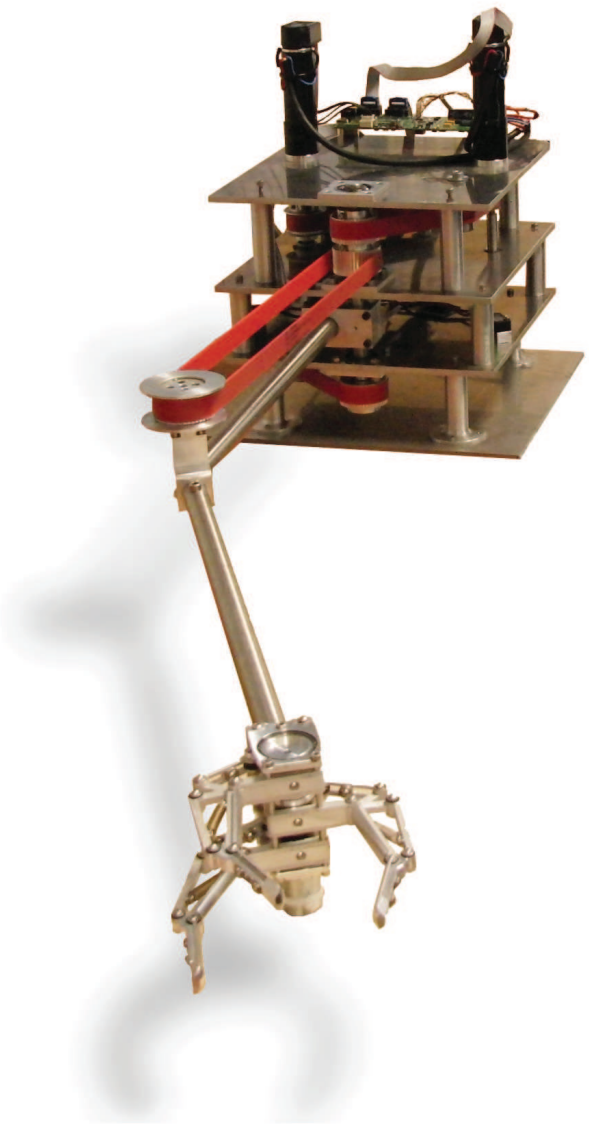
# Evolutionary Co-Optimisation of Control and System Parameters for a Resonating Robot Arm

S.J. Pen

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

# Evolutionary Co-Optimisation of Control and System Parameters for a Resonating Robot Arm

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

S.J. Pen

March 9, 2012

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology


Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Evolutionary Co-Optimisation of Control and System Parameters for a Resonating Robot Arm

by

S.J. Pen

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: March 9, 2012

Supervisor(s):

prof.dr. R. Babuška

dr.ir. M. Wisse

Reader(s):

dr.ir. G.A. Delgado Lopes

ir. M.C. Plooij

# Abstract

This study investigates the evolutionary co-optimisation of fuzzy control and system parameters for the Resonating robot Arm (RA). The RA is a novel concept for a pick-and-place manipulator that uses a spring mechanism to reduce the required actuator torques. Since the performance of the total system depends on the combination of the spring mechanism and the controller it is difficult to find (near) optimal solutions using conventional design approaches in which the system and the controller are optimised separately. Therefore evolutionary co-optimisation is proposed in which Evolutionary Algorithms (EAs) are used to optimise the RA system as a whole.

Three experiments were conducted in which the first experiment validated the use of fuzzy control and EAs to find near optimal control solutions, and the second and third experiment considered the co-optimisation of the RA with one and two degrees-of-freedom (DOF), respectively. Two types of EAs (*CoSyNE* and *CMA-ES*) and two types of fuzzy controllers (with fixed and free membership functions) were applied and their performances compared.

The results revealed that evolutionary co-optimisation yields near optimal solutions for the 1-DOF RA, which require 43% less torque than the solution found through a separate optimisation of the system and control parameters. In case of the 2-DOF RA, evolutionary co-optimisation resulted in working solutions, however, no consistent convergence to near optimal solutions was found. Additionally, it was shown that for all experiments the best solutions came from the *CMA-ES* algorithm in combination with the fuzzy controller with free membership functions.

The main conclusion drawn from this study is that evolutionary co-optimisation is an effective approach to find near optimal solutions for the 1-DOF RA, however more research is needed for it to be effectively applied to the 2-DOF RA.

# Contents

# List of Figures

# List of Tables

# Acknowledgements

I would like to thank my supervisors prof.dr. Robert Babuška, dr.ir. Martijn Wisse and my mentor dr.ir. Wouter Caarls for all the time they invested in me and the many things I learned from them while I was working on this thesis.

I would also like to thank the other members of the exam committee, dr.ir. Gabriel Lopes and ir. Michiel Plooij, for taking the time to evaluate my thesis.

I am grateful for my fellow students in the Resonating Arm Project and my friends studying with me in the basement. They have always shown interest in my work and even took the time to provide me with their feedback.

At last I owe my deepest gratitude to my parents and girlfriend who have shown great understanding, patience and care, and just never stop believing in me.

Delft, University of Technology S.J. Pen
March 9, 2012

For my grandfather and grandmother,
who would have loved to see me graduate.

# Chapter 1

# Introduction

Robots are playing an increasingly large role in both society and industry, and the demands and requirements for their design are becoming more and more complex. Engineers are therefore constantly challenged to find better and more advanced robotic solutions for the complex modern day problems. This drives the development of more effective and efficient design approaches.

Traditionally, a robot design process starts with the design and optimisation of a mechanical system. Thereafter a mathematical model is derived, which is used to design and tune a corresponding controller. A problem with this approach is that during the optimisation of the system the impact of the design variables on the performance of the whole system is difficult to grasp. Since during the design of the mechanical system the controller still needs to be designed, human designers must often rely on intuition and experience in order to tune the system parameters. Therefore this approach can easily lead to suboptimal system designs.

Co-optimisation approaches solve this problem by optimising the system and controller in parallel. By considering the system as a whole, parameters of both the system and the controller are tuned based on their influence on the overall performance.

## 1-1 Evolutionary Co-Optimisation

The parallel optimisation of system and control generally results in high dimensional and complex optimisation problems. These problems cannot be solved by analytical methods and heuristic search methods have to be applied. Most often Evolutionary Algorithms (EAs) are applied to solve these problems due to their robustness to local minima and to search spaces that are highly nonlinear and of varying dimensions.

One of the pioneers in evolutionary co-optimisation is Karl Sims [1] who showed that EAs can be used to evolve both the morphology and control of walking, jumping or swimming creatures in a virtual environment. Later, others used the same approach to co-optimise the sensor positioning and control of flying and driving robots [2–4] or to design walking and swimming robots from modular parts [5–7].

It appears that evolutionary co-optimisation has mainly attracted the attention of researchers in the field of autonomous robots, while its benefits are generally applicable to the development of other types of robots, such as manipulators used in the industry. The software Darwin2K from Leger [8] is one of the few applications concerning the optimisation of robot manipulators. This software is able to synthesise the robot morphology and optimise the system and control parameters. In order to keep the complexity of the problem manageable, simple local control strategies (i.e. PID) are applied. Unfortunately, these simple local controllers also limit the search for an optimal solution [8]. Therefore this research concerns the optimisation of more complex and global controllers.

## 1-2 Resonating Robot Arm

In this research, evolutionary co-optimisation is applied to the optimisation of a Resonating robot Arm (RA). This arm is a novel concept of a pick-an-place robot and aims to reduce the actuation power needed to fulfil the desired pick-and-place tasks through the use of a spring mechanism. This spring mechanism fulfils a task normally fulfilled by the controller, namely the acceleration and deceleration of the arm when moving between the pick-and-place positions. The task of the original controller can therefore be reduced to small actions needed to steer the system, reject disturbances and account for friction losses.

The spring mechanism can be considered as a mechanical subcontroller, which works in parallel with the actuator controller to perform the pick-and-place tasks. Both of these controllers need to be carefully tuned to obtain the best results; a parallel or co-optimisation of both system and control is therefore clearly desired.

## 1-3 Research Goals

This thesis addresses the use of evolutionary co-optimisation in the development of the RA. The general goal is to develop an environment that allows co-optimisation of the RA and to investigate its benefits and limitations. Other than research such as performed by Leger [8], this thesis does not consider the optimisation of a system configuration in combination with linear local controllers. Instead, it aims to optimise the system parameter of a predefined configuration and a nonlinear global controller. This nonlinear feedback controller is defined by a fuzzy control architecture. This thesis only considers the movements in the horizontal plane and does not address the influence of external disturbances or obstacles when performing the tasks.

The research question that will be answered in this thesis is:

*To what extent is co-optimisation through evolutionary algorithms able to find (near) optimal solutions when optimising both the system parameters and the parameters of a fuzzy controller for a resonating robot arm?*

This research question will be answered by conducting three experiments in which the following three research subquestions are considered:

*Exp. 1: Is the fuzzy control approach optimised by EAs suitable to generate (near) optimal feedback control solutions?*

*Exp. 2: Is evolutionary co-optimisation an effective approach to find (near) optimal solutions for Resonating Arm with one degree-of-freedom?*

*Exp. 3: Is evolutionary co-optimisation an effective approach to find (near) optimal solutions for Resonating Arm with two degrees-of-freedom?*

## 1-4 Approach

Two state-of-the-art evolutionary algorithms (CoSyNE and CMA-ES) will be applied and two types of fuzzy control architectures will be considered in which one has membership functions with fixed locations and the other uses free membership function with variable locations to be optimised by the EAs. These different EAs and controllers will be used to investigate the influence of the EA and controller selection on the effectiveness of the evolutionary co-optimisation.

Next to providing an answer to the research question and subquestions, this work will conclude which evolutionary algorithm (CoSyNE or CMA-ES) and which fuzzy control architecture (with fixed or free membership functions) appeared to be the most effective.

## 1-5 Thesis Outline

Chapter 2 of this thesis gives an introduction to the field of evolutionary algorithms. It explains why evolutionary algorithms are especially useful for solving engineering problems and gives an overview of the different components of which an evolutionary algorithm is constructed. Thereafter two state-of-the-art evolutionary algorithms CoSyNE and CMA-ES, which have been used in this research, are presented and their general structure is discussed.

Chapter 3 contains details about the RA system designed for pick-and-place tasks. This chapter presents the purpose of the RA, explains its mechanical structure and discusses the specific properties. At last, the derivation of the system model is discussed which has been used to simulate the system during optimisation.

Chapter 4 introduces the fuzzy control strategy used for nonlinear feedback control. It presents the control requirements which led to the controller selection and explains the basic principles. Thereafter the two types of fuzzy control (with fixed and free membership functions), which are investigated in this research, are presented and their expected advantages and disadvantages are discussed.

In Chapter 5 the execution and results of three experiments are discussed in which evolutionary algorithms have been applied on problems of different complexity. The first experiment validates the use of fuzzy control in combination with evolutionary algorithms by solving a minimal-time control problem on a single mass. The second experiment validates the effectiveness of evolutionary co-optimisation by optimising a simpler form of the RA system in which only one degree-of-freedom is considered. The third experiment analyses the results found after co-optimising the control and system parameters of the complete RA with two degrees-of-freedom. The chapter is ended with a discussion and analysis on the performance of the different fuzzy controllers and EAs used in the experiments.

Finally, in Chapter 6 we conclude and discuss our work and present recommendations for further research.

# Chapter 2

# Evolutionary Algorithms

In this research evolutionary algorithms (EAs) are used to co-optimise control and system parameters. This chapter introduces EAs, discusses why these algorithms are being used and explains the basic concepts on which EAs are based. Finally, two types of EAs are presented, which will both be used in this research.

## 2-1 Introduction

Evolutionary algorithms is a class of optimisation heuristic that mimics the process of Darwinian evolution [9] by copying natural mechanisms such as *selection*, *reproduction* and *mutation*. Since its inception in the 1950s the field of EAs has strongly matured making EAs widely recognised as powerful problem solving methods [10].

Successful applications of EAs can be found in a large array of fields [11] such as mathematics, chemistry, biology, finance and arts and also in the field of engineering the use of EAs is becoming more and more popular [12].

## 2-2 Why Evolutionary Algorithms?

Engineering problems, such as the one solved in this research, are in general characterised by their high dimensionality, complexity and constraints causing a large and nonlinear search space with many peaks and discontinuities [13]. In Figure 2-1 a classification structure of existing optimisation methods is presented. With this structure the two main advantages of using EAs in engineering problems are presented.

The first advantage of EAs is the fact that they are *non-gradient-based* optimisation methods and thus do not require a gradient of the search space. Gradient-based methods need the gradient of an objective function in order to apply so called 'hill-climbing' techniques (i.e. steepest descent method) where the local optima are sought by moving the search in the

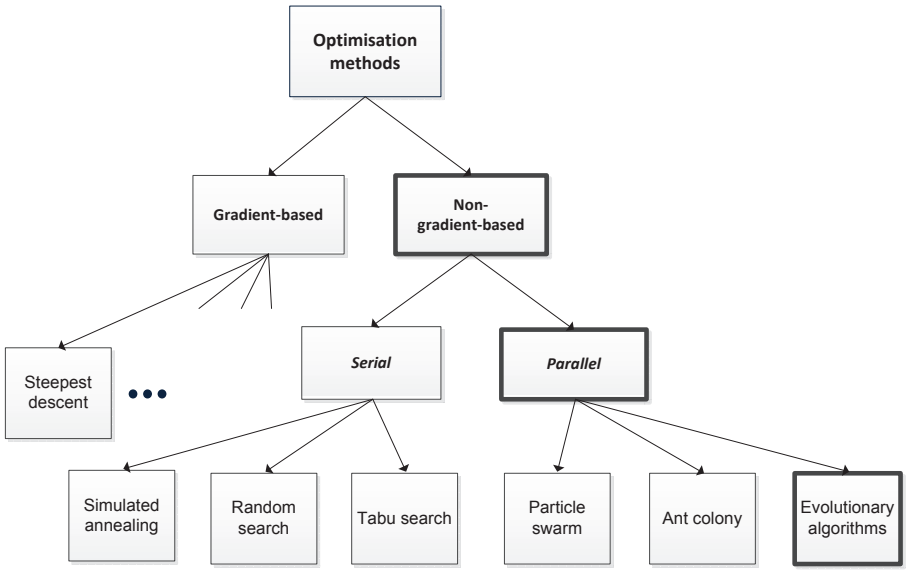



**Figure 2-1:** A classification of optimisation methods in gradient-based, non-gradient-base, serial and parallel methods with examples of common methods.

direction of the local gradient. These methods are powerful when the objective functions have low complexity and continuous derivatives, however, when this is not the case they lack robustness and are not able to perform an effective search [14]. The high complexity and possible discontinuities in engineering problems therefore require the use of non-gradient-based methods.

A second advantage is the fact that EAs are performing a *parallel search* of the search space. Serial optimisation algorithms (e.g. simulated annealing and tabu search) evaluate only one point in a search space at a time, whereas parallel optimisation algorithms (e.g. particle swarm optimisation, ant colony optimisation and evolutionary algorithms) gather information from multiple points. When unfortunately initialised a serial optimisation methods could start-off in a local optimum from which it may not be able to get out. Multiple restarts are therefore required to find the global optimum. The main disadvantage of the multiple restart approach is that no search space information is shared between the different searches. Therefore the same area in the search space may be explored multiple times, which makes the multiple restart approach inefficient. In a parallel search the search space is sampled at multiple points at the same time and all information is used to direct the search in the most promising area of the search space. Since the search space of the co-optimisation problems will most likely have many local optima parallel optimisation methods seem to be the more appropriate since they are less sensitive to their initialisation and therefore more effective.

Another advantage of EAs is the fact that they are able to optimise solutions, which consist of both real- and discrete valued parameters. Although in this research we will only focus on the optimisation of real valued parameters it could very well be that future optimisations require the optimisation of discrete parameters (e.g. choosing between different actuator types).

An often mentioned limitation of EAs is their sensitivity to the various parameters that define them. In this research we will try to overcome this limitation by applying two different types

of EAs of which one has almost no parameters to be tuned and for the other the parameters have been applied as published.

## 2-3 Fundamentals of Evolutionary Algorithms

Although the term EA comprises many different algorithms the structure of the algorithms can be represented as the pseudocode in Algorithm 1. At first, an *initial* population of multiple solutions is created. These initial solutions are traditionally created randomly, but it is also possible to use predefined solutions. Each solution is *evaluated* for its ability to solve the problem, which is indicated by a fitness score. The evolutionary mechanism *selection* uses the fitness scores to choose solutions for *reproduction*, in which new solutions are created based on the information of the selected solutions. Another selection procedure brings these new solutions into the next generation by replacing worse performing solutions from the last generation. This process is repeated until a solution with satisfactory fitness has been found or until the number of cycles exceeds a certain limit.

**Algorithm 1** Pseudocode of an evolutionary algorithm

1: $t \leftarrow 1$
2: $P(t) \leftarrow$ `initialise`
3: **repeat**
4: `evaluate`$(P(t))$
5: $S(t) \leftarrow$ `select`$(P(t))$
6: $R(t) \leftarrow$ `reproduce`$(S(t))$
7: $P(t+1) \leftarrow$ `replace`$(P(t), R(t))$
8: $t \leftarrow t+1$
9: **until** termination criteria

In this section the basic components defining an EA are discussed:

- representation
- fitness evaluation
- population
- parent selection mechanism
- variation operators
- replacement mechanism
- initialisation
- termination condition

Each EA is defined by these components, only the detailed implementation of these components varies between the different types of EAs.

### 2-3-1 Representation

An EA can only find possible solutions to a problem when the solutions are transformed into a representation the EA can handle. This means that a mapping is needed between possible solutions within the original problem context, called phenotypes, onto individuals within the context of the EA, called genotypes.

In EAs this encoding can be done as a string of binary code [15], real-valued numbers [6], integers [16], or as a tree structure [17]. However, for mechanical problems the most effective representation is a string of real-valued numbers [18]. In order to limit the search space the parameters are often bounded to a maximum and minimum value.

The EA will search for optima in the genotype space and will deliver a good genotype solutions, if the mapping between the genotype space and the phenotype space has been done correctly this can be decoded into a more meaningful phenotype solution.

### 2-3-2 Fitness Evaluation

The fitness evaluation step evaluates each solution and determines the relative quality of a solution in comparison to other solutions using the so called fitness function. The quality of a solution is called the fitness score and it facilitates the selection mechanism with a measure to determine the best solutions in a set of solutions. The fitness function is based on the objective function of the optimisation problem to be solved. The evaluation of the solutions is therefore often applied in the phenotype space by decoding the solutions before evaluation.

### 2-3-3 Population

The group of solutions handled by the EA is called the population. In order to define the population one needs to determine the number of genotypes that are in it. For most EAs this number is kept constant and is not changed during the evolutionary search. For some sophisticated EAs a population also has an additional spatial structure, with a distance measure or a neighbourhood relation [19]. In these cases this measures or relations have to be defined as well.

### 2-3-4 Parent Selection Mechanism

The parent selection mechanism operates at the population level and is used to select only the best solutions in the population as a seed for the production of new solutions. The selection is always made relative to what currently is present in the complete population and it supports quality improvement by favouring better solutions above solutions that perform worse. The selected solutions are called parents since they will be used in the variation operators to produce the new solutions, also called offspring or children.

Often the parent selection is probabilistic giving high quality individuals a higher chance to become parents than those with low quality. However, low quality are still able to be selected in order to prevent the search from becoming too greedy, which could lead to premature convergence to local optima.

### 2-3-5 Variation Operators

The variation operators are responsible for creating offspring from a group of selected parents. Two types of variation operators can be recognised; mutation and recombination.

#### Mutation

Mutation takes one parent and slightly modifies the genotype representation in order to create a mutant of the original parent (Figure 2-2). The modification applied is always stochastic and its purpose is to discover potential better solutions based on the current best solutions.

Many mutations with large effects will be beneficial to get out of sub optima, but make convergence to a global optimum more difficult [18]. The need for both small and large mutations is often solved by using Gaussian or Cauchy distributions, these distributions favour small mutations but still allow bigger mutations to happen.

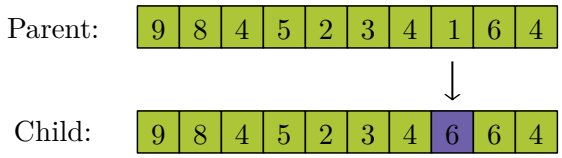



**Figure 2-2:** Mutation operator.

#### Recombination

Recombination or crossover merges information from two parent genotypes into one or two offspring genotypes. The idea behind recombination is that by mating two individuals with different but desirable features, it is possible to produce offspring that combines both of those features. Since one cannot tell which information of a solution is responsible for the desired behaviour, the offspring is constructed by combining randomly selected partial information from both parents.

Three main recombination approaches are: single-point, two-point and uniform crossover. In single-point and two-point crossover the array of information from the parents is cut at, respectively, one or two random points and the offspring is created by combining the sliced information of both parents (Figure 2-3). In uniform crossover each parental variable is randomly assigned to either of the two children.

### 2-3-6 Replacement Mechanism

The replacement mechanism prepares the next generation of the population by replacing solutions from the current generation with the new offspring produced by the variation operators. The selection of the solutions to be replaced is often based on the quality of a solution expressed by its fitness score.

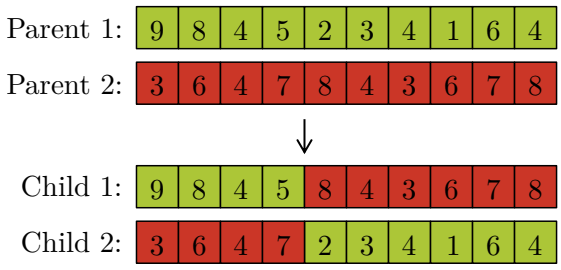



**Figure 2-3:** Single-point crossover operator.

### 2-3-7 Initialisation

In the initialisation step one defines the solutions in the first generation of the evolutionary process. Generally this is done randomly since one desires the first individuals to be spread over the complete search space before converging to more promising areas. However, when prior knowledge is available about the approximated location of the optimum one could initiate the algorithm around this location in order to point the algorithm in a promising direction. This is often done to speed up the convergence.

### 2-3-8 Termination Condition

A trivial termination condition for the evolutionary process is the moment when the optimum has been found. However, for many optimisation problems the optimum is unknown and one is therefore never sure whether a solution is the best solution or whether there still exist better solutions. Moreover EAs are stochastic and might take a long time to converge to the optimum or may never reach the optimum at all.

Commonly used termination conditions are therefore based on the maximum allowed CPU time elapses, total number of fitness evaluations or the improvement of the solutions in a given period of time.

## 2-4 Algorithm Design

Two different EAs have been used in this research in order to analyse and compare their performances. Additionally an incremental fitness function has been constructed to be used for the optimisation of the Resonating Arm (RA). In this section we will discuss the design of the fitness function and present the two algorithms CMA-ES and CoSyNE used in this research.

### 2-4-1 Fitness Function

The fitness functions used in this work are defined in such a way that a minimum amount of a priori knowledge is required to construct them. This is done by using incremental aggregate

fitness functions, which, in contrast to the often used behavioural fitness functions, evaluate solutions on their ability to fulfil the desired task and not on their behaviour.

Most of the fitness functions used in EAs are behavioural fitness functions [20], which determine the fitness of a solution on its ability to perform the behaviour that is expected to lead to an optimal fulfilment of the task. An example of such a fitness function is given in (2-1) in which the minimisation of the settling time of a controlled system is translated into the weighted integral of the error $e$ and error derivative $\dot{e}$.

$$f = \int_{t_0}^{t_f} a \cdot |e(t)| + b \cdot |\dot{e}(t)| dt \tag{2-1}$$

The main reason for the use of behavioural fitness functions is that even if none of the solutions is able to fulfil the task (e.g. converge) their behaviour can still be used to rank the solutions. However, their disadvantage lies in the fact that *a priori* knowledge is required of the optimal behaviour that corresponds to a certain task. A mismatch between the expected optimal behaviour and the real optimal behaviour may prevent the optimal solutions from being found.

A better approach is to score the solutions on their ability to perform the task, regardless of their behaviour. These types of fitness functions are called aggregate fitness functions [20]. The aggregate fitness function for the minimal settling time problem used before would therefore translate into a direct relation between the fitness $f$ and the settling time $t_{settle}$

$$f = t_{settle} \tag{2-2}$$

Although aggregate fitness functions do not impose any restrictions on the search for the optimal solution, their disadvantage lies in the fact that at early stages in the optimisation process none of the solutions might be able to fulfil the task and thus no distinction can be made between good and bad performing solutions. This problem, called the 'bootstrap problem' [21], results in the fact that the evolutionary mechanisms of selection and replacement cannot effectively be applied.

A solution to this problem is the use of incremental aggregate fitness functions tailored for the desired task at hand [20]. These fitness functions eliminate the 'bootstrap problem' by ranking the solutions that do not fulfil the main task by using sub fitness functions. These sub fitness functions determine the solution's ability to fulfil a sub task, which is required to fulfil the main task. In the fitness functions used in this research this often translates into the sub task of reaching a desired state, which is fundamental for the main task of using a minimal torque or convergence time.

Although for many EAs the shape of the fitness function has an influence on their performance, this is not the case for the algorithms used in this work. This is because both the CoSyNE and CMA-ES algorithm are rank-based algorithms [22, 23], which means that any fitness function that preserves the rank will be considered equal from the EA's point of view [24].

Detailed descriptions of the fitness functions used for the conducted experiments are given in Chapter 5.

### 2-4-2 CoSyNE algorithm

The CoSyNE (COoperative SYnapse NeuroEvolution) algorithm is one of the two EAs used in this research. CoSyNE has been proposed by Gomez, Schmidhuber and Miikkulainen in

2008 [22] and was originally designed for the optimisation of neural networks, but has also been successfully applied for the optimisation of system parameters of a car setup [25]. It is a relatively new algorithm and experiments have shown that it is able to match-up and in some cases beat the current state-of-the-art algorithm CMA-ES (which is the second algorithm used in this research) [22] [25].

CoSyNE is part of the family of cooperative co-evolutionary algorithms, which are inspired by natural ecosystems where several species cooperate for their survival (i.e. pilot fish that clean the skin of a shark in return for protection). In this class large optimisation problems are decomposed into sub components, which are subject to local evolutionary processes. This decomposition and evolution of sub solutions is expected to create EAs, which are capable of optimising problems with high dimensionality.

Since the co-optimisation of control and system parameters often results in high dimensional search spaces, the cooperative co-evolutionary approach is expected to be beneficial to optimise both the control and system parameters of the RA.

#### General Framework

The cooperating species or sub populations in the CoSyNE algorithm are the different sub components of a complete solution. Since the fitness function cannot determine a fitness score for a single sub component complete solutions are assembled using one member from each species. The fitness scores at the species level is defined in terms of the fitness of the complete solutions in which the species members participated. The evolution of each sub population is then handled independently by a standard evolutionary process of selection, reproduction, mutation and replacement [22].

To explain these steps more clearly a pseudo-code of the CoSyNE algorithm is presented in Algorithm 2. First (line 1), a population $P$ consisting of $n$ sub populations $P_i$ $i = 1, \ldots, n$ is initialised randomly, where $n$ is the number of variables needed to define a complete solution. Each sub population is initialised to contain $m$ real numbers chosen from a uniform probability distribution in the interval $[0, 1]$. Starting with these initial sub populations the CoSyNE algorithm loops through a sequence of generations until the stopping criteria are met (lines 2-11). Each generation starts by constructing a complete solution by combining one individual from each sub population. In CoSyNE this is done by taking the $j^{th}$ row of the population matrix $P$ (line 4). The complete solution is now evaluated and in this way a fitness score is determined for all $m$ combinations (line 5). The fitness score of an individual is set equal to the fitness score of the complete solution it was part of. A quarter of the best performing solutions is selected as parents and used to create offspring by applying the variation operators mutation and crossover (line 7-8). The newly created individuals are now used to replace the worst performing individuals (line 9).

Up to now the algorithm is equivalent to a conventional genetic algorithm that evolves complete solutions. In order to incorporate co-evolution into the algorithm the sub populations are permuted probabilistically so that each individual forms part of a potentially different solution in the next generation (line 10). Gomez proposed a permutation mechanism in which individuals are marked randomly according to probabilities assigned by a user-defined

**Algorithm 2** CoSyNE algorithm

1: $P \leftarrow$ `initialise`$(P^1, P^2, P^3, \ldots, P^n)$ {initialise all $n$ sub population with $m$ variables}
2: **repeat**
3: **for** $j = 1$ to $m$ **do**
4: $P_j \leftarrow (P_j^1 P_j^2 P_j^3 \ldots P_j^n)$
5: `evaluate`$(P_j)$
6: **end for**
7: $P_{\text{parents}} \leftarrow$ `SelectBestQuarter`$(P)$
8: $P_{\text{offspring}} \leftarrow$ `CreateOffspring`$(P^1_{\text{parents}}, P^2_{\text{parents}}, P^3_{\text{parents}}, \ldots, P^n_{\text{parents}})$
9: $P_{\text{new}} \leftarrow$ `ReplaceWorstQuarter`$(P, P_{\text{offspring}})$
10: $P \leftarrow$ `Permutate`$(P^1_{\text{new}}, P^2_{\text{new}}, P^3_{\text{new}}, \ldots, P^n_{\text{new}})$
11: **until** end criteria

function $f()$ [22]. One possible probability function presented in his paper is:

$$\text{mutprob}(x_{ij}) = 1 - \sqrt{\frac{f(x_{ij}) - f_i^{min}}{f_i^{max} - f_i^{min}}} \tag{2-3}$$

where $f(x_{ij})$ is the fitness of individual $x_{ij}$, and $f_i^{min}$ and $f_i^{max}$ are, respectively, the fitness of the least and most fit individuals in sub population $i$. In this permutation process the probability of an individual to be permuted is inversely proportional to its relative fitness, which means that the combinations with high fitness are more likely to be preserved, while those with low fitness are more likely to be disrupted.

However, Gomez does not use this function in his publication and rather uses the less sophisticated probability function; $\text{prob}(x_{ij}) = 1, \forall i, j$, which he only applies on the old individuals. Both in [22] and [25] this approach was found to work well and will therefore also be applied in this research. Moreover the sophisticated function would make the algorithm dependent on the distance between the fitness values, which is undesirable since a rank-dependent algorithm is preferred [24].

After permutation the process is repeated until the stopping criteria have been met.

### 2-4-3 CMA-ES algorithm

The performance of the CoSyNE algorithm will be compared with a second EA used in this research called CMA-ES (Covariance Matrix Adaptation Evolution Strategy). CMA-ES is a well-known state-of-the-art optimisation algorithm proposed by Gawelczyk, Hansen and Ostermeier [23]. Many empirical results show the effectiveness of CMA-ES and it is considered to be particularly useful on non-convex, non-separable, ill-conditioned, multi-model or noisy objective functions.

The main advantages of CMA-ES is that it is quasi parameter-free. Finding good (default) strategy parameters is considered as a part of the algorithm design, which saves the user from endless tuning of the different algorithm parameters. The only used-defined parameter of importance is the population size, which influences the difference between a more local search with a fast convergence or a global search with a longer convergence time.

#### General Framework

CMA-ES was developed with the idea that the evolutionary operators of selection, recombination and mutation implicitly define a distribution from which the next generation is sampled. Therefore an evolutionary search in its simplest form can be described by a three step process of sampling, evaluation and an update of the distribution parameters as shown in Algorithm 3.

**Algorithm 3** Basis structure of the CMA-ES algorithm

1: `set population size` $\lambda \in \mathbb{N}$
2: `initialise distribution parameters` $\gamma$
3: **repeat**
4: `sample distribution` $P(\mathbf{x}|\gamma) \rightarrow \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \in \mathbb{R}^n$
5: `evaluate` $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ `on fitness function` $f$
6: `update distribution parameters` $\gamma \leftarrow F(\gamma, x_1, \ldots, \mathbf{x}_\lambda, f(x_1), \ldots, f(\mathbf{x}_\lambda))$
7: **until** end criteria

The sample distribution used in CMA-ES is a multi-variate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ that is uniquely determined by its mean value $\mathbf{m} \in \mathbb{R}^n$ and its symmetric positive definite $n \times n$ covariance matrix $\mathbf{C}$. This distribution can be geometrically interpreted as an iso-density ellipsoid, which shape is determined by the covariance matrix $\mathbf{C}$. Its position is determined by the mean value $\mathbf{m}$ around which the distribution is symmetric.

The mean value $\mathbf{m}$ of this distribution is updated after each generation by selecting the best points from the evaluated population. The weighted average of these points is then used to determine the updated mean value $\mathbf{m}$, where the weight of each point is determined by its ranking position. Also the covariance matrix $\mathbf{C}$ is updated by finding a covariance matrix that fits the distribution of the best points of the new populations best. Since CMA-ES uses a non-elitist selection only the newly created solutions are used for these updates.

The step-size determines how far new solutions are sampled from the mean value. This step-size is also automatically updated in CMA-ES by using a cumulative step-size adaptation, which increases the step-size when the mean vector $\mathbf{m}$ is updated in the expected direction based on the previous update directions. When this is not the case the step-size is reduced in order to facilitate a finer search.

### 2-4-4 Algorithm Parameters

The user-defined parameters of both the CoSyNE and CMA-ES algorithm are given in Table 2-1.

The basic CoSyNE framework as presented by Gomez [22] describes the use of conventional genetic operators acting on the different sub populations, but does not specify which genetic operators are used. In this research uniform crossover is used and each individual has a probability of 0.3 to be mutated by a normal distributed noise. The population size was set to 40 after investigating the convergences performances of different population sizes.

As mentioned the CMA-ES algorithm is designed to be a quasi-parameter free optimisation algorithm and therefore many parameters are already identified by the developers. The only

**Table 2-1:** The user defined parameters of the CoSyNE and CMA-ES algorithm.

| Parameter | CoSyNE | CMA-ES |
|---|---|---|
| Population size | 40 | 40 |
| Crossover type | Uniform | - |
| Mutation | Normal distributed noise ($\sigma = 0.15$) | - |
| Mutation probability | 0.3 | - |

variable that can be varied by the user is the population size $\lambda$, which in CMA-ES stands for the number of new solutions sampled after each generation. In order to get a fair comparison between both algorithms the population size was set equal to the population size used in the CoSyNE algorithm.

## 2-5 Summary

Evolutionary algorithms are a class of optimisation approaches that mimic the process of Darwinian evolution. Their main advantages are the fact that they do not need a gradient an objective function (non-gradient-based) and explore the search space at multiple points at the same time (parallel search). This makes them suitable and efficient optimisation algorithms for engineering problems, for which the derivatives of the objective function may often be discontinues and the search space are likely to be highly complex with many local minima.

The basic structure of EAs is based on evaluation, selection, reproduction and replacement. First a population of different solutions is initialised. Every solution is evaluated in the fitness function, which determines the quality of the solution and gives it a fitness score. The fitness scores of all solutions are used to select the better performing solutions (parents) for reproduction. New solutions are created by the variation operators, which combine or slightly mutate the information of the selected solutions. These new solutions are then place back into the population by replacing the worst performing solutions.

The fitness functions used in this research are aggregate fitness functions that score the solutions on their ability to perform the desired task. In contrast to the often used behavioural fitness functions, which limit the search for an optimal tasks fulfilment by imposing restrictions on the behaviour. The 'bootstrap problem' of aggregate fitness functions is avoided by an incremental structure in which solutions that are not able to fulfil the main task are scored on their ability to fulfil sub tasks.

The CoSyNE algorithm is one of the two EAs used in this research. It is a cooperative co-evolutionary algorithm in which the solutions are decomposed into sub solutions. The fitness of a sub solution is determined by combining different sub solutions into one complete solutions and evaluating the complete solution in the fitness function. Each sub solution is thereafter evolved using conventional EA operators. The decomposition of the complete solution is expected to make the algorithm more efficient on high dimensional problems.

The second EA is CMA-ES, which is a well-known state-of-the-art optimisation algorithm. It translates the selection, recombination and mutation processes into the updating of an adaptive covariance matrix, which is thereafter used to sample new solutions for the next generation.

# Chapter 3

# Resonating Arm System

In this chapter the *Resonating Arm* (RA) will be discussed by explaining its purpose, background and working principles. Thereafter the mathematical model of the RA is derived, and the friction model and simulation method are discussed.

## 3-1 Purpose and Background

This research is part of the 'Resonating Arm Project' which concerns the development of robots arms that have the design and skills to perform their tasks in a natural dynamic manner. This is achieved by creating a natural oscillatory motion through a smart combination of springs, mechanisms and actuators. In this section we will discuss the purpose of the RA and how this research builds upon previous results achieved in this project.

### Purpose

The Resonating Arm is a novel concept for a robot arm with low-power actuators designed for pick-and-place tasks. These tasks are in general highly repetitive which favours the use of robots above human workers. Therefore robot arms are already widely being used to perform pick-and-place tasks in numerous industries (e.g. the food handling industry). These robot arms are typically equipped with high-power actuators to meet the fast handling speeds required by the industry. The resonating arm is being developed to reducing this need for powerful actuators while being fast enough to be used in an industrial environment.

The reduction of actuation power has a positive influence on many aspect. One of those aspects is a possible increase in energy efficiency due to the fact that low-power actuators demand less energy. Another aspect is safety which can be improved by the decreased maximum torque the actuators are able to generate. Furthermore the cost and mass of the robot could be reduced by the implementation of low-power actuators, which are generally cheaper and lighter.

In this research the reduction of actuation power is translated into the minimisation of the maximum torque needed to perform the demanded pick-and-place tasks within a limited time frame. The minimisation of this maximum torque enables the use of less powerful actuators in the final design of the robot arm.

#### Background

The development of the Resonating Arm started in early 2011 when a research group was formed by dr.ir. Martijn Wisse at the Delft University of Technology (TU Delft). The first step in the development of the RA was taken by ir. Michiel Plooij who designed and built the first RA prototype during his graduation project [26]. This research builds upon Plooij's graduation work and uses the morphological structure of the prototype to investigate whether evolutionary algorithms could help to improve the system parameter values and optimise a minimal torque controller.

## 3-2 Working Principles

The reduction of actuation power without a loss of handling speed can only be done by designing a clever mechanism that does not rely on actuation power to generate the high accelerations needed, the RA is such a mechanism. Its working principles will be explained in this section by presenting the basic layout and thereafter the most important part of the RA; the spring mechanism.

### 3-2-1 Basic Layout

The basic layout of the RA is equal to a SCARA type robot [27], it has a parallel-axis joint layout in which the arms can move in the X-Y plane but are rigid in the Z-direction. Only the end effector located at the end of the robot arm is able to move up and down in the Z-direction allowing movement in all three dimensions.

In Figure 3-1 a drawing of the RA is presented and the most important parts of the system are indicated. One of the parts is the spring which is attached to a small pulley and a large pulley. A timing belt connects the two pulleys and creates a ratio between the angular displacement of both pulleys. This timing belt together with the two pulleys and the spring is called the spring mechanism and forms the core of the RA. Attached to the large pulley is the upper arm and the lower arm is connected to the upper arm. To position the upper arm a motor is connected to the large pulley, the lower arm is actuated by a second motor which is placed at the pivot point between both arms. The end effector is not shown but would be located at the end of the lower arm.

In the proceeding sections we will investigate the spring mechanism which is one of the most important parts of the RA.

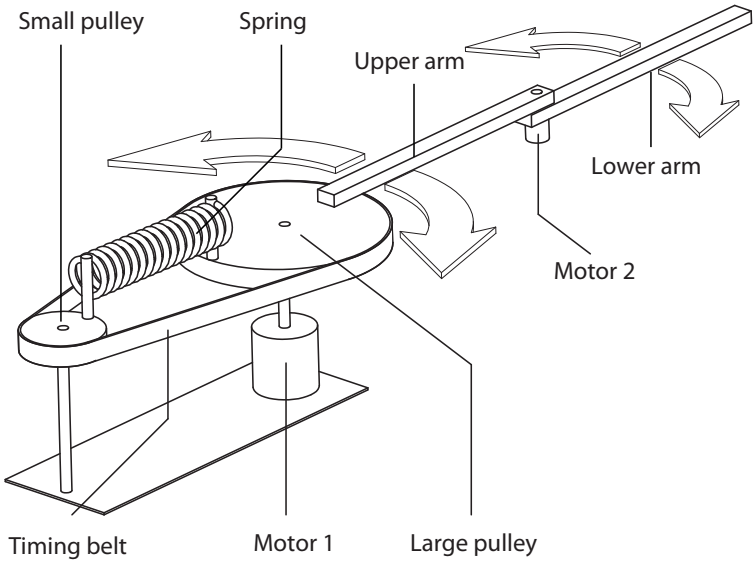



**Figure 3-1:** A drawing of the controlled mechanical system, called the Resonating Arm, which will be optimised in this research. An important part of the Resonating Arm is its spring mechanism, which is represented by the two pulleys, the spring and a timing belt connecting them. Furthermore the systems consists of an upper and lower arm plus two actuators to control both degrees of freedom. (source: [26])

### 3-2-2 Spring Mechanism

The spring mechanism is responsible for the high accelerations in the system needed to perform the pick-and-place tasks quick enough. The high accelerations cannot be generated by the low-power actuators since they are too weak to generate the required torques. A RA with only low-power actuators and no spring mechanism would be very slow and therefore useless for industrial applications.

The basic idea behind the use of a spring is the possibility of storing and transforming energy from potential energy into the kinetic energy and vise versa. The potential energy stored in the spring can be used to accelerate the upper arm by transforming it into kinetic energy and deceleration is done by transforming kinetic energy back into potential energy. The spring is stretched and potential energy is stored when the upper arm is moved away from its rest position (in one line with the spring). When the upper arm is released this potential energy will cause the upper arm to move in the direction of the rest position at which the potential energy level is the lowest. When the arm accelerates it will pass the rest position at full speed and from that moment on the spring mechanism will start to decelerate the arm by storing potential energy.

The elongation of the spring is correlated with the potential energy stored in the spring and is a non-linear function of the arm's position. Since the angular displacement of the small pulley depends on the angular displacement of the large pulley it is possible to derive the functions describing the elongation and the stored potential energy given the angular displacement of the large pulley. This angular displacement of the large pulley is equivalent to the angular position of the upper arm.

The relation between the potential energy stored in the spring and the angular position of the upper arm can be derived by determining the relation between spring elongation and angular position of the upper arm. In Figure 3-2 the spring mechanism is depicted together with the system parameters influencing this relation.

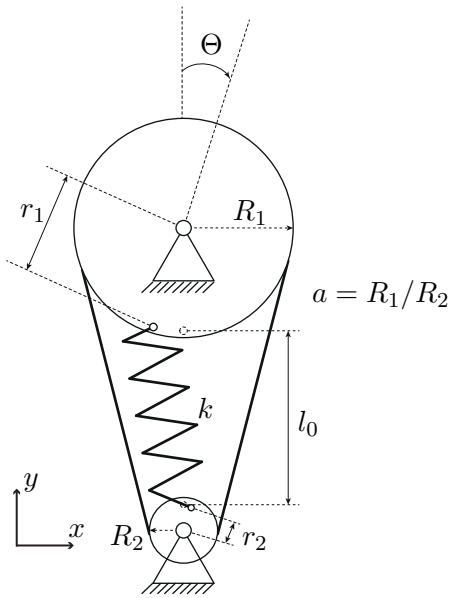



**Figure 3-2:** The spring mechanism driving the Resonating Arm. This mechanism consists of two pulleys interconnected by a timing belt and a spring. The most important variables describing this mechanism are the angular position of the large pulley $\Theta$ and the system parameters $l_0$ $r_1$, $r_2$, $a$ and $k$, respectively the original length of the spring, the radii at which the spring ends are attached to the large and small pulley, the ratio between the outer radii of the pulleys ($R_1/R_2$) and the spring constant.

This relation is described by the difference between the spring length $l(\Theta)$ $(m)$ at angle $\Theta$ $(rad)$ and the original spring length $l_0$ $(m)$ at its rest position $\Theta = 0$,

$$\Delta l(\Theta) = l(\Theta) - l_0$$

with $l(\Theta)$ depending on the system parameters $r_1$, $r_2$, $a$ and $l_0$ as follows,

$$l(\Theta) = \sqrt{l_x(\Theta)^2 + l_y(\Theta)^2} \tag{3-1}$$

$$\tag{3-2}$$

$$l_x(\Theta) = r_1 \sin(\Theta) + r_2 \sin(a\,\Theta) \tag{3-3}$$

$$l_y(\Theta) = l_0 + r_1 - r_1 \cos(\Theta) + r_2 - r_2 \cos(a\,\Theta)\,. \tag{3-4}$$

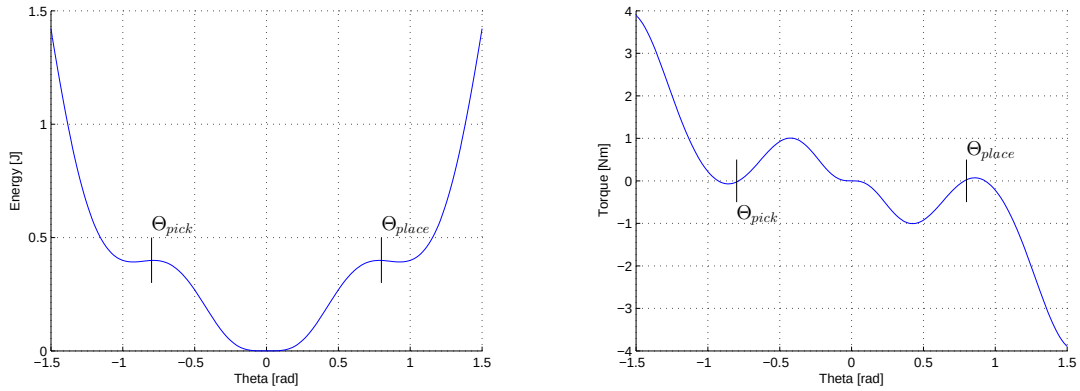
In order to calculate the potential energy stored in the spring we use Hooke's law $F = -k \cdot \Delta l$ and approximate the elasticity or spring force by stating that the elongation of a spring is in direct proportion with the load applied to it by a constant $k$ $(N/m)$, called the spring constant. As long as the material's elastic limit is not exceeded this relation can be generally applied to most springs.

After integrating the spring force $F$ over the spring elongation $\Delta l$ we are finally able to derive the relation between the potential energy $E_p$ and the angle $\Theta$ of the upper arm as shown below,

$$\begin{aligned} E_p &= -\int F \cdot d\Delta l = -\int -k \cdot \Delta l \cdot d\Delta l \\ &= \frac{1}{2} \cdot k \cdot \Delta l^2 \\ &= \frac{1}{2} k \left( l_0 - \sqrt{\begin{array}{r} (l_0 + r_1 + r_2 - r_2 \cos(a\,\Theta) - r_1 \cos(\Theta))^2 + \dots \\ (r_2 \sin(a\,\Theta) + r_1 \sin(\Theta))^2 \end{array}} \right)^2 \end{aligned} \tag{3-5}$$

where $k$ is the spring constant and $r_1$, $r_2$, $a$ and $l_0$ the variables defining the morphology of the spring mechanism. In this relation we assume that the original spring length is equal to the natural length of the spring, in reality this might not be the case but this will only increase the total potential energy by a constant value and does not have any effect on the dynamic behaviour of the spring mechanism.

In Figure 3-3(a) the relation between the potential energy and the angle (as derived in 3-5) is plotted from $-1.5$ to $+1.5$ radians. the system parameter values used correspond with the system parameter values of the first prototype, namely: $r_1 = 0.1\ m$, $r_2 = 0.02\ m$, $a = 5\ m/m$, $l_0 = 0.1\ m$ and $k = 150\ N/m$. Also the desired pick angle $\Theta_{pick} = -0.8\ (rad)$ and desired place angle $\Theta_{place} = 0.8\ (rad)$ of the prototype are indicated.




(a) Potential energy curve

(b) Torque curve

**Figure 3-3:** Potential energy (a) and torque (b) as a function of angle $\Theta$ of the upper arm for morphological values. The spring mechanism in the Resonating Arm allows the existence of flat plateaus in the potential energy curve (a) near the desired pick angle $\Theta_{pick} = -0.8$ and desired place angle $\Theta_{place} = 0.8$ at which energy is stored in the system but the derivative of the potential energy curve is close to zero. This implies that at these angles the torque applied by the spring mechanism is close to zero as visible in (b), this is ideal when the system has to be kept at these positions using a low-power actuator, which maximum torque is limited.

In this plot two specific characteristics of the spring mechanism can be observed. First we see the expected behaviour that the potential energy increases when the system moves away

from the rest position at $\Theta = 0$ due to the spring elongation. But more important are the two plateaus in the potential energy curve at which the change in potential energy is relative small compared to angles outside these plateaus. At these angles the derivative of potential energy $E_p$ over the angle $\Theta$ is close to zero and this equals spring mechanisms torque $\tau_s$ applied on the upper arm as shown below,

$$E_p = \int -\tau_s \cdot d\Theta$$
$$\frac{\partial}{\partial \Theta} E_p = \frac{\partial}{\partial \Theta} \int -\tau_s \cdot \partial\Theta$$
$$\tau_s = -\frac{\partial E_p}{\partial \Theta}$$

the equation describing the torque $\tau_s$ is given in Appendix C.

In Figure 3-3(b) the negative derivative of the potential energy curve is plotted as a function of the angle $\Theta$, this equals the torque applied by the spring mechanism on the upper arm. In this plot we can verify that the torque applied to the upper arm is approximately zero at the pick and place angles, this gives the system the possibility to story energy at these positions and stay at them without having to use a strong actuator torque to counteract the spring force. This property is extremely useful for the RA since it enables the system to stay at the pick or place positions until a product is fully grabbed or placed and use the stored energy to quickly accelerate the system in the direction of the other pick or place position.

## 3-3 Model

A model of the Resonating Arm is needed to simulate the many different control and system parameter solutions. In this section the assumptions which have been made to limit the complexity of the model are presented together with parameters used in the model. Thereafter the equations of motion are derived using the TMT-method and the friction model is presented.

### 3-3-1 Assumptions and Model Parameters

#### Assumptions

The following assumptions have been made in order to construct a model of the Resonating Arm with manageable complexity:

- The arms of the robot can be described as straight rods with infinite stiffness.
- The spring mechanism can be described as an inertia and an applied torque around the pivot point of the large pulley.
- The spring used in the system has a linear spring characteristic and the elongation is zero when the angle of the upper arm equals zero.
- The timing belt has no elastic properties and does not slip around the pulleys.

- The total friction in the system can be modelled by the total sum of a coulomb friction model and a viscous friction model.
- The friction coefficients in both joints are equal to the estimated friction coefficients of the Resonating Arm prototype.
- The friction coefficients do not vary when the forces on the pivot points changes.
- The controller has a direct influence on the torques applied by the actuators.
- The mass of the end effector (e.g. gripper with product) is 1 kg.

#### Model Parameters

Based on the assumptions, the Resonating Arm can be modelled as a double pendulum with actuation on both pivot points. The configuration of the system can be described by two generalised coordinates, namely the angles $\Theta_1$ and $\Theta_2$ as shown in Figure 3-4(a).

In Figure 3-4(b) 11 parameters are presented which define the masses, position of the centers of mass and the inertia acting on the double pendulum model.

The two arms with lengths $L_1$ and $L_2$ are modelled as straight rods with uniform distributed masses $m_1$ and $m_2$, negligible thickness and their centers of mass located at $c_1 = \frac{1}{2}L_1$ and $c_2 = \frac{1}{2}L_2$. The inertia of the upper arm ($I_1$) and the lower arm ($I_2$) around their center of mass has been determined by

$$I_i = \frac{1}{12} m_i L_i^2 \quad i = 1, 2\,. \tag{3-6}$$

The mass of the pivot mechanism ($m_a$) and the end effector ($m_b$) are seen as point masses with a moment of inertia equal to zero. The variable $I_0$ present the inertia of the spring mechanism around the pivot point of the large pulley.

Together with the 5 parameters which define the spring mechanism (Figure 3-2) the total amount of parameters sums up to 16 morphological parameters as summarised in Table 3-1.

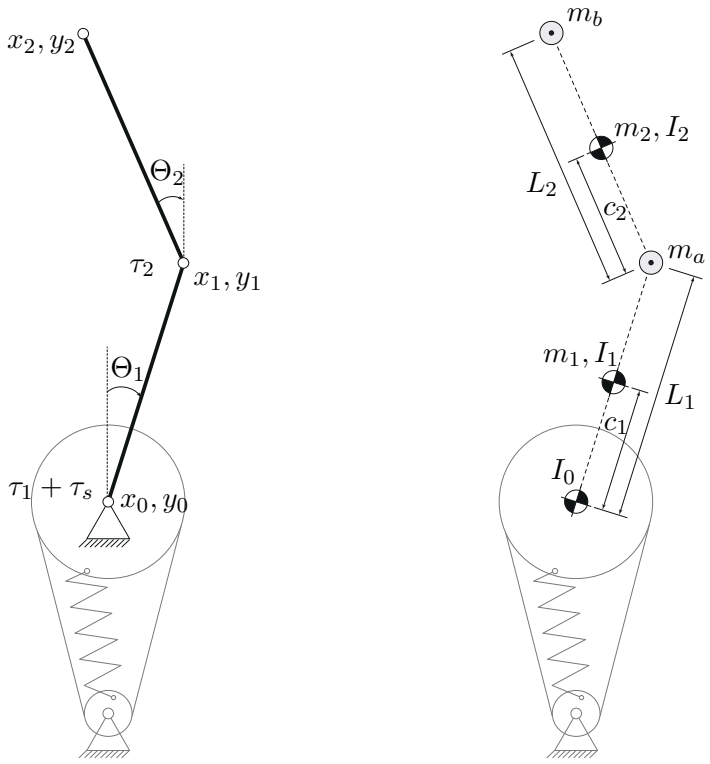
In order to focus the optimisation on the complex parameter interaction of the spring mechanism, which have the most substantial influence on the dynamics of the system, the parameters describing the arms are kept constant during the optimisation. In Table 3-1 the values at which the arm parameters will be set are presented, these values are chosen equal to the parameter values used in the current prototype.

### 3-3-2 Equations of Motion

The equations of motion describe the dynamics of the system as a set of differential equations. The differential equations describing the RA have been derived by applying the TMT-method [28], this method is explained in Appendix A.

The TMT-method derives the equations of motion as a set of acceleration equations in the form

$$\bar{\mathbf{M}}\ddot{\mathbf{q}} = \bar{\mathbf{f}} \tag{3-7}$$




(a) State variables and torques (b) Center of masses and inertia

**Figure 3-4:** Top view of the Resonating Arm model with in (a) the two degrees of freedom of the system $\Theta_1$ and $\Theta_2$ and the torques acting on the joints with $\tau_1$ and $\tau_2$ the actuator torques and $\tau_s$ the torque caused by the spring mechanism. In (b) the position of the center of masses and inertia are presented. The masses and inertia $m_1$, $I_1$ and $m_2$, $I_2$ represent the arm rods which are assumed to have uniform distributed masses. Parameters $m_a$ and $m_b$ represent the pivot mechanism and the end effector respectively, both are assumed to be point masses. The inertia $I_0$ is the inertia caused by the spring mechanism.

where $\ddot{\mathbf{q}}$ holds the independent generalised accelerations and matrices $\bar{\mathbf{M}}$ and $\bar{\mathbf{f}}$ represent the generalised mass matrix and the generalised force vector, respectively.

The matrices $\bar{\mathbf{M}}$ and $\bar{\mathbf{f}}$ are constructed by transforming the 'normal' Newton-Euler equation matrices $\mathbf{M}$ and $\mathbf{f}$ as follows

$$\bar{\mathbf{M}} = \mathbf{T}^T \mathbf{M} \mathbf{T} \,, \tag{3-8}$$

$$\bar{\mathbf{f}} = \mathbf{T}^T [\Sigma \mathbf{f} - \mathbf{M}\mathbf{h}] \,, \tag{3-9}$$

with the transformation matrix $\mathbf{T}$ and vector $\mathbf{h}$ holding the convective accelerations.

The RA system can be defined by two independent variables; the angle of the upper arm $\Theta_1$ and the angle of the lower arm $\Theta_2$. These variables are called independent generalised

**Table 3-1:** Parameters for the Resonating Arm system corresponding to Figure 3-4 and 3-2. The given parameter values indicate parameters which will not be optimised and will be predefined at a value equal to that of the current prototype. The parameters without a given values will be subject to optimisation and are able to vary during the optimisation process in order to find their optimal values.

| | | | |
|---|---|---|---|
| **Spring mechanism** | | | |
| Radius at which the spring is connected to the large pulley | $r_1$ | - | $m$ |
| Radius at which the spring is connected to the small pulley | $r_2$ | - | $m$ |
| Ratio between the outer radii of the pulleys ($R_1/R_2$) | $a$ | - | $m/m$ |
| Natural length of the spring | $l_0$ | - | $m$ |
| Spring constant | $k$ | - | $N/m$ |
| Momemt of inertia of spring mechanism | $I_0$ | - | $kgm^2$ |
| **Arms** | | | |
| Arm lengths | $L_1$, $L_2$ | 0.4 | $m$ |
| Arm masses | $m_1$, $m_1$ | 0.2 | $kg$ |
| C.o.m. location of arms | $c_1$, $c_2$ | 0.2 | $m$ |
| Moment of inertia of arms | $I_1$, $I_2$ | $5 \cdot 10^{-4}$ | $kgm^2$ |
| Pivot mass | $m_a$ | 0.2 | $kg$ |
| End effector mass | $m_b$ | 1.0 | $kg$ |

coordinates and are written as a vector $\mathbf{q}$

$$\mathbf{q} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \end{bmatrix}. \tag{3-10}$$

The 'normal' mass matrix $\mathbf{M}$ for the RA system is a square matrix with all masses and inertia on the diagonal

$$\mathbf{M} = \begin{bmatrix} I_0 + I_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & m_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_b & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m_b \end{bmatrix}. \tag{3-11}$$

The 'normal' force vector $\mathbf{f}$ defines all the torques and forces acting on the masses and inertia present in the mass matrix, in our case these are the control torques on the upper arm ($\tau_1$), the lower arm ($\tau_2$) and the torque from the spring mechanism ($\tau_s$) also acting on the upper

arm

$$\mathbf{f} = \begin{bmatrix} \tau_1 + \tau_s \\ 0 \\ 0 \\ 0 \\ 0 \\ \tau_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3-12}$$

The torque $\tau_s$ applied by the spring mechanism is derived by taking the partial derivative of the potential energy $E_p$ equation (3-5) with respect to the generalised coordinate $\Theta_1$

$$\tau_s = -\frac{\partial E_p}{\partial \Theta_1}. \tag{3-13}$$

The mass matrix $\mathbf{M}$ and force vector $\mathbf{f}$ can now be transformed by a matrix $\mathbf{T}$ which equals the partial derivative of the transformation matrix $\mathbf{T}^*$ with respect to $\mathbf{q}$. This transformation matrix $\mathbf{T}^*$ expresses the positions and orientations of the centre of masses as a function of the generalised coordinates

$$\mathbf{T}^*(\mathbf{q}) = \begin{bmatrix} \Theta_1 \\ x_{m1} \\ y_{m1} \\ x_{ma} \\ y_{ma} \\ \Theta_2 \\ x_{m2} \\ y_{m2} \\ x_{mb} \\ y_{mb} \end{bmatrix} = \begin{bmatrix} \Theta_1 \\ c_1 \sin\Theta_1 \\ c_1 \cos\Theta_1 \\ L_1 \sin\Theta_1 \\ L_1 \cos\Theta_1 \\ \Theta_2 \\ L_1 \sin\Theta_1 + c_2 \sin\Theta_2 \\ L_1 \cos\Theta_1 + c_2 \cos\Theta_2 \\ L_1 \sin\Theta_1 + L_2 \sin\Theta_2 \\ L_1 \cos\Theta_1 + L_2 \cos\Theta_2 \end{bmatrix} \tag{3-14}$$

and by taking the derivative with respect to $\mathbf{q}$ we are able to find transformation matrix $\mathbf{T}$

$$\mathbf{T} = \frac{\partial \mathbf{T}^*}{\partial \mathbf{q}}. \tag{3-15}$$

The convective accelerations matrix $\mathbf{h}$ is defined as

$$\mathbf{h} = \frac{\partial \mathbf{T}\dot{\mathbf{q}}}{\partial \mathbf{q}}\dot{\mathbf{q}} \tag{3-16}$$

Now all necessary matrices are constructed we are able to construct the generalised mass matrix $\bar{\mathbf{M}}$ and the generalised force vector $\bar{\mathbf{f}}$ as explained at the start;

$$\bar{\mathbf{M}} = \mathbf{T}^T \mathbf{M} \mathbf{T} \tag{3-17}$$

$$\bar{\mathbf{f}} = \mathbf{T}^T [\Sigma \mathbf{f} - \mathbf{M}\mathbf{h}] \tag{3-18}$$

The equations of motion are now found and can be solved for $\ddot{\mathbf{q}}$ as shown below,

$$\bar{\mathbf{M}}\ddot{\mathbf{q}} = \bar{\mathbf{f}} \tag{3-19}$$

$$\bar{\mathbf{M}}^{-1}\bar{\mathbf{M}}\ddot{\mathbf{q}} = \bar{\mathbf{M}}^{-1}\bar{\mathbf{M}}^{-1}\bar{\mathbf{f}} \tag{3-20}$$

$$\ddot{\mathbf{q}} = \bar{\mathbf{M}}^{-1}\bar{\mathbf{f}}. \tag{3-21}$$

The complete equation of motion can be found in Appendix B.

### 3-3-3 Friction Model

The friction torque $\tau_f$ in the arm joints is separately modelled as a combination of coulomb friction $\tau_c$ and viscous friction $\tau_v$

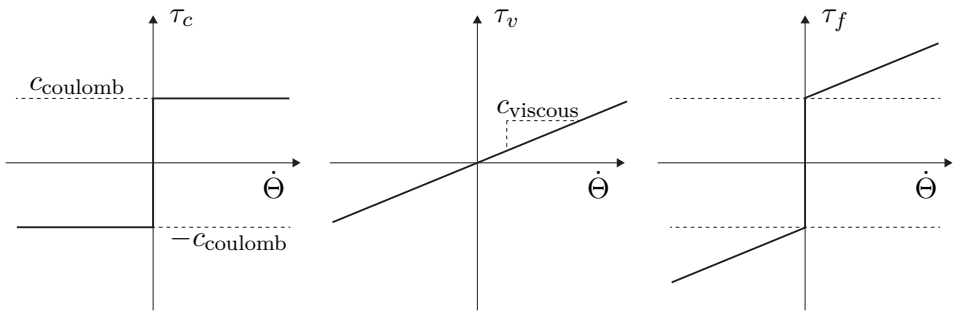
$$\tau_f = \tau_c + \tau_v \tag{3-22}$$

with

$$\tau_c = \begin{cases} \text{sign}(\dot{\Theta}) \cdot c_{\text{coulomb}} & \text{if } \dot{\Theta} \neq 0 \\ \text{sign}(\tau_{\text{ext}}) \cdot c_{\text{coulomb}} & \text{if } \dot{\Theta} = 0 \text{ and } |\tau_{\text{ext}}| > c_{\text{coulomb}} \\ \tau_{\text{ext}} & \text{if } \dot{\Theta} = 0 \text{ and } |\tau_{\text{ext}}| \leq c_{\text{coulomb}} \end{cases} \tag{3-23}$$

$$\tau_v = c_{\text{viscous}} \cdot \dot{\Theta} \tag{3-24}$$

The coulomb friction is a static phenomena and equals the coulomb friction coefficient $c_{\text{coulomb}}$ when the velocity $\dot{\Theta}$ is not equal to zero and its sign is depending on the sign of the velocity. When the velocity is zero and the applied external torque $\tau_{\text{ext}}$ is bigger than $c_{\text{coulomb}}$ then the friction torque will be equal to the coulomb friction coefficient with its sign equal to the of the external torque. In case the velocity is zero and the external torque is smaller than the friction coefficient then friction will be equal to the external torque. The viscous friction is a dynamic friction which is determined by the viscous friction coefficient $c_{\text{viscous}}$ multiplied by the velocity. The sum of these friction torques defines the friction model as shown in Figure 3-5.




(a) Coulomb friction (b) Viscous friction (c) Total friction

**Figure 3-5:** The friction in the Resonating Arm is modelled as a static coulomb friction (a) with friction coefficient $c_{\text{coulomb}}$ and a dynamic viscous friction (b) with friction coefficient $c_{\text{viscous}}$. The coulomb friction is a constant friction independent of the velocity $\dot{\Theta}$, while the viscous friction is depending on the velocity $\dot{\Theta}$. The total friction (c) is the sum of both friction types.

It is difficult to determine the right values for the friction coefficients in the friction model since the actual values of the robot will be influenced by many different aspects (e.g. the type of bearings or the lubrication used). Therefore the friction coefficients used in this model are equal to the estimated friction coefficients of the current prototype as presented in Table 3-2. From these friction coefficients it can be seen that the coulomb friction has the biggest influence on the total friction in the system and that the influence of the viscous friction is almost neglectable. Still the viscous friction is incorporated in the model to provide a more comprehensive friction model for future optimisations with different friction coefficients.

| **Friction type** | **Friction coefficient** |
|---|---|
| Coulomb friction | 0.48 $Nm$ |
| Viscous friction | 0.01 $Nm/(rad/s)$ |

**Table 3-2:** Coefficients of friction used in the friction model.

## 3-4 Summary

The Resonating Arm (RA) is a novel concept for a robot arm with low-power actuators designed for pick-and-place tasks. It has SCARA type configuration with two arms that are able to move in a horizontal plane. A special spring mechanism causes a nonlinear natural behaviour of the system in which potential energy is stored at the pick and place positions. This stored energy is used to perform the accelerations and decelerations of the arm, which liberates the controller from this task and permits the use of smaller actuator torques.

A model of the RA has been constructed by deriving the equations of motion and adding a friction model of coulomb and viscous friction.

# Chapter 4

# Fuzzy Control

In this research fuzzy control has been used to generate a feedback control behaviour for the RA. In this chapter we will shortly introduce fuzzy logic, explain why fuzzy control has been used and briefly explain the basic principles of fuzzy control. In the last section we will present two fuzzy controllers which will be used in this research and discuss their properties and expected advantages and disadvantages.

## 4-1 Introduction

Fuzzy control is a common type of knowledge-based control in which the control laws, corresponding to particular conditions of the system, are described in terms of **if-then** rules. It makes use of the fuzzy set theory, introduced by Zadeh in 1965 [29], in which sets are defined by its elements and their degree of membership.

Fuzzy control was first successfully applied by Mamdani and Assilian in 1975 [30] and developed to describe the control heuristics from human operators, expressed in (fuzzy) linguistic input and output terms (e.g. 'big' and 'small'), into a form usable for automated control. Later in 1985 Takagi and Sugeno [31] recognised that these fuzzy terms could also be used to describe a nonlinear control process in terms of interacting linear subcontrollers acting locally on a state of the system. This latter type of fuzzy controller will be used in this research.

The combination of fuzzy control and reinforcement learning techniques became more and more popular due to the fact that the tuning of fuzzy controllers appeared to be a complex and tedious task. Also the use of evolutionary algorithms to tune fuzzy controllers has been frequently applied and has been published in numerous papers [32–51].

## 4-2 Why Fuzzy Control?

In order to control the Resonating Arm (RA) system we require a nonlinear feedback controller. Nonlinear since the RA is a highly nonlinear system for which linear control approaches

will not be able to generate an optimal control. And feedback is required to control the system from different starting positions.

To control the system optimal a nonlinear function $f$ has to be found

$$\mathbf{u} = f(\mathbf{x}) \tag{4-1}$$

which maps the current state inputs $\mathbf{x}$ to the optimal control outputs $\mathbf{u}$.

One approach to find the optimal relation between state inputs and control outputs is by performing an online optimisation of the control output based on the current states and the predicted states of a model. This type of control, called model predictive control has been effectively used for relatively slow systems, but is less effective when the system is quick and the CPU budget for optimisation and prediction is small [52]. Since the RA is a relatively fast system (with angular velocities up to 4 rad/s) online techniques are less practical and we will thus have to rely on static mappings which are found offline.

Neural networks and fuzzy control are two often used techniques to provide a nonlinear mapping between states and control output. Both of them are tuned offline and belong to the class of universal approximators, meaning that they are able to approximate any input-output mapping [53, 54].

Neural networks are inspired by the way our brains process information and create an input-output mapping through interconnecting artificial neurons. Due to the complex interaction between the artificial neurons neural networks are considered black-box controllers, which means that it is considered difficult to extract general control rules from them or incorporate general rules into them.

Fuzzy control does not have this problem since it is based on human reasoning and uses logic rules to map inputs to outputs. This makes it much easier to understand the control behaviour and to distill general control rules. Moreover, it becomes possible to incorporate human knowledge into the controller. When human knowledge is available this could be used to initialise the optimisation and increase the speed of convergence.

The fact that fuzzy control is a nonlinear feedback controller which uses a static input-output mapping, together with the advantages of extracting and incorporating human knowledge makes fuzzy control a suitable and convenient type of control to be used in the RA system.

## 4-3 Basic Concepts

In Figure 4-1 the general structure of a fuzzy controller is presented, which basically consists of a rule base, an inferencing mechanism and two input/output interfaces that scale, fuzzify and defuzzify. In this section we will discuss each of these elements.

### 4-3-1 Scaling and Fuzzification

Fuzzification is the process of transforming crisp input variables into fuzzy output variables using fuzzy membership functions. Before this fuzzification process is applied, the inputs of the controller are often scaled into a more convenient interval between -1 and 1.

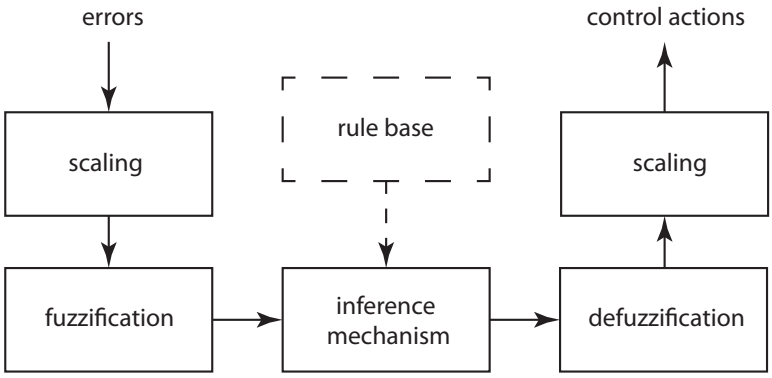



**Figure 4-1:** General fuzzy control structure.

In order to understand the fuzzification process we first need to define what fuzzy variables are and how they are defined by fuzzy membership functions. Fuzzy variables differ from 'normal' variables in the way that they are able to represent a vague set of values. One can compare this to the way humans use a vague definition (e.g. 'very high') when they want to express the vague set of room temperatures they consider to be uncomfortable.

In Figure 4-2 an example is given of fuzzy variables defining different room temperature intervals. As mentioned before fuzzy variables do not describe a precise set of parameter values, but instead make use of so called membership degrees to define how much a parameter value is part of that fuzzy variable. These membership degrees can be determined by evaluating the membership functions of the fuzzy variables which give a value ranging from 0 *'not a member'* to 1 *'fully a member'*. In the room temperature example these membership functions are described by the trapezoidals plotted below the different fuzzy variables.

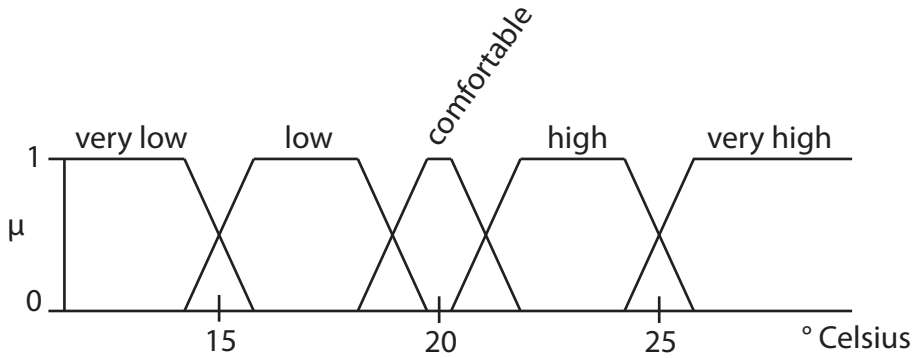



**Figure 4-2:** Five different linguistic variables or fuzzy variables covering different room temperature intervals. Each fuzzy variable is described by a membership function $\mu(\cdot)$ that defines the degree (0 to 1) to which a certain temperature belongs to that fuzzy variable, where 0 corresponds with *not a member* and 1 with *fully a member*. In this example the membership functions are trapezoids and the sum of membership degree is always 1, however, this is not necessarily true for all fuzzy controllers.

### 4-3-2 Rule Base and Inference

The rule base defines the input-output behaviour of the fuzzy controller. In this research a zero order Takagi-Sugeno type fuzzy controller is used which means that the consequences of the rules are defined as singletons (constants) and a rule $R_r$ is described by the following form:

$$\begin{aligned} R_r : \textbf{IF } & x_1 \text{ is } A_r^1 \textbf{ and } \ldots \textbf{ and } x_k \text{ is } A_r^k \\ \textbf{THEN } & u_r^1 = \omega_r^1 \textbf{ and } \ldots \textbf{ and } u_r^l = \omega_r^l \end{aligned}$$

where $x_i$ represent the inputs for $i = 1, \ldots, k$, $A_r^i$ the fuzzy variables on the domains of these inputs and $u_r^j$ the control outputs for $j = 1, \ldots, l$ with $\omega_r^j$ the corresponding constant consequent variables.

In the inference process the truth value of each rule is determined using the membership degrees obtained from the fuzzification process. Many different types of inference mechanisms exist but an often used method in combination with Takagi-Sugeno controllers is the *product-sum* method, also called the *weighted average*, which is defined by

$$\mathbf{u} = \frac{\sum_{r=1}^{m} \mu(x_r) \cdot \boldsymbol{\omega}_r}{\sum_{r=1}^{m} \mu(x_r)} \tag{4-2}$$

with

$$\mu(x_r) = \Pi_{i=1}^{k} \mu_{A_r^i}(x_i) \tag{4-3}$$

in which the output $\mathbf{u} = [u_1, \ldots, u_l]$ is determined by multiplying rule truth values $\mu(x_r)$ with the rule consequent part $\boldsymbol{\omega}_r = [\omega_1, \ldots, \omega_l]$, taking the sum and dividing this value by the sum of all rule truth values. The truth value of a rule $\mu(x_r)$ is in this case determined by multiplying all the different membership degrees $\mu_{A_r^i}(x_i)$ of the fuzzy variables $A^i$ with $i = 1, \ldots, k$ in rule $r$.

### 4-3-3 Defuzzification and Scaling

Defuzzification is used to transform a fuzzy output into a crisp value, this is needed when the outputs of the rules are defined as fuzzy variables, which is the case in Mamdani controllers. Since in this research Takagi-Sugeno controllers with crisp consequents are used this defuzzification step is not needed. The only step that remains is scaling the output of the inference mechanism onto the working domain of the control output.

## 4-4 Design of Two Fuzzy Control Variations

Two different fuzzy controllers denoted by *fixed fuzzy controller* and *free fuzzy controller* have been designed and tested. In this section we will discuss the properties of these controllers by presenting their inputs and outputs, the difference in the positioning of the membership functions and their expected advantages and disadvantages.

### 4-4-1 Inputs, Outputs and Scaling

The controllers used in this research are fuzzy variants of the linear proportional-derivative (PD) controllers used in many control applications, which determine their control output $\mathbf{u}$ based on the error $\mathbf{e}$ and the error derivative $\dot{\mathbf{e}}$

$$\mathbf{u} = f(\mathbf{x}) \tag{4-4}$$

with

$$\mathbf{x} = \begin{bmatrix} \mathbf{e} \\ \dot{\mathbf{e}} \end{bmatrix} \tag{4-5}$$

where in fuzzy control the function $f(\cdot)$ will be determined by the fuzzy rules and membership functions.

The number of inputs of the controllers used to control the 2-DOF RA is equal to four, defining the errors between the desired angular positions and velocity of both arms as

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \Theta_{1d} - \Theta_1 \\ \Theta_{2d} - \Theta_2 \end{bmatrix} \tag{4-6}$$

$$\dot{\mathbf{e}} = \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} \dot{\Theta}_{1d} - \dot{\Theta}_1 \\ \dot{\Theta}_{2d} - \dot{\Theta}_2 \end{bmatrix} \tag{4-7}$$

where $\Theta_{1d}$, $\Theta_{2d}$, $\dot{\Theta}_{1d}$ and $\dot{\Theta}_{2d}$ represent the desired angles and angular velocities, respectively.

The output $\mathbf{u}$ is defined as

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{4-8}$$

with $\tau_1$ and $\tau_2$ the control torques acting on the upper and lower arm.

A reduced version of this controller is used to control the 1-DOF systems considered in the first and second experiment. For these system the control input is defined by $e_1$ and $\dot{e}_1$, and the control output by $u_1$.

Before fuzzification the $\mathbf{e}$ and $\dot{\mathbf{e}}$ are scaled by a factor between 1/3 and 3. This scaling factor is used to bring the real inputs into the domain used by the fuzzy controller (between -1 and 1), but also provides the optimisation algorithm with a variable that has a global influence on the relative widths of all membership functions. The same holds for the outputs which are also scaled by a factor between 1/3 and 3.

### 4-4-2 Membership Functions

The membership functions are defined by Gaussian functions (Figure 4-3) which determine the membership degree $\mu$ of a one-dimensional input variable $x$ for fuzzy variable $A^i$ as

$$\mu_{A^i}(x) = \exp\left(-\frac{(c^i - x)^2}{2(\sigma^i)^2}\right), \tag{4-9}$$

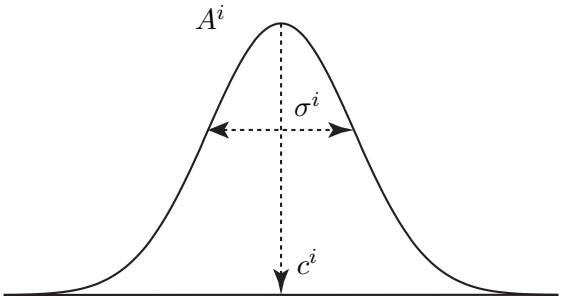



**Figure 4-3:** Gaussian function of the fuzzy term $A^i$ on the $i$th input defined by its center $c^i$ and its width $\sigma^i$.

where $c^i$ and $\sigma^i$ are the centre and width of the Gaussian curve, respectively.

Gaussian functions are used since it has been shown that these yield the best approximation property [55, 56]. Also for optimisation purposes it has been expected to be beneficial since Gaussian functions have a non-zero output for all outputs meaning that each rule will have some influence on the control output of each state in the state space.

In order to limit the size of the search space, both the values for the centers and widths of the membership functions have been limited. The centers of the membership functions are bounded between -1 and 1 relative to the scaled domains of the inputs. And the widths are bounded between 0.01 and 1, which allows the membership functions to cover a relative small or large part of the scaled input domain.

### 4-4-3 Fixed vs Free Fuzzy Control

The *fixed fuzzy controller* and *free fuzzy controller* have been constructed to investigate the trade off between the approximation capabilities of the controller and the complexity of optimising its parameters. The fixed fuzzy controller has a lower complexity than the free fuzzy controller at the cost of approximation capabilities.

#### Fixed Fuzzy Control

The fixed fuzzy controller has its membership functions evenly distributed at fixed positions on the interval [-1 1]. This facilitates a grid partitioning of the state space as shown in Figure 4-4 at which each crosspoint corresponds with a consequent of one of the rules in the rule base. This means that membership functions are shared between the different rules and that the total number of rules is equal to the number of combinations possible when considering the different membership functions defined on the different inputs.

The $N$ rules in the rule base of the fixed fuzzy controller have the following form

$$\begin{aligned} R_r : &\textbf{IF } x_1 \text{ is } A^1 \textbf{ and } \ldots \textbf{ and } x_k \text{ is } A^k \\ &\textbf{THEN } \mathbf{u}_r = \boldsymbol{\omega}_r \end{aligned}$$

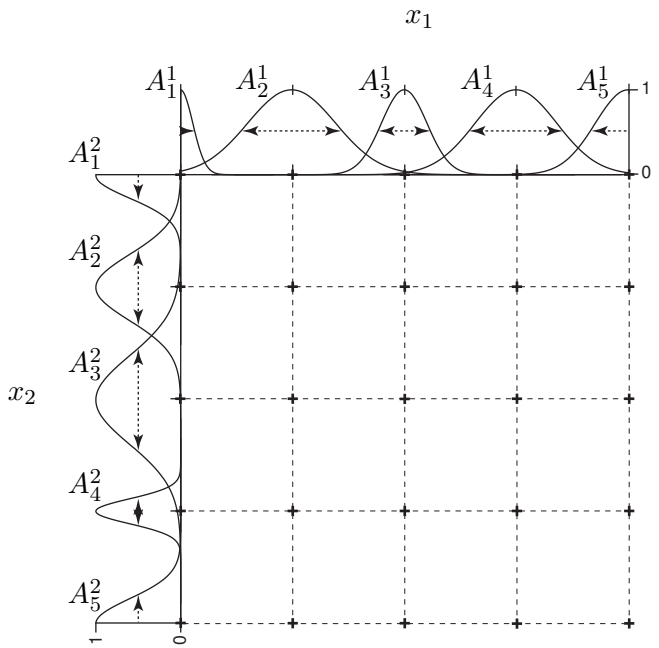



**Figure 4-4:** In the fixed fuzzy controller a grid partitioning is created through fixed membership functions. Each crosspoint corresponds with a rule consequent which defines the control output.

where $r = 1, 2, \ldots, N$, $x_i$ $(i = 1, \ldots, k)$ represent the $k$ scaled inputs, $A^i$ represent the fuzzy terms characterised by the Gaussian membership functions $\mu_{A_r^i}(x_i)$, $\mathbf{u}_r = [u_1, \ldots, u_l]$ the $l$ control outputs and $\omega_r = [\omega_1, \ldots, \omega_l]$ the corresponding constant consequent variables.

The parameters of the fixed fuzzy controller that still need to be optimised are the width of the membership functions, the constant consequent variables and the scaling factors for the inputs and outputs. The total number of parameters to be optimised for the fixed fuzzy controller depends on the number of inputs, outputs and the number of membership functions defined on each input. For simplicity the number of membership functions for each input is set equal to $m$ which yields,

$$\text{parameters} = \underbrace{m \cdot \text{inputs}}_{\text{membership functions}} + \underbrace{m^{\text{inputs}} \cdot \text{outputs}}_{\text{consequent variables}} + \underbrace{\text{inputs} + \text{outputs}}_{\text{scaling}} \tag{4-10}$$

where variables 'inputs' and 'outputs' represent the number of inputs and outputs, respectively. In order to facilitate a fair comparison, the number of parameters which need to be optimised are kept the same or nearly the same for both fuzzy controllers.

The advantage of this controller lies in the fact that the complete state space is evenly partitioned from the start and therefore the optimisation does not have to consider the position of the membership functions. This keeps the complexity of the optimisation low. Secondly, the fixation of the membership functions and the fact that they are shared between rules reduces the number of parameters needed to define a rule, therefore more rules can be defined in comparison to the free fuzzy control when the same number of parameters is used. Of course fixed membership functions will also decrease the flexibility of the controller, which

means that certain control behaviours can only be approximated with large approximation errors.

### Free Fuzzy Control

The positions of the membership functions of the free fuzzy controller are variable and need to be optimised by the EA. Additionally, membership functions are not being shared between rules. This means that changing the position or width of one membership function will only have an effect on one of the rules in the rule base. Therefore one can combine the membership functions $\mu_A$ of each rule to describe a multidimensional fuzzy basis function $\mu_{\mathbf{B}}$ in the complete state space that corresponds to that rule,

$$\mu_{\mathbf{B}}(\mathbf{x}) = \Pi_{i=1}^{k} \mu_{A_j^i}(x_i) \tag{4-11}$$

$$= \Pi_{i=1}^{k} \exp\left(-\frac{(c_j^i - x_i)^2}{2(\sigma_j^i)^2}\right) \tag{4-12}$$

Each rule has its own fuzzy basis function and it represents the area of the state space in which that rule is most active, as graphically shown in Figure 4-5.

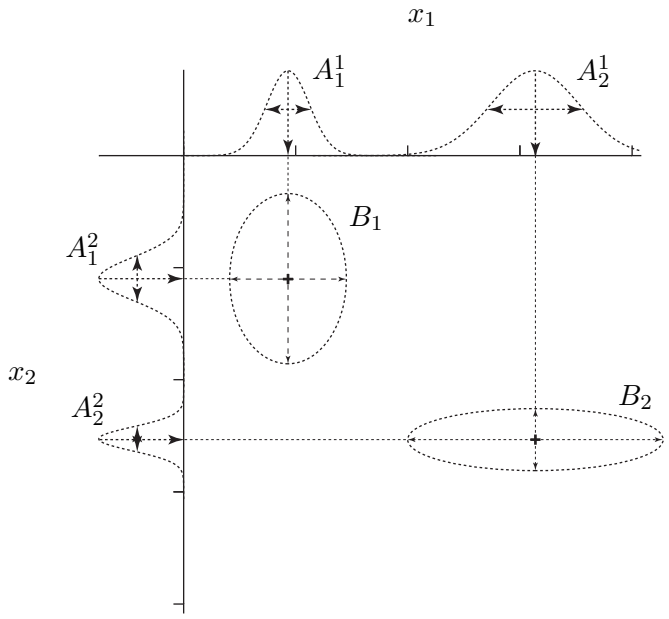



**Figure 4-5:** The free fuzzy controller has membership functions which positions are part of the optimisation problem. Each rule consequent is defined by its own membership functions on the inputs.

The $N$ rules in the rule base of the fixed fuzzy controller have the following form

$$R_r : \textbf{IF } \mathbf{x} \text{ is } \mathbf{B}_r$$
$$\textbf{THEN } \mathbf{u}_r = \boldsymbol{\omega}_r$$

where $r = 1, 2, \ldots, N$, $\mathbf{x}$ represent the scaled input vector, $\mathbf{B}_j$ represents a local fuzzy area in the state space characterised by the Gaussian fuzzy basis function $\mu_{\mathbf{B}_j}(\mathbf{x})$, $\mathbf{u}_r = [u_1, \ldots, u_l]$, the $l$ control outputs and $\boldsymbol{\omega}_r = [\omega_1, \ldots, \omega_l]$ the corresponding constant consequent variables.

The total number of parameters to define a free fuzzy controller with $b$ fuzzy basis functions equals,

$$\text{parameters} = \underbrace{b \cdot \text{inputs} \cdot 2}_{\text{basis functions}} + \underbrace{b \cdot \text{outputs}}_{\text{consequent variables}} + \underbrace{\text{inputs} + \text{outputs}}_{\text{scaling}} \tag{4-13}$$

where variables 'inputs' and 'outputs' represent the number of inputs and outputs, respectively.

The advantages of using the free fuzzy controller is that the evolutionary algorithms will have more influence on the positioning of the rules. Therefore all rules can be shifted towards places in the state space where they are most valuable. This can also be seen as a disadvantage since the use of free membership functions will make the optimisation problem more complex and therefore harder to solve. This could lead to a slow convergence.

### 4-4-4 Control Equations

For both the fixed and free fuzzy controllers the *weighted average* method has been used to determine the output of the controller. Therefore the control equation for both controllers can be written as

$$f(\mathbf{x}) = \frac{\sum_{j=1}^{N} \boldsymbol{\omega}_j \left( \Pi_{i=1}^{k} \mu_{A_j^i}(x_i) \right)}{\sum_{j=1}^{N} \left( \Pi_{i=1}^{k} \mu_{A_j^i}(x_i) \right)} \tag{4-14}$$

with

$$\mu_{A_j^i}(x_i) = \exp\left( -\frac{(c_j^i - x_i)^2}{2(\sigma_j^i)^2} \right) \tag{4-15}$$

where the centres $c_j^i$ are predefined in the fixed fuzzy controller and variable in the free fuzzy controller and the widths $\sigma_j^i$ and consequent singletons $\boldsymbol{\omega}_j$ are to be optimised by the algorithm for both controllers.

## 4-5 Summary

Fuzzy control is a nonlinear feedback control strategy based on fuzzy variables and logic rules. The advantages of fuzzy control above other nonlinear feedback control techniques such as model predictive control and neural networks is that it is not limited by the speed of the system and has the ability to extract and incorporate human knowledge from and into the controller.

The basic structure of a fuzzy controller is composed of a scaling and fuzzification step in which the control input is scaled into a domain between -1 and 1 and a membership degree is determined for each of the fuzzy variables. These membership degrees are then used in an inference mechanism which uses a rule base to determine the input-output relation of the controller. At last the output of the inference mechanism is defuzzified when necessary and scaled back into the appropriate domain of the control output.

Two types of fuzzy controllers will be used in this research. One is the fixed fuzzy controller for which the positions of the membership functions have been predefined. The other is the free fuzzy controller for which the positions of the membership function are variable and need to be optimised.

# Chapter 5

# Experiments and Results

This chapter is divided into four sections. The first three sections present the three conducted experiments and their results. The last section gives an analysis and discussion of these results. In the first experiment, presented in Section 5-1, the effectiveness of fuzzy control in combination with EAs is validated. The other two experiments concern the co-optimisation of the 1 and 2-DOF RA, as presented in Section 5-2 and Section 5-3 respectively. The results from each experiments give an answer to the three research sub questions, which are used to answer the main research question to what extent (near) optimal solutions can be found when co-optimising fuzzy control and system parameters for the RA.

In each experiment the two types of evolutionary algorithms (EAs) and two types of fuzzy controllers (as presented in Chapter 2 and 4) will be used, allowing four different combinations to be evaluated (Table 5-1). In Sections 5-1, 5-2 and 5-3 the best result found is presented and used to answer the research question. In Section 5-4 all the obtained results from the the different combinations are compared and discussed in order conclude which of the fuzzy controls and EAs is the most effective.

**Table 5-1:** The four different combinations possible when combining the two types of evolutionary algorithms (CMA-ES and CoSyNE) and the two types of fuzzy controllers (fixed and free). In each of the three experiments all four combinations have been used to solve the optimisation problems.

| Combination | Evolutionary Algorithm | Fuzzy Logic Controller |
|---|---|---|
| 1 | CMA-ES | Fixed fuzzy control |
| 2 | CoSyNE | Fixed fuzzy control |
| 3 | CMA-ES | Free fuzzy control |
| 4 | CoSyNE | Free fuzzy control |

## 5-1 Evolutionary Algorithms and Fuzzy Control Validation

The first experiment is conducted to validate whether the fuzzy control approach in combination with the EAs is suitable for finding (near) optimal control solutions. This is done by solving a minimum-time control problem in the form of a single mass on a frictionless surface with a bounded control output. For this problem the optimal feedback control solution can be derived analytically, which provides an useful baseline against which the solution found by the EA can be compared.

In Figure 5-1 a schematic drawing of the system is given. At time $t_0$ $(s)$ the mass is positioned away from the origin $z(t_0) \neq 0$ $(m)$ and has an initial speed $\dot{z}(t_0)$ $(m/s)$. An external actuation force $u(t)$ $(N)$, which is bounded between $-1$ and 1, acts on the mass. The equation of motion can be derived using Newton's Law $F = ma$ with $F = u(t)$, $m = 1$ $(kg)$ and $a = \ddot{z}(t)$;

$$\ddot{z}(t) = u(t) \tag{5-1}$$

where the control signal $u(t)$ is unspecified and bounded $-1 \leq u(t) \leq 1$.

The minimum-time control problem to be solved is written as:
*Given any initial state with position $z(t_0) \neq 0$ at time $t_0$, find the optimal feedback control signal $u(t) = f(z(t), \dot{z}(t))$ that minimises the final time $t_f$ needed to bring the state of the system close to the origin where $|z(t_f)| \leq \epsilon$ and $|\dot{z}(t_f)| \leq \epsilon$ with $\epsilon = 0.03$.*

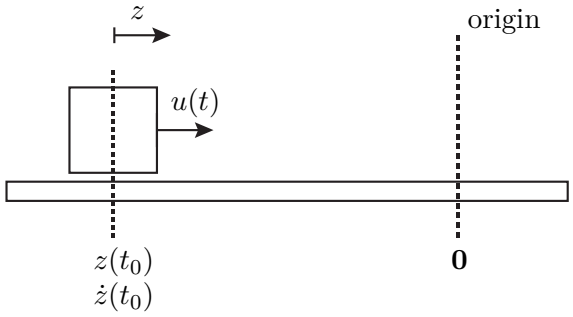



**Figure 5-1:** A single mass system is an often used example to illustrate the optimal control principle. In this system an actuation force equal to $u(t)$ acts upon a mass, which slides on a frictionless surface. The aim of this optimisation problem is to find a controller with a bounded control output that minimises the final time $t_f$ needed to bring the the mass from state $[z(t) \neq 0, \dot{z}(t)]$ at $t_0$ to a state close to the origin $\mathbf{0}$ at $t_f$. Using optimal control theory it is possible to derive the optimal feedback controller for this simple problem.

Note that the goal of the optimisation problem does not require the system to converge exactly to the origin, but instead demands a convergence to an area near the origin. This is done since it is very difficult to find a fuzzy controller that during simulation will exactly converge to the origin. Therefore the system is assumed to be converged when the absolute error between the states and the origin is smaller than or equal to 0.03 for both the position $(m)$ and the velocity $(m/s)$. The states for which this is valid are said to be within the *convergence bounds*. The value of 0.03 was found large enough for the optimised fuzzy controllers to let the system converge for all initial states simulated in the optimisation.

### 5-1-1 Analytical Derivation of the Optimal Solution

The control problem stated in this experiment can be solved analytically by applying the Pontryagin's Minimum Principle (PMP) for optimal control [57]. The advantage of the PMP is that it simplifies the optimal control problem into a two-point boundary value problem (BVP). Therefore, instead of finding a time varying control output, a set of ordinary differential equations have to be solved that satisfy the stated boundary values (Appendix D).

#### Solving the Minimum-Time Control Problem

In order to solve the single mass minimum-time problem with the PMP the system is rewritten as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \end{aligned} \tag{5-2}$$

with

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} z(t) \\ \dot{z}(t) \end{bmatrix}.$$

The performance index of a minimum-time problem is

$$J(u) = \int_{t_0}^{t_f} 1dt = t_f - t_0 \tag{5-3}$$

where $t_0$ is the fixed starting time and $t_f$ the final time needed to reach the origin.

With this performance index $J(u)$ the Hamiltonian can be constructed as

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t)) = 1 + \lambda_1(t)x_2(t) + \lambda_2(t)u(t). \tag{5-4}$$

with $\lambda_1(t)$ and $\lambda_2(t)$ the costates.

With the obtained Hamiltonian (5-4) and the inequality property of the Hamiltonian, as defined by the PMP [57], a minimisation problem is derived depending on the optimal costate $\lambda_2^*$ and the control signal $u(t)$,

$$\mathcal{H}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t)) \leq \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}(t), \boldsymbol{\lambda}^*(t)) \tag{5-5}$$

$$\begin{aligned} 1 + \lambda_1^*(t)x_2^*(t) + \lambda_2^*(t)u^*(t) &\leq 1 + \lambda_1^*(t)x_2^*(t) + \lambda_2^*(t)u(t) \\ \lambda_2^*(t)u^*(t) &\leq \lambda_2^*(t)u(t) \\ &= \min_{|\mathbf{u}(t)|\leq 1} [\lambda_2^*(t)u(t)] \end{aligned} \tag{5-6}$$

Since $u(t)$ is bounded between +1 and -1 this function is minimised for $u(t) = u^*(t)$ with

$$u^*(t) = -\text{sign}(\lambda_2^*(t)). \tag{5-7}$$

Substituting $u^*(t)$ in the costate equations, as defined by the PMP [57], yields

$$\dot{\lambda}_1^*(t) = -\left(\frac{\partial \mathcal{H}}{\partial x_1}\right)_* = 0 \tag{5-8}$$

$$\dot{\lambda}_2^*(t) = -\left(\frac{\partial \mathcal{H}}{\partial x_2}\right)_* = -\lambda_1^*(t) \tag{5-9}$$

which are solved to derive $\lambda_1^*(t)$ and $\lambda_2^*(t)$,

$$\lambda_1^*(t) = \lambda_1^*(0), \tag{5-10}$$

$$\lambda_2^*(t) = \lambda_2^*(0) - \lambda_1^*(0)t. \tag{5-11}$$

Substituting the optimal costates in (5-7) yields the optimal feedforward controller

$$u^*(t) = -\text{sign}(\lambda_2^*(0) - \lambda_1^*(0) \cdot t) \tag{5-12}$$

with two unknown initial values $\lambda_1^*(0)$ and $\lambda_2^*(0)$. When $\lambda_1^*(0)$ and $\lambda_2^*(0)$ have different signs the control output will remain $+1$ or $-1$. When both $\lambda_1^*(0)$ and $\lambda_2^*(0)$ have equal signs this controller will switch between the values $-1$ and $+1$ exactly once at $t = \lambda_2^*(0)/\lambda_1^*(0)$. This type of control is called bang-bang control.

The optimal control solution for a given initial position can now be solved by finding the initial values of the costates that will bring the system from the initial state to the fastest reachable state located within the convergence bounds. However, in this experiment we are interested in finding the optimal feedback controller defined by a feedback law that will give the optimal control output for all initial states.

The optimal feedback controller can be found by solving the state equations (5-2) for both $x_1^*(t)$ and $x_2^*(t)$

$$\dot{x}_1^*(t) = x_2^*(t) \tag{5-13}$$

$$\dot{x}_2^*(t) = u^*(t) \tag{5-14}$$

which yields

$$x_1^*(t) = x_1^*(0) + x_2^*(0)t + \frac{1}{2}u^*(t)t^2 \tag{5-15}$$

$$x_2^*(t) = x_2^*(0) + u^*(t)t \tag{5-16}$$

and eliminate $t$ to get the state trajectories independent of time

$$x_1^*(t) = x_1^*(0) + \frac{1}{2\ u^*}(x_2^*(t)^2 - x_2^*(0)^2) \tag{5-17}$$

$$t = (x_2^*(t) - x_2^*(0))/u^*\ , \tag{5-18}$$

where $u^* = -\text{sign}(\lambda_2^*(0) - \lambda_1^*(0) \cdot t) = \pm 1$.

At some point in time the system will have to follow the state trajectories that will lead the system towards the fastest reachable state located at the convergence bounds. This fastest

final state is dependent on the initial state of the system, therefore two optimal final state variables are used,

$$x_1^*(t_f) = x_{1f} \tag{5-19}$$
$$x_2^*(t_f) = x_{2f}. \tag{5-20}$$

Substituting the final state variables in the state trajectories equation (5-18) yields

$$x_{1f} = x_1^*(0) + \frac{1}{2\,u^*}(x_{2f}^2 - x_2^*(0)^2) \tag{5-21}$$

with $u^* = \pm 1$.

In Figure 5-2(a) the initial state trajectories are plotted for different values of $x_{1f}$ and $x_{2f}$ and for both $u^* = +1$ and $u^* = -1$. In this figure one can see that either two states are on the same trajectory or the trajectories of the two states intersect. This intersection is the state at which the controller should switch between the control actions -1 and +1 when moving between two states that are not located on the same trajectories. The switch state $[x_{1s}\ x_{2s}]$ when moving from initial state $[x_{10}\ x_{20}]$ to the final state $[x_{1f}\ x_{2f}]$ can be calculated by setting the two state trajectories equations equal to each other

$$x_{10} + \frac{1}{2\,u_1^*}(x_{2s}^2 - x_{20}^2) = x_{1f} - \frac{1}{2\,u_2^*}(x_{2f}^2 - x_{2s}^2) \tag{5-22}$$

which yields the switch states when switching from $u_1^* = +1$ to $u_2^* = -1$

$$x_{2s} = \pm\sqrt{(x_{1f} - x_{10}) + \frac{1}{2}(x_{2f}^2 + x_{20}^2)} \tag{5-23}$$
$$x_{1s} = x_{1f} + \frac{1}{2}(x_{2f}^2 - x_{2s}^2) \tag{5-24}$$

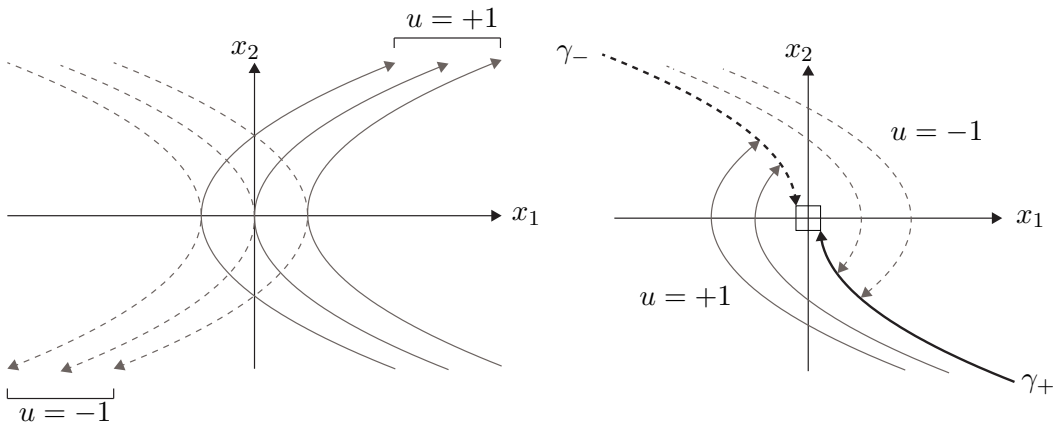
and when switching from $u_1^* = -1$ to $u_2^* = +1$

$$x_{2s} = \pm\sqrt{(x_{10} - x_{1f}) + \frac{1}{2}(x_{2f}^2 + x_{20}^2)} \tag{5-25}$$
$$x_{1s} = x_{1f} - \frac{1}{2}(x_{2f}^2 - x_{2s}^2)\ . \tag{5-26}$$

In Figure 5-2(b) two switching lines $\gamma_+$ and $\gamma_-$ are presented which represent the optimal switching states for all initial positions considering the fastest reachable state at the convergence bounds. With a convergence bound of $\epsilon = 0.03$ these lines are defined by:

$$\gamma_-:\ x_1 = -0.03 + \frac{1}{2}(0.03^2 - x_2^2) = -0.0296 - \frac{1}{2}x_2^2 \quad \text{for } x_2 \geq 0.03 \tag{5-27}$$
$$\gamma_+:\ x_1 = 0.03 - \frac{1}{2}((-0.03)^2 - x_2^2) = 0.0296 + \frac{1}{2}x_2^2 \quad \text{for } x_2 \leq -0.03\ . \tag{5-28}$$

An insightful way of presenting the feedback control behaviour is shown in Figure 5-3 in which the control action at multiple states in the state space is plotted as a control surface with different colours corresponding to the different control outputs. In Figure 5-4 the optimal control and state response are plotted over time when starting from the initial state $[z, \dot{z}] = [-0.8, 0]$. These plots together with the derived switching line will be used as a comparison for the plots and results from the fuzzy controllers obtained after optimising them with EAs.




(a) State trajectories (b) Optimal switching line

**Figure 5-2:** Two phase planes with state trajectories corresponding to the frictionless single mass system presented in Figure 5-1. Variable $x_1$ represents the position $z$ and $x_2$ the velocity $\dot{z}$. In (a) the state trajectories are plotted for different desired final states and for both $u^* = +1$ (solid lines) and $u^* = -1$ (dashed lines). In (b) the optimal switching lines $\gamma_-$ and $\gamma_+$ are presented which represent the states at which the controller should switch between the control actions $u^* = +1$ and $u^* = -1$ in order to ensure a minimal time in which the convergence boundary (represented by the square around the origin) is reached.

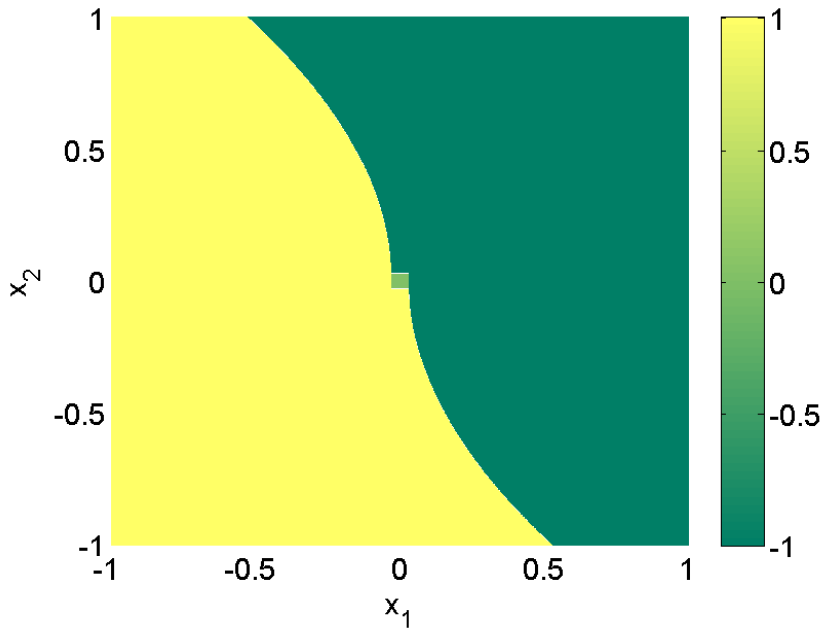
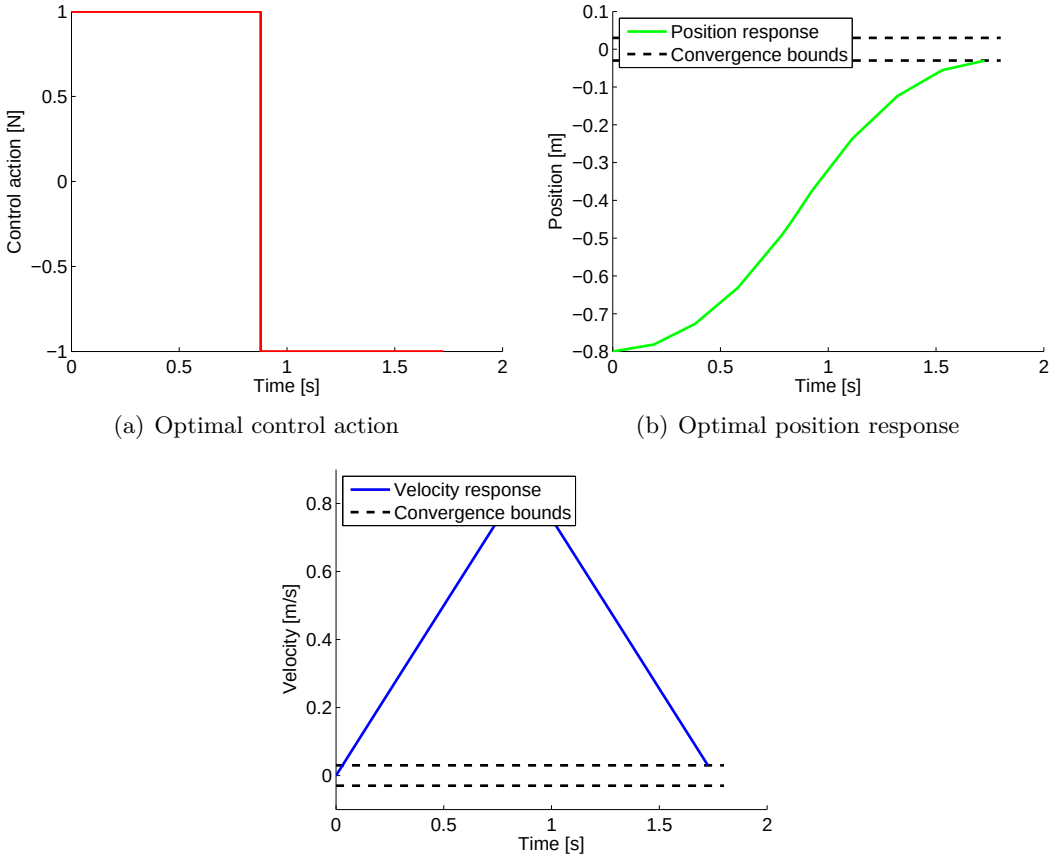



**Figure 5-3:** The optimal control surface for a minimum-time frictionless mass control problem, in which $x_1 = z$ and $x_2 = \dot{z}$. This plot is similar to Figure 5-2(b), but here the value of the control action is presented as a colour. Dark green stands for a control action of -1 and bright yellow represents a control action of $+1$ (see colour scale).




(a) Optimal control action

(b) Optimal position response

(c) Optimal velocity response

**Figure 5-4:** The optimal control output and state response for a minimal time frictionless mass control problem when starting from $[z, \dot{z}] = [-0.8, 0]$. The bang-bang controller generates a switching behaviour from +1 to -1 at half of the convergence time.

## 5-1-2 Experimental Setup

The same minimal-time control problem has been solved in an experimental setup using the fuzzy controllers and EAs. This section presents all the parameter values specific for this experiment, the initial states and fitness functions used in the experiment.

### Parameter Values

In Table 5-2 the experimental parameters and their values are presented. The number of generations is set to 3000, which allowed both EAs to converge to a stable fitness score that did not change significantly for the last 100 generations. The simulation time of 3 seconds gives the system enough time to converge to the origin for all simulated initial states (presented in the next section). The system was solved using the Runge-Kutta fourth order method (RK4) with a step size of 0.01 seconds. This step size was found suitable since smaller step sizes did not provide significant better results. For the fixed fuzzy controller 5 membership functions are defined on each of the two inputs, which results in an optimisation problem of

38 parameters in total. The free fuzzy controller uses 7 fuzzy basis functions and also results in an optimisation problem of 38 parameters.

**Table 5-2:** Specific parameters used in the single mass experiment. Variable $m$ represents the number of fuzzy membership functions at each input for the fixed fuzzy controller and variable $b$ the total number of fuzzy basis functions used in the free fuzzy controller.

| Parameter | Value | Units |
|---|---|---|
| Number of generations | 3000 | |
| Simulation time | 3 | $s$ |
| RK4 step size | 0.01 | $s$ |
| Convergence bounds | $\pm 0.03$ | $m$ |
| | $\pm 0.03$ | $m/s$ |
| Membership functions (free fuzzy contr.): $m$ | 5 | |
| Basis functions (fixed fuzzy contr.): $b$ | 7 | |

### Initial States

The feedback quality of a controller is determined by evaluating the system responses from eight different initial states. At these initial states the system is at rest ($\dot{z}_{\text{init}} = 0$), which is also the case for the initial states of the RA when performing the pick-and-place task. The initial positions $z_{\text{init}}$ are evenly distributed between -1 and 1 (except the position 0, which is equal to the desired position),

$$[z_{\text{init}}, \dot{z}_{\text{init}}] \in \left\{ \begin{array}{cccc} [0.2,0], & [0.4,0], & [0.6,0], & [0.8,0], \\ [-0.2,0], & [-0.4,0], & [-0.6,0], & [-0.8,0] \end{array} \right\} \tag{5-29}$$

The difference in starting position will cause each state trajectory to reach the optimal switching line at a different point in the state space. This will force the EA to find a feedback controller that approximates the optimal control for each of the initial positions and thus will give an approximation of the overall optimal feedback behaviour.

### Fitness Function

The fitness function used to solve the minimum-time control problem is shown in Algorithm 4. The fitness function determines a fitness score depending on the time needed to get within the convergence bounds. Two sub fitness functions are used because it is possible that the simulated system did not converge within the simulation time. This makes the fitness function an incremental fitness function, which is able to avoid the bootstrap problem when all solutions resulted in simulation errors or did not converge to the desired state (as discussed in Section 2-4-1). The fitness function is constructed in such a way that the fitness score given to a solution is always between 0 and 3 for which a lower score corresponds to a better solution. A detailed description of the fitness function is given below:

**Line 1-2:** First it is checked whether the simulation failed due to numerical limitations. If so, the time at which the first error occurred ($t_{\text{fail}}$) and the total simulation time ($t_{\text{simulation}}$)

are used to calculate the fitness score and a penalty of 2 is added.
**Line 3-4:** Second, if the simulation did not fail, it is checked whether the absolute error $|e|$ between the final states and the desired states at time $t_{\text{final}}$ was larger than the predefined convergence bound $\epsilon$. If so, the system is considered to be not converged and the error is taken as a measure for the fitness plus a penalty score of 1.
**Line 5-6:** Finally, if the system did converge, the time needed for convergence ($t_{\text{converged}}$) is used to determine the fitness.

**Algorithm 4** Fitness function for the single mass optimisation

1: **if** `simulation fails at some time` $t_{\text{fail}}$ **then**
2: fitness $\leftarrow \left(1 - \frac{t_{\text{fail}}}{t_{\text{simulation}}}\right) + 2$
3: **else if** `absolute error` $|e_{t_{\text{final}}}| >$ `convergence bound` $\epsilon$ **then**
4: fitness $\leftarrow \left(1 - \frac{1}{1+e^2}\right) + 1$
5: **else**
6: fitness $\leftarrow \left(1 - \frac{1}{1+t_{\text{converged}}}\right)$
7: **end if**

Each initial state will be simulated in a separate simulation and will receive a fitness score $f_{\text{sim}}$ from the fitness function. The final fitness score of the controller $f_{\text{ctrl}}$ is set equal to the sum of all fitness scores obtained from the eight simulations,

$$f_{\text{ctrl}} = \sum_{n=1}^{8} f_{\text{sim}}(n) \tag{5-30}$$

where $n$ represents the simulation number.

This fitness function together with the initial states and experimental parameters mentioned before define the optimisation of the minimum-time problem using the fuzzy controllers and EAs. The obtained results of this experiment will be discussed in the following section.

### 5-1-3 Analysis of Results

The goal of the single mass experiment is to answer the question whether a (near) optimal feedback control solution in the form of a fuzzy controller can be found when EAs are used for optimisation. A solution is considered near optimal when the average convergence time is at maximum 5% off from the optimal convergence time. Each controller-algorithm combination has been tested 20 times and the best performing solution was found by the CMA-ES algorithm in combination with the free fuzzy controller.

#### Analysing the Control Surface

The control surface of the best performing fuzzy controller is presented in Figure 5-5 and one can see a clear resemblance to the optimal control surface derived before (Figure 5-3). All eight state trajectories converge to the convergence bounds and their switching states are close the optimal switching line (represented by the dashed line).

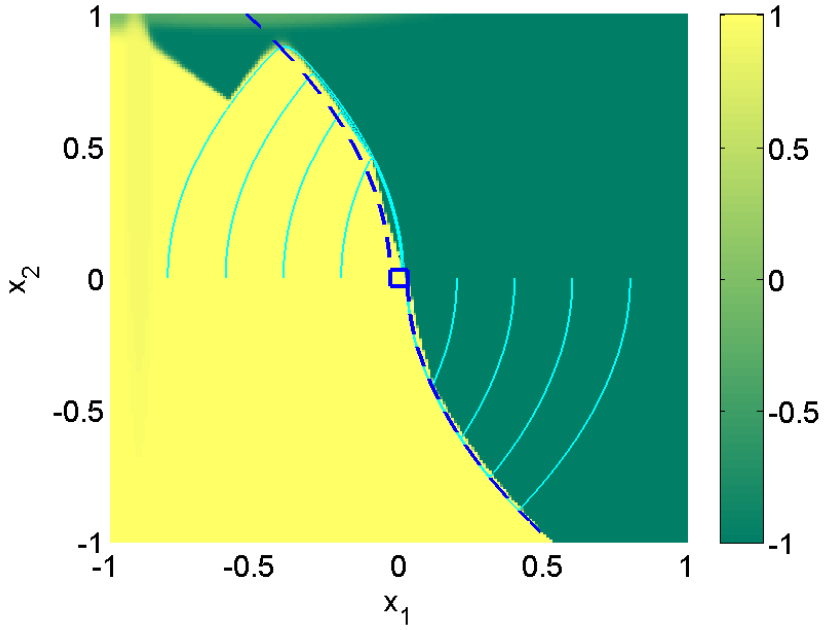



**Figure 5-5:** The control surface of the best controller found during the optimisation of fuzzy controllers for a single mass control problem. Variable $x_1$ represents the position $z$ of the mass and $x_2$ its velocity $\dot{z}$. The value of the control action at each state is presented by a colour, where dark green stands for a control action of -1 and bright yellow represents a control action of $+1$. All control actions in between are indicated by the colour scale. The eight solid lines represent the state trajectories of the system simulations starting from eight different initial positions. Additionally, the convergence bounds and the optimal switching line (dashed) are presented. From this figure it can be seen that the optimised control surface is a close approximation of the optimal control surface presented in Figure 5-3.

This visual analysis also endorses the fact that if a part of the state space is not reached during the simulations there exist a possibility that it will not be optimised towards the optimal solution. This is visible in the upper left corner of the top view where a part of the state space has a control solution far from the optimal solution. None of the eight state trajectories is influenced by this control area and thus did it not influence the fitness score of the controller. For this reason the control output at this area was not optimised.

### Analysing the Simulation Runs

To investigate the actual quality of the control solution found by the EA, the state responses from the eight initial positions are analysed. In Figure 5-6 a comparison is made between the derived control action and state responses for one of the initial positions and the optimal solution. It shows that the fuzzy controller output approximates the discrete step function of the optimal control output as a smooth and continues function. The fact that this approximation is smooth and not discrete is due to the use of the Gaussian membership functions, which are continues functions with a positive output for all inputs.

A numerical comparison of all eight simulations is presented in Table 5-3. Different conver-

**Table 5-3:** The convergence time of the eight different simulation runs starting from eight different initial states. Next to them the optimal convergence times are given and the difference (error) between both. In both cases the system is assumed to be converged when the absolute position and velocity errors are equal or below the convergence bound $\epsilon = 0.03$.

| **Nr.** | **Initial state** $[z, \dot{z}]$ | **Convergence time** [seconds] | **Optimal convergence time** [seconds] | **Difference** [seconds] |
|---|---|---|---|---|
| 1 | [-0.8,0] | 1.79 | 1.73 | 0.06 |
| 2 | [-0.6,0] | 1.55 | 1.48 | 0.07 |
| 3 | [-0.4,0] | 1.27 | 1.18 | 0.09 |
| 4 | [-0.2,0] | 0.90 | 0.80 | 0.1 |
| 5 | [0.2,0] | 0.82 | 0.80 | 0.02 |
| 6 | [0.4,0] | 1.21 | 1.18 | 0.03 |
| 7 | [0.6,0] | 1.50 | 1.48 | 0.02 |
| 8 | [0.8,0] | 1.74 | 1.73 | 0.01 |
| **Average:** | | 1.3475 | 1.2975 | 0.05 (4%) |

gence times are compared against the optimal convergence times and the difference between them is given. Also the average convergence time of all runs is presented. From the analysis of the different convergence times we can derive that the optimised controller has an average convergence time error of about 4% when considering the convergence times of all eight simulation runs. This is below the stated maximum error of 5% and therefore the solution is considered near optimal.

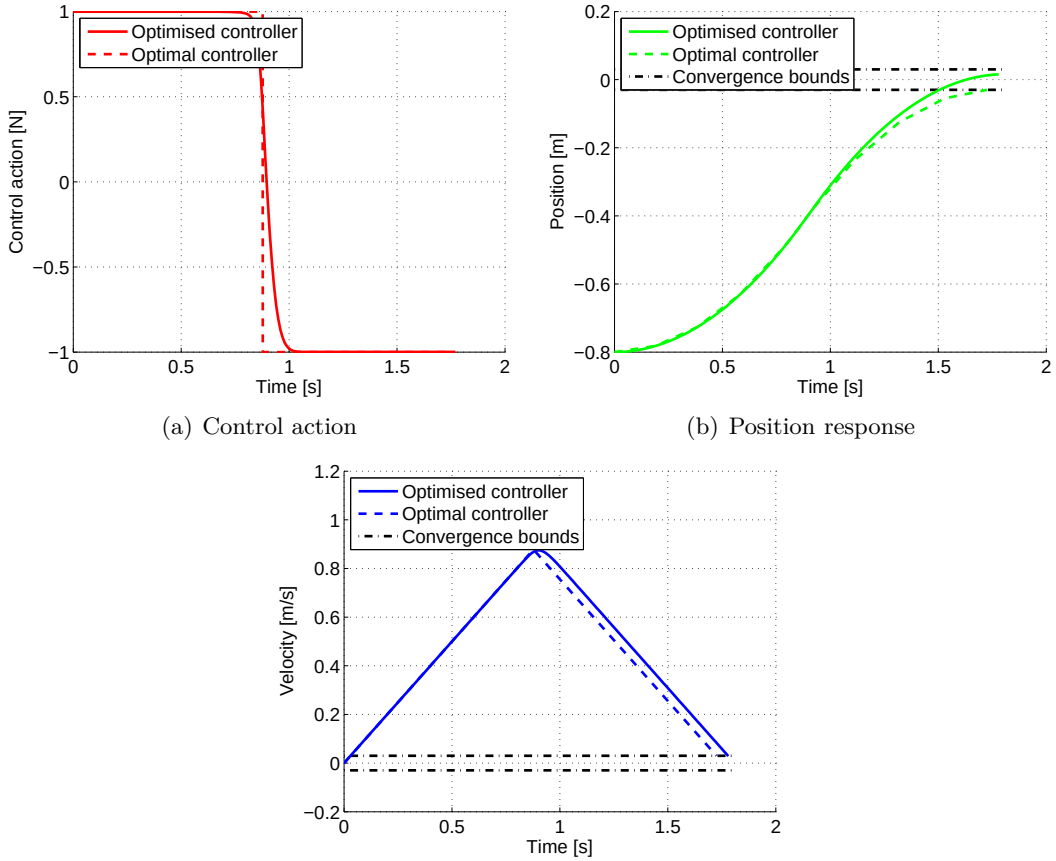
### 5-1-4 Conclusion

From the results of this experiment it can be concluded that an evolutionary algorithm is able to find near optimal control solutions in the form of a fuzzy controller. It has been observed that the control actions for the different simulations are close approximations of the optimal control actions. The average convergence time error is 4%, which is below the maximum error of 5% stated at the beginning of the experiment. Additionally it has been observed that the total control surface of the solution is very similar to the optimal control surface, however, some areas which are not reached during the simulation of the different initial states are not optimised since they do not influence the fitness score.

## 5-2 Co-Optimisation of 1-DOF Resonating Arm Validation

The second experiment concerns the optimisation of the RA with only one degree of freedom. This 1-DOF RA consists of only one arm and the pivot mechanism has been replaced by the gripper.

The focus of this experiment is on the advantages of *evolutionary co-optimisation* compared to a more conventional *stepwise optimisation*. In this stepwise optimisation the system parameters are optimised first and thereafter the optimal controller is found. The effectiveness of evolutionary co-optimisation is shown by optimising the RA in a stepwise optimisation and comparing this solution with the solution found through evolutionary co-optimisation.




(a) Control action

(b) Position response

(c) Velocity response

**Figure 5-6:** The control output and state response from the initial state starting at a position of -0.8 $m$ and a velocity of 0 $m/s$ when simulating with the best control solution found during the experiment. The dashed lines represent the optimal control output and corresponding state response, which are very similar to the simulated output.

The optimisation goal of this second experiment is stated to be:
*Find the fuzzy control and system parameters that minimise the maximum actuator torque required to perform certain pick-and-place tasks. This actuator torque should allow the fuzzy controller to move the system to a predefined region in the state space and it should be large enough to stay at all positions in this region.*

In this optimisation goal two criteria are defined which the maximum actuator torque has to satisfy. The first criterion demands that the minimised actuator torque is large enough for the fuzzy controller to move to a predefined region in the state space. This predefined region is defined as a bounded area around the desired state, which is similar to the convergence bounds used in the first experiment. The task of the fuzzy controller is to bring the state of the system into this region, where a second controller (e.g. a PID controller) is assumed to take over and provide a more precise positioning of the system. The control structure and tuning of this second controller are not considered in this work, however, it is important to realise that this controller should be able to counteract the torque of the spring mechanism. If the maximum actuator torque is too low the second controller will not be able to keep the

arm still during a pick or place handling. Therefore the second criterion demands that the minimised actuator torque is large enough to counteract the spring mechanism torque at all positions in the predefined region.

The predefined region in the state space is defined by a position interval and a maximum velocity. This position interval represents all the desired positions at which the arm should be able to stand still in order to pick or place an object. In this research this position interval is defined as $[\Theta_a, \Theta_b] = [0.75, 0.85]$ $(rad)$. As mentioned before, the fuzzy controller will only be optimised to bring the position of the arm in this interval and a second controller will then perform the precise positioning. In order to support a smooth positioning, the second controller will only take over the control when the velocity of the arm is below a certain maximum. In this research the maximum is set at 0.05 $(rad/s)$, which is a relative low velocity.

## 5-2-1 Stepwise Optimisation of Control and System Parameters

A conventional stepwise optimisation will be applied in which the system parameters and control parameters are optimised in a step-by-step process. First the system parameters are analysed and optimised, and thereafter the optimal control is derived in order to find a complete optimised 1-DOF RA solution.

### Optimising the System Parameters

In Table 5-4 the system parameters to be optimised are summarised. In order to optimise these parameters the general optimisation goal is transformed into a form which only depends on the systems parameters. This simplified optimisation goal reads: *Find the system parameters that minimise the maximum actuator torque required to stay at all positions in the predefined position interval.*

**Table 5-4:** Six parameters describing the dynamic behaviour of the 1-DOF RA.

| Parameter description | Symbol | Units |
|---|---|---|
| Radius at which the spring is connected to the large pulley | $r_1$ | $m$ |
| Radius at which the spring is connected to the small pulley | $r_2$ | $m$ |
| Ratio between the outer radii of the pulleys $(R_1/R_2)$ | $a$ | $m/m$ |
| Initial length of the spring | $l_0$ | $m$ |
| Spring constant | $k$ | $N/m$ |
| Moment of inertia (spring mechanism) | $I_0$ | $kgm^2$ |

The problem with this optimisation goal is that the optimal solution has a spring constant equal to zero $(k = 0\ N/m)$, which is equivalent to having no spring mechanism. No spring mechanism means that the actuator torque needed to stay at the positions in the position interval is zero, but also that no potential energy is stored to reduce the torque required to move between the pick-and-place positions.

A possible solution to this problem is putting a constraint on the potential energy stored at a certain position. This will lead to a optimisation problem that demands a minimised

actuator torque to stay in the position interval while providing a certain level of potential energy stored to accelerate and decelerate the arm between the pick-and-place positions.

This transforms the previously stated optimisation problem into:
*Find the system parameters that minimise the maximum actuator torque required to stay at all positions in the predefined position interval, while providing a predefined level of potential energy at a predefined position.*

In mathematical form this reads:

$$\text{min: } f(r_1, r_2, a, l_0, k) = \max(|\tau_s(\Theta)|) \quad \text{for all } \Theta \in [\Theta_a, \Theta_b] \tag{5-31}$$

$$\text{subject to: } E_p(\Theta_d) = E_{p,\text{desired}}. \tag{5-32}$$

in which the function $f(\cdot)$ is minimised given a constraint on the potential energy $E_p$ at the angle $\Theta_d$. The function $f(\cdot)$ equals the maximum torque needed to keep the system at all positions in the interval $[\Theta_a, \Theta_b]$ and depends on the parameters $r_1$, $r_2$, $a$, $l_0$ and $k$. The inertia $I_0$ of the spring mechanism does not have an influence on the potential energy curve and will have to be optimised separately.

The constraint optimisation problem was solved for a desired potential energy level of 0.40 Joule at $\Theta_d = 0.8$ and the presented position interval $[\Theta_a, \Theta_b] = [0.75, 0.85]$. The desired potential energy level of 0.40 Joule at $\Theta_d = 0.8$ is equal to the amount of potential energy stored in the prototype at that angle. In order to limit the size of the search space, the parameter values of $r_1$, $r_2$, $a$, $l_0$ and $k$ were bounded around the parameter values of the prototype. The optimised values, bounds and prototype values are summarised in 5-5.

**Table 5-5:** The six variables describing the 1-DOF RA. Given certain bounds, the values are optimised to minimise the torque applied by the spring mechanism in an interval of $0.75$ to $0.85$ radians while maintaining a stored energy of $0.4$ Joules at an angle of $0.8$ radians. The value of parameter $k$ is indicated with a '*' since its optimality only holds for a desired energy level of $0.4$ Joules, when the desired energy is changed this value will change, however, all other values will still remain the same.

| Parameter | Symbol | Prototype value | Optim. bounds | Optim. value | Units |
|---|---|---|---|---|---|
| Spring radius at large pulley | $r_1$ | 0.1 | $[0.05 - 0.20]$ | 0.20 | $m$ |
| Spring radius at small pulley | $r_2$ | 0.02 | $[0.01 - 0.04]$ | 0.0389 | $m$ |
| Transfer ratio | $a$ | 5 | $[2.5 - 10]$ | 5.1153 | $m/m$ |
| Initial length of spring | $l_0$ | 0.1 | $[0.05 - 0.20]$ | 0.05 | $m$ |
| Spring stiffness | $k$ | 150 | $[0 - 200]$ | 33.1861* | $N/m$ |
| **Max. abs. torque at $[\Theta_a, \Theta_b]$ :** | | 0.0817 | | 0.0218 | $Nm$ |

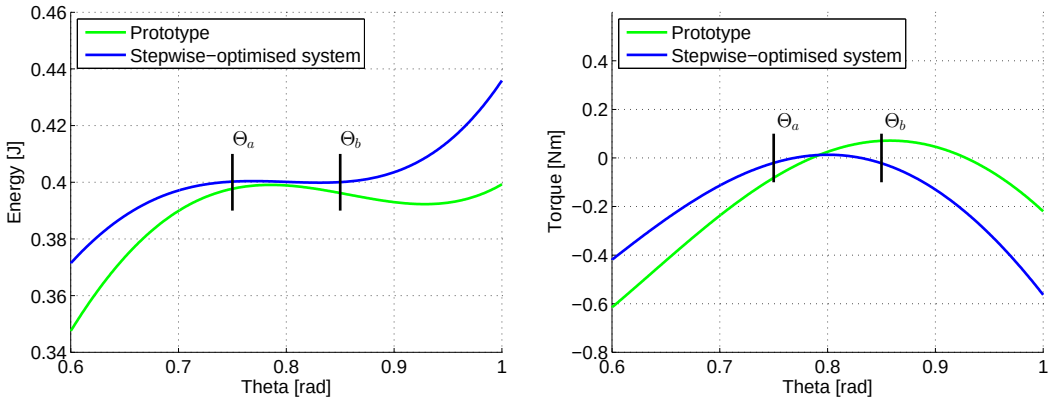
Through optimisation of the prototype parameters the maximum absolute torque at the position interval was reduced from 0.0817 $Nm$ to 0.0218 $Nm$. The optimisation of the system parameters is also reflected in the potential energy and torque curve as presented in Figure 5-7. One can observe that at the indicated interval $[\Theta_a, \Theta_b]$ the potential energy curve has been flattened; this corresponds with a smaller absolute torque as clearly visible in the torque curve.

The inertia $I$ of the spring mechanism does not influence the potential energy curve, but has an influence on the torque needed to attain a certain acceleration. As known from Euler's

equation $\tau = I\ddot{\Theta}$ a higher inertia requires a higher torque to obtain the same acceleration. Therefore the optimal value of the inertia equals its lower bound, which is equivalent to the inertia of the prototype, namely 0.16 $Nm^2$.

At this point it seems that all parameters have successfully been set to their optimal values. However, it should be noted that this has been done for one specific potential energy level $E_{p,desired}$, which was not proven to be optimal. Fortunately, the potential energy level is directly correlated with the spring constant $k$ (Equation 3-5). This means that any desired potential energy level used in the optimisation will yield the same optimal values for $r_1$, $r_2$, $a$ and $l_0$, and only changes the spring constant $k$.

The optimal potential energy level $E_{p,desired}$, and thus the optimal spring constant $k$, cannot be determined without investigating the influence on the actuator torque required to move between the pick-and-place positions. Since the optimal controller is not yet derived, an optimal potential energy level has to be assumed. At this point we assume that a highest level of potential energy will yield the lowest torque and thus the spring constant $k$ is set to its upper bound of 200 $N/m$. This assumption was later verified by solving the the optimal control problem for different values of $k$.




(a) Potential energy curve

(b) Torque curve

**Figure 5-7:** The potential energy and torque curves of the spring mechanism with both the prototype and the two optimised system parameter values. The goal of the optimisation was to minimise torque applied by the spring mechanism between the angles $\Theta_a$ and $\Theta_b$. This corresponds with a minimisation of the derivative of the potential energy curve.

### Optimising the Controller

The optimal controller is found by transforming the control part of the general optimisation goal into the following form: *Find the optimal control that minimises the maximum actuator torque required to move the system to a predefined region in the state space.*

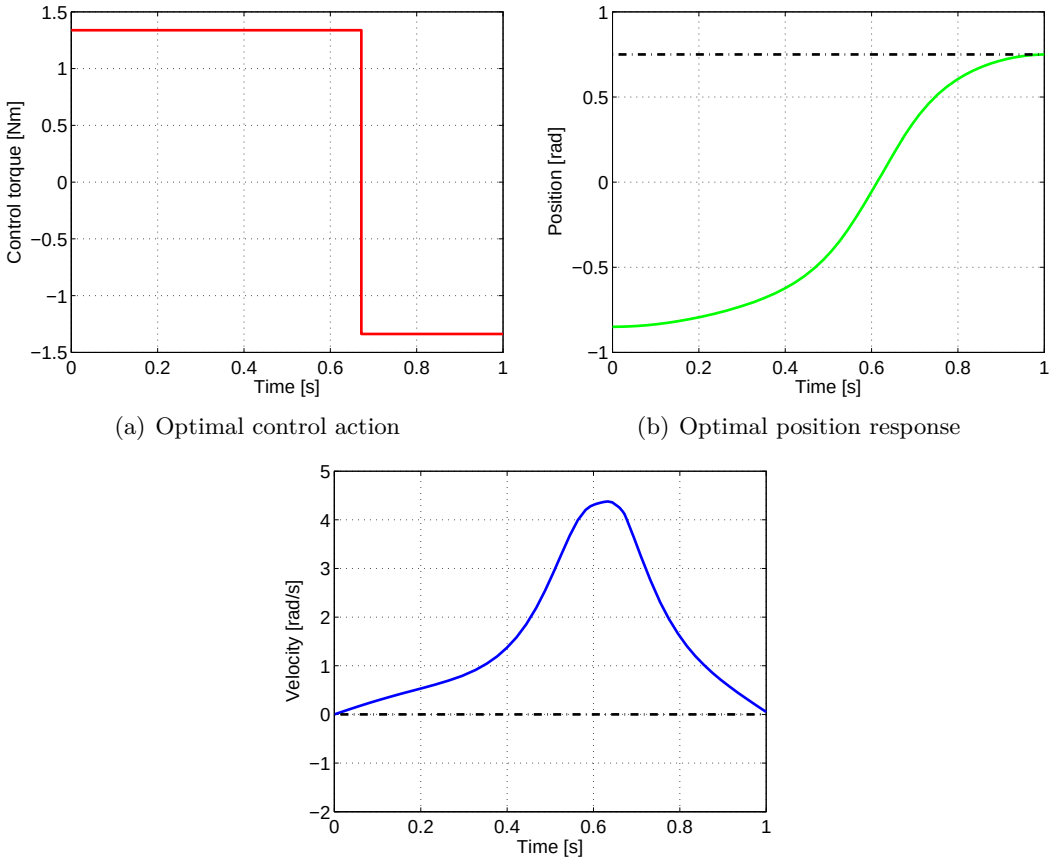
Unfortunately, this problem cannot directly be solved by applying the Pontryagin's Minimum Principle as done in the first experiment. This is due to the fact that no cost-to-go or final-cost can be assigned for the minimisation of the maximum used torque.

This problem has been solved by applying an iterative approach in which a minimum-time problem is solved for a decreasing set of control bounds. When the control bounds are set

too high the minimal-time problem will yield an optimal time below the time constraint of 1 second. By iteratively decreasing the bounds until the optimal time is equal to 1 second the minimal-maximum-torque problem can be solved.

### The Stepwise-Optimised Solution

The stepwise optimisation has provided an optimised solution for the system parameters and control. In Figure 5-8 an example is given of the optimal control solution and state response for an initial state of $[\Theta, \dot{\Theta}] = [-0.85, 0]$ given the optimised system parameters and a time constraint of 1 second. This example will be used to compare the stepwise solutions with the solutions found through evolutionary co-optimisation.




(a) Optimal control action

(b) Optimal position response

(c) Optimal velocity response

**Figure 5-8:** The optimal control solution for the 1-DOF RA with optimised system parameters when starting from an initial state of $[\Theta, \dot{\Theta}] = [-0.85, 0]$ and moving towards the closest boundary of the place state at $[\Theta_d, \dot{\Theta}_d] = [0.75, 0.05]$.

## 5-2-2 Experimental Setup

After the stepwise optimisation the 1-DOF RA will be optimised through evolutionary co-optimisation. This section presents the parameter values which are used to conduct this

**Table 5-6:** Specific parameters used in the 1-DOF RA experiment. Variable $m$ represents the number of fuzzy membership functions at each input for the fixed fuzzy controller and variable $b$ the total number of fuzzy basis functions used in the free fuzzy controller.

| Parameter | Value | Units |
|---|---|---|
| Number of generations | 6000 | |
| Simulation time | 1 | $s$ |
| RK4 step size | 0.01 | $s$ |
| Membership functions (free fuzzy contr.): $m$ | 5 | |
| Basis functions (fixed fuzzy contr.): $b$ | 7 | |

experiment together with the initial states and fitness functions.

### Parameter Values

In Table 5-6 the different parameters corresponding to the 1-DOF RA experiment are presented. The number of generations is chosen to be 6000, which is large enough for the algorithms to converge. The simulation time is set equal to the time constrain of 1 second. The step size of the RK4 integration method is set to 0.01 seconds after validating that no significant better results were found with smaller step sizes. The number of membership functions per input for the fixed fuzzy controller is set to 5 and the number of basis functions for the free fuzzy controller is set to 7. For both controllers this results in an optimisation problem of 44 dimensions, in which 6 dimensions are due to the system parameters.

### Initial States

The feedback quality of a controller is determined by evaluating the responses of the system from three different simulations, each starting from a different initial state. These initial states have an angular velocity $\dot{\Theta}_{\text{init}}$ equal to zero and their angular positions $\Theta_{\text{init}}$ are located in the middle and edges of the pick-position interval $[-0.85, -0.75]$, opposite to the (place-)position interval discussed earlier;

$$[\Theta_{\text{init}}, \dot{\Theta}_{\text{init}}] \in \{[-0.85, 0],\ [-0.8, 0],\ [-0.75, 0]\} \tag{5-33}$$

These points have been selected to optimise a feedback controller that is able to bring the arm to the place-position interval starting from different initial positions in the pick-position interval. Other initial positions and disturbances have not been considered in order to keep the required simulation time low.

### Fitness Function

The fitness function used for the optimisation of the 1-DOF RA is shown in Algorithm 5. Before the fitness is determined using the maximum used torque during the simulation, two checks are present to make sure that the solutions can still be given a score when the simulation did fail or if the system did not converge to the desired state (see Section 2-4-1 about

incremental fitness functions). The fitness score of each solution is always between 0 and 3. A detailed description of the fitness function is given below:

**Line 1-2:** First it is checked whether the simulation failed due to numerical limitations. If so, the time at which the first error occurred ($t_{\text{fail}}$) and the total simulation time $t_{\text{simulation}}$ are used to calculate the fitness score and a penalty of 2 is added.
**Line 3-4:** Second, if the simulation did not fail, it is checked whether the absolute error $|e|$ between the final state and the desired state at time $t_{\text{final}}$ was larger than the predefined convergence bound $\epsilon$. If so, the system did not converge and the error is taken as a measure for the fitness plus a penalty score of 1.
**Line 5-8:** Finally, if the system did converge, the maximum absolute torque used to move between the pick-and-place positions and the maximum torque used to stay at the place-position interval are determined. The maximum torque needed to stay or move is used to determine the fitness.

**Algorithm 5** Fitness function of the 1-DOF RA optimisation

1: **if** `simulation fails at some time` $t_{fail}$ **then**
2: fitness $\leftarrow \left(1 - \frac{t_{\text{fail}}}{t_{\text{simulation}}}\right) + 2$
3: **else if** `absolute error` $|e_{t_{\text{simulation}}}| >$ `convergence bound` $\epsilon$ **then**
4: fitness $\leftarrow \left(1 - \frac{1}{1+e^2}\right) + 1$
5: **else**
6: $\tau_m \leftarrow$ `max. abs. torque-to-move from pick position to place range`
7: $\tau_p \leftarrow$ `max. abs. torque-to-stay in place range`
8: fitness $\leftarrow \left(1 - \frac{1}{1+\max(\tau_m, \tau_p)}\right)$
9: **end if**

Similar to the first experiment the total fitness score $f_{\text{total}}$ of a solution is defined by the sum of the fitness scores $f_{\text{sim}}$ obtained from the performed simulations,

$$f_{\text{total}} = \sum_{n=1}^{3} f_{\text{sim}}(n) \tag{5-34}$$

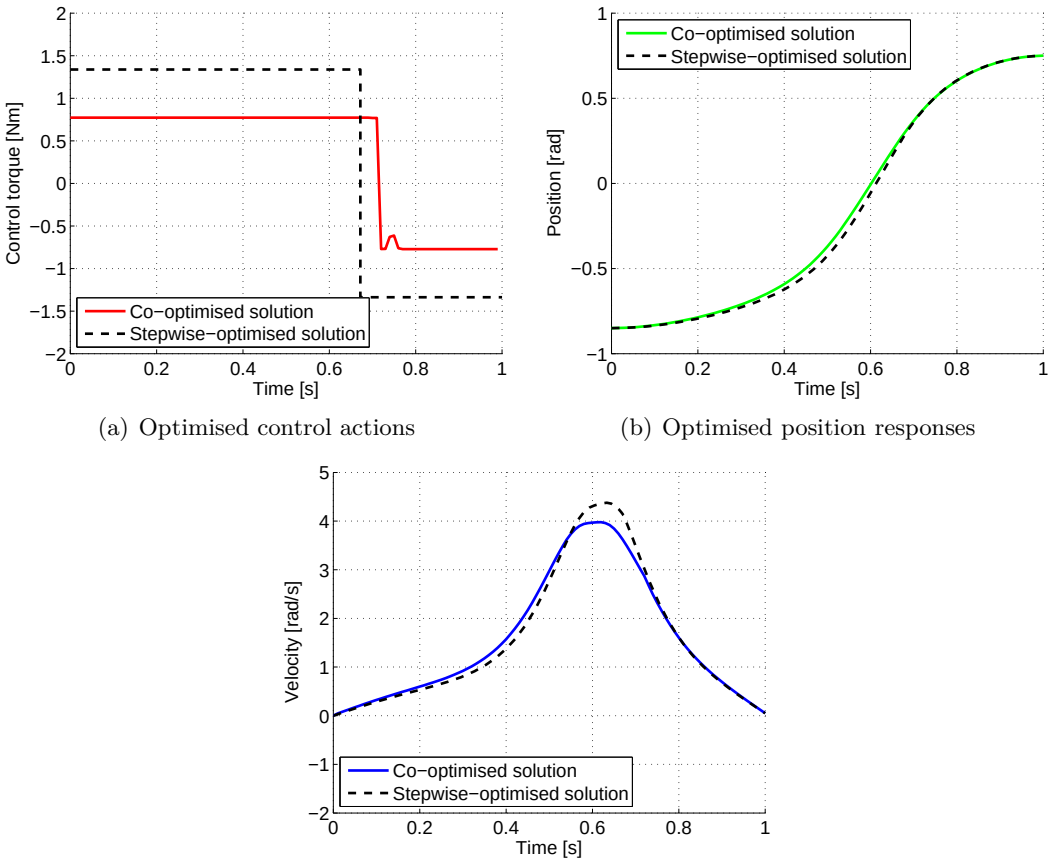
in which $n$ represents the number of the simulation.

The experimental setup has now been defined. In the next section the results of this experiment are discussed.

### 5-2-3 Analysis of Results

The goal of the 1-DOF RA experiment is to answer the question whether a (near) optimal solution can be found when co-optimising both a fuzzy controller and the system parameters of the RA using an EA. This questions is answered in this section by presenting the best co-optimised solution found after 20 optimisations with all controller-algorithm combinations. The best solutions was found by the CMA-ES algorithm in combination with the free fuzzy controller. An analysis of the performance of the other controller-algorithm combinations can be found in Section 5-4.

A first comparison between the co-optimised solution and the stepwise-optimised solution is presented in Figure 5-9, where the dashed lines correspond with stepwise-optimised solution and the solid lines correspond with the co-optimised solution. From this figure we have to conclude that the co-optimised solution is able to use an even lower maximum torque than the stepwise-optimised solution. In fact, when compared to the stepwise-optimised solution, the co-optimised solution is able to reduce the required maximum absolute control torque by 43%, from 1.338 $Nm$ to 0.768 $Nm$. In the next section the system parameters are analysed to see how this is possible.




(a) Optimised control actions

(b) Optimised position responses

(c) Optimised velocity responses

**Figure 5-9:** The control output and state response of the solution found by the EA starting from $[\Theta_{\text{init}}, \dot{\Theta}_{\text{init}}] = [-0.85, 0]$. The dashed lines represent the optimal control solution belonging to the stepwise-optimised system. It can be seen that the solution found by the EA is able to use a maximum torque which is less than the solution found in the stepwise optimisation. From this result we can conclude that the EA found better system parameter values, which allow a lower control torque.

### Analysing the System Parameters

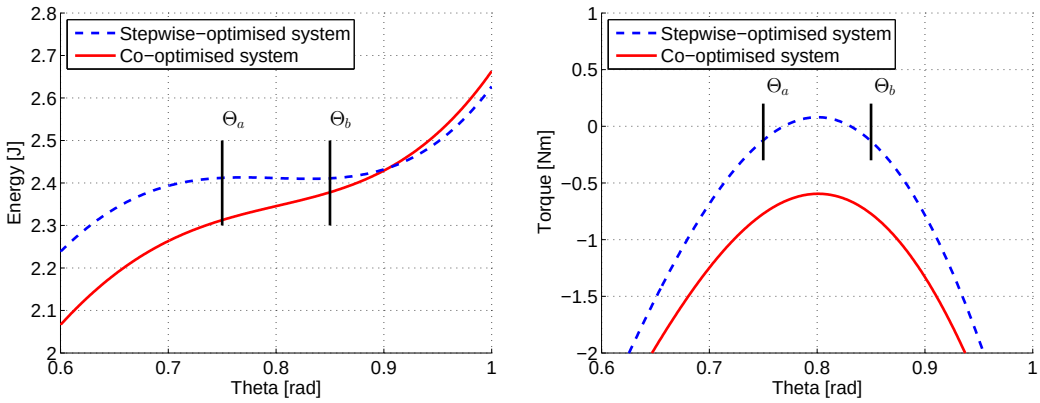
In Table 5-7 the parameter values of both solutions are presented and it can be seen that the parameter values of the spring radius at the large pulley $r_2$ and the transfer ration between

the pulleys $a$ found through evolutionary co-optimisation differ from the parameter values found after the stepwise optimisation.

**Table 5-7:** The system parameters values found through evolutionary co-optimisation next to the values found after a stepwise optimisation. The EA was able to find better values for the system parameters, which allowed a lower maximum torque to be used during the pick-and-place tasks.

| Parameter | Symbol | Stepwise optim. | Co-optim. | Units |
|---|---|---|---|---|
| Spring radius at large pulley | $r_1$ | 0.20 | 0.20 | $m$ |
| Spring radius at small pulley | $r_2$ | 0.0389 | 0.0351 | $m$ |
| Transfer ratio | $a$ | 5.1153 | 5.0745 | $m/m$ |
| Initial length of spring | $l_0$ | 0.05 | 0.05 | $m$ |
| Spring stiffness | $k$ | 200 | 200 | $N/m$ |
| Inertia (spring mechanism) | $I_0$ | 0.16 | 0.16 | $kgm^2$ |
| **Max. abs. control torque:** | | 1.338 | 0.768 | $Nm$ |

The influence of these parameters is shown in Figure 5-10 in which the potential energy and torque curves are plotted for both the stepwise-optimised solution and the co-optimised solution. Although the co-optimised solution uses the nonlinearity of the spring mechanism to minimise the torque at the position interval $[\Theta_a, \Theta_b]$, it did not minimise these torques to a minimum. Instead the maximum torque at the position interval is close to the torque used to move the system between the pick-and-place positions.




(a) Potential energy curve

(b) Torque curve

**Figure 5-10:** The potential energy and torque curves of the spring mechanism with parameter values obtained from a stepwise optimisation and an evolutionary optimisation. The EA seems to prefer a bigger torque at the pick-and-place intervals than the minimised torque found in the stepwise optimisation.

This shows that the co-optimised solution was able to decrease the overall maximum torque by increasing the torques at the position interval. The co-optimised solution uses the increased spring mechanism torques to accelerate the arm directly from the start and therefore less actuation torque is needed to perform the movement in time. This is a valuable insight, which was not discovered before the evolutionary co-optimisation was applied.

Eventually, this insight could also have been found through a further analysis of the mechanism. However, even when this was the case, this would result in an optimisation problem that is difficult to solve in a stepwise optimisation. This is caused by the fact that the optimal torque at the pick-and-place positions has to be equal to the optimal torque used to move between the pick-and-place positions. Therefore the dependency of the system parameters on the control solution is even stronger, which has to be solved through a cumbersome iterative process of system optimisation and control optimisation.

#### Analysing the Control Solution

To investigate the fuzzy control solution found by the EA, the optimal control solution for each of the three initial states has been solved for the co-optimised system.

In Figure 5-11 the actual control outputs and the optimal control outputs are shown. It can be seen that for all initial states the same maximum torque is used, even though a lower torque would have been able to bring the system to the desired state within the time constraint of 1 second. Moreover, the maximum torque used for all initial states is about equal to the optimal torque needed to bring the system from the furthest state to the desired state.

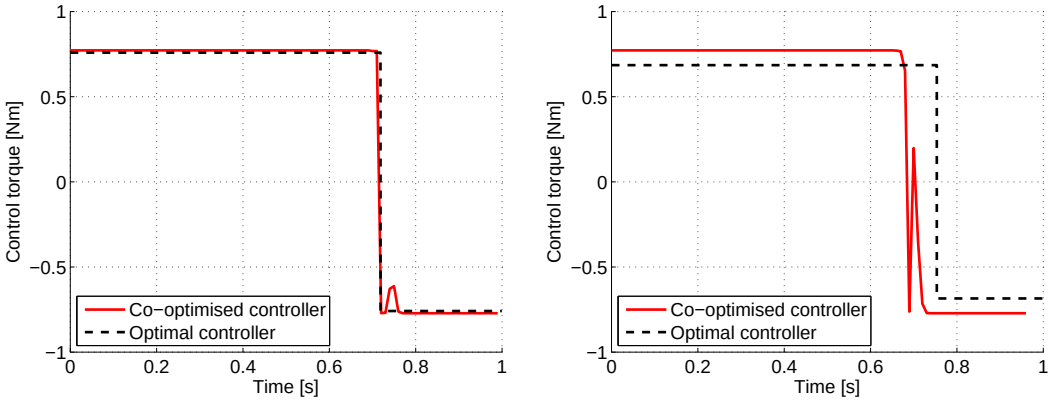
The reason why the EA only optimised the control for the furthest initial state can be found by analysing the optimisation goal stated at the beginning of this experiment. This goal demands a minimised actuator torque that allows the controller to move the system to a predefined region in the state space and is at the same time large enough to keep the system at all positions in this predefined region. The initial state furthest from the desired region requires the highest actuator torque and the EA will reduce this torque-to-move by increasing the torque needed to stay at the position interval. The minimal actuator torque for that initial state is found when the torque-to-move and the torque-to-stay have the same size. This torque-to-stay will now dominate the minimal actuation torque of the other initial states, and therefore a further minimisation of their torque-to-move does not influence the optimisation outcome. After optimising the fuzzy controller this resulted in equal torques-to-move for all initial states.

From a practical point of view this does not matter since the highest minimal maximum torque for one of the initial positions will set the requirements for the actuators used in the robot arm.

### 5-2-4 Conclusion

The results obtained in the 1-DOF RA experiments show that the co-optimisation of system and control parameters using evolutionary algorithms is an effective optimisation approach. The solutions found through evolutionary co-optimisation used a maximum absolute torque of 0.768 $Nm$, which is 43% less torque compared to the solution found through a conventional stepwise optimisation (1.338 $Nm$). Moreover, in the evolutionary co-optimisation no a priori knowledge about the advantageous system dynamics was required, while the stepwise optimisation required a thorough system analysis and assumptions to be made.

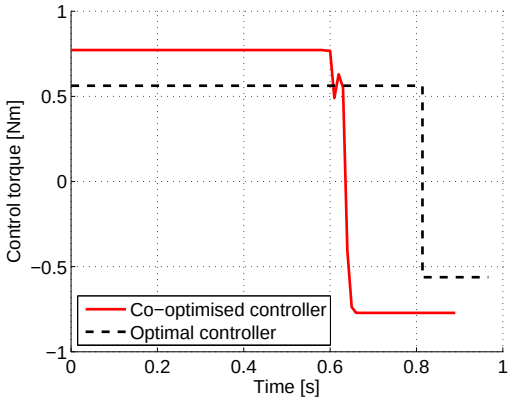
The solution found through evolutionary co-optimisation let to the insight that an increase of the maximum torque required to stay at the pick-and-place positions, can result in a lower




(a) Initial state: $\Theta = -0.85rad$, $\dot{\Theta} = 0rad/s$




(b) Initial state: $\Theta = -0.80rad$, $\dot{\Theta} = 0rad/s$




(c) Initial state: $\Theta = -0.75rad$, $\dot{\Theta} = 0rad/s$

**Figure 5-11:** The control output and state response starting from the three initial states. The dashed lines represent the optimal control solution and the optimal state response for each initial state. From the three plots we can see that the EA converged to a control solution at which an equal torque is used for all initial states, this torque equals the minimal torque needed to bring the system from the furthest state to the desired state in the desired time as can be seen in Figure (a). In Figures (b) and (c) the used torque is higher than the optimal maximal torque and therefore the system converges faster.

maximum torque needed to move between the pick-and-place positions. And that the overall torque used by the system is minimised when both of the torque-to-stay and the torque-to-move have the same size. Additionally, it was found that the minimal torque required by the system is fully determined by the minimised torque needed to move from the initial position that has the greatest distance from the desired position.

The near optimal solution found and the obtained insights lead to the conclusion that evolutionary co-optimisation is an effective approach to optimise the 1-DOF RA.

## 5-3 Optimising the 2-DOF Resonating Arm

In the previous two experiments it was shown that the combination of EAs and fuzzy control can effectively be used to find near optimal control and system parameter solutions. Based

**Table 5-8:** Specific parameters used in the 2-DOF RA experiment. Variable $m$ represents the number of fuzzy membership functions at each input for the fixed fuzzy controller and variable $b$ the total number of fuzzy basis functions used in the free fuzzy controller.

| Parameter | Value | Units |
|---|---|---|
| Number of generations | 10 000 | |
| Simulation time | 1 | $s$ |
| RK4 step size | 0.01 | $s$ |
| Membership functions (free fuzzy contr.): $m$ | 3 | |
| Basis functions (fixed fuzzy contr.): $b$ | 17 | |

on these results evolutionary co-optimisation will be used to optimise the fuzzy control and system parameters for the 2-DOF RA .

The optimisation goal of the 2-DOF RA experiment is equal to the goal stated for the 1-DOF RA experiment, which is:
*Find the fuzzy control and system parameters that minimise the maximum actuator torque required to perform certain pick-and-place tasks. This actuator torque should allow the fuzzy controller to move the system to a predefined region in the state space and it should be large enough to stay at all positions in this region.*

The predefined region in the state space is defined by the interval $[\Theta_a, \Theta_b] = [0.75, 0.85]$ $(rad)$ for the upper arm, an interval $[-0.05, 0.05]$ $(rad)$ for the lower arm and a maximum velocity of 0.05 $(rad/s)$ for both arms.

The experimental setup will be presented first, thereafter the best solution found during this experiment is presented and discussed.

## 5-3-1 Experimental Setup

Since the addition of the second arm makes the problem considerably more complex the number of generations and the controller parameters have been increased. These parameters together with the initial positions and the fitness function are discussed in this section.

### Parameter Values

In Table 5-8 the different parameters corresponding to the final experiment are presented. The extra arm mainly increases the complexity of the controller, due to the extra control output and two extra state inputs. This increased complexity makes the problem harder to solve and therefore the number of generations is increased to 10 000. The simulation time is still 1 second and a step size of 0.01 seconds for the RK4 method was still found suitable to simulate the system. The number of membership functions per input is set to 3, which yields a total optimisation problem of 186 variables. For the free fuzzy controller 17 fuzzy basis functions are used, which corresponds to a total of 182 variables.

#### Initial States

In this experiment the initial state of the extra lower arm is set to zero, which means that it is always pointing away from the spring mechanism. The initial states of the upper arm are equal to the ones used in the 1-DOF RA experiment. These starting positions are:

$$[\Theta_{1,\text{init}}, \dot{\Theta}_{1,\text{init}}, \Theta_{2,\text{init}}, \dot{\Theta}_{2,\text{init}}] \in \{[-0.85, 0, 0, 0],\ [-0.8, 0, 0, 0],\ [-0.75, 0, 0, 0]\} \tag{5-35}$$

#### Fitness Function

The fitness function used for this optimisation is similar to the one used in the 1-DOF RA optimisation, which was presented in Algorithm 5. The only changes are the fact that the error now consists of four values and that two torques are used to move the system from the initial positions to the desired positions. The maximum of the two torques-to-move and the torque-to-stay is used to determine the fitness score, this means that the overall maximum torque used in the system is minimised and no distinction is made between the upper and lower arm actuators.

### 5-3-2 Analysis of Results

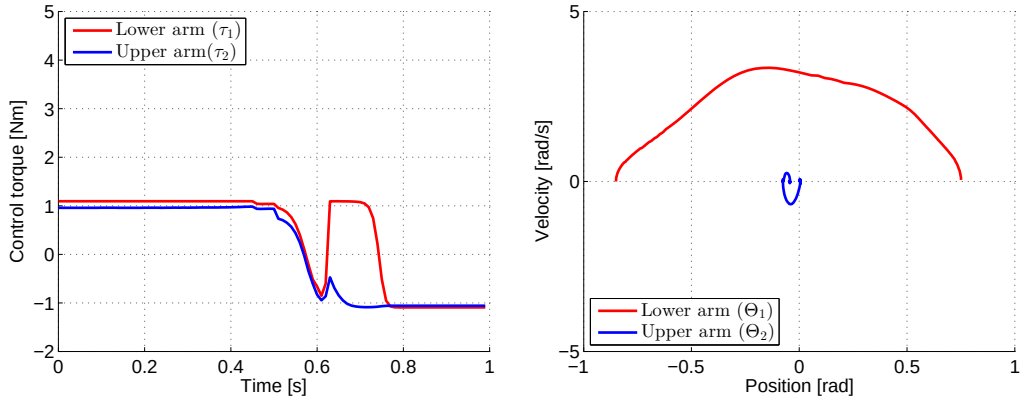
The goal of the 2-DOF RA experiment is to investigate the results found when co-optimisation is applied on the complete system with both the upper and lower arm. In this section the best solution is presented, which was found by the CMA-ES algorithm in combination with the free fuzzy controller. A comparison between the two fuzzy controllers and EAs will be given in the next section (Section 5-4).

In Figure 5-12(a) the control output for one of the initial states is plotted. The maximum absolute torque used during the pick-and-place movement is 1.1 $Nm$ and the maximum absolute torque-to-stay at the pick-and-place locations is also approximately 1.1 $Nm$ as shown in Figure 5-13(b). The fact that these torques have the same size indicates that the optimisation did indeed optimise the system and control parameters to a near optimal solutions, by finding the right balance between the torque-to-move and the torque-to-stay.

In Figure 5-13(a) the potential energy curve corresponding to the 2-DOF system is plotted and compared with the potential energy curve of the 1-DOF system. Here it can be seen that the level of potential energy stored in the optimised 2-DOF RA is considerably lower than the potential energy stored in the optimised 1-DOF RA. This may be the result of a sub optimal convergence, but it is also possible that a higher level of potential energy will not allow a lower torque-to-move. This latter explanation finds support in the findings of Plooij, who observed that the sudden acceleration of the upper arm caused by the spring mechanism increases the torque required to control the lower arm.

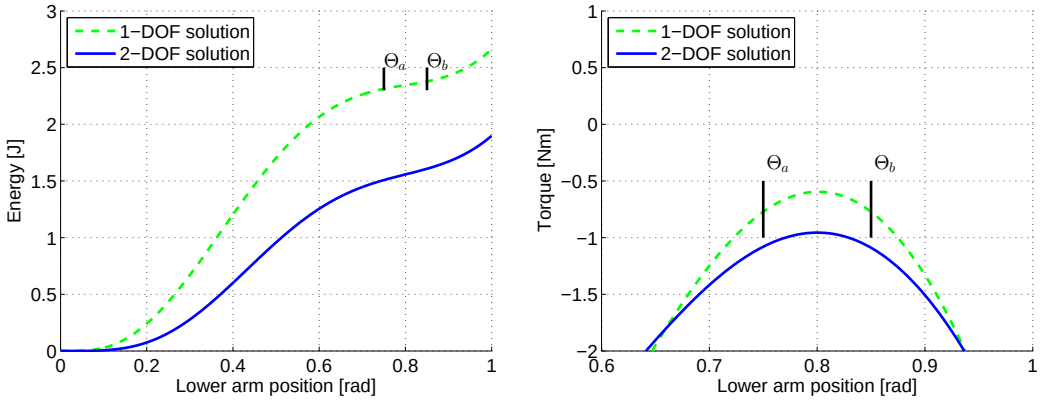
### 5-3-3 Conclusion

Co-optimisation of fuzzy control and system parameters of the 2-DOF RA results in working solutions that are able to bring the upper and lower arm from their initial position to the desired position.




(a) Control outputs (b) Phase plane

**Figure 5-12:** The control outputs and the phase trajectories of the the actuators and arm angles, respectively, from an initial state of $\Theta_1 = -0.85$, $\dot{\Theta}_1 = 0$, $\Theta_2 = 0$ and $\dot{\Theta}_2 = 0$.




(a) Potential energy (b) Torque

**Figure 5-13:** The potential energy and torque curves corresponding to the best solutions found for the 1-DOF system and the 2-DOF system.

Similar to the 1-DOF RA co-optimised solution, the value of the maximum absolute torque required to stay at the pick-and-place positions is similar to the maximum absolute torques needed to move the upper and lower arm to the desired state. This indicates that the algorithm did balance the torque-to-stay and the torque-to-move in order to decrease the overall torque of the system.

The level of potential energy stored in the 2-DOF RA solution is considerably less than the potential energy stored in the 1-DOF RA solution. This might indicate that a high potential energy level is less beneficial for the minimisation of the torques for both arms.

## 5-4 Discussion and Analysis

While the previous sections focused on the best solutions found in the experiments this last section will discuss and analyse the performance of the different fuzzy controllers and EAs

used in each of the conducted experiments.

### 5-4-1 Comparing the Fuzzy Controllers

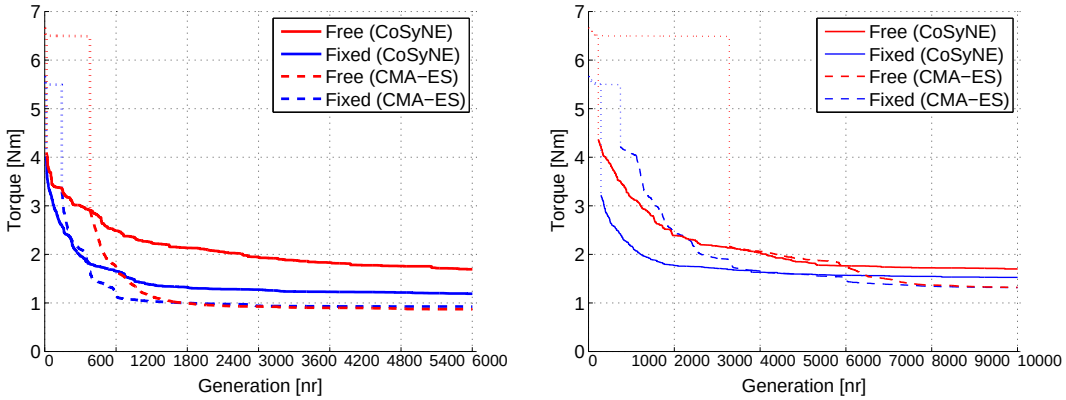
For all three experiments conducted in this research the best control solution were found when using the *free fuzzy controller*.

The limited approximation capabilities of the fixed fuzzy controller in comparison to the free fuzzy controller are shown in Figure 5-14. In this figure the best control surfaces of the free and fixed fuzzy controllers are presented for both the single mass and 1-DOF RA experiment. In Figure 5-14(b) a clear stair-shaped approximation of the switching line is visible which indicates the limitation of the fixed membership functions and for this reason the fixed fuzzy controller performed considerably worse in comparison to the free fuzzy control. However, for the 1-DOF RA experiment the performances differences between the fixed and free fuzzy controllers are considerably less. An explanation can be found in Figure 5-14(c) and 5-14(d) where the control surfaces of the free and fixed fuzzy controller optimised for the 1-DOF RA are presented. The state trajectories in the 1-DOF RA experiment are much closer together than the state trajectories in the single mass experiment, therefore the required approximation of the optimal control surface is reduced to a smaller area of the state. This is relatively easier to approximate with the fixed fuzzy controller.

It appears to be difficult to derive general control rules from the optimised fuzzy rules. In Figure 5-14(a) and 5-14(c) the locations of the optimised fuzzy rules are indicated by crosses. The dashed ellipses represent the area in which the rule is most active (two times the width parameter $\sigma$ of the Gaussian membership functions). In order to approximate the step function of the optimal controller the membership functions are positioned far from each other and their widths are in most cases considerably small. This, however, does not mean that their overall influence in the state space is small. This is due to the weighted average inference method which makes the influence of one rule dependent on the locations and widths of all other rules. Therefore little control information can be attained from analysing each control rule separately.

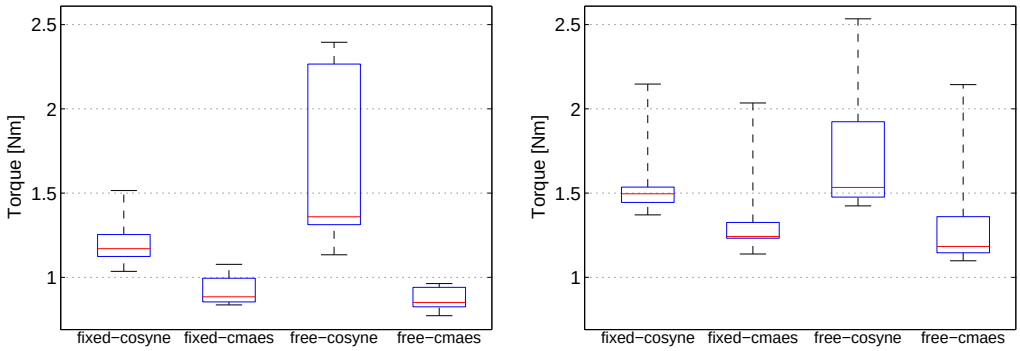
The reduced complexity of the fixed fuzzy controller in comparison to the free fuzzy controller results in a faster initial convergence as presented in Figure 5-15. In this figure an average is given of the torques used by the solutions found during 20 optimisations. It can be seen that for both the 1-DOF and 2-DOF RA the average torque of the fixed fuzzy controllers (blue lines) show a faster convergence than the average torque of the free fuzzy controllers (red lines). In this figure it can also be seen that for the CoSyNE algorithm the fixed fuzzy controller yields an even better average performance than the free fuzzy controller.

The reason why the best solutions where still found with the free fuzzy controller can be seen in Figure 5-16 in which the torques of the optimised solutions from the different controller-algorithm combinations are presented in box plots. Although the average values of the solutions found by the fixed controller with CMA-ES and free controller with CMA-ES combinations are not so far apart, the lowest torque value is always found by the CMA-ES algorithm in combination with the free fuzzy controller for both the 1-DOF and 2-DOF RA. It also appears that the consistency of the fixed control solutions is better in comparison to the free control solutions. This may be caused by a smaller amount of local optima due to the decreased complexity of the fixed fuzzy controller.




(a) 1-DOF RA (b) 2-DOF RA

**Figure 5-15:** The average torques of the best solutions found during the optimisation runs of the four controller-algorithm combinations. For both the 1-DOF and 2-DOF RA each combination was tested 20 times. The blue lines represent the optimisation of the fixed fuzzy controller and the red lines that of the free fuzzy controller. One can see that on average the fixed fuzzy controller shows a faster initial converges than the free fuzzy controller.




(a) 1-DOF RA (b) 2-DOF RA

**Figure 5-16:** Boxplots representing the lower quartile, median, upper quartile, the smallest and largest sample of the minimised torques of 20 solutions found for each of the four different controller-algorithm combinations.

## 5-4-2 Comparing the Evolutionary Algorithms

For all experiments the best results where found by the *CMA-ES algorithm* and it therefore appears to be the most effective algorithm for the optimisation of fuzzy controllers and system parameters.

In Figure 5-17 the convergence of both algorithms is presented when optimising the 1-DOF RA with both types of controllers. The dotted lines indicate that the average fitness does not equal the average maximum torque used since some of the simulations obtained a penalty for not converging to the desired states. When the convergence line is solid the values correspond with the average maximum torque of all optimisation runs.

From the convergence plots it can indeed be see that the CMA-ES algorithm obtained a better solution than the CoSyNE algorithm for both controller types, however, the CoSyNE algorithm shows a faster initial convergence.

The best 1-DOF RA solutions were found when applying the free fuzzy controller in combination with the CMA-ES algorithm. The cpu time required to perform the 6000 generations was between 3 and 4.5 hours on a 3.10 GHz Intel Core i5-2400 desktop computer. In the convergence plot (Figure 5-17(b)) it can be seen that after 1500 generations 75% of the runs has already converged to a torque value below 1.3 Nm. This means that after 1 to 1.5 hours of evolutionary co-optimisation 3/4 of the runs will already produce solutions which outperform the stepwise optimised solution.

In Figure 5-18 the convergence of both algorithms is presented when optimising the 2-DOF RA. Again, it can be seen that the CMA-ES algorithm yields the deepest convergence when compared to the CoSyNE algorithm.

Also for the 2-DOF RA problem the best solutions were found when applying the free fuzzy controller in combination with the CMA-ES algorithm. The cpu time required to perform the 10 000 generations was between 25 and 30 hours on a desktop computer (3.10 GHz Intel Core i5-2400). This increase in CPU time is not only caused by the higher number of generations, but also due to the higher complexity of the model and the controller, which requires more parameters to be optimised and more time to for simulation. Although the consistency of the solutions found after 10 000 generations is considerably low, it can also be seen that after 5 000 generations (12.5 to 15 hours) already 50% of the optimisation runs yielded in solutions with a fitness score close to the final fitness score.

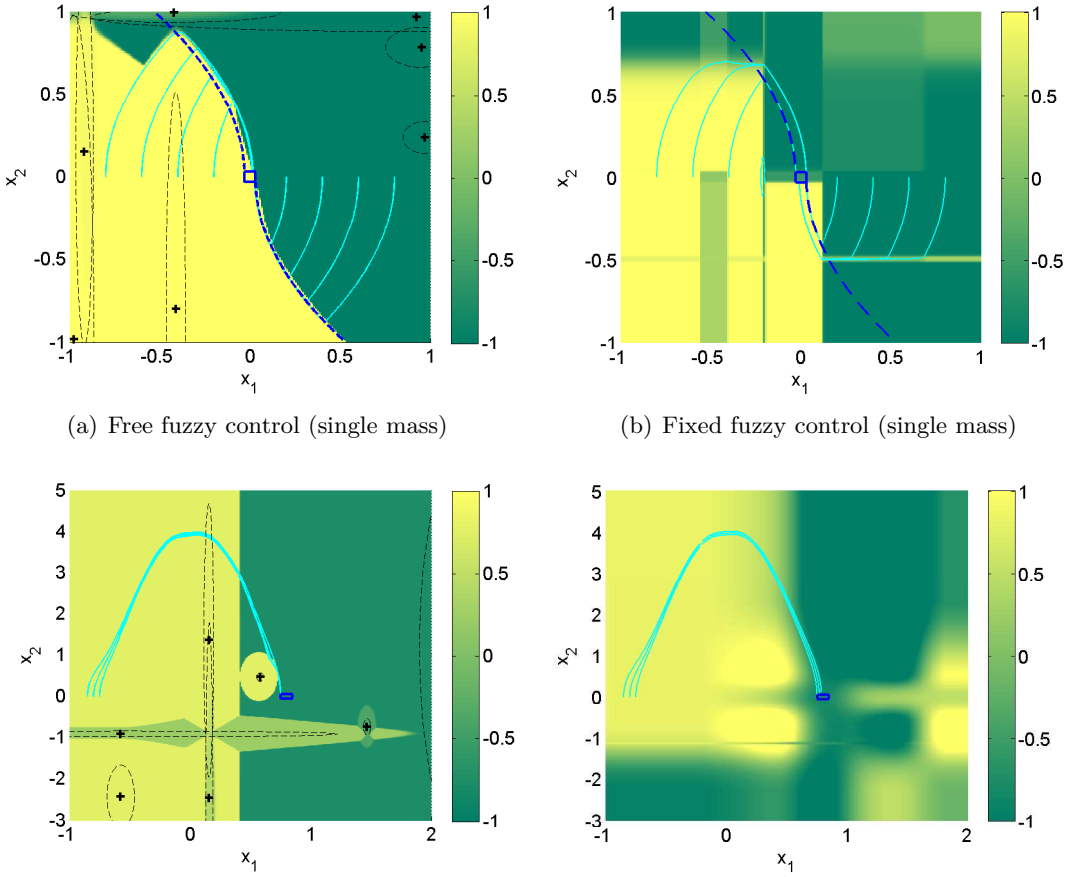
The deeper convergence debt of the CMA-ES algorithm can be explained by the fact that the CMA-ES algorithm seems to be better in optimising the system parameters. In Figure 5-19 the normalised parameter values are plotted and one can see that the CMA-ES algorithm is constantly finding better systems while the CoSyNE algorithm changes the best found system parameters less often. This is assumed to be the result of the adaptive mutation size used in the CMA-ES algorithm, which allows the algorithm to change the system parameters more gradually.

In Figure 5-20(a) the potential energy curves corresponding to the different system parameter solutions found by the CMA-ES and CoSyNE algorithm for the 1-DOF RA are presented. This plot shows that the CMA-ES algorithm gives a more consistent convergence of the system parameters than the CoSyNE algorithm. For the 2-DOF RA this consistency is lower as can be seen in Figure 5-20(b), however the CMA-ES algorithm is still better than the CoSyNE algorithm in reducing the slope of the potential energy curve in the place interval, which might explain why it is able to reduce the torque further than the CoSyNE algorithm.

### 5-4-3 Conclusion

The best combination of fuzzy controllers and evolutionary algorithms used in this research is the *free fuzzy controller* in combination with the the *CMA-ES algorithm*. The free membership functions allowed a better approximation of the optimal control surface for the single mass experiment and yielded better control solutions for the 1-DOF and 2-DOF RA experiments. However, in general the decreased complexity of the fixed fuzzy controller resulted in a faster convergence with a better consistency of the final performance of the solutions.

The CMA-ES algorithm showed a deeper convergence in all experiments and has shown to give more consistent solutions for the parameter values. Although CoSyNE is showing a faster initial convergence than CMA-ES at the beginning of the optimisation, it is not able to converge as deep as CMA-ES.



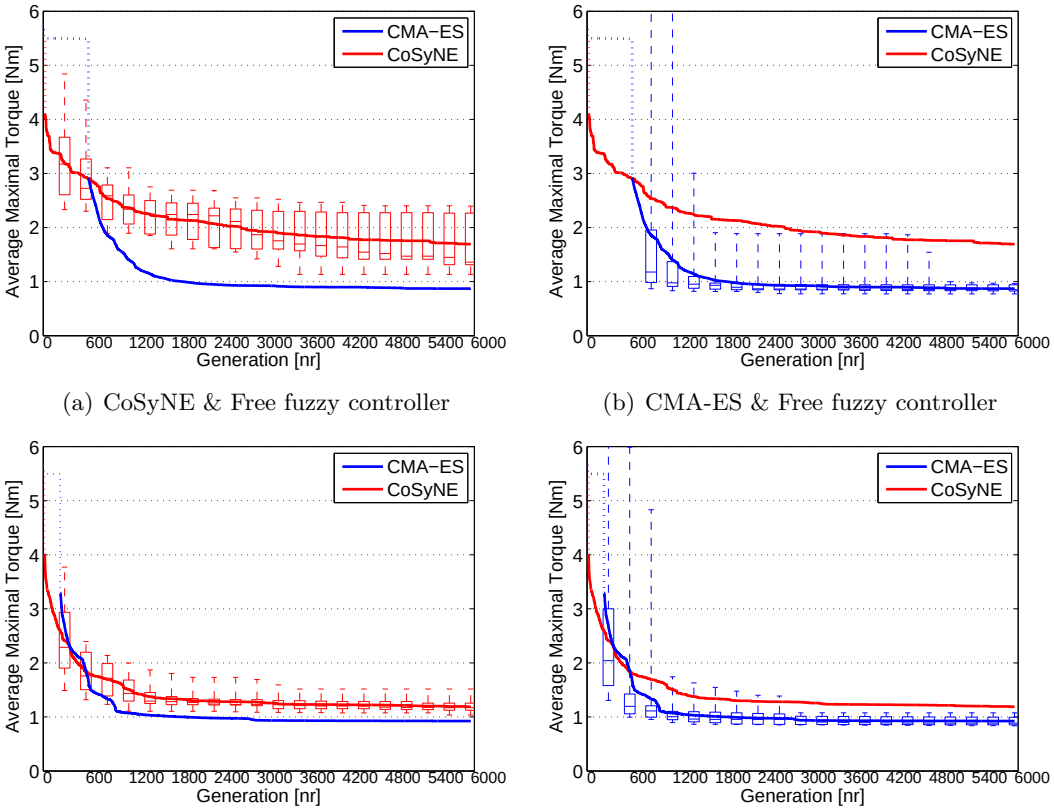

(a) Free fuzzy control (single mass)

(b) Fixed fuzzy control (single mass)

(c) Free fuzzy control (1-DOF RA)

(d) Fixed fuzzy control (1-DOF RA)

**Figure 5-14:** The control surfaces generated by the best solutions found for the free and fixed fuzzy controllers in the single mass and 1-DOF RA experiment. The value of the control action at each state is presented by a colour, where dark green stands for a control action of -1 and bright yellow represents a control action of $+1$. All control actions in between are indicated by the colour scales. The solid lines represent the state trajectories of the simulated systems starting from the different initial positions. Additionally, the optimal switching line is plotted for the single mass controllers and for the free fuzzy controllers the locations of the optimised fuzzy base functions are shown (crosses). The dashed ellipses represent the shape of the fuzzy base functions.



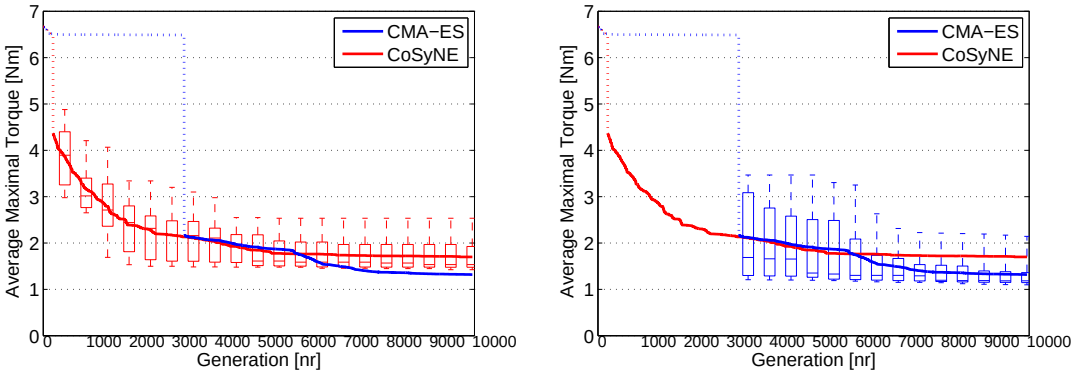

(a) CoSyNE & Free fuzzy controller

(b) CMA-ES & Free fuzzy controller

(c) CoSyNE & Fixed fuzzy controller

(d) CMA-ES & Fixed fuzzy controller

**Figure 5-17:** The convergence of the four controller-algorithm combinations for the 1-DOF RA experiment. The plotted line represents the median of 20 runs and the box represents the lower quartile and the upper quartile, the lines connected to the box show the smallest and largest sample. Although CoSyNE shows a faster initial convergence for the free fuzzy controller, the CMA-ES algorithm is able to converge to a better solution for both types of controllers.



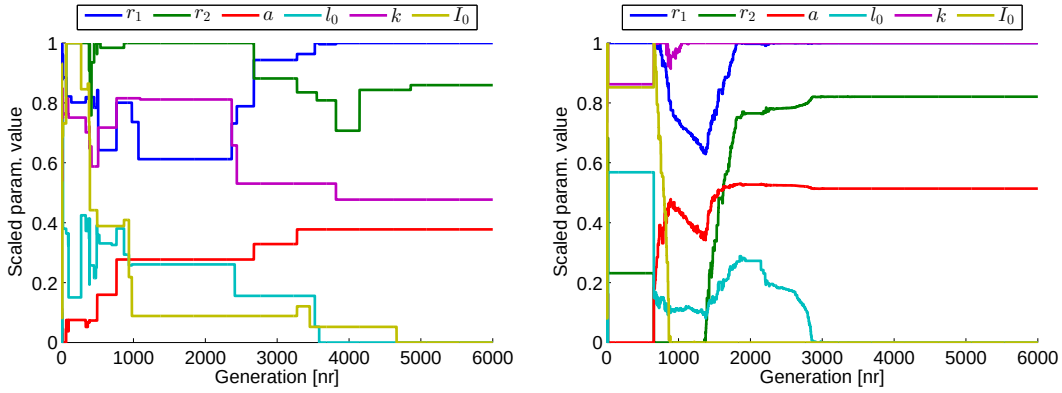

(a) CoSyNE & Free Gaussian MFs

(b) CMA-ES & Free Gaussian MFs

(c) CoSyNE & Fixed Gaussian MFs

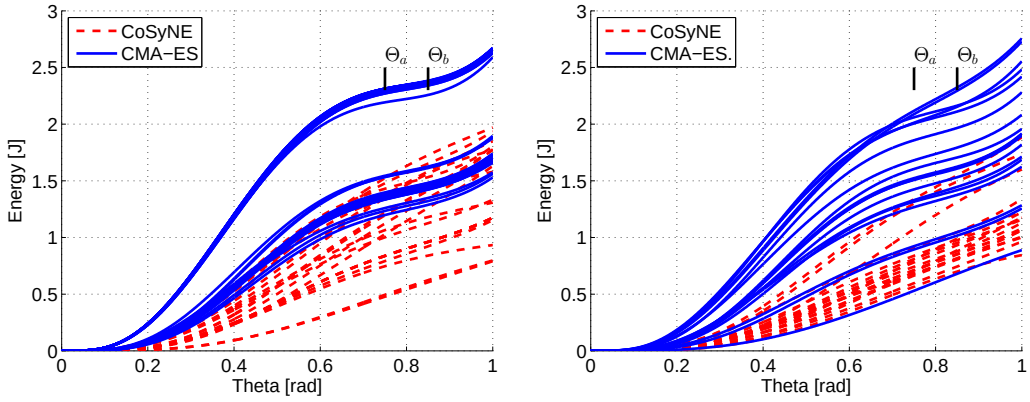
(d) CMA-ES & Fixed Gaussian MFs

**Figure 5-18:** The convergence of the four controller-algorithm combinations for the 2-DOF RA experiment. The plotted line represents the median of 20 runs and the box represents the lower quartile and the upper quartile, the lines connected to the box show the smallest and largest sample. Although CoSyNE shows a quicker convergence for the free fuzzy controller, the CMA-ES algorithm is able to converge to a better solution for both types of controllers.




(a) System parameters 1DOF - CoSyNE (b) System parameters 1DOF - CMA-ES

**Figure 5-19:** The normalised system parameters plotted for each generation when optimised by the CoSyNE and CMA-ES algorithm. In these plots 0 represents the lower bound and 1 the upper bound of the particular system parameter. The figures show that the CMA-ES algorithm is able to give a faster convergence to a system parameter solution while the CoSyNE algorithm is changing the system parameters with bigger steps and does not converge as quickly.




(a) 1-DOF RA (b) 2-DOF RA

**Figure 5-20:** The potential energy curves of all the different system parameters combinations found after optimising the 1-DOF RA (a) and the 2-DOF RA (b) with the CMA-ES (20 runs) and CoSyNE algorithm (20 runs). It can be seen that the CMA-ES algorithm gives more consistent results in comparison to the CoSyNE algorithm when optimising the 1-DOF RA. Considering the results of the 2-DOF RA we see that the both algorithms lack consistency but the CMA-ES algorithm performed better in shaping the potential energy curve to decrease the torques at the pick-and-place positions.

# Chapter 6

# Conclusion and Recommendations

This thesis introduced the use of evolutionary co-optimisation of control and system parameters in the development of the Resonating Arm (RA); a novel concept for a low-power pick-and-place robot arm. Co-optimisation of robotic designs combines the optimisation of control and system parameters into one optimisation problem in order to find the optimal combination of the system as a whole. To solve this parallel optimisation problem Evolutionary Algorithms (EAs) have been used which are known for their ability to find near optimal solutions to complex and high dimensional optimisation problems. The optimisation goal of the RA is set equal to the minimisation of the maximum torque needed to stay and move between the pick-and-place positions.

Three experiments have been conducted to investigate the benefits and limitations of evolutionary co-optimisation of the RA. In each experiment two types of fuzzy controllers (with free and fixed membership positions) and two types of EAs (CoSyNE and CMA-ES) were applied and each of the four combinations was investigated.

## 6-1 Conclusion

In the beginning of this thesis the question has been raised to what extent evolutionary co-optimisation is able to find (near) optimal system and fuzzy control solutions for the Resonating Arm. From the obtained results we conclude that evolutionary co-optimisation is an effective approach to find near optimal solutions for the RA with one degree of freedom (1-DOF), however more research is needed to effectively use it in the optimisation of the 2-DOF RA.

The results from the *first experiment*, in which a single mass minimal-time control problem was solved, showed that the combination of fuzzy control and EAs is well able to generate near optimal feedback control behaviours.

In the *second experiment* the 1-DOF RA has been optimised both in a conventional stepwise manner and through evolutionary co-optimisation. It showed that the stepwise optimisation

of control and system parameters requires a deep knowledge of the system and results in a tedious and iterative process, which can easily end in suboptimal solutions. On the contrary, evolutionary co-optimisation resulted in near optimal solutions requiring little knowledge of the system. The 1-DOF RA solution found through evolutionary co-evolution required 43% less torque to perform the pick-and-place tasks when compared to the stepwise optimised 1-DOF RA solution.

Two new insights could be derived after analysing the 1-DOF RA solutions found through evolutionary co-optimisation. First, it was shown that an increase of the torques needed to stay at the pick-and-place positions allows a decrease of the torques needed to move between the pick-and-place positions, which in the optimal case are equal to each other. Secondly, it was found that the maximum torque required by the system is fully determined by the torque needed to fulfil the task beginning from the initial position that is the furthest from the final state.

The *third experiment* considered the optimisation of the 2-DOF RA and it was shown that evolutionary co-optimisation will generate solutions that are able to fulfil the pick-and-place tasks. However, the consistency of the solutions was low and it is therefore difficult to say whether these solutions are near optimal.

In order to answer the *research subquestions*, the performances of the two fuzzy controllers and EAs have been compared. Although the membership functions with fixed positions resulted in a faster convergence, better control solutions where found when using free membership functions. Between the two EAs, CMA-ES yielded the best and most consistent solutions for all experiments. CoSyNE did show a faster convergence in the beginning of the optimisation but was not able to converge to near optimal solutions.

Although the findings in this work are the results of one specific optimisation problem, the obtained conclusions are expected to be generally applicable to problems with slightly different optimisation goals or system properties (i.e. the minimisation of energy use or a different positioning of actuators). Moreover, the optimisation environment developed in this work can easily be altered to solve these problems. This makes it a useful tool for future optimisations of the Resonating Arm and systems alike.

## 6-2 Recommendations

More research is still needed in order to improve the effectiveness of evolutionary co-optimisation in the optimisation of industrial manipulators. Especially considering the low consistency of the solutions found when optimising the 2-DOF RA.

One aspect which needs further attention is the population size. When more solutions are evaluated in one generation the convergence speed will decrease, but the global search performance will improve. This might yield better solutions for the more complex 2-DOF RA optimisation problem.

The performance could also be improved by implementing restarting techniques. These techniques restart the EA when it is assumed to be stuck in a local optimum, in order to reinitiate a broader search for the global optimum. Information about the search space can be preserved by initialising the new generation with some of the old solutions from the last optimisation [58].

Another aspect that could improve the effectiveness of evolutionary co-optimisation is the selection of the controller. Since the rule base structure of the fuzzy control approach did not seem to bring much advantages, other control strategies, such as neural networks and fuzzy neural networks should be considered.

Moreover, evolutionary co-optimisation could be applied on more detailed models of the RA. The detail of these models could be increased by; incorporating the actuator dynamics, using a different friction model for the lower arm and letting the friction of the upper arm be dependent on the force of the spring mechanism acting on the pivot point. Also other morphological concepts could be investigated by using evolutionary co-optimisation. For example concepts that use a nonlinear spring or have a second spring mechanism to control the lower arm.

This work sets one of the first steps in the application of evolutionary co-optimisation for the development of industrial manipulators. By the means of future research more steps can be taken in order to turn evolutionary co-optimisation into a generally applicable and fruitful way of designing the industrial robots of the future.

# Appendix A

# Equations of Motion Derivation using TMT method

The TMT method is a simple, clear and computational efficient approach to derive the equations of motion for a multibody system. It combines the Lagrangian approach with the Newton-Euler method, which allows the use of independent generalised coordinates while using the simple Newton-Euler mass and force matrices [28].

By using this TMT method it becomes relatively easy to adjust the configuration of masses and arms in order to derive the equations of motion for different arm and/or mass configurations. This can become beneficial for future optimisations in which a more detailed mass distribution is demanded or extra arms have to be added to the system.

In this appendix the derivation of the TMT-method is presented.

From Newton we know that the sum of the applied forces equals the mass times the acceleration in all directions and Euler showed us that the sum of applied moments at the centre of mass equals the moment of inertia at the centre of mass times the angular acceleration, therefore a mass moving in two dimensions can be described by

$$\begin{bmatrix} \Sigma F_x \\ \Sigma F_y \\ \Sigma \tau \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\Theta} \end{bmatrix}. \tag{A-1}$$

If the system is described by multiple masses we can write their Newton-Euler relations as

$$\Sigma \mathbf{f} = \mathbf{M}\ddot{\mathbf{x}} \quad \text{or} \tag{A-2}$$

$$\Sigma \mathbf{f} - \mathbf{M}\ddot{\mathbf{x}} = \mathbf{0} \tag{A-3}$$

where $\mathbf{x}$ represent all positions and angles, $\mathbf{f}$ the force vector holding all forces and torques acting on the masses and $\mathbf{M}$ the mass matrix with on its diagonal all masses and inertia.

Generally the masses in a system are connected to each other in some way, this implies the existence of constraints on some of the positions and angles in the system. Using the concept

of virtual power we know that a mechanical system is in equilibrium if the virtual power is zero for all virtual velocities that satisfy the constraints

$$\delta\dot{\mathbf{x}}\left\{\Sigma\mathbf{f} - \mathbf{M}\ddot{\mathbf{x}}\right\} = 0. \tag{A-4}$$

As mentioned we would like to express all coordinates present in state $\mathbf{x}$ in terms of the independent generalised coordinates $\mathbf{q}$. This is done by constructing a kinematic transformation $\mathbf{T}$ as in

$$\mathbf{x} = \mathbf{T}(\mathbf{q}) \tag{A-5}$$

and the corresponding velocities become

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{T}}{\partial \mathbf{q}} = \mathbf{T}_\mathbf{q}\dot{\mathbf{q}}, \text{ and the virtual velocities} \tag{A-6}$$

$$\delta\dot{\mathbf{x}} = \mathbf{T}_\mathbf{q}\delta\dot{\mathbf{q}} \tag{A-7}$$

Replacing the virtual velocities in (A-4) yields

$$\mathbf{T}_\mathbf{q}\delta\dot{\mathbf{q}}\left\{\Sigma\mathbf{f} - \mathbf{M}\ddot{\mathbf{x}}\right\} = 0 \tag{A-8}$$

since the generalised coordinates $\delta\dot{\mathbf{q}}$ are independent it must be that

$$\mathbf{T}_\mathbf{q}\left\{\Sigma\mathbf{f} - \mathbf{M}\ddot{\mathbf{x}}\right\} = 0 \tag{A-9}$$

Also the accelerations $\ddot{\mathbf{x}}$ can be expressed in independent generalised coordinates by differentiating equation (A-5) twice yielding

$$\ddot{\mathbf{x}} = \frac{\partial \mathbf{T}_\mathbf{q}\dot{\mathbf{q}}}{\partial \mathbf{q}} \tag{A-10}$$

$$= \mathbf{T}_\mathbf{q}\frac{\dot{\mathbf{q}}}{\partial \mathbf{q}} + \frac{\partial \mathbf{T}_\mathbf{q}}{\partial \mathbf{q}}\dot{\mathbf{q}} \tag{A-11}$$

$$= \mathbf{T}_\mathbf{q}\ddot{\mathbf{q}} + \mathbf{h} \tag{A-12}$$

in which $\mathbf{h} = \frac{\partial \mathbf{T}_\mathbf{q}}{\partial \mathbf{q}}\dot{\mathbf{q}}$ is called the convective acceleration.

By substituting (A-12) in (A-8) we are now able to express the complete equations of motion in terms of independent coordinates

$$\mathbf{T}_\mathbf{q}\left\{\Sigma\mathbf{f} - \mathbf{M}\{\mathbf{T}_\mathbf{q}\ddot{\mathbf{q}} + \mathbf{h}\}\right\} = 0 \tag{A-13}$$

or in matrix vector notation:

$$\bar{\mathbf{M}}\ddot{\mathbf{q}} = \bar{\mathbf{f}} \tag{A-14}$$

| | |
|---|---|
| with the reduced mass matrix: | $\bar{\mathbf{M}} = \mathbf{T}^T\mathbf{M}\mathbf{T}$ |
| the first order kinematic transfer function: | $\mathbf{T} = \mathbf{T}_\mathbf{q}$ |
| and the reduced force vector: | $\bar{\mathbf{f}} = \mathbf{T}^T[\Sigma\mathbf{f} - \mathbf{M}\mathbf{h}]$ |

# Appendix B

# Equations of Motion

The equations of motion for the Resonating Arm with one and two degrees-of-freedom:

**1-DOF Resonating Arm:**

$$\begin{bmatrix} \Theta_1 \\ \dot{\Theta}_1 \end{bmatrix} \frac{d}{dt} = \begin{bmatrix} \dot{\Theta}_1 \\ \frac{(\tau_1+\tau_s)}{I_0+\frac{64}{375}} \end{bmatrix}$$

**2-DOF Resonating Arm:**

$$\begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix} \frac{d}{dt} = \begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \\ \frac{-\sin(2\,\Theta_1-2\,\Theta_2)\,\dot{\Theta}_1^2\,1089-\sin(\Theta_1-\Theta_2)\,\dot{\Theta}_2^2\,2112+12000\,(\tau_1+\tau_s)+12375\,\tau_2\left(2\sin\left(\frac{\Theta_1}{2}-\frac{\Theta_2}{2}\right)^2-1\right)}{2178\sin(\Theta_1-\Theta_2)^2+12000\,I_0+638} \\ \frac{\sin(2\,\Theta_1-2\,\Theta_2)\,\dot{\Theta}_2^2\,2178+33000\,\tau_2+140625\,I_0\,\tau_2+24750\,(\tau_1+\tau_s)\left(2\sin\left(\frac{\Theta_1}{2}-\frac{\Theta_2}{2}\right)^2-1\right)+\sin(\Theta_1-\Theta_2)\left(24750\,I_0\,\dot{\Theta}_1^2+5808\,\dot{\Theta}_1^2\right)}{4356\sin(\Theta_1-\Theta_2)^2+24000\,I_0+1276} \end{bmatrix}$$

where $\Theta_1$, $\dot{\Theta}_1$, $\Theta_2$ and $\dot{\Theta}_2$ represent the generalised coordinates of the Resonating Arm system and their derivatives. The variables $\tau_1$, $\tau_2$ and $\tau_s$ represent the two actuator torques and the torque applied by the spring mechanism, respectively.

# Appendix C

# Spring Mechanism Equation

The equation describing the torque $\tau_s$ of the spring mechanism:

$$\tau_s = \; k\,A\,\left(\left({r_1}^2 + l_0\,r_1 + r_1\,r_2\right)\,\sin(\Theta_1) + \left(a\,{r_2}^2 + a\,l_0\,r_2 + a\,r_1\,r_2\right)\,\sin(a\,\Theta_1) + \left(a\,r_1\,r_2 - r_1\,r_2\right)\,\sin(\Theta_1 - a\,\Theta_1)\right)$$

$$A = \left(\frac{l_0}{\sqrt{\left(l_0 + r_1 + r_2 - r_2\,\cos(a\,\Theta_1) - r_1\,\cos(\Theta_1)\right)^2 + \left(r_2\,\sin(a\,\Theta_1) + r_1\,\sin(\Theta_1)\right)^2}} - 1\right)$$

# Appendix D

# Pontryagin's Minimum Principle

The Pontryagin's Minimum Principle (PMP) is used in optimal control [57] to determine an analytical solution to control problems of the form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{D-1}$$

with $\mathbf{x}$ the state of the system and $\mathbf{u}$ the control action to be optimised.

The PMP minimises a certain performance index $J(\mathbf{u}(t))$, which is defined by a cost-to-go $V(\mathbf{x}(t), \mathbf{u}(t))$ at each time instance $t_0 \leq t \leq t_f$ and a final-cost $S(\mathbf{x}(t_f))$ of the end state at time $t_f$ as shown below,

$$J(\mathbf{u}(t)) = S(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} V(\mathbf{x}(t), \mathbf{u}(t))\, dt. \tag{D-2}$$

This performance index is used to construct the Hamiltonian $\mathcal{H}$

$$\mathcal{H}(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t)) = V(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda}^T(t)\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{D-3}$$

with $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ the system equations and $\boldsymbol{\lambda}^T(t)$ the transpose of the the costate vector $\boldsymbol{\lambda}(t)$, which is yet unknown. This costate vector can be interpreted as a Lagrange multiplier or dummy variable often used in the minimisation of functions subject to constraints.

For this Hamiltonian it holds that

$$\mathcal{H}(\mathbf{x}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t)) \leq \mathcal{H}(\mathbf{x}^*(t), \mathbf{u}(t), \boldsymbol{\lambda}^*(t)) \tag{D-4}$$

for all time $t \in [t_0, t_f]$ and for all permissible control actions $\mathbf{u} \in \mathcal{U}$. For a minimum-time optimal control problem this equation can be solved to find the optimal control $\mathbf{u}^*$ as a function of the optimal costate vector $\boldsymbol{\lambda}^*$.

The PMP describes the optimal trajectory of the state $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$ as

$$\dot{\mathbf{x}}^*(t) = +\left(\frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}\right)_* \tag{D-5}$$

$$\dot{\boldsymbol{\lambda}}^*(t) = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)_* \tag{D-6}$$

in which the optimal control $\mathbf{u}^*$ inside the optimal Hamiltonian $\mathcal{H}^*$ can be replaced by the derived function of the costate vector $\boldsymbol{\lambda}^*$.

The strength of this new form lies in the fact that the only unknown variables in this system are the initial values of the state vector $\mathbf{x}$ and costate vector $\boldsymbol{\lambda}$.

Since for a minimum-time problem the initial and desired final states are known, the optimal control problem can be solved by finding the initial costates that yield an optimal state trajectory from the initial state to the desired final state. Thus the optimal control problem has been transformed into a two-point boundary value problem.

# References


[1] K. Sims, "Evolving 3D Morphology and Behavior by Competition," *Artificial Life*, vol. 1, pp. 353–372, Jan. 1994.

[2] W. L. Hallam, J. Lund, and H.H., "A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pp. 384–389, 1996.

[3] M. Sato and K. Ishii, "Simultaneous Optimization of Robot Structure and Control System Using Evolutionary Algorithm," *Journal of Bionic Engineering*, vol. 7, pp. S185–S190, Sept. 2010.

[4] M. D. Bugajska and A. C. Schultz, "Autonomous, Co-Evolution of Form and Function in the Design of Project, Agents: Micro Air Vehicle," in *Workshop on Evolution of Sensors GECCO 2000, IEEE*, pp. 240—-244, 2000.

[5] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms.," *Nature*, vol. 406, pp. 974–8, Aug. 2000.

[6] B. von Haller, A. J. Ijspeert, and D. Floreano, "Co-evolution of structures and controllers for Neubot underwater modular robots," in *Proceedings of the VIIIth European Conference on Artificial Life {ECAL}*, Lecture Notes in Artificial Intelligence, pp. 189–199, Springer Verlag, 2005.

[7] G. S. Hornby, H. Lipson, and J. B. Pollack, "Generative representations for the automated design of modular physical robots," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 4, pp. 703–719, 2003.

[8] C. Leger, *Darwin2K: an evolutionary approach to automated design for robotics*. Kluwer Academic Publishers, 2000.

[9] C. Darwin, *The origin of species by means of natural selection*. New York Hurst, 1872.

[10] M. Mitchell and C. E. Taylor, "Evolutionary Computation: An Overview," *Annual Review of Ecology and Systematics*, vol. 30, pp. 593–616, 1999.

[11] A. Marczyk, "Genetic Algorithms and Evolutionary Computation." http://www.talkorigins.org/faqs/genalg/genalg.html, Apr. 2004.

[12] S. J. Pen, "Literature Survey: Using Evolutionary Algorithms in the development of Resonating Robot Arms," tech. rep., Delft University of Technology, Delft, 2011.

[13] T. Bäck and M. Emmerich, "Evolution Strategies for Optimisation in Engineering Applications," 2002.

[14] J. Andersson, "A Survey of Multiobjective Optimization in Engineering Design," 2000.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[16] A. Jafari, M. Safavi, and A. Fadaei, "A Genetic Algorithm to Optimum Dynamic Performance of Industrial Robots in the Conceptual Design Phase," in *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, pp. 1129–1135, 2007.

[17] P. Bentley and S. Kumar, "Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem," in *Genetic and Evolutionary Computation Conference*, pp. 35–43, 1999.

[18] R. Hinterding, "Representation, mutation and crossover issues in evolutionary computation," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2, pp. 916 –923 vol.2, 2000.

[19] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer, corr. 2nd ed., 2007.

[20] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, pp. 345–370, Apr. 2009.

[21] K. Kawai, A. Ishiguro, and P. Eggenberger, "Incremental evolution of neurocontrollers with a diffusion-reaction mechanism of neuromodulators," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, pp. 2384 –2391 vol.4, 2001.

[22] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated Neural Evolution through Cooperatively Coevolved Synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.

[23] A. Ostermeier, A. Gawelczyk, and N. Hansen, "Step-size adaptation based on non-local use of selection information," in *Parallel Problem Solving from Nature - PPSN III*, pp. 189—-198, Springer, 1994.

[24] D. Whitley, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 116—-121, Morgan Kaufmann, 1989.

[25] L. Cardamone and D. Loiacono, "Applying cooperative coevolution to compete in the 2009 torcs endurance world championship," *Computation (CEC), 2010*, pp. 1–8, 2010.

[26] M. C. Plooij and M. Wisse, "Using a Resonant Mechanism to Reduce Energy Consumption in Robotic Arms," tech. rep., TUDelft, Delft, 2011.

[27] N. Chivarov, N. Shivarov, and P. Kopacek, "Educational Robot - Robco SCARA," *serviceroboticsgroup.com*, 2010.

[28] R. van der Linde and A. Schwab, "Lecture notes multibody b, version 3 February 2011," 2011.

[29] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8 (3), pp. 338–353, 1965.

[30] E. Mamdani, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, 1975.

[31] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. Volume No., pp. 116–132, 1985.

[32] D. Pelusi, "Optimization of a Fuzzy Logic Controller Using Genetic Algorithms," *2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, pp. 143–146, Aug. 2011.

[33] D. Srinivasan, "Hybrid fuzzy logic-genetic algorithm technique for automated detection of traffic incidents on freeways," *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, pp. 352–357, 2001.

[34] N. Baine, "A simple multi-chromosome genetic algorithm optimization of a Proportional-plus-Derivative Fuzzy Logic Controller," *NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 1–5, May 2008.

[35] S.-j. Huang and J.-s. Lee, "A stable self-organizing fuzzy controller for robotic motion control," *IEEE Transactions on Industrial Electronics*, vol. 47, pp. 421–428, Apr. 2000.

[36] J. Velasco and L. Magdalena, "Genetic algorithms in fuzzy control systems," *Genetic Algorithms in Engineering and Computer Science*, vol. 28040, pp. 141–165, 1995.

[37] C.-h. Chou, "Genetic algorithm-based optimal fuzzy controller design in the linguistic space," *IEEE Transactions on Fuzzy Systems*, vol. 14, pp. 372–385, June 2006.

[38] Q. Liu, Y. Yu, Q. Xia, and L. Su, "A new fuzzy method for the motion control of underactuated robots based on genetic algorithm," *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pp. 999–1003, June 2008.

[39] P. Shill, K. Pal, and M. Amin, "Genetic algorithm based fully automated and adaptive fuzzy logic controller," *Fuzzy Systems (FUZZ),*, pp. 3–9, 2011.

[40] S. A. H. Sheroz Khan , Salami Femi Abdulazeez , Lawal Wahab Adetunji , AHM Zahirul Alam , Momoh Jimoh E . Salami, A. H. A. Isl, and M. Rafiqul, "Design and Implementation of an Optimal Fuzzy Logic Controller Using Genetic Algorithm," *Journal of Computer Science*, vol. 4, no. 10, pp. 799–806, 2008.

[41] J. Zhang, Z. Zhao, Y. Sun, and W. Han, "Optimal design of fuzzy logic controller for multi-body model of semi-active suspension based on genetic algorithm," in *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, pp. 1478–1483, IEEE, 2010.

[42] C. Jie, "The Application of Float-Encoding Genetic Algorithm in Optimizing Parameters of Fuzzy Controller," *2010 International Conference on Digital Manufacturing & Automation*, pp. 621–624, Dec. 2010.

[43] Y. Petrenko and S. Alavi, "Fuzzy logic and genetic algorithm technique for non-liner system of overhead crane," in *Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), 2010 IEEE Region 8 International Conference on*, pp. 848–851, IEEE, 2010.

[44] M. Dubey, "Design of Genetic Algorithm Based Fuzzy Logic Power System Stabilizers in Multimachine Power System," *2008 Joint International Conference on Power System Technology and IEEE Power India Conference*, pp. 1–6, Oct. 2008.

[45] D. Devaraj and B. Selvabala, "Real-coded genetic algorithm and fuzzy logic approach for real-time tuning of proportional-integral-derivative controller in automatic voltage regulator system," *IET Generation, Transmission & Distribution*, vol. 3, no. 7, p. 641, 2009.

[46] M. Uddin, M. Abido, and M. Rahman, "Real-Time Performance Evaluation of a Genetic-Algorithm-Based Fuzzy Logic Controller for IPM Motor Drives," *IEEE Transactions on Industry Applications*, vol. 41, pp. 246–252, Jan. 2005.

[47] F. Leung, H. Lam, S. Ling, and P. Tam, "Optimal and Stable Fuzzy Controllers for Nonlinear Systems Based on an Improved Genetic Algorithm," *IEEE Transactions on Industrial Electronics*, vol. 51, pp. 172–182, Feb. 2004.

[48] J. Velagic, "Design of fuzzy logic based mobile robot position controller using genetic algorithm," *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, pp. 1–6, 2007.

[49] F. Naderi, a. a. Gharaveisi, and M. Rashidinejad, "Optimal Design of Type_1 TSK Fuzzy Controller Using GRLA for AVR System," *2007 Large Engineering Systems Conference on Power Engineering*, pp. 106–111, Oct. 2007.

[50] A. A. Eftekharian and H. Sayyaadi, "Design of Mixed Fuzzy-GA Controller For SCARA Type Robot," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2173–2178, 2006.

[51] Y. Jiang, J. Cao, and G. Yang, "Learning Technique for TSK Fuzzy Model Based on Cooperative Coevolution," in *Machine Learning and Cybernetics, 2006 International Conference on*, no. August, pp. 1924–1929, IEEE, 2006.

[52] R. Findeisen, "An introduction to nonlinear model predictive control," *Benelux Meeting on Systems and Control*, 2002.

[53] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[54] J. L. Castro, "Fuzzy logic controllers are universal approximators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, pp. 629–635, Apr. 1995.

[55] T. Poggio and F. Girosi, "Networks for approximation and learning," in *IEEE 78*, pp. 1479–1481, 1990.

[56] B. Liu and J. Si, "The best approximation to C2 function and its error bounds using regular-center Gaussian networks," in *IEEE Trans. Neural Networks 5*, pp. 847–848, 1994.

[57] D. S. Naidu, *Optimal Control Systems*. New York: CRC Press, 2003.

[58] G. Beligiannis and G. Tsirogiannis, "Restartings: A technique to improve classic genetic algorithms' performance," *World Academy of Science,*, vol. 018, no. Epeaek Ii, pp. 144–147, 2005.

# Index