AUTOMATICALLY ADDRESSING UNCERTAINTY IN AUTONOMOUS ROBOTS WITH COMPUTATIONAL EVOLUTION

By

Anthony Joseph Clark

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor Of Philosophy

2016

ABSTRACT

AUTOMATICALLY ADDRESSING UNCERTAINTY IN AUTONOMOUS ROBOTS WITH COMPUTATIONAL EVOLUTION

By

Anthony Joseph Clark

Autonomous robotic systems are becoming prevalent in our daily lives. Many robots are still restricted to manufacturing settings where precision and repetition are paramount. However, autonomous devices are increasingly being designed for applications such as search and rescue, remote sensing, and tasks considered too dangerous for people. In these cases, it is crucial to continue operation even when some unforeseen adversity decreases performance levels—a robot with diminished performance is still successful if it is able to deal with uncertainty, which includes any unexpected change due to unmodeled dynamics, changing control strategies, or changes in functionality resulting from damage or aging.

The research presented in this dissertation seeks to improve such autonomous systems through three evolution-based techniques. First, robots are optimized offline so that they best exploit available material characteristics, for instance flexible materials, with respect to multiple objectives (e.g., speed and efficiency). Second, adaptive controllers are evolved, which enable robots to better respond to unforeseen changes to themselves and their environments. Finally, adaptation limits are discovered using a proposed mode discovery algorithm. Once the boundaries of adaptation are known, self-modeling is applied online to determine the current operating mode and select/generate an appropriate controller.

These three techniques work together to create a holistic method, which will enable autonomous robotic systems to automatically handle uncertainty. The proposed methods are evaluated using robotic fish as a test platform. Such systems can benefit in multiple ways from the integration of flexible materials. Moreover, robotic fish operate in complex, nonlinear environments, enabling thorough testing of the proposed methods.

Copyright by
ANTHONY JOSEPH CLARK
2016

This dissertation is dedicated to Megan, Trapp, and Beckett.

ACKNOWLEDGEMENTS

Many thanks to my doctoral advisor, Dr. Philip McKinley, as well as the members of my committee: Drs. Erik Goodman, Charles Ofria, Bill Punch, and Xiaobo Tan. I would also like to acknowledge the support and feedback provided by my coworkers and members of the BEACON Center at Michigan State University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Increasingly, robotic devices are being deployed in uncontrolled, unstructured environments. Unlike industrial manufacturing settings, where robots often perform a single repetitive task, autonomous robots must operate in remote, hazardous, and highly dynamic environments in which they are expected to perform multiple complex tasks. Examples include search and rescue, remote sensing, package delivery, and assisting humans in dangerous occupations such as mining and firefighting. With this increase in required autonomy and complexity comes an increase in design difficulty. Accordingly, in the past decade the engineering community has seen many advances in cyber-physical technologies, including MEMS-based physical sensing (accelerometers, gyroscopes, etc.), rechargeable batteries (e.g., lithium polymer), and increased computing power of embedded systems. All of these technologies have become readily available as commodity hardware for researchers, hobbyists, and students.

Despite these advances, however, most autonomous systems still lack the abilities exhibited by even simple biological organisms. First, the materials, sensors, and actuators comprising robotic systems are not as effective as biological tissue, and second, the control systems governing robotic systems do not yet exhibit the adaptability and robustness of biological systems. In this dissertation, we develop and apply evolutionary computation (EC) methods to improve such autonomous robotic systems. EC codifies the basic princi-

ples of genetic evolution in computer software [35]. The open-ended nature of evolutionary algorithms has been shown to discover novel, often counter-intuitive, solutions to complex engineering problems [19], sometimes surpassing human designers [1].

One approach to mimicking the fluid nature of natural organisms in robotic systems is to integrate soft, flexible materials into robots [126, 15]. In this case, flexible components are intended to partially compensate for the relatively primitive actuation capabilities of robots compared to biological organisms. An example is the "whipping" action exhibited by the flexible fin of a robotic fish [26], which can increase thrust compared to that generated by a rigid fin. However, the integration of flexible materials comes at the cost of increased design complexity, specifically in discovering how to best exploit their properties. For example, how should the flexible fin be actuated/controlled to take advantage of its spring-like nature? Evolutionary methods have been shown to work well with such problems, and produce a favorable matching between morphology (i.e., the physical components of a robot) and control [93].

A majority of evolution-based robotics studies, referred to as evolutionary robotics (ER), focus on a single objective; for example, the objective is often to *move as fast as possible*. In practice however, autonomous robots often have limited energy capacity. This means that if the most effective way to move fast consumes a lot of power the robot will soon be unable to move at all. For this reason, it can be beneficial to consider a second objective such as power efficiency. A robot can then be optimized with a multiobjective approach in order to, for example, both move fast and be efficient. Multiobjective techniques are particularly helpful when two or more of the objectives are in competition, meaning that to improve the first objective, the second objective must be performed at a deteriorated level. Instead of a single optimal solution, evolutionary multiobjective algorithms (EMOs) find a set of Pareto-optimal solutions where each individual can be considered optimal depending on how much weight is given to each objective [37]. Moreover, the information attained from an EMO can be utilized both offline during design and on-board during runtime. This enables

a system to adjust the relative importance of competing objectives as its internal state and its environment evolve through time.

Although evolutionary methods often find effective, if not optimal, designs for robotic systems, the systems themselves can change over time. For example, motor performance may degrade or the characteristics of flexible materials may change as a component ages. Thus, it is unlikely that any static controller will remain effective for the life of an autonomous system. The field of control theory has produced a set of techniques specifically aimed at alleviating this issue of changing dynamics. *Adaptive control* methods combine parameter estimation with control law in order to control systems with dynamics that are either unknown or changing unpredictably [63]. Traditional adaptive control relies on sufficiently accurate models of the target system (and modeling the desired response for the system), allowing for variations in the underlying dynamics through a set of adaptive laws. Over the past sixty years such adaptive control methods have proven to be effective, particularly in the domains of aircraft control, power systems, and process control [63, 96]. More recently, however, researchers have started to develop and refine *model-free* and *data-driven* techniques, which do not rely on the development of a specific dynamic model, but rather only on input-output data [59]. These techniques still require significant domain knowledge and design expertise. In this dissertation, however, we use evolutionary methods to develop adaptive controllers and enhance their adaptability.

Adaptive controllers enable cyber-physical systems, such as autonomous robots, to manage uncertain conditions during execution. However, there is a limit to the range of conditions that can be handled by a given controller. When this limit is exceeded, a controller might fail to respond as expected, not only rendering it ineffective but possibly putting the entire system at risk. For this reason, it is important to discover the boundaries of a given adaptive controller. Collectively, these boundaries define an *execution mode* for that controller. Explicit specification of mode boundaries facilitates the development of decision logic that determines, based on system state and sensed conditions, when to switch to a different execu-

tion mode (and typically a different controller), such as one for providing fail-safe operation. Once mode boundaries have been determined, however, it is still an open question as to how a robotic system should determine its current mode. How does the system identify compensatory behaviors? One approach to handling such a case is for the system to develop a new representation of its principle dynamics. Such techniques are referred to as *model-learning* or *self-modeling* methods. These techniques are intended to enable autonomous robots to automatically generate models based solely on information extracted from the accessible data streams (i.e., data from sensors) [97]. Model-learning techniques commonly use standard regression techniques including least-squares [79]. However, Bongard et al. have also shown that an on-board evolutionary algorithm is capable of discovering a self-model through a series of evolved actions and measured responses [12]. In general, the idea is to first learn the behavior of the system from observations, and then determine how to manipulate the system (via the newly generated self-model) to achieve a desired response [97].

1.1 Problem Description.

This dissertation addresses three issues of increasing complexity. (1) *How can the body-plan and control pattern of robotic systems be simultaneously optimized while taking into account flexible components?* It is advantageous to use an optimization process that automatically matches morphology with control patterns. (2) *How can robotic systems be optimized with respect to multiple, competing objectives?* Most complex systems must necessarily balance the importance of multiple objectives (e.g., speed and energy usage). (3) *How can autonomous robotic systems adapt to modeling inaccuracies and to subtle variations in physical dynamics, and then use knowledge of adaptive boundaries to switch to (or generate new) behaviors via a self-modeling process?* Any offline (simulation-based) optimization will infuse error due to modeling inaccuracies, and furthermore, degradation of materials and motors can cause conventional (nonadaptive) controllers to detune. Beyond adapting to

subtle changes, an autonomous system may encounter more catastrophic changes, such as physical damage and motor failure, that require more drastic response. Such measures, while perhaps not restoring complete functionality, should at least enable the robot to return to a repair station.

1.2 Thesis Statement.

Employing evolutionary optimization methods can enable autonomous robotic systems to leverage the benefits of flexible components with respect to control patterns and multiple objectives, automatically adapt to subtle changes in operating dynamics, and overcome more significant changes that might otherwise cause a complete failure.

The methods developed in this research are applied to robotic fish. Robotic fish benefit from the use of flexible components, and they operate in a highly nonlinear environment that is well suited for testing adaptive control. Furthermore, self-modeling and self-healing techniques can be tested by changing the caudal fin actuator (i.e., the tail fin) or structure of the caudal fin itself.

1.3 Research Contributions.

The broad contribution of our research is a holistic method for developing autonomous systems capable of mitigating the uncertainty of an unstructured environment. More specifically, this research has produced results regarding the design of cyber-physical systems, including the co-development of robust embedded computing systems and robot morphologies with flexible components. This work can be broken into three parts. First, we leveraged dynamic models of flexible components such that material properties were optimized in simulation. The goals were to increase the maneuverability, speed and agility, of autonomous robots while considering both morphology and control patterns. Second, we explored optimization processes that balance trade-offs among the competing objectives of performance

and efficiency. Finally, we enhanced control systems capable of adapting to changes in environmental conditions and to the electrical and mechanical components; an example of the latter is degrading performance of flexible components. During this process, we also developed methods for determining mode boundaries of adaptive behaviors, real-time failure diagnosis, and switching to compensatory behaviors.

1.4 Document Layout.

The remaining chapters are organized as follows: In Chapter 2 we cover the relevant background and related materials. Chapter 3 describes our investigations in which morphology and control patterns are simultaneously evolved. Chapters 4 and 5 discuss our research involving multiobjective evolution. Likewise, in Chapters 6 and 7 we describe studies involving adaptive control, mode discovery, and self-modeling. Finally, Chapter 8 contains conclusions for this document.

Chapter 2

Background

Research into autonomous robotic systems requires modeling, simulation, and design of both control and morphology. In addition to these topics, our research includes integrating flexible materials and the use of evolutionary optimization. Moreover, to produce systems capable of handling adverse conditions we make use of adaptive control (including the discovery of boundaries of adaptation), self-modeling, and self-healing techniques. The key theories and methods related to the aforementioned topics are introduced in this chapter.

2.1 Flexible Materials in Robots

Although many recent robot designs are said to be *bio-inspired* or *biomimetic*, many of these electro-mechanical systems fail to reach levels of performance, adaptability, and robustness comparable to those of their natural counterparts. One reason is that most robotic systems comprise only hard, rigid materials, while natural organisms include soft tissues that enhance performance and "soften" interaction with the environment. Integrating flexible components into robot morphologies is one method for bridging this performance gap between artificial and natural creatures.

With the proliferation of 3D printers, the design, rapid fabrication, and testing of flexible components has become much easier. For example, Richter et al. have fabricated wings of a

robotic dragonfly composed of both flexible and rigid components [108], and in our work we have shown that, when compared to a rigid fin, a passively flexible caudal fin can improve thrust of a robotic fish [26]. Both of these studies rely on 3D printers for construction of robotic devices. Aside from autonomous robots, flexible and compliant materials can benefit application domains such as the design of prostheses, safety in robot-human interaction, and automated manufacturing [101, 15].

Despite the many advantages of incorporating flexible components into robots, they do come at the cost of increased design complexity and increased design effort for the control schemes governing their motion. This is particularly important if the controller is required to handle changes in flexibility over time. For example, the characteristics of a flexible fin may change depending on water temperature and the effects of material aging. In such cases, it might be appropriate to use an automated design process capable of *testing* many different parameter combinations. However, designing a robotic system in such a way can lead to a very large search space—possibly too many parameter combinations to perform a brute-force, exhaustive search. Likewise, with complex parameter interactions a simple gradient-based search algorithm is likely to find only local optima. In the research presented in this dissertation, we apply evolutionary search techniques, which are known for their ability to widely search the space of possible answers as well as find *novel* solutions beyond the preconceptions of human designers [57, 50, 36].

2.2 Evolutionary Computation

Evolutionary computation (EC) methods have proven to be effective for a variety of problems [77]. The most well-known variant is the genetic algorithm (GA) [57], a stochastic search technique in which a population evolves toward a solution to an optimization problem. The genome of each individual in the population is an encoding of a candidate solution. In optimizing a robot, for example, the genome might include values for the robot's controller

as well as characteristics of the robot's morphology (e.g., oscillating frequency of the motor and the dimensions of the fin). An initial population of individuals is either generated randomly or seeded around a point of interest, such as a solution to the problem that is known to be effective. Genome performance is quantified via a fitness function, and evolutionary operators such as crossover and mutation are applied to high-performing genomes, producing a new generation of possible solutions. This procedure is repeated until a maximum number of generations is reached or until an optimal (or adequate) solution has been found. Genetic programming (GP) [50, 76] is a related method where the individuals are actual computer programs. Both GAs and GPs have been shown to be effective in a wide variety of science and engineering domains, rivaling and even surpassing human designers [1], including the design of an antenna for a NASA satellite [84] and to automatically find and repair software bugs [48]. Another form of EC is neuroevolution [131], a machine learning method in which evolutionary algorithms (EAs) are used to evolve artificial neural networks (ANNs). Neuroevolution has been applied to the design of controllers for finless rockets [51] and mobile robots [17, 45].

Evolutionary robotics (ER) [17, 114, 98, 81, 45, 13], is a subfield of EC concerned with the design of robotic systems. One of the primary benefits of ER methods is that they can concurrently optimize both control and morphology of the robot, which typically produces better coupling (or synergistic relationship) between behavior and physical form [21, 90, 58, 127, 103], and hence better performance. Evolved controllers usually take the form of an array of control parameters, central pattern generators [107, 9], or genetic programs [77, 67]. ANNs in particular have been extensively studied by the ER community [114, 131, 51, 29], primarily due to their biological nature and their ability to model any nonlinear mapping from input to output. NEAT [119] and HyperNEAT [118], developed by Stanely et al., are two of the most common algorithms for evolving ANNs. In NEAT, ANNs are gradually "complexified" by starting with a simple network and randomly adding new neurons and

connections between neurons. In contrast, HyperNEAT ANNs are generally fixed in topology, but the values for connections among neurons is based on the *physical* layout of the network.

Recently, ER has been applied to problems in *soft robotics* [56]. Similar to the robotic fish developed for this study, soft robots comprise (or contain) malleable, flexible components, which are meant to improve performance and/or safety [104, 86, 70, 82, 109]. For example, Cheney et al. [19] evolved locomotion for voxel-based simulated robots, where each robot is made up of a 3D-grid of cubic voxels and each voxel can be evolved with different material properties. That study demonstrated that evolving soft robots with a generative encoding, based on principles from developmental biology, dramatically improves locomotion when compared to direct encodings. The research discussed in this dissertation leverages the benefits of *both* ER and traditional control paradigms. Specifically, in Chapter 3 we investigate how EC can be used to discover optimal control patterns for the flexible caudal fins of a robotic fish, and in Chapter 6 we enhance the adaptability of an adaptive controller.

2.2.1 Multiobjective Optimization.

In many ER studies, algorithm variants are demonstrated on relatively simple tasks. For example, a robot may be optimized to simply move as fast or as far as possible [26]. However, in a real-world application the task is likely to be more complex, such as way-point following or station holding. Furthermore, multiple competing objectives might need to be simultaneously optimized. One such example is an autonomous system that needs to be both effective at locomotion and able to carry out tasks with limited power (i.e., it needs to be energy efficient).

In such cases, using a evolutionary multiobjective algorithm (EMO) can be advantageous. EMO algorithms operate using similar principles as their single objective counterparts. However, unlike traditional approaches to managing multiple objectives in which fitness values are a weighted sum of different goals (a technique sometimes referred to as scalarization) [39], EMO fitness functions return a sequence of values, where each value rep-

resents fitness with respect to a different objective. In addition, instead of locating a single optimal set of parameter values, EMOs converge to a *set* of Pareto-optimal solutions. Individuals belonging to a Pareto front are said to be *nondominated*; that is, each of the solutions is optimal with respect to some combination of the objectives. The most common EMO algorithms, such as NSGA-II/III [37, 112] and SPEA2 [140], use an elitism approach for driving solutions toward the optimal Pareto front, and a niching or crowding mechanism to ensure that the entire set of Pareto-optimal solutions can be found. Indicator-based EMOs [139, 66] also produce a set of solutions, but do so by maximizing an indicator variable, which acts as a single objective for evolving the entire population. One of the most popular indicator variables is *hypervolume* [8]. By maximizing the hypervolume of a population, the algorithm effectively drives search toward the user-defined goals while maintaining a distance between evolved individuals in the solution space. The advantages of EMO algorithms, when compared to single objective EAs and parameter sweeps, include: (1) locating a Pareto front with fewer evaluations, (2) automatically handling of constraints, (3) not needing to specify the relative importance among multiple objectives, and (4) automatically sorting solutions according to feasibility and domination. In Chapters 4 and 5 we demonstrate that EMOs are effective in optimizing flexile robotic fish caudal fins for both thrust (speed) and efficiency (mechanical and electrical).

2.2.2 Modeling.

To apply EC algorithms to robot design, solutions are typically evolved entirely in simulation. The primary advantage is time savings—simulations can easily be executed in parallel and much faster than physical experiments. Many ER simulations are conducted with a rigid-body dynamics engine. For example, the Open Dynamics Engine (ODE) [116] is widely used by the ER community. ODE, which was developed by the computer gaming community, has the advantage of being efficient, albeit at the cost of accuracy. Other physics engines, such as the Bullet Physics Library [32], Chrono::Engine [22], and Newton Game

Dynamics [65], are also used by the ER community, depending on specific features of the target system or environment under study. For example, Bullet Physics includes support for simulating *soft bodies*, but at this time uses a less stable algorithm for handling the constraints between rigid bodies, which can be important for articulated robotic components.

Use of such physics engines is referred to as physical simulation. In physical simulation, a simulated world is constructed from basic primitives (ellipsoids, cuboids, cylinders, etc.) and these components interact with one another through collisions and applied forces/torques. An alternative to physical simulation is called numerical (or mathematical) simulation, where all computations are based on mathematical formula (including differential equations). Numerical simulation environments (e.g., Matlab and Simulink [115]) are often more accurate than physical simulations, however, they have increased computational costs, are more difficult to parallelize, and are limited in terms of making the environment interactive and dynamic. For example, simulating collisions and adding new obstacles mid-simulation is straightforward in a physical simulation, but is often infeasible for numerical simulation.

Although all of the above simulation tools are helpful to ER research, none of them includes sufficient support for flexible materials or hydrodynamics. When such features are required, dynamic models must be developed. For example, Gomez et al. [51] relied on an accurate aerodynamics engine to optimize a guidance system for finless rockets in simulation. As described in Chapter 3, we incorporated a hydrodynamics model, as well as a model for flexible caudal fins, into a physical simulation in order to optimize the propulsion of a robotic fish [26]. An important feature of such models is that they include only critical aspects of the underlying dynamics. Simplifying assumptions make the models more efficient to develop and execute. The latter is particularly important in evolutionary algorithms, where the cost of fitness evaluation often dominates execution time. However, these assumptions, also give rise to discrepancies between behaviors seen in simulation and in reality, the so-called *reality gap*.

2.2.3 Reality gap.

Despite numerous impressive results in the field of ER, the general applicability of EC to robotics is still limited, as most ER techniques are developed and demonstrated in simulation. The reality gap often leads to difficulties when trying to realize an evolved solution in a physical system. Over the past two decades, several methods have been developed aimed at circumventing the reality gap. The most obvious approach is to perform evolution on the devices themselves. This is referred to as on-board (or evolving physical systems) evolution and has proven to be quite effective [46, 16]. However, on-board evolution is both time consuming and potentially hazardous to the physical device. For instance, randomly evolved solutions may put too much stress on a robot's motors, or cause it to move into a dangerous location (e.g., drive off of a ledge). An improvement is to begin evolution in simulation and then finish on the device [106]. A similar approach is to incorporate physical trials, at set intervals, into the evolutionary algorithm [75]. These methods assume that the reality gap is small enough such that simulation-evolved individuals can *cross* the gap; that is, the performance in simulation is *close enough* to that in reality. Additionally, it is generally infeasible to evolve morphologies using an on-board process.

Since it is desirable to do as much work in simulation as possible, some methods attempt to account for the reality gap through additional modeling or control complexity. For example, Miglino et al. [91] modeled simulation inaccuracies as environmental noise; doing so places an evolutionary pressure on evolved robots to handle certain levels of error between simulation and reality. However, an open question is how much noise should be modeled. Another approach, proposed by Floreano et al. [45], is to evolve flexible controllers capable of adapting online. These controllers use "plastic" artificial neural networks [47, 94], meaning that some neurons (and some connections) are allowed to change their behavior according to an evolved, static set of rules. However, such networks still have trouble when facing large disparities between simulation and reality.

Other methods for crossing the reality gap use physical trials as a way to improve the simulation itself [74, 134]. Bongard et al. [14] developed the estimation-exploration algorithm (EEA) to iterate between evolving a neural controller and improving the simulation. In the exploration phase of EEA, a new *test* is found that best exposes a disparity between reality and the current model. While in the estimation phase, the chosen test is used to improve a population of models so that they better match reality. A benefit of this approach is that it does not require any morphological model to be specified *a priori*, (although such a model does improve performance). The main drawback of EEA is that it assumes the simulation can be parameterized and subsequently improved, which may become too difficult as robot and environmental complexity are increased.

These studies demonstrate the importance of the reality gap to ER. Regardless of how accurate the simulation, there will likely exist disparities between simulation and reality. The most effective ER approaches include a mixture of both simulated and physical evaluations. In this dissertation, we rely on controller-based techniques to cross the reality gap. Specifically, one benefit of using the adaptive controllers described in the next section is that they treat any unmodeled (or poorly modeled) dynamics as something to which they can automatically adapt.

2.3 Adaptive Control

Different from the techniques addressed in the previous section, our research relies in part on adaptive control to aid in crossing the reality gap. Adaptive control techniques have been studied since the early 1950s, starting with the design of autopilots for high-performance aircraft [63]. These methods combine parameter estimation with a control law in order to control target systems with dynamics that are either unknown or changing unpredictably. For the purpose of this document, control techniques can be divided into two categories: *model-based* and *model-free* (also referred to as data-driven). As their names suggest, model-

based methods rely on a dynamic model developed with first principles, whereas model-free control techniques do not necessitate any modeling.

Figure 2.1 illustrates the difference between a conventional feedback controller and an adaptive controller. Both use the error e between the reference signal and measured output in order to produce an input u for the target system (or *plant*). While a non-adaptive controller enables the system to respond to dynamic conditions by adjusting the value of u, the manner in which it does so is fixed (specifically, once tuned for a particular system and expected set of conditions, the parameters to the equations defining u are constants). An example is the widely used PID controller, where u is the weighted sum of terms involving e (**P**roportional term), its integral (**I**ntegral term), and its derivative (**D**erivative term).

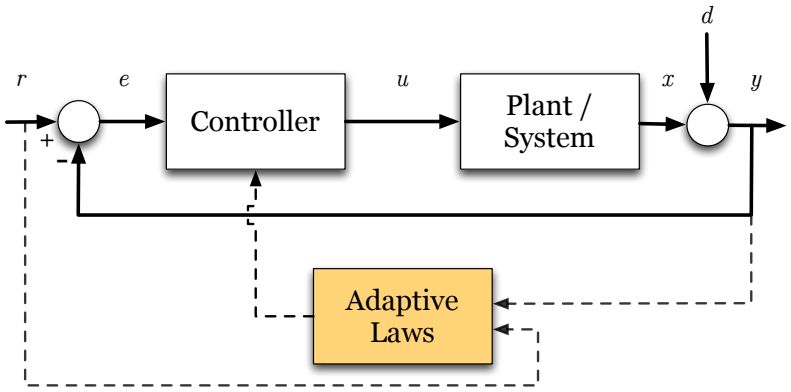


Figure 2.1: A block diagram for a feedback control system. The shaded block shows the additional components necessary for adaptive control. The signals r, x, y, and e are the reference signal (desired output), the current output of the system, the measured output of the system (including any measurement error due to d), and the error between the input and the measured output, respectively, while u is the control signal. All of these signals vary as a function of time.

The main shortcomings of PID control are that it requires parameter tuning and typically cannot respond to changes in the controlled system. In contrast, an adaptive controller automatically adjusts control parameters online according to adaptive laws (shown by the shaded box in the figure). Effectively, an adaptive controller can adjust how it responds to a particular set of inputs based on the situations that it has already encountered.

2.3.1 Model-based Adaptive Control.

Two of the most well-studied adaptive control techniques are *model-reference adaptive control* (MRAC) and *adaptive pole-placement control* (APPC). A diagram of MRAC control is shown in Figure 2.2. For both MRAC and APPC, the desired behavior is based on a hand-designed reference model (outlined by a red, dashed box in the figure), and a controller is adapted online such that the target system behaves similarly to the reference model. Specifically, error between the model and system outputs is used by the adaptive laws to drive the system toward the behavior of the model. Adaptive laws are differential equations that include error as a term; these equations are typically derived using Lyapunov stability criteria such that they guarantee convergence within some predetermined amount of time [63].

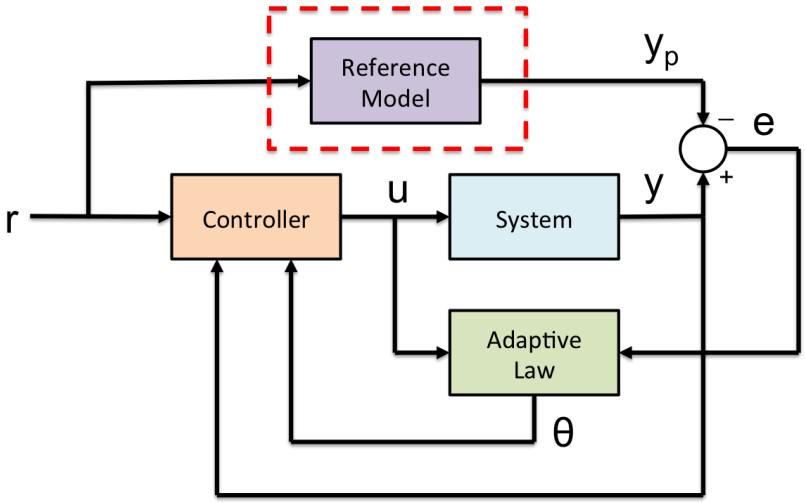


Figure 2.2: A graphical representation of the MRAC method. A reference input r is fed to both the reference model and the adaptive controller. The controller produces an output signal u and is adapted using Θ, which is based on the error e between the reference model and the controlled system outputs, y_p and y respectively.

MRAC approaches have a restriction that the plant must be minimum-phase (i.e., all zeros have to be stable). Practically, this restriction means that MRAC cannot be applied in many realistic settings, including systems with time delays (when a system is slow to respond to varying input commands). APPC relaxes the assumption, and does not require unstable zero-pole cancellation during controller design. Accordingly, MRAC can be considered a

special case of APPC. At a minimum, however, both of these techniques require a designer to specify the poles of a reference model. Although there are many techniques and tools that aid in pole placement, much of the process is still dependent upon the expertise of a designer.

Another approach to adaptive control, unrelated to APPC or MRAC, is neuroadaptive control [63]. This set of techniques is particularly useful for nonlinear systems, especially when any nonlinearities are difficult to identify. In neuroadaptive control, nonlinearities in the system are approximated with ANNs whose weights are adaptive. An overview of these techniques can be found in [63, 64].

Although the above control methods have been extensively studied and applied, they all require the development of a suitable model of the desired behavior. Reliance on a specific reference model makes it difficult to accommodate unexpected conditions that induce a change to the underlying dynamics. Additionally, the high order associated with many physical systems, such as robots with flexible components, can render the design and implementation of a model-based controller intractable. An alternative to model-based control is to use more general purpose *model-free* control techniques.

2.3.2 Model-free Adaptive Control.

A model-free adaptive controller is meant to be as general-purpose as PID while also being capable of adapting to changes online. Model-free adaptive control (MFAC) techniques are intended for "gray box" situations where only partial and possibly inaccurate information about the system is known. As with traditional approaches to adaptive control [64], MFAC attempts to minimize the error between desired and actual outcomes. Hou and Huang first proposed the idea of MFAC based on the concept of a pseudo-partial-derivative and reliance on only system inputs and outputs [60]. Later, Aidong et al. [3] and Karoń [68] proposed modifications to MFAC that enhance the flexibility and responsiveness of the MFAC algorithm, respectively. The MFAC approach employed in this dissertation,

developed by Cheng [20], combines a traditional proportional controller with an adaptive ANN. The MFAC ANN, shown in Figure 2.3, computes its output in the same manner as a conventional feed-forward neural network, and operates similarly to a standard feedback controller (see Figure 2.1) where the goal is to minimize the input error signal. Unlike a nonadaptive controller, however, the values of links connecting nodes are adjusted during operation according to a set of adaptation laws, which are discussed in Chapter 6.

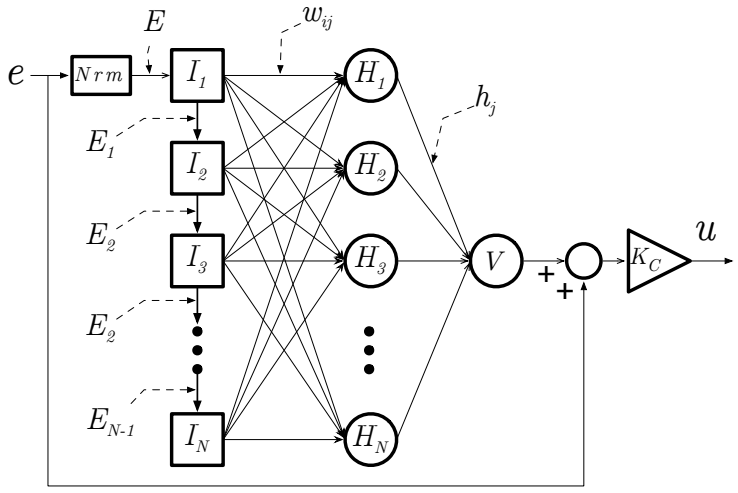


Figure 2.3: A graphical representation of the MFAC ANN. A continuous time error signal e is first normalized (via the Nrm block) and then propagated through the input neurons I_i. The ANN is then activated as a feed-forward network to produce an output V. The final controller output u is an amplified summation of V and e.

2.3.3 Evolving Adaptive Control.

The ER community has explored the evolution of adaptive control [47, 124], however, these approaches typically use ANNs with synaptic plasticity to handle online adaptation. An evolutionary pressure for adaptive behavior is incorporated by subjecting evolved individuals to adverse conditions during simulation. In contrast, traditional adaptive control techniques such as MRAC and APPC are generally not optimized in simulation and adapt according to designed adaptive laws. An exception is the work of Coelho et al. [30], which extends traditional adaptive control methods by incorporating an evolved neural compensator, which

acts as an adaptive gain module and improves MFAC's ability to handle nonlinear systems. In our research, described in Chapter 6, we further expand upon established model-free adaptive control methods by optimizing adaptive control parameters in simulation. Specifically, we subject adaptive controllers to a variety of different conditions during fitness evaluation in order to put an evolutionary pressure on adaptive controllers to handle a wide range of scenarios.

2.4 Mode Discovery and Self-modeling

Adaptive control methods are effective for handling the dynamic conditions that robotic systems experience throughout the lifetime of a physical device. This includes degradation of materials and changing environmental conditions such as temperature. However, once these conditions vary beyond a certain threshold (e.g., due to a damaged component) an adaptive controller will fail. For example, an adaptive controller can typically adapt to a servomotor acting differently under low-power conditions, but is not able to control a device with a seized servo. In such cases, more extreme measures must be taken to to prevent the robotic system from failing completely. This scenario is particularly difficult for any system that is not always under direct human supervision.

Thus, if the system is expected to handle more extreme changes, it is essential to explicitly define the boundaries of an adaptive controller. More specifically, it is important to know by how much can any given physical property change before a controller fails and the system needs to switch to a different mode of execution driven by a different controller. Explicit knowledge of such mode boundaries facilitates the development of higher-level decision logic, where each mode can have its own unique control strategy (e.g., adaptive controller). The automated discovery of execution modes reduces the reliance on a system designer to *manually* anticipate failure cases. Alternative behaviors and "recovery" modes can be pre-programmed (e.g., using evolutionary methods), which provides an efficient means of addressing problems

during operation. Traditional solutions to this problem include control theory techniques such as gain scheduling and model adaptive control with switching [49, 95, 111, 53, 55, 54] in which a set of controllers are designed to handle a finite set of possible conditions. For this dissertation, however, we rely on EC-based methods.

Even if the execution modes have been defined, however, the system still must be able to determine the current mode. Based purely on system state information and sensed conditions, how should a cyber-physical system decide which execution mode is correct (or best-fit)? The problem can also be stated as follows: *how can the system discover a new model for itself during runtime?* With a new model, the system can switch to the correct mode (and accompanying controller). One approach to achieving such runtime model discovery is *self-modeling*, where the system maintains an internal "mental image" of itself [12].

The search capability of computational evolution has been shown to be effective in discovering both models and corresponding behaviors for robots. Bongard and Lipson [14] introduced the Estimation-Exploration Algorithm (EEA), a general purpose algorithm for reverse engineering complex, nonlinear systems. The authors demonstrated the use of EEA to automatically diagnose failures in terrestrial robots (quadrupeds and hexapods) and generate compensatory behaviors without human intervention [83]. However, there are areas of potential improvement. First, the algorithm is computationally expensive to run on a robot, which may have limited resources. Second, even with an accurate self-model it is likely that the reality gap will make it difficult to evolve compensatory behaviors. A more recent algorithm, the *T-Resilience* algorithm by Koos et al. [73], has been developed to address these shortcomings as well as other concerns. This new algorithm incorporates techniques specifically developed to cross the reality gap, including the Koos's previous work involving a multiobjective approach to promote transferable controllers [75].

Our approach, combining self-modeling with offline optimization and adaptive control will allow an autonomous robot to both quickly recover from minor issues and compensate for potentially crippling damage. However, unlike the related studies described above, our

work does not rely on explicitly addressing the reality gap. Instead, our use of an adaptive controller will allow robotic systems to adapt to both unforeseen changes that occur throughout the lifetime of a robot as well as any poorly modeled or unmodeled dynamics that typically account for the reality gap. In addition, the self-modeling process has an added benefit of providing a means to automatically generate a new compensatory behaviors even when faced with completely unanticipated scenarios (i.e., scenarios not considered during the mode discovery process). In the case of a robot, if the system has an accurate simulation model of its own morphology, including sensors and actuators, it can derive new behaviors dynamically using the model rather than the physical system. Our research into mode discovery and self-modeling are discussed in Chapter 7.

2.5 Robotic Fish Platform

Techniques discussed in the previous sections of this chapter provide a foundation upon which to build a holistic approach for autonomous robots to handle adverse conditions. However, each of the techniques must be extended to enable this integration. To conduct our research, we used robotic fish as the target device.

Similar to live fish, robotic fish accomplish swimming by deforming their bodies or fin-like appendages. This form of locomotion offers certain key advantages relative to traditional propeller-driven aquatic vehicles. First, robotic fish are potentially more maneuverable, which is critical when operating in cluttered underwater environments [2, 123]. Second, since robotic fish produce very low acoustic noises and exhibit wake signatures similar to those of live fish, they are less intrusive to aquatic ecosystems and offer stealth in security-related applications. Third, with fin/body movements occurring at relatively low frequencies (typically only a few Hz), these systems are less likely to harm aquatic animals or become jammed with foreign objects. Given these characteristics, robotic fish are anticipated to play an important role in environmental monitoring [123], inspection of underwater

structures [61], tracking of hazardous wastes and oil spills [130], and the study of natural systems [84, 43, 122, 105].

Several recent studies have addressed the design of aquatic robots with compliant components [87, 5, 62, 85, 18, 42, 102]. For example, Low and Chong [85] used statistical methods to investigate the effect of control and morphological design parameters on the resulting thrust of a robotic fish with a compliant caudal fin. Esposito et al. [42] performed a similar analysis of a caudal fin with six independently actuated fin rays. While studying a single kinematic parameter, the phase difference between the driving angle at the base of a flexible caudal fin and the fin-bending angle, Park et al. [102] discovered that maximal thrust occurs at a specific phase difference, even when the morphology of the caudal fin is changed. Although many recent works in this area focus on flexible caudal fins, as evidenced by the studies mentioned above, Daou et al. [40] have investigated a compliant body, where both the head and caudal fin were rigid.

Autonomy and adaptation are particularly important in aquatic environments, where human oversight is often limited, if not impossible. However, while investigations of robotic fish have produced many advances over the past two decades [125, 69, 6, 133, 41, 78, 85, 132], robotic fish still do not rival their biological counterparts in terms of robust swimming abilities. Different from the above studies, our work is concerned with improving, through evolutionary optimization, thrust (which is necessary for maximizing average speed), energy efficiency (which is important to deployed systems), and adaptability.

In Chapter 3 we demonstrate how evolutionary algorithms can be utilized to optimize morphological characteristics and control patterns. The interaction between geometry and flexibility complicates the underlying dynamics. For example, determining the so-called "optimal" *foil*-shape for robotic fish tail fins depends on a given performance metric (e.g., speed, power) [44]. In Chapters 4 and 5 we extend this work to include multiobjective optimization of swimming abilities and energy efficiency. In Chapter 6, we describe our investigations into evolving adaptive control and applying it to robotic fish. Furthermore,

in Chapter 7 we show that adding a layer above adaptive control—in which we define mode boundaries and monitor for more severe changes, triggering a self-modeling routine when needed—enables aquatic robots to select/evolve new behaviors that compensate for possible damage. By combining offline optimization, adaptive control, and self-modeling on robotic fish, we are be able to study how autonomous robotic systems can be made more robust in the face of uncertain environments.

Chapter 3

Evolving Caudal Fin Morphology and Control Patterns

We begin our study by exploring an evolution-based methodology for the design of a robotic fish caudal fin and the control pattern that governs its motion. Evolutionary optimization occurs in a rigid-body dynamics engine that incorporates a mathematical model of the hydrodynamics associated with a robotic fish. The chief contribution of this chapter is an evolution-based design method, integrating recently developed dynamic models, of flexible materials, that can be adapted into a general robotics engineering process.

3.1 Methods

To create a hydrodynamics environment, we built a simulator atop a mathematical model of flexible materials proposed by Wang et al. [128] and an open source rigid-body dynamics engine, the Open Dynamics Engine (ODE) [116]. Using rigid-body dynamics, natural caudal fin motion is approximated by dividing the fin into multiple discrete segments connected by a spring and damping system [128]. The mathematical model we use to com-

Some of the results and descriptions in this chapter were published in [26] and [23].

pute the forces produced by fins is based on Lighthill's theory [80], for which the motion of each point on a body can be used to determine how much propulsion is generated. For most of this chapter, we consider only rectangular fins; however, later we show an extension to Wang's model in which we investigate the simulation of "irregular" fins (i.e., fins that are not rectangular and have nonuniform flexibility).

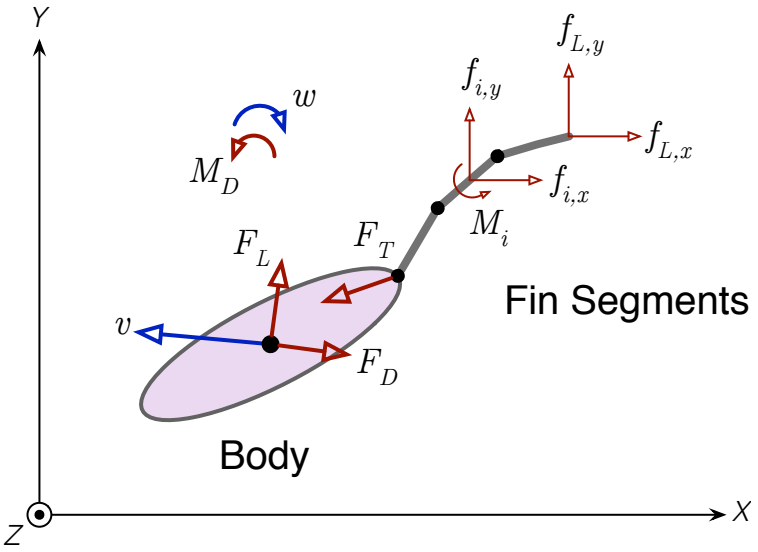


Figure 3.1: Graphical representation of the simulated hydrodynamics. Linear velocity v and angular velocity w are the result of thrust force F_T, drag force F_D, lift force F_L, and drag moment M_D. F_T is calculated as the sum of all forces acting on the fin segments.

The hydrodynamics and spring forces are shown in Figure 3.1. The force acting on each fin segment f_i can be calculated independently, and the resulting thrust force F_T is a summation of all segment forces, including an additional force that acts at the tip of the final segment f_L. In the figure, three segments are shown (although the model generalizes to any number of segments), along with the forces that apply to each individual segment. According to the mathematical model, each fin segment generates two component forces, a resistive component and a propulsive component, both of which are described by Equation 3.1:

$$\vec{f}_i(\tau) = \begin{pmatrix} f_{i,x}(\tau), \\ f_{i,y}(\tau) \end{pmatrix} = -m\frac{d}{dt}(v_\perp \hat{n}), \tag{3.1}$$

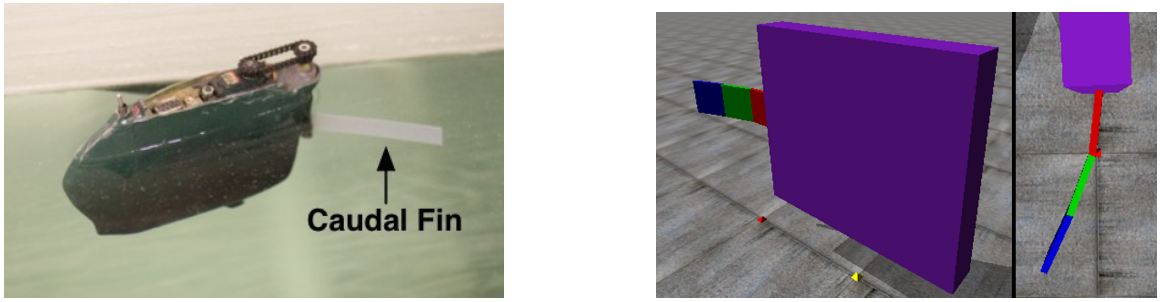
where m denotes the mass per unit length (typically taken as $\frac{1}{4}\pi\rho d^2$ where ρ is the density of water and d is the height of the fin), τ is the location on the fin where the force acts, and $\hat{n}$ and $v_\perp$, respectively, are the unit direction and velocity perpendicular to the fin. The tip of the final segment experiences an additional force described by Equation 3.2:

$$\vec{f_L} = \begin{pmatrix} F_{L,x} \\ F_{L,y} \end{pmatrix} = \left[-\frac{1}{2} m v_\perp^2 \hat{m} + m v_\perp v_\parallel \hat{n} \right]_{\tau=L}, \tag{3.2}$$

where τ=L represents the posterior end of the fin, and $\hat{m}$ and $v_\parallel$, respectively, are the unit direction and velocity parallel to the fin. These hydrodynamic forces can be calculated given the X and Y positions of each fin segment over time. At the base of the fin, which is attached to the body, a motor drives the rhythmic motion in a oscillatory pattern.

Along with caudal fin dimensions, the Young's modulus of elasticity, which is captured in the parameters for the springs and dampers, determines flexibility. This relationship provides a means of transferring simulated designs into real materials using known and inferred material properties. This topic is discussed further in Chapters 4 and 5.

The simulated robotic fish is modeled after a physical robotic fish prototype (shown in Figure 3.2(a)), which was originally constructed to test the performance of different fin dimensions and material stiffnesses. A representation of the virtual model can be seen in Figure 3.2(b), showing the main body and a three-segment caudal fin.



(a) (b)

Figure 3.2: (a) The robotic fish prototype. (b) Depiction of the virtual fish model with a three-segment rigid-body caudal fin. Movement of the 3D-printed rectangular caudal fin is accomplished using a servomotor with a set range of motion and period of oscillation.

3.1.1 3D-Printed Fins.

Test fins were fabricated using an Objet Connex350 multi-material 3D printer. Fins were printed with a combination of different physical materials to yield flexibilities that resemble the motion observed in simulation. As demonstrated in [108], a 3D printer can considerably improve the efficiency of an experimental design process. Several iterations of printed parts can be fabricated in a matter of hours. The printed fins were attached to the robotic fish prototype and evaluated in an aquatic test environment. An image of the physical robot with attached fin is shown in Figure 3.2(a).

Time trials were used to determine the average velocity achieved by each fin, while visual observations helped determine the flexibility of fins during movement. In these physical trials, the height, length, and thickness of each fin were fixed at 2.5, 8.0, and 0.1 cm, respectively, and the Young's modulus of elasticity was provided by the manufacturer data sheets. For each of the printed fins, the robot was placed in a test tank and allowed to reach a stable swimming speed before the average velocity was computed. The stiffness of each fin can be calculated with Equation 3.3:

$$K_s = \frac{Edh^3}{12l}, \tag{3.3}$$

where K_s represents a material's torsion spring constant, d and l denote the height and length of the fin, respectively, E represents Young's modulus of elasticity for the material itself, and h is the thickness of a fin. These values can be directly used in simulation during optimization trials and provide a means of comparing simulation and physical results.

3.1.2 Oscillatory Controllers.

CPGs [71, 135, 33] and other oscillatory controllers have been studied for controlling robotic fish [2]. CPGs are able to coordinate multiple actuation devices, an important feature for robots with more than one servomotor [2, 137, 138]. Crespi et al. [33] demonstrated this

concept when they controlled an amphibious robot by producing locomotion patterns for both swimming and crawling behaviors. In this chapter we describe experiments in which we implemented the neural oscillator proposed by Matsuoka [88, 89], referred to as a Matsuoka Neural Oscillator (MNO), and compared it to sinusoidal controllers. An MNO is composed of two identical, coupled artificial neurons that generate an oscillation based on a fourth-order system of differential equations. The amplitude and frequency of an MNO output can be modulated by adjusting the dynamic parameters.

To produce rhythmic caudal fin motion, we tested both sinusoidal waves and MNOs. With both of these controllers, the amplitude and frequency of the pattern determine the average velocity of the robotic fish, while a bias (i.e., vertical shift) affects the turning rate. Equation 3.4 shows how a sinusoidal controller governs the angle of between a body and caudal fin:

$$\theta(t) = \theta_A \, sin(2\pi f t) + \theta_b, \tag{3.4}$$

where θ represents the fin angle, θ_A and θ_b are the amplitude and bias, respectively, and f is the frequency.

Figure 3.3 depicts the block diagram of an MNO. In contrast to a pure sinusoid, the output of an MNO contains an initial transient response prior to steady-state oscillations. Furthermore, a signal generated by an MNO will comprise a more diverse frequency response. These two MNO characteristics may provide an advantage over a simple pure sinusoid. Figure 3.3 depicts the two identical, interconnected neurons. The neurons provide a mutually inhibiting feedback, which is the source of stable oscillation.

For an MNO, τ and T represent the rising and the inhibitory time constants respectively, a represents the strength of mutual inhibition, b denotes the degree of adaptation, and c is a varying continuous input. An in-depth overview of these parameters and the governing differential equations can be found in [89].

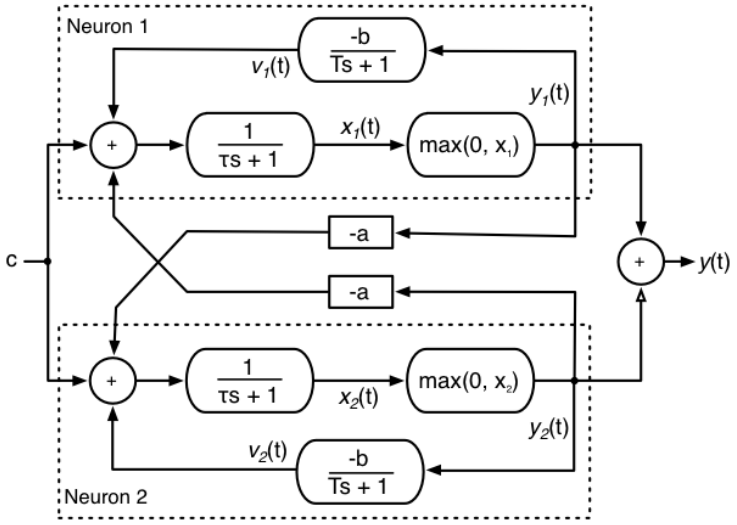


Figure 3.3: The MNO comprises two mutually-inhibiting neurons. The output of the the MNO is equal to the output of neuron 1 minus the output of neuron 2 (Adapted from [89]).

It is important to note that the output generated by either type of controller can exceed the physical limits of the simulated robotic fish motor. When the control signal is above the motor angle limits (± 55 degrees), the motor will reach the limit and hold. Likewise, when the required angular velocity is too high, the motor will reach its maximum angular velocity and maintain that velocity until the control signal rate-of-change decreases below the angular velocity threshold. By not limiting control signals to physical constraints, evolved controllers will be able to produce non-sinusoidal caudal fin motion. This behavior can be seen in the experiments described below (see Figure 3.9 for an example).

3.1.3 Genetic Algorithm.

The first experiments were performed using a variant of the conventional GA, and differential evolution is used for the irregular fins. In general, every individual in the population can encode the caudal fin morphology as well as the controller parameters depending on the focus of the experiment. Each individual genome is represented as a set of genes, where each gene encodes a real value in the range [0,1]. The translation from gene to parameter is as follows: MNO parameters a, b, c, τ, and T are mapped into the range [0,10], sinusoidal

control parameters *amplitude* and *bias* into [0,720] degrees, $frequency$ into [0,20] Hz, and fin flexibility s is mapped into the range [0,1.8e-2] Nm.

As with most GAs, we start by randomly initializing the population, and then continue through a sequence of generations in which each individual is evaluated and stochastically allowed to reproduce. Different from a conventional GA, our variant employs a 1D trivial geography [117] in which every location in the population has geographical meaning (i.e., the population is divided into overlapping subpopulations). Our GA uses a size 3 tournament selection for recombination. This implementation maintains the population's diversity with little overhead. Additionally, we implemented single-point crossover and mutation. In the event of a mutation, the affected gene is randomly displaced by adding a value drawn uniformly from a Gaussian distribution with a zero mean and a variance of 0.01; values above or below the range [0,1] after mutation are set to 0 or 1, respectively.

3.2 Evolving Fin Morphology

This section can be divided into three parts: mathematical model validation, physical validation, and evolutionary optimization. We first compare our simulation results with data derived directly from the mathematical model. Next, we perform a similar comparison between simulation and data gathered from physical experiments. Once our simulation environment was validated, we applied evolutionary algorithms to optimize fin for shape and flexibility. For all experiments in this section, the body-fin motor oscillates at a fixed rate of 0.9Hz in a 30 degree symmetrical range of motion. In later sections, we also evolve the oscillatory motion.

3.2.1 Mathematical Model Validation.

Prior to physical validation and evolutionary experiments, it was important to ensure that our simulation environment matched the mathematical model. Any disparity between

simulation and model could signify an error that would make evolutionary results meaningless. Two algorithms were employed to optimize the stiffness of the simulated caudal fin. In both experiments, only the Young's modulus value was altered.

The first algorithm was a basic hill-climber. For this experiment, 100 independent runs were conducted. Every run was initialized with a different seed and a Young's modulus value chosen uniformly at random from the range [0, 5 GPa]. Every Young's modulus value was evaluated by translating it, with Equation 3.3, to the spring coefficients that govern caudal fin flexibility. For evaluation, the simulated robotic fish was allowed to swim for 10 seconds. The fitness of each Young's modulus was computed as the average velocity achieved over this evaluation period. Each hill-climber run began with the evaluation of a randomly-chosen initial Young's modulus value. Subsequent values were generated by shifting the current value by a random number chosen uniformly from a Gaussian distribution with a mean of 0 and a variance of 0.1. The resulting Young's modulus was then evaluated, and the better performing (higher average velocity) value was kept and used to generate the next test case. In each run, this process was repeated until 100 candidate values had been evaluated. Every hill-climber instance converged to an optimum Young's modulus of roughly 1.9 GPa.

The second algorithm deployed was the aforementioned genetic algorithm. The primary purpose of this experiment was to confirm that the simulation environment could be used effectively with an evolutionary algorithm. This experiment comprised 30 independent runs. Each run was seeded with a different value and a population of 125 randomly generated individuals. Every individual was evaluated in a process identical to that used in the hill-climber experiment, and the populations were evolved for 100 generations.

Results from the evolutionary experiment closely resembled those of the hill-climber, with the most fit individuals in every run having a Young's modulus near 1.9 GPa. Data generated from the mathematical model can be seen in Figure 3.4, and results from the two simulation experiments are shown Figure 3.5. The experimental results show that both the hill climber and evolutionary approaches yield near identical solutions (i.e. a Young's

modulus of 1.9 GPa). This is an expected result, as both experiments rely on the same simulation environment.

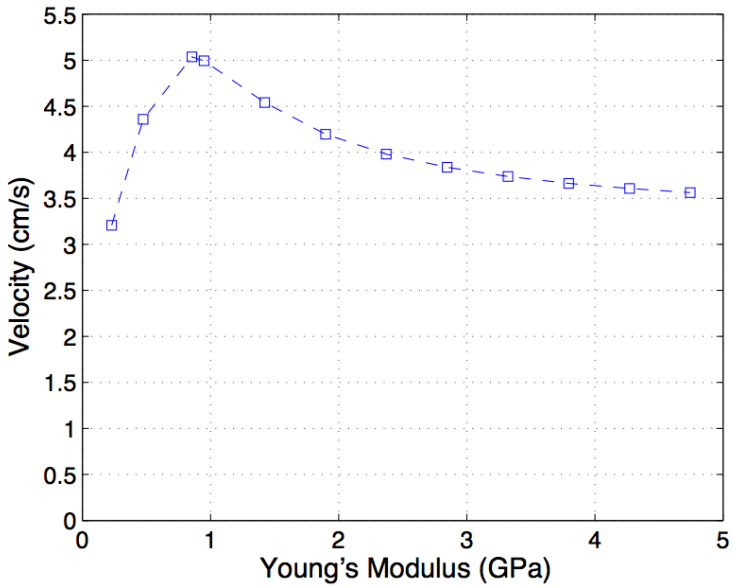


Figure 3.4: Predicted velocities for different Young's Modulus values from the mathematical model calculations. Note that this assumes that the body is anchored.

Comparing Figures 3.4 and 3.5, a disparity between model and simulation results is apparent. Specifically, the model predicts a maximum velocity of roughly 5.1 cm/s at a Young's modulus near 0.9 GPa, while simulation results achieve a maximum average velocity closer to 1.4 cm/s at a Young's modulus near 1.9 GPa. Despite the differences, both figures show the same trend, in which intermediate values of the Young's modulus produce the fastest robotic fish. Additionally, the disparity between figures can be explained by closer examination of the model and simulator. The most marked differences are that the mathematical model assumes the robotic fish body does not affect caudal fin motion, and the caudal fin segments are without mass. Neither of these assumptions is carried over into the simulation environment, and both of these factors would cause simulated robotic fish to appear *slower* than model data would predict.

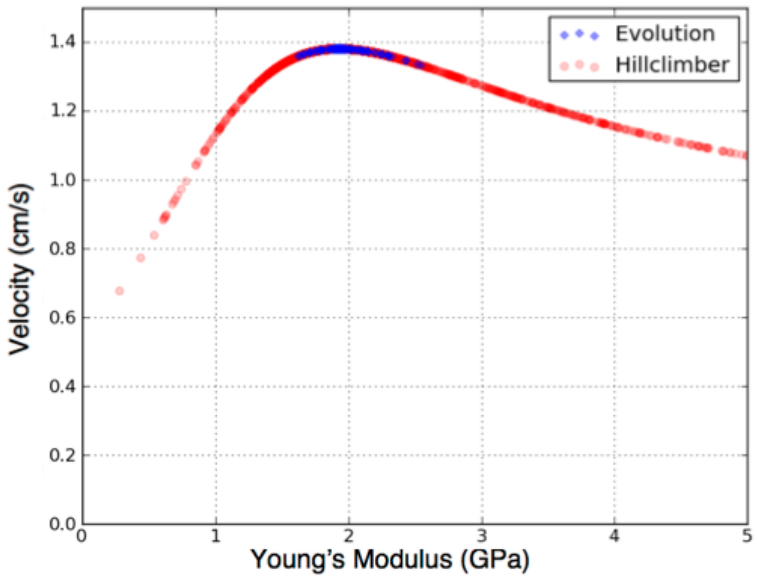


Figure 3.5: Results of the hill climber and evolutionary runs for determining the optimum stiffness of a fixed dimension fin. Both methods converged on a common stiffness yielding the highest average velocity. Darker shades indicate clustered results from different trials.

3.2.2 Physical Validation.

Next, we fabricated caudal fins with a 3D printer and tested them on the robotic fish prototype described earlier. Six unique fins were printed, each with a different Young's modulus. The materials ranged from extremely flexible (TangoBlackPlus) to nearly inflexible (VeroWhite). Each printed fin was attached to the robot and tested in the aquatic environment; the average velocity was measured over 5 separate trials. The results of this experiment are plotted in Figure 3.6. Consistent with the predicted performance, the plot shows that an intermediate flexibility produces the highest average velocity. However, direct comparisons between simulation and reality are not possible due to current limitations of the 3D printed materials. Specifically, the materials do not have an exact Young's modulus value, but rather the manufacturer provides a range of possible values for each material (materials properties are not guaranteed to remain constant between print jobs). For example, VeroWhite has a modulus in the range of 2-3 GPa, while the other materials have lower-value ranges.

However, since the mathematical model, simulation, and physical data are all for fins of identical shape, some comparisons can be made. First, the velocity values of the physical robotic fish are closer to mathematical model predictions than they are to simulation results. In addition, the optimal Young's modulus for all results is in the range of 1-2 GPa. The reason for the disparity in the model predictions was discussed in the previous section, however it is also apparent that simulation results do not match reality. The maximum velocity of 3.7 cm/s in the physical experiments is nearly twice the maximum simulation velocity. As with the mathematical model, certain approximations were made in the simulation environment. For instance, distributed forces were treated as single point forces, and the flexible fin was split into just three segments. By decreasing the size of each segment and increasing the number of segments, the motion and discretization of forces will likely be more realistic and increase the accuracy of the simulation.

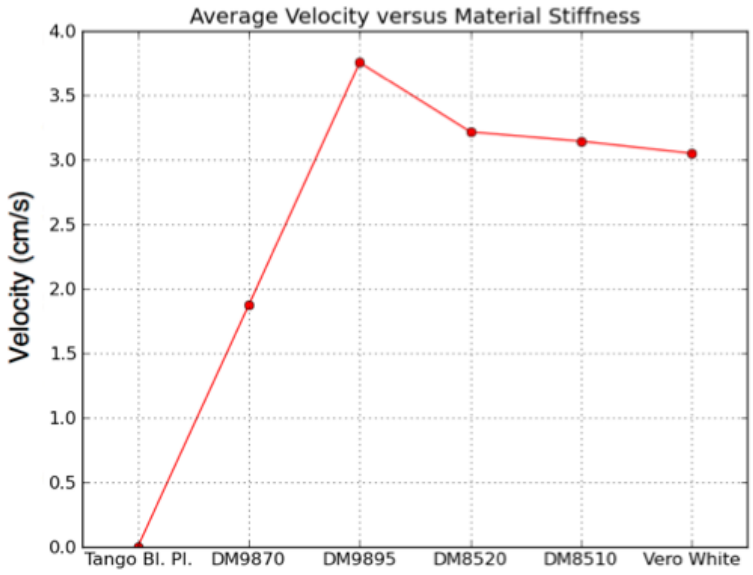


Figure 3.6: Observed average velocity for different materials used in printed fins. Stiffness increases from left to right in the plot.

As a secondary measure of performance between the simulation and physical experiments, we recorded the shape of fins as they oscillated. Figure 3.7 presents a side by side

comparison between a simulated flexible caudal fin and the 3D printed version on the robot. This visual observation helps to reinforce the viability of simulating flexible caudal fins.

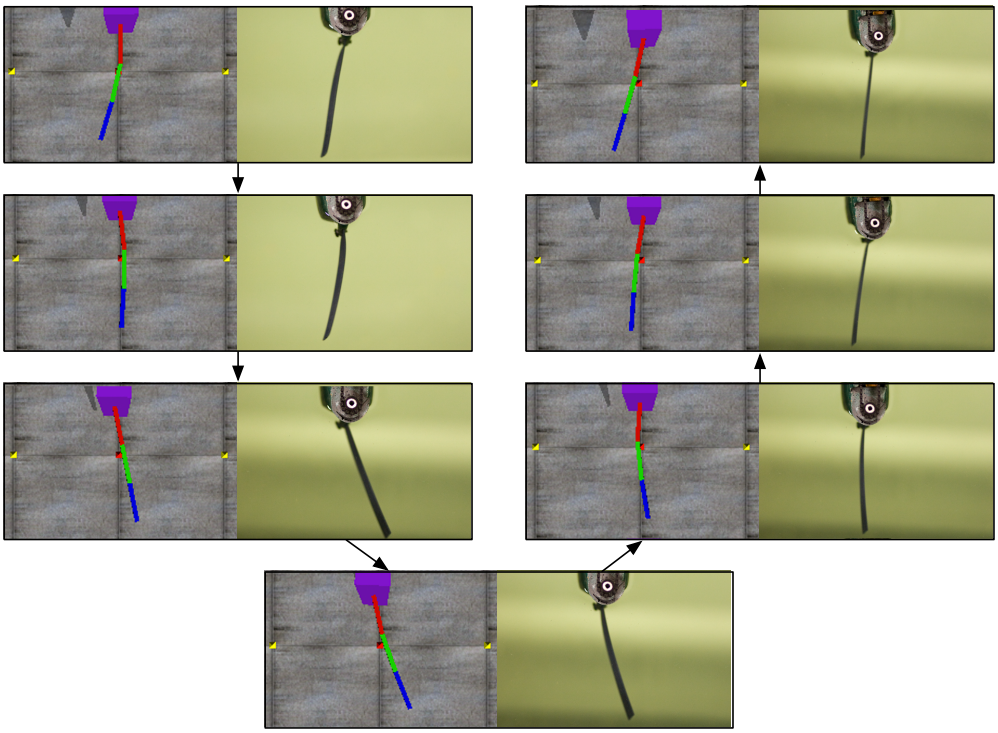


Figure 3.7: Visual performance of the evolved flexible fin in simulation (left) versus a fabricated flexible fin tested on the prototype robot (right).

3.2.3 Evolutionary Optimization.

Upon completion of comparisons between the mathematical model, simulation, and physical results, optimization was expanded into a full evolutionary computation run in which the Young's modulus and dimensions of a rectangular caudal fin were simultaneously evolved. Fin shape was allowed to evolve under the constraint that the overall area of the length-height face and the thickness of the fin remain fixed. This created a state in which the height of the fin was dependent upon the length of the fin. As such, the two parameters to evolve were the Young's modulus and length of a fin. Practical considerations on the overall dimensions of the fin were also taken into account as a maximum length of 14 cm (length of the robotic fish body) and a minimum length of 4 cm (half the length of previous experiments)

were imposed upon evolution. Values outside of this range could suffer from transferability issues given electromechanical constraints such as the maximum torque exerted by a servo. Again, an individual run consisted of 125 individuals evolving for 100 generations. Similar to the previous evolutionary experiments, tournament selection of size 3 and elitism were used to select the parents for the next generation. Unlike earlier experiments, however, single point crossover was added so that individuals could be generated as a combination of two selected parents. In total, 30 replicate runs were conducted to find the relationship between fin stiffness, fin shape, and average velocity.

From the evolutionary runs, a set of optimum values was found for both the Young's modulus and dimensions of the fin. The Young's modulus found in the trial was 7.55 GPa, and the caudal fin length and height were 14 and 1.43 cm respectively. Hence, the fittest solutions reached the maximum fin length allowed at a cost of fin width. This result was expected, as a longer fin will be able to generate larger propulsive forces, while width has a lesser effect on this force. This characteristic can be seen by close examination of Equation 3.2, where the length of a fin is a linear factor, and longer fins will have a higher angular velocity near the posterior of the fin.

While the Young's modulus found in the trial is larger than that found in prior experiments, the resulting material stiffness is similar: 1.35×10^{-3} N m for the original experiments, and 1.73×10^{-3} N m for the full evolutionary experiments. This result suggests that a single stiffness value may be adequate for any rectangular caudal fin dimensions. The reason these stiffness values are similar is that as length increased, the Young's modulus also increased to maintain a fairly constant value. Figure 3.8 presents the three dimensional fitness landscape found in the evolutionary run. As shown, a peak is located at a modulus of elasticity of 7.55 GPa and a length of 14 cm. This combination yielded an average velocity of 2.2 cm/s. This landscape would suggest that for each set of dimensions there is a specific Young's modulus that correlates to the overall best performance for a fin.

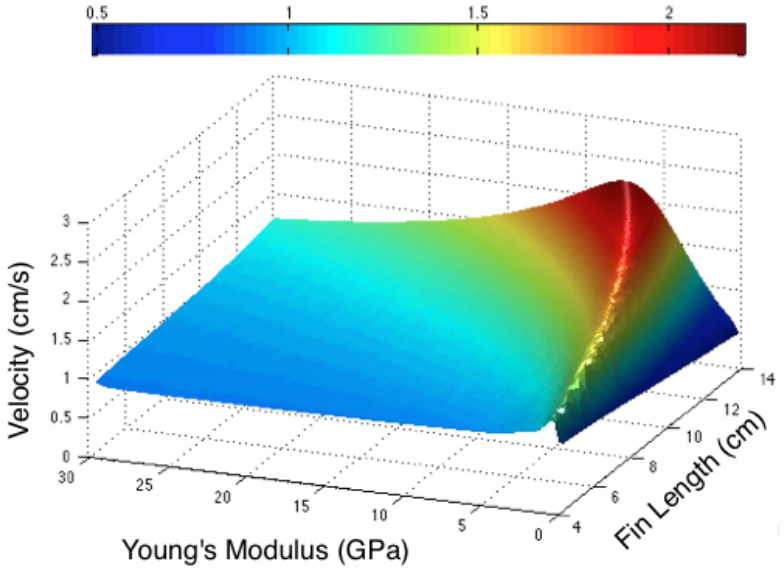


Figure 3.8: Visualization of the fitness landscape for different shape and stiffness fins. Note that height is dependent upon length in determining shape, therefore, height has been omitted from the data. As the length of the fin increases, the Young's Modulus increases as well to maintain similar stiffness fins for different lengths.

The complex dynamics of an underwater environment make designing efficient robotic fish a challenging engineering endeavor. Considering the difficulty, it is desirable to create an automated design process by which robotic fish can be optimized for a specific task. Making use of the hydrodynamic model for a robotic fish caudal fin, we have shown that an *in silico* process can be used to optimize the Young's modulus of a flexible fin. In simulation, we observed that the optimum Young's modulus is dependent on both the caudal fin motion and dimensions. Specifically, for any combination of fin frequency, amplitude, height, width and length there will be a unique Young's modulus optimum. However, when the Young's modulus was simultaneously evolved with fin shape, we found that the overall resulting fin stiffness exhibited comparable characteristics. Generally, higher values of length and Young's modulus produced faster swimmers.

3.3 Evolving Fin Morphology and Control Patterns

In the preceding section, we investigated flexible caudal fins using a fixed sinusoidal controller. In this section, evolution is responsible for optimizing both control and morphology in order to improve locomotion. Through controller-morphology evolution, the complex internal interactions between actuation and material properties can be exploited to produce a system that matches the needs of its environment. Giving an EA the freedom to adjust both morphology and control allows solutions to have a tight coupling between motion and physical form.

We first performed a series of experiments in which we compare MNOs to pure sinusoidal signals. In these preliminary experiments, the target of evolution is to reach a maximal average velocity. To ensure that initial MNO transients and starting bias do not affect the stable average velocity measurement, all evaluations began after a 5 second start-up period; the total evaluation period is 15 seconds. To provide a basis for the evolution experiments we, first separately evolved morphology and control. Following these velocity-only experiments, we proceeded to evolving sinusoidal controllers for minimal turning radius.

In the previous section, we observed that for a given control pattern there is likely a unique optimal morphology. This phenomenon is due to a matching between the natural frequency of the fin and the fundamental frequency of the rhythmic control oscillations. For a given fin flexibility, the natural frequency can be defined as the frequency at which the fin would vibrate once placed into motion and is independent of actuation. The fundamental frequency of a control signal is best described as the frequency with the highest power in the frequency domain. To confirm this finding, we performed experiments in which the flexibility of the caudal fin is acted on by evolution, but control patterns are fixed. Additionally, we performed complementary experiments in which fin-flexibility is fixed, but control patterns are acted on by evolution. Each is described in turn.

3.3.1 Fixed Control, Evolved Morphology.

In these experiments, pure sinusoidal controllers govern the motion of the caudal-fin while the spring coefficient is adjusted by evolution. The amplitude of sinusoidal controllers was set at 30 degrees, while the frequency was set at 0.3Hz, 0.6Hz, and 0.9Hz in three independent experiments. These frequencies were chosen based on typical robotic operating modes. The results from these experiments are summarized in Table 3.1.

Table 3.1: Fixed-Controller Experimental Results

Controller		Spring	Velocity
Freq.	Ampl.		
0.3 Hz	30°	3.58e-4 Nm	0.92 cm/s
0.6 Hz	30°	7.80e-4 Nm	2.37 cm/s
0.9 Hz	30°	1.38e-3 Nm	3.02 cm/s

From Table 3.1 it can be seen that as the frequency of the control signal increases from 0.3 Hz to 0.9 Hz, the optimal spring coefficient also increases. From this data, we can determine that the natural frequency of the caudal fin is greater than or equal to 0.9 Hz. Another takeaway from this data is that as the frequency increases so to does the resulting velocity. It is likely that frequency will increase only until the natural frequency of the fin is reached, at which point the average velocity will begin to decrease once again. Fundamentally, as the stiffness of a fin increases it can sustain more angular momentum, and the faster a fin oscillates the higher the resulting thrust force.

Control signals are not dependent upon the performance of the actuated hinge. This behavior is displayed in the 0.9 Hz experiment. Due to the angular velocity constraint, the actuated hinge is not able to reach the 30 degree amplitude before the controller signal begins the next cycle. This behavior is depicted in Figure 3.9. An interesting side effect of the hinge lagging behind the signal is that resulting hinge motion does not necessarily resemble a pure sinusoid, but rather a triangular wave.

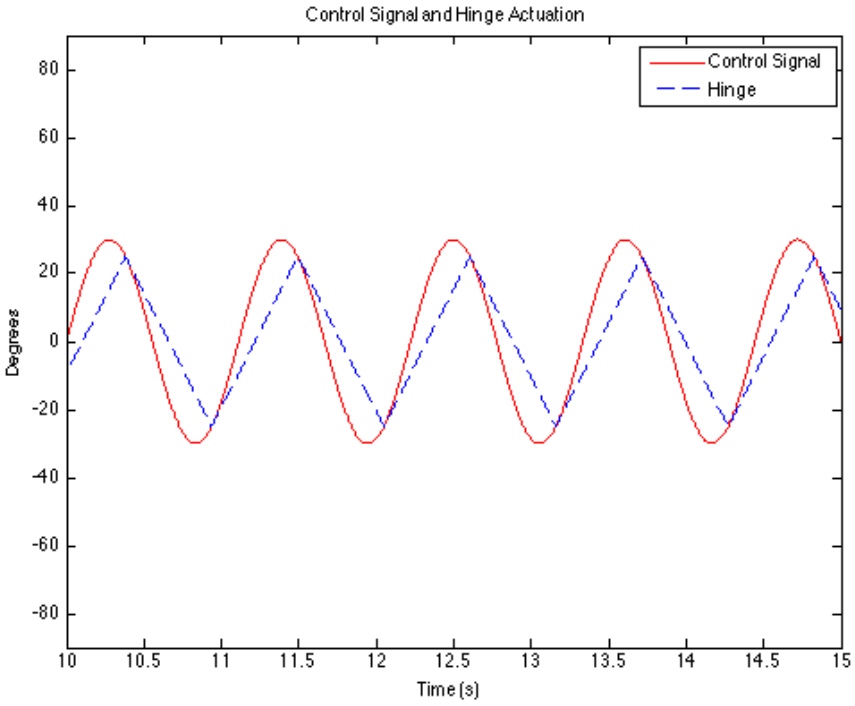


Figure 3.9: An example of a controller signal that is beyond the capabilities of the simulated motor. The solid red line represents the signal generated by the 0.9Hz sinusoidal controller, and the dashed blue line the simulated motor hinge.

3.3.2 Evolved Control, Fixed Morphology.

In the next set of experiments we evolved control signals for a fixed caudal fin flexibility. This value (1.8e-3 Nm) represents the flexibility of the default robotic fish prototype caudal fin. The purpose of this experiment is to confirm that for a given spring coefficient, the caudal fin will have a natural frequency, based on material properties, at which it will produce the most thrust. First, we conducted 30 replicate runs in which sinusoidal controllers were evolved, such that the amplitude and frequency of the sinusoidal signal were adjusted by the genetic algorithm. The most fit sinusoidal controllers produced an average velocity of 3.9 cm/s, and converged to a signal with an amplitude and frequency of roughly 110 degrees and 1.26 Hz. After the sinusoid experiments, we conducted 30 more replicate runs with the MNO controller. Unlike the sinusoidal controllers, the best solutions varied among the different runs producing similar signals with different parameter sets. Figure 3.10 shows how widely

MNO parameters varied. Under such dynamics, it is impractical to hand-select parameters and expect to produce an oscillation, much less an oscillation capable of generating optimal behavior. This is an important benefit of genetic algorithms: many unique parameter combinations can be efficiently tested in a manner not possible for a traditional gradient-descent algorithm.

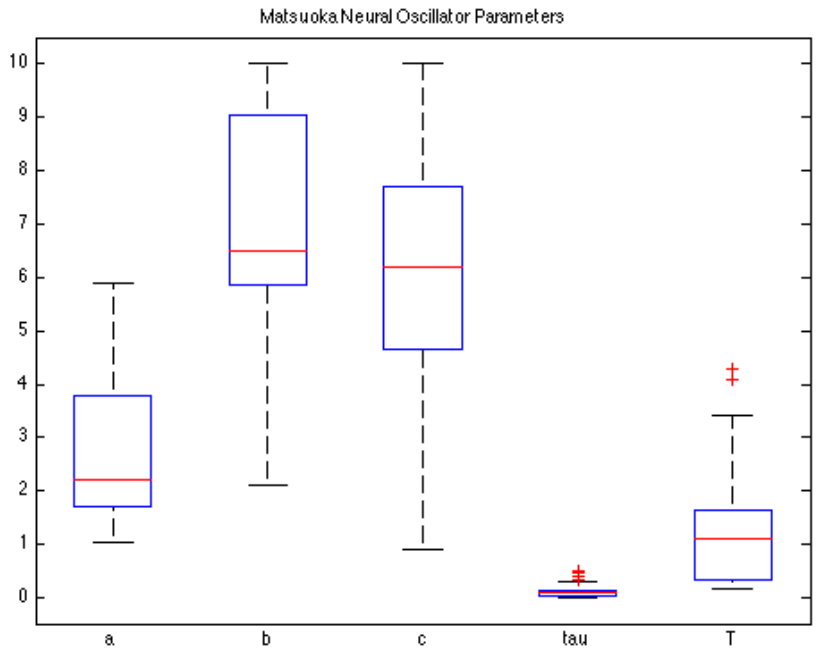


Figure 3.10: Boxplots for each MNO parameter from the *best* performing MNOs of each of 30 independent runs.

The fundamental frequencies of the *best* MNO solutions were in the same range (1.26 Hz) and produced similar velocities (3.9 cm/s). Frequency domain data for the best solutions from both experiments are shown in Figure 3.11. This plot demonstrates that for the given morphological characteristics, including the spring coefficient, the two controller types converged on a similar solution. This is an expected result, as a single control pattern will match the given caudal fin natural frequency.

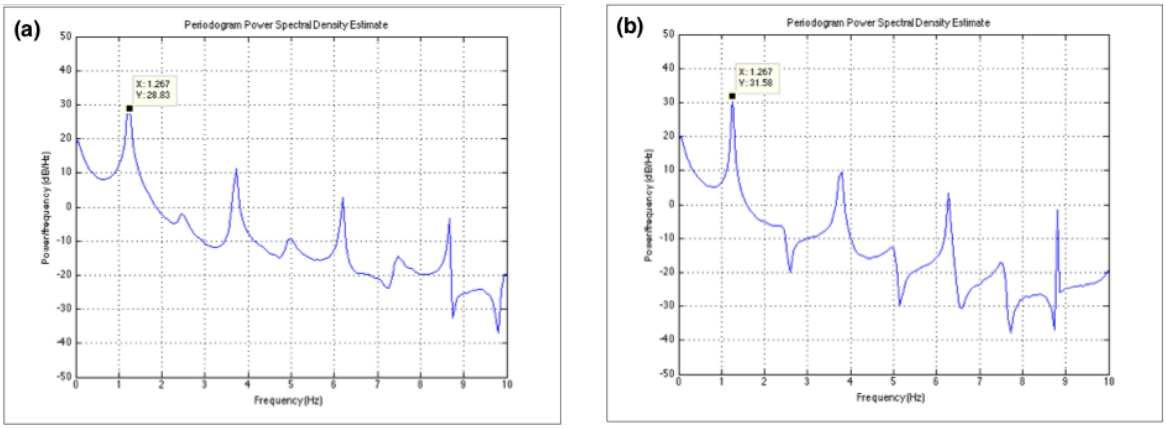


Figure 3.11: The frequency-domain response of the overall best sinusoidal (A) and MNO controllers (B).

3.3.3 Evolution of Control and Morphology.

Next, we turn to the main objective of our study: to investigate the optimization of robotic fish through the simultaneous evolution of morphology and control. As such, all following experiments evolve both control and fin flexibility. First, we evolved sinusoidal controllers with the spring coefficient governing flexibility. Again, the sinusoidal controllers converged to a single optimal solution. In this case, evolved sinusoids had a frequency of 4.2 Hz and an amplitude of 100 degrees. All optimal solutions also converged on a single spring coefficient of 1.7e-2 Nm, which is near the maximum value set for these experiments (1.8e-2 Nm). Given the higher frequency, it is not surprising that a higher spring coefficient evolved. As shown in the flexibility-only experiments, higher frequencies require a lower fin flexibility. Maximal average velocity also continued the same trend, as the most fit solutions produce enough thrust to average 11 cm/s, which is significantly higher than the tests in which only morphology or only control were evolved. This result demonstrates the importance of simultaneously optimizing control and morphology.

The final velocity-targeted experiments involve the joint evolution of MNO parameters with caudal fin flexibility. As before, the parameters and *best* solutions from one run to another varied widely (similar to the ranges displayed in Figure 3.10). The most fit solutions

from the 30 replicate runs also differed in final fitness. This result speaks to the difficulty in selecting the correct MNO parameters. As with the sinusoidal controller, flexibility was lower than in previous experiments, with the overall best solution having a spring coefficient of 9.2e-3 Nm. However, this MNO controller was capable of producing only enough thrust for an average velocity of 8.3 cm/s.

Figure 3.12 plots average trajectories for each of the evolutionary experiments in which maximal velocity is the target of evolution. From the figure, it can be seen that systems produced through the simultaneous evolution of morphology and control clearly outperform systems in which only control *or* morphology are evolved. A somewhat surprising result is that sinusoidal controllers outperform MNO-based controllers. Even when the population size and number of generations are increased, MNO controllers only come closer to matching the sinusoidal performance; they do not surpass it. For open-loop single actuation devices, it appears that controllers based on MNOs do not have any advantage over traditional sinusoids. In light of this result, the following experiments on the evolution of turning motions focus on sinusoidal-based controllers.

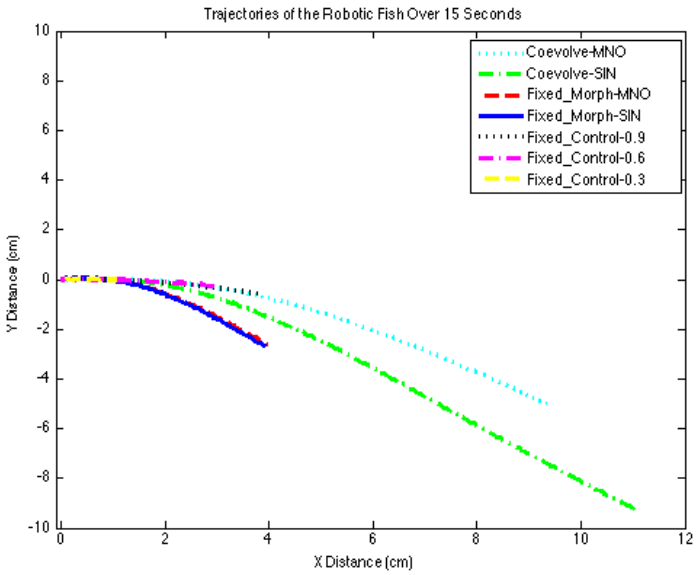


Figure 3.12: Paths of the robotic fish while being controlled by different oscillatory controllers and having different flexibilities. Note: plots overlap significantly at the origin.

3.3.4 Evolution of Turning.

Experiments targeted at turning were similar to the velocity experiments with the fitness being evaluated as average angular velocity in place of linear velocity. Solutions were rewarded for having a high average angular velocity and were evaluated over a period of 30 seconds. The increase in time allows a circular pattern to develop, which is helpful for the analysis. Evolved sinusoidal controllers produce locomotion in which the robotic fish begins moving straight forward during a transient phase, and then begins to turn in a tight circle. The best evolved sinusoidal controller produced a path of radius 4.5 cm, as shown in Figure 3.13. Both frequency and the spring coefficient were reduced compared to sinusoidal velocity experiments; the frequency and spring coefficient were 2.2 Hz and 1.2e-2 Nm compared to 4.2 and 1.7e-2. Also shown in Figure 3.13, the evolved control signal forces the fin to the maximum hinge angle with regular changes of less than 10 degrees. In Figure 3.13(b), the control signal (red dashed line) and resulting hinge motion (solid blue line) demonstrate that a pure sinusoidal control signal can result in non-sinusoidal motion.

3.3.5 Evolution of Velocity and Turning.

To explore more general robotic fish behavior, our final experiment is to evolve evolving control and morphology for both both velocity and turning. This is done by evolving two separate sinusoidal controllers: one specific to forward locomotion and another to turning. The challenge for the evolutionary algorithm is to strike a balance between the higher flexibility conducive to tight turning, and the higher stiffness values that produce greater thrust for maximal average velocity. Other than a shared spring coefficient for the driven caudal fin, the two controllers are independent of each other, and are each evaluated for 20 seconds under identical initial conditions.

For multiple objectives, typically a specialized algorithm, such as the multiobjective evolutionary algorithm NSGA-II [37], is utilized to form a complete Pareto front of *non-dominated* solutions. We perform such experiments in Chapters 4 and 5. However, for this

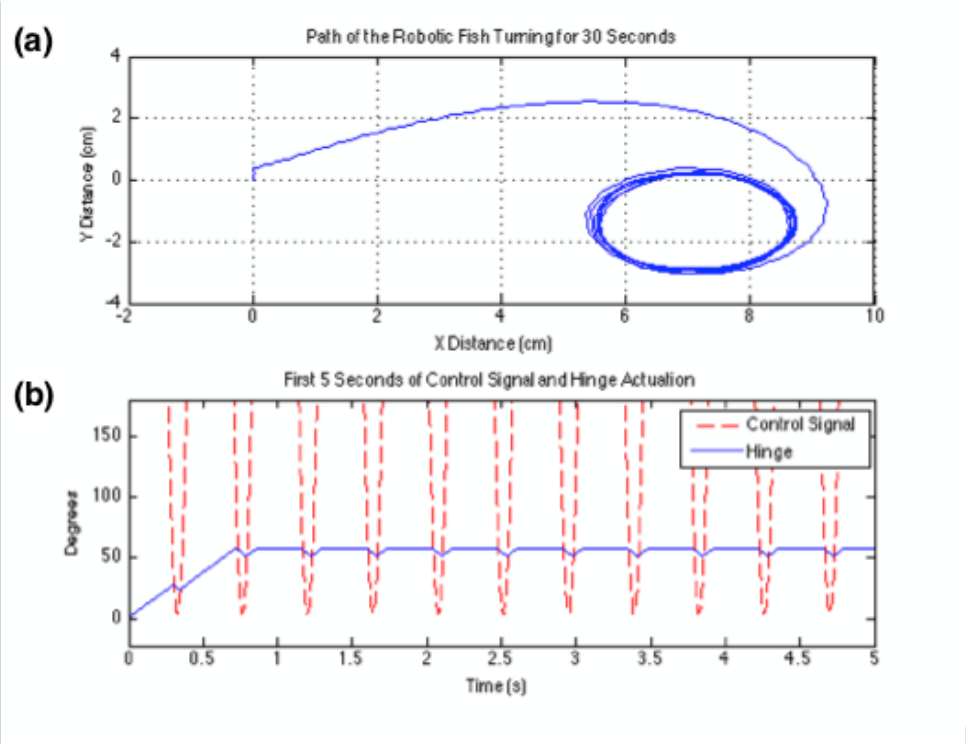


Figure 3.13: Evolved turning behavior: (a) path taken by the best solution, and (b) control signal and resulting servo angle.

study we focused on analyzing a simple set of data to find the trade-offs between optimizing turning and velocity. Some solutions may be optimal for turning, others for velocity, and still more which effect a compromise between these two objectives. Figure 3.14 shows the performance of two high-fitness solutions. The top portion of the figure shows the turning and velocity paths of a solution that is comparatively better at turning than achieving high velocity. The radius of the turning path is approximately 7 cm and the average velocity is 1 cm/s. The robotic fish performs adequately when compared to our preliminary studies, however it is less efficient than solutions evolved exclusively for turning or velocity. The evolved spring coefficient of 1.5e-2 Nm is in between previously found optimal values for turning (1.2e-2 Nm) and velocity (1.7e-2 Nm). This result indicates that the solution is a compromise.

The bottom portion of Figure 3.14 shows the paths of a different solution, which was better optimized for maximal average velocity rather than turning. This solution is very

similar to the highest fitness solution of the velocity evolution experiment. The frequency and spring coefficient were 4.6 Hz and 1.7e-2 Nm compared to the 4.2 and 1.7 seen before. As shown in the figure, however, the turning behavior has a significant wobble, that is, the center of the turning radius is constantly moving. This shift is due to a poor matching between the materials natural frequency, and the optimal frequency of the sinusoidal controller responsible for turning. This final experiment demonstrates that evolutionary search is capable of exploring the complex interactions between morphology and control, while at the same time balancing competing objectives, all in a nonlinear environment.

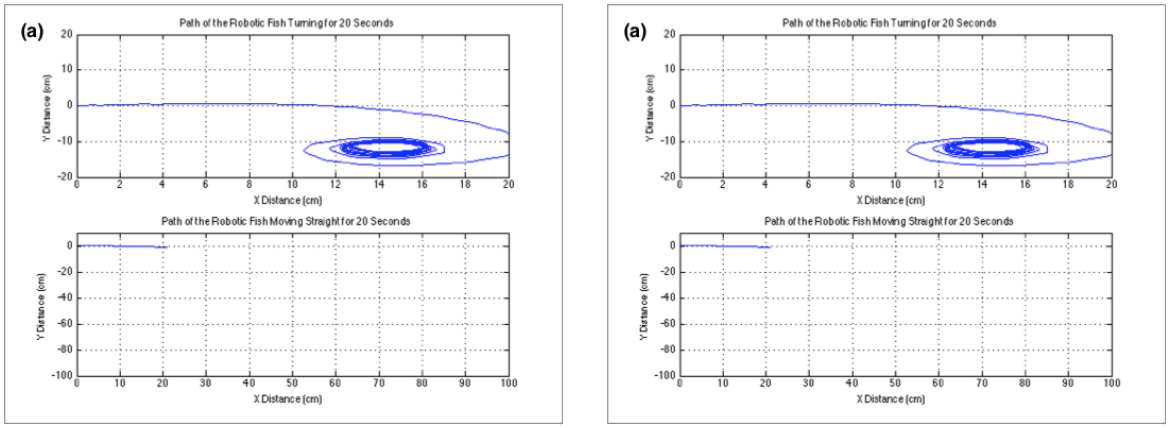


Figure 3.14: The paths produced by two evolved controllers: in (a) fin flexibility is more optimal for turning rather than velocity, and in (b) flexibility is more conducive to achieving linear velocity.

3.4 Fins With Nonuniform Flexibility and Non-Rectangular Shapes

The simulation model described in Section 3.1, based on work done by Wang et al. [128], has several limitations. First, all motion occurs in the two-dimensional surface of water. Effectively, the model does not consider the ability to dive or rise. Second, fins are required to be "elongated." That is, fins must be at least three times longer than they are tall. Finally, the model only considers fins of uniform flexibility and rectangular shape. While eliminating

the first two items would require the development of a new model, here we briefly explore fins of nonuniform flexibility and non-rectangular shape.

Figure 3.15 depicts the variables associated with simulating the irregular fins. In the diagram we show a caudal fin with five segments—we also use five segments in all experiments discussed in this section. Each segment has a unique value for length (L1..L5) and depth (D1..D5), and the joint between each segment has a value for flexibility (E1..E4). We performed five experiments in which we evolved a simulated robotic fish for maximum average velocity using the differential evolution algorithm [120] (details regarding DE will be presented in Chapter 6). We conducted 40 replicate simulations for each of the five experiments described in this section. Each replicate ran for 100 generations and included 100 individuals.

Evolved parameters include those listed in the figure, as well as the frequency and amplitude of the sinusoidal caudal fin motion. For these experiments, we used a numerical simulation, in Simulink [115], based on the dynamics described in Section 3.1. In the simulation, each segment and each joint can be configured separately.

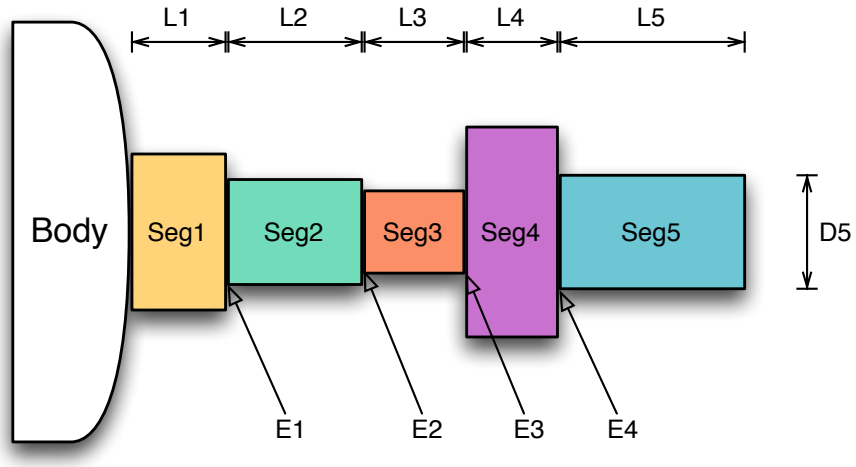


Figure 3.15: Diagram for irregular fins. Each segment has a unique length and height, and the joint between each segment has its own value for flexibility.

The five experiments include: (1) *baseline* (segments are identical and flexibility is uniform), (2) *length* (segment lengths are evolved separately), (3) *depth* (segment depths

are evolved separately), (4) *flexibility* (joint flexibilities are evolved separately), and (5) *irregular* (lengths, depths, and flexibilities are evolved separately). The best speeds for these experiments are shown Table 3.2, and the best results are achieved when all parameters are evolved individual (the *irregular* experiment).

Table 3.2: Speed Comparison Among Experiments

Baseline	22.1 cm/s
Depth	33.5 cm/s
Length	38.9 cm/s
Flexibility	22.1 cm/s
Irregular	51.8 cm/s

Figure 3.16 shows boxplots for the five parameters evolved in the baseline experiment. For this boxplot (and in the remaining boxplots described in this section) when only one value is given for a type of parameter (e.g., L1 is provided, but not L2, L3, L4, or L5) the remaining segments (or joints) are configured with the same value. For rectangular fins, an intermediate total fin length of 7.8 cm (the sum of all segment lengths), a Young's modulus of 3 GPa, and an amplitude of approximately 20° are fixed in the final population. Values for fin depth and control frequency converge to relatively small ranges of values.

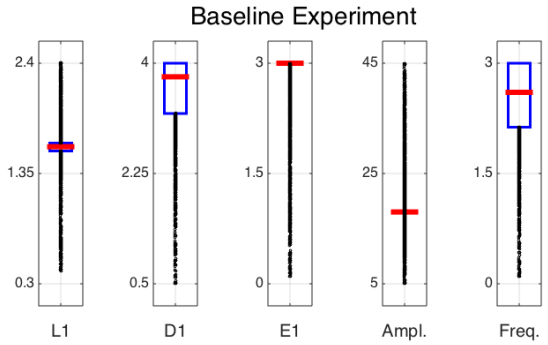


Figure 3.16: Boxplots for the baseline experiment. Parameters (from left-to-right) include L1 (the length of each segment), D1 (the depth of each segment), E1 (flexibility at each joint), and the control pattern's amplitude and frequency. The morphology parameters are as depicted in Figure 3.15.

Figure 3.17 shows boxplots for the length, depth, and flexibility experiments. In these experiments, only one type of morphological parameter (i.e., length, depth, and flexibility)

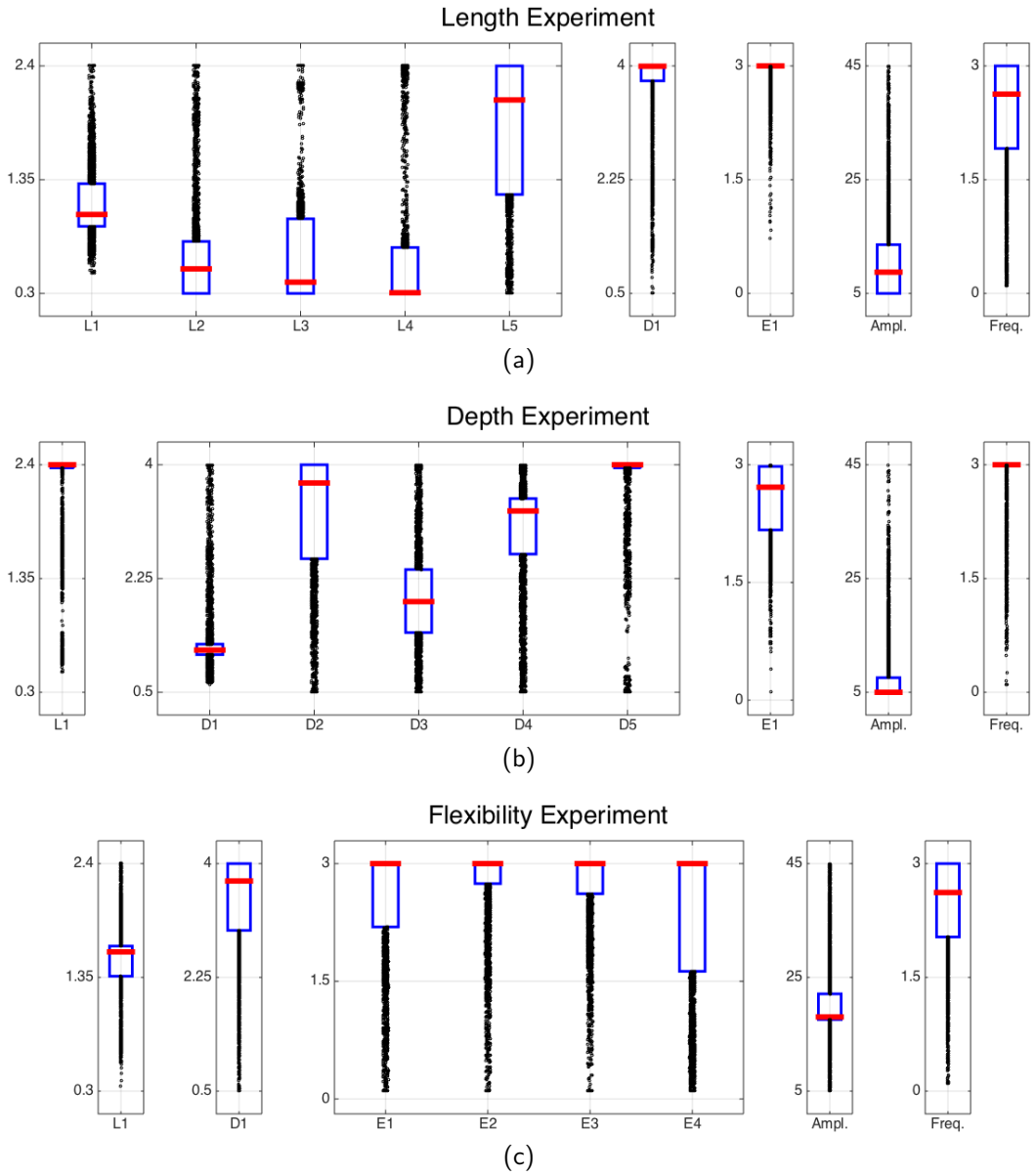


Figure 3.17: Boxplots for the length experiment (a), the depth experiment (b), and the flexibility experiment (c). For each set of boxplots, if a parameter value is unspecified (for example, L2 in (b)) then all segments or joints share the same value.

is evolved independently for each segment or joint. Among these experiments we find that the amplitude, frequency, and flexibility values have similar trends. Specifically, amplitude evolves to relatively low values, frequency evolves to higher values, and flexibility values are relatively high across all replicates. Aside from these trends, however, it appears that each experiment evolves fundamentally different values for length and depth. For example, in the

length experiment most segments have relatively low values, but in the depth experiment the value for length evolves near the maximum value.

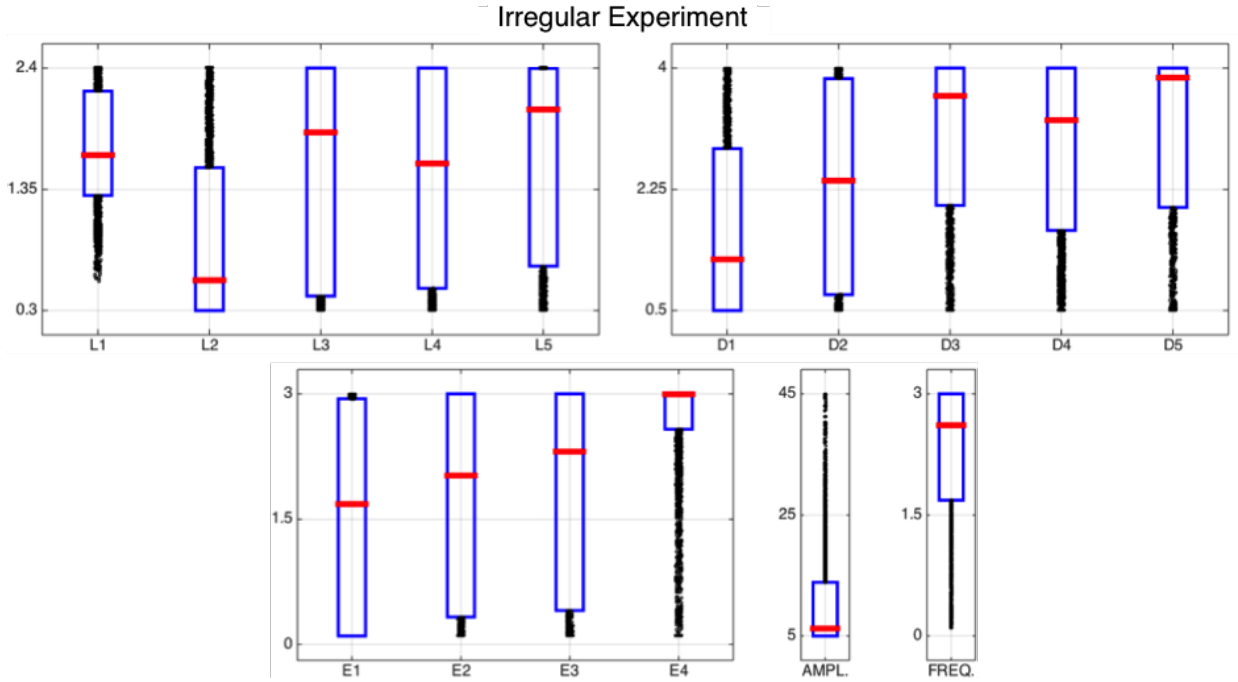


Figure 3.18: Boxplots for the irregular experiment. For this experiment all possible parameters are subject to evolution.

Figure 3.18 show boxplots for the irregular experiment. In this experiment, each segment has a uniquely evolved length and depth value, and each joint has its own value for flexibility. Additionally, the control amplitude and frequency are evolved. The final speeds achieved in this experiment (51.8 cm/s) shows that for maximum speed it is better to evolve all available morphological parameters. Even so, the boxplots show that none of the evolved parameters converge to final values, thus it is difficult to draw any further conclusions. It appears that DE was unable to solve this problem within the allowed number of generations. Increasing the number of generations and the population size may improve convergence.

While the results presented in this section are promising, for the remainder of this document we focus on rectangular fins with uniform flexibility. We made this decision for two reasons: (1) it enables us to maintain consistency between studies, and (2) the additional complexity (and parameters) required for the more complex fin shapes makes the analysis

of our multiobjective experiments and control algorithms less straightforward. Specifically, when we are considering adaptive control, it is worth keeping the simple rectangular fins discussed in Section 3.1 so that we can focus on the process of evolving control parameters.

3.5 Conclusions

In this chapter, we demonstrated an evolutionary design method for robotic fish caudal fins and control patterns. We first developed a simulation environment in which unique fin configurations could be tested. The simulation environment was created by combining a rigid-body dynamics engine with a mathematical model of a flexible caudal fin's hydrodynamics. To test the simulation environment, we first implemented a hill-climber algorithm. Given a fixed fin shape and control pattern, the hill-climber algorithm mapped out the fitness landscape for fin stiffness vs. velocity. These results were compared to data generated directly from the model, which confirmed that the simulation and the mathematical model have comparable dynamics, although the absolute values differ.

Hill-climber results were further validated through comparisons with physical experiments. With the aid of a 3D printer, an aquatic test environment, and a robotic fish prototype, we conducted a series of velocity tests for several 3D-printed fins. All fins were identical in shape, but had Young's modulus values ranging from very low to nearly inflexible. Plots of stiffness vs. velocity for the mathematical model, simulation, and physical experiments all showed a similar trend in which average velocity was maximal for intermediate caudal fin flexibility. This result demonstrates that it is possible for a simulation environment to capture key aspects of the dynamics of flexible materials.

To simultaneously optimize several fin parameters, we progressed from the hill-climber experiments to an evolutionary algorithm. A genetic algorithm was used to evolve both the Young's modulus and shape of a fin. From this series of experiments, we found that the most fit fins generally evolved to be as long as possible while maintaining a fairly constant

stiffness value. This result is consistent with the fact that longer fins generally produce larger propulsive forces. Additionally, our results show that for each fin shape, and presumably for each control pattern, there is an associated optimal Young's modulus. The simulated and physical results presented in this study demonstrate the effectiveness of an evolutionary based approach given the high dimensionality of the solution space.

Next, we simultaneously evolved of morphology and control. Evolutionary pressures were based on achieving maximal average velocity or maximal turning rate by adjusting control parameters and the flexibility of a caudal fin. We assessed the utility of the proposed method by evolving behaviors for both turning and achieving high velocity, producing two controllers. The algorithm successfully found solutions able to perform both behaviors. As these results indicate, an evolutionary optimization approach is particularly well suited to the nonlinearity of the aquatic domain. Specifically, this study demonstrates that evolutionary algorithms can handle the complex interactions found between material properties, physical form, and control patterns. In future chapters, we expand on evolving both morphologies (by considering multiple objectives) and controllers (by using feedback controllers that are adaptive).

Chapter 4

Evolving Swimming Performance with Mechanical Efficiency

Many studies in evolutionary robotics focus on optimizing the system for a single task. Likewise, existing work on optimizing performance of robotic fish with flexible fin or body has typically dealt with a single optimization objective, for example speed or thrust. However, in practical applications robotic fish (and other types of robots) are often required to meet multiple objectives, either simultaneously or within different tasks or environments. For example, while speed is in general an important specification, energy efficiency (and thus operating duration) is often equally as important. Maneuverability, the ability of the robot to make tight turns or deal with disturbances, is also a particularly significant objective. In this chapter, we consider energy efficiency as the ratio between the "useful" and total mechanical power exerted by the caudal fin (these terms are discussed in detail in Section 4.1).

We apply evolutionary multiobjective optimization (EMO) to the design of a robotic fish with a flexible caudal fin, shown in Figure 4.1. EMO algorithms typically use an elitism approach for driving solutions toward the optimal Pareto front, and a niching or crowding

Some of the results and descriptions in this chapter were published in [28].

mechanism to ensure that the entire set of Pareto-optimal solutions can be found [140, 37]. The advantages of EMO algorithms include: (1) locating a Pareto front with fewer evaluations when compared to a parameter sweep, (2) automatically handling constraints, and (3) automatically sorting solutions according to feasibility and domination. For this study, we apply the NSGA-II algorithm [37], which is widely used in both research and real-world applications. Compared with other EMO algorithms, the main advantages of NSGA-II include a faster sorting operation and a more effective method for maintaining diversity (i.e., reducing premature convergence) [140, 31, 139, 72, 136].

The results of simulations reveal several general principles that can be applied in the design of robotic fish morphology and control. To verify that the simulation results are physically relevant, we selected several of the evolved solutions, fabricated flexible caudal fins using a multi-material 3D printer, and attached them to a robotic fish prototype. Experimental results, conducted in a large water tank, correspond reasonably well to simulation results in both swimming performance and power efficiency, demonstrating the usefulness of evolutionary computation methods to this application domain. The contribution of this chapter is a demonstration that multiple objectives can be simultaneously optimized for a robotic fish, and that in performing such an optimization we can extract meaningful design criteria that will be applicable to similar robotic systems.

4.1 Modeling and Simulation

Finding combinations of morphological and control parameters that effectively balance speed and energy consumption is challenging. As demonstrated in Chapter 3, in robotic fish that incorporate compliant materials, morphological and control parameters are highly interrelated. Specifically, any change to the flexibility of the caudal fin will require a corresponding change to the control signal to ensure that the frequency of oscillation matches the natural motion of the fin.

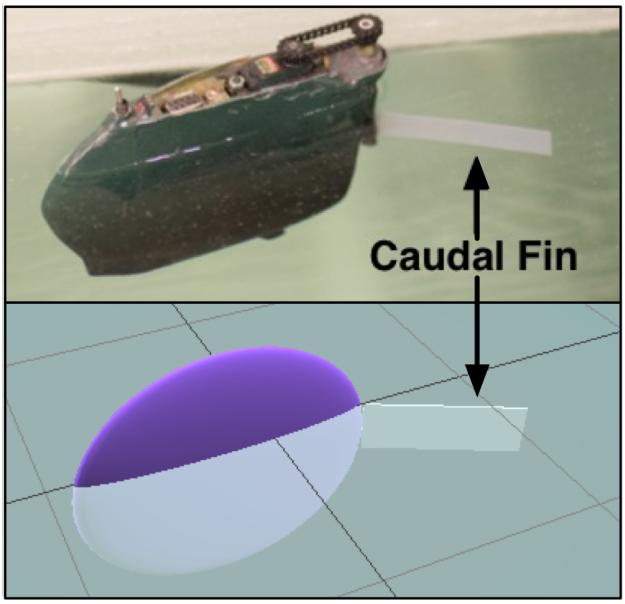


Figure 4.1: (top) The robotic fish prototype used in this study, and (bottom) the virtual representation evolved during simulation.

In this study, we again use a dynamics model developed by Wang et al. [128] based on Lighthill's Large-Amplitude Elongated Body Theory of Locomotion [80]. Unlike the work presented in Chapter 3, however, we do not utilize a rigid body dynamics engine (i.e., physical simulation) for this work. Instead, simulation of the robotic fish is conducted in Simulink [115], which allows for a straightforward translation of dynamic equations into simulation (i.e., numerical/mathematical simulation). Numerical simulation provides greater accuracy for our simulated robotic fish. Details regarding the modeled dynamics can be found in Section 3.1.

4.1.1 Evolutionary Optimization.

The numerical simulation of the robotic fish above is parameterized by several terms, including the amplitude and frequency of a sinusoidal control signal and the length, height, and flexibility (spring and damping coefficients) of the caudal fin. The ranges for each of these values, aside from the spring coefficient, are listed in Table 4.1.

Table 4.1: Range of Evolved Parameters

	Min	**Max**
Amplitude (rad)	0.08	0.5
Frequency (Hz)	0.5	3.0
Fin Length (cm)	3.0	15.0
Fin Height (cm)	1.0	5.0
Young's Modulus (GPa)	0.1	3.0

Unlike the other parameters, the range of spring coefficient values is not chosen by a designer. Instead, it is limited by the properties of available materials. For this study, the spring coefficient is restricted to what our 3D printer, an Objet Connex350, is capable of fabricating. Moreover, for physical materials it is more common to consider their Young's modulus, an inherent material property relating to elasticity and flexibility measured in Pascals (Pa). Consequently, this quantity is displayed in the table in place of a range for spring coefficients. The printed materials, and resulting Young's modulus range, are discussed in the next section (Section 4.2).

For evolving genomes comprising real values, NSGA-II requires a user to set the four following parameters (selected values are in parenthesis): the probabilities of crossover (90%) and mutation (33%), and the distribution index for both simulated binary crossover (10) and polynomial mutation (10).

4.1.2 Constraints.

Along with evolving the parameters as real-valued numbers, NSGA-II also accommodates the following two limitations on the dynamic model. First, the dynamic model is only valid for an elongated fin in which fin length is at least roughly three times fin height, which corresponds to a minimum length-height ratio of 3:1. Second, since the dynamic model itself does not limit power supplied at the base of the caudal fin, presumably by a servomotor, we

impose a maximum power constraint. Practically, this maximum power constraint limits the top speed of the robotic fish. These two constraints are given by the following equations:

$$length - 3height \geq 0 \tag{4.1}$$

$$MAX\_POWER - power \geq 0 \tag{4.2}$$

where *length*, *height*, and *power* refer to properties of the robotic fish caudal fin, and the maximum power constant MAX_POWER was determined experimentally.

In NSGA-II these limitations are treated as constraints, which enables the algorithm to smoothly follow a gradient from *infeasible* (i.e., a solution that violates a constraint) to *feasible* solutions. For the constraint equations, a negative value denotes an infeasible solution.

4.1.3 Fitness Evaluation.

Each evolved individual is evaluated for 10 seconds of simulation time, however, only the second half of this period determines fitness. This allows the robotic fish to reach a "cruising speed" and final heading, with average speed and power efficiency calculated over the final 5 seconds. Power efficiency is the ratio of two terms: *effective power* as the numerator, and *total power* as the denominator. Both of these terms are mechanically based (as opposed to the electrical power of the motor) and are calculated using Equations (3.1), (3.2), and the following definition of mechanical power:

$$\vec{P}(t) = \vec{F}\vec{v}, \tag{4.3}$$

where $\vec{P}$ is mechanical power, and $\vec{F}$ and $\vec{v}$ can be taken as the instantaneous force and velocity, respectively, of a point on the caudal fin.

Effective power, sometimes called *useful power*, is illustrated in Figure 4.2. Effective power takes the robotic fish's velocity and total force and projects it along the average direction of motion (labeled d_{5to10s}) before using Equation (4.3) to calculate an average power. Practically, effective power includes only the surge (forward-to-back) force produced by the caudal fin. Total power, on the other hand, includes the force exerted to create *both* sway (side-to-side) and surge motions. The resulting power efficiency ratio has a range from 0 to 100%, with higher numbers being desirable.

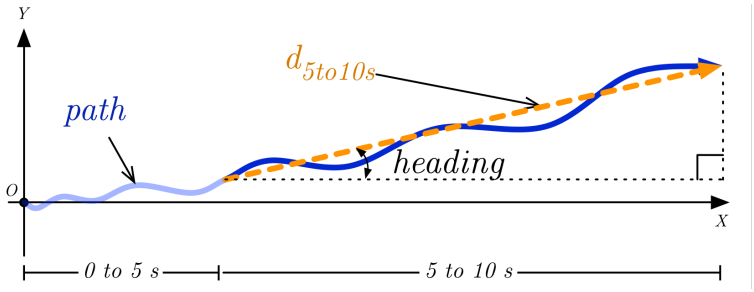


Figure 4.2: An illustration of the dynamics involved in calculating fitness. The *path* of the robotic fish includes two parts: a light-blue segment from 0 *to* 5 seconds, which does not directly affect fitness, and a dark-blue segment from 5 *to* 10 seconds used to evaluate fitness. The path settles to an average *heading*, which is not in line with the X-axis, due to a bias caused by the initial rotation of the caudal fin. The dashed, orange line (d_{5to10s}) represents displacement over the final 5 seconds of simulation.

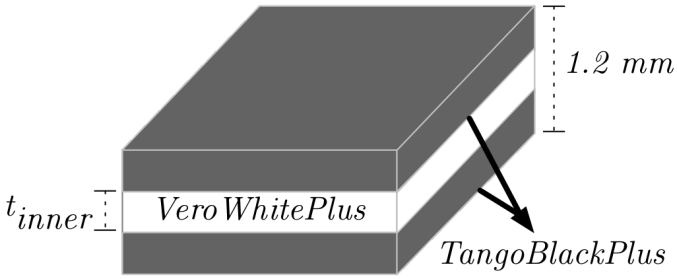
4.2 Fin Fabrication and Testing

In simulation, flexibility of the caudal fin is determined by spring coefficients. However, the flexibility for actual materials is expressed as a physical property such as the Young's modulus. Therefore, we require the following equation, which relates a spring coefficient to the Young's modulus of a 3D-printed component:

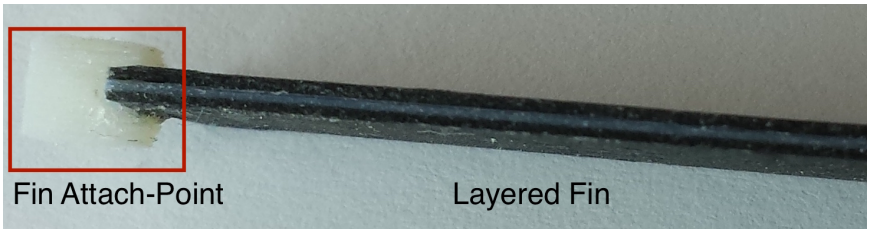
$$K_s = \frac{Edh^3}{12l}, \tag{4.4}$$

where K_s and E refer to the spring coefficient and Young's modulus values, respectively, and d, h, and l represent the height, thickness, and length of a rectangular fin, respectively.

To match the Young's modulus of printed materials with evolved spring coefficients, we needed a way to fabricate a fin for a given Young's modulus value. To do so, we designed composite fins in which flexibility is adjusted by varying the relative thickness of two different materials, as shown in Figure 4.3. The fin comprises two outer layers of a rubber-like polymer, and an inner layer of a more rigid plastic. When discussing composite materials, flexibility is often referred to as an *effective* Young's modulus to distinguish from uniformly fabricated materials. Thus, specifying the thickness of the inner layer, t_{inner}, and fixing the overall thickness to 1.2 mm limits the range of possible effective Young's modulus values.



(a)



(b)

Figure 4.3: (a) Diagram of a composite material for a 3D-printed flexible caudal fin (note: the caudal fin would be on its side). (b) Top-view photograph of a 3D-printed caudal fin with an inner thickness of 0.38 mm. The overall thickness is a constant 1.2 mm. The effective Young's modulus value for the composite material depends on the relative thickness (t_{inner}) of the inner VeroWhitePlus layer with respect to the two flexible TangoBlackPlus outer layers.

To determine the 3D-printable range of effective Young's modulus values, we set up the experiment shown in Figure 4.4 and measured the Young's modulus values for a series of composite materials with different values of t_{inner}. For this experimental setup, the Young's modulus E is evaluated with:

$$E = \frac{L_b^3 P_L}{3 I_b w_L} \tag{4.5}$$

where L_b and I_b are the length and area of moment inertia of the test composite, respectively, and P_L and w_L are the load and displacement at the tip of the composite, respectively.

As shown in Figure 4.4, a sample of composite material is fixed to a harness while its tip rests against a load cell. Displacement at the tip is adjusted using a sliding rail and measured with a laser sensor. Three replicate sets of load and displacement data are gathered for each composite material, in which each set comprises five data points at different displacements. A least square error method is adopted to find the slope between force and displacement for each of the three sets, and the effective Young's modulus for each composite is evaluated as the average of the three replicate experiments.

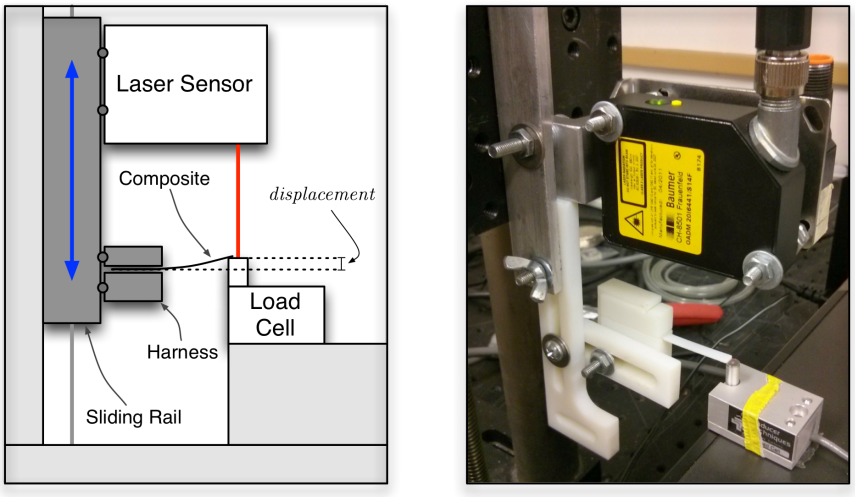


Figure 4.4: Testing of physical fins. (a) Diagram of the experiment showing the testing process. (b) Photograph of the experimental setup for measuring the effective Young's modulus of 3D-printed composite materials.

Figure 4.5 plots the results of these experiments for different values of t_{inner}. The evolvable range of spring coefficients corresponds to an effective Young's modulus of approximately 100 MPa to 3 GPa. These values correspond to a range of materials roughly from rubber, which typically has a Young's modulus of 10 to 100 MPa, to hard plastics, which

have a Young's modulus of 1 to 5 GPa. Here, we use the *best fit* line in Figure 4.5 to find the required t_{inner} value for a given Young's modulus.

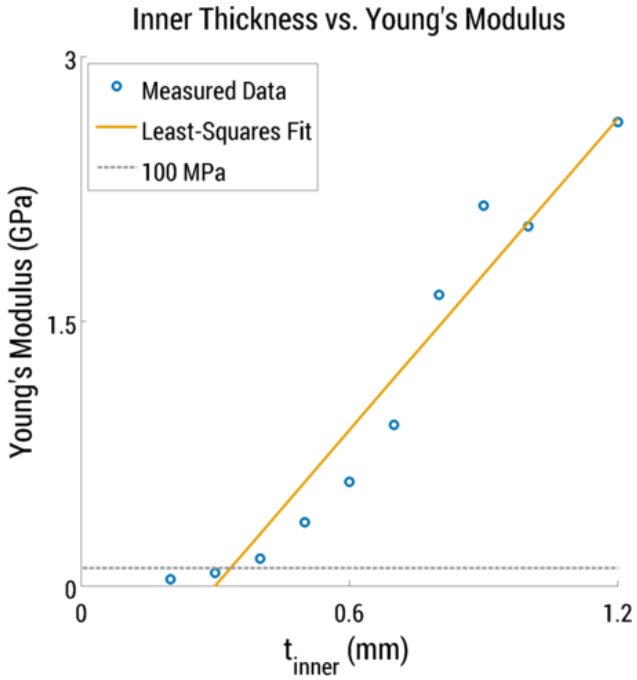


Figure 4.5: Effective Young's modulus of composite materials for different values of t_{inner}.

4.3 Experiments and Results

We conducted 20 replicate evolutionary runs, each containing 200 individuals evolving for 500 generations. All of the 20 replicates converged to a similar Pareto front. In total, each replicate simulation requires roughly 5×10^4 evaluations to converge to a final Pareto front, which is significantly less than what could be required for a parameter sweep over the same parameter ranges. For example, testing each of the 6 parameters (i.e., the 5 parameters shown in Table 4.1 and an additional spring damping coefficient) at 10 evenly distributed points would require exactly 10^6 evaluations. Furthermore, unlike an EMO algorithm, such a parameter sweep would not consider values in between the 10 fixed points, resulting in a more coarse optimization.

Figure 4.6 shows the combined Pareto front for the 20 replicate runs; each of the subsequent figures in this section will correspond to the data found in Figure 4.6. The combined Pareto front reveals that for the robotic fish prototype an evolved caudal fin and control signal can produce a maximum average speed (given the maximum power constraint) in the range of approximately 4.8 to 5.8 cm/s and an efficiency in the range of 35 to 42%. As expected, we see that an increase in speed is accompanied by a decrease in efficiency. Although the Pareto front is clearly formed, we emphasize that it covers only a small range of values (4.8 - 5.8cm/s and 35 - 42% efficiency). The Pareto front is more clearly visible in Figure 4.7, which plots every feasible solution evolved in each of the 20 replicate experiments.

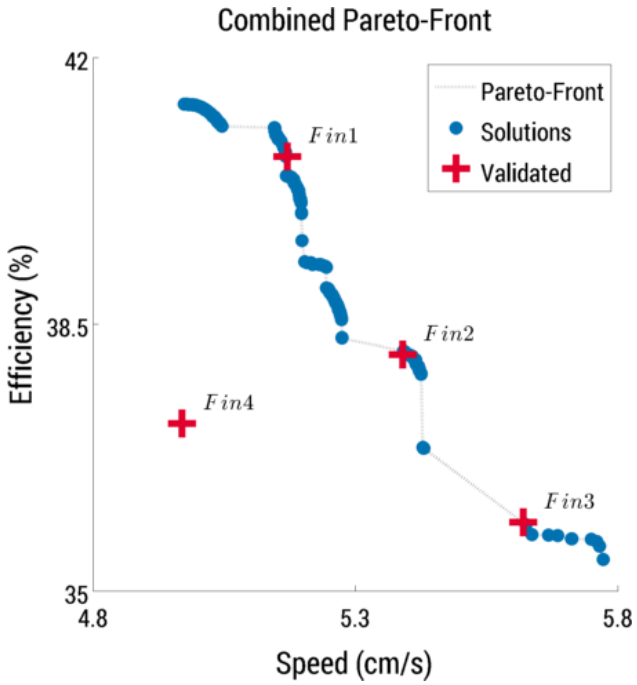


Figure 4.6: A combined Pareto front including the best solutions from each of the 20 replicate evolutionary simulations. The labeled, red "+" symbols denote solutions that were physically fabricated and validated.

Inspecting the affects of each parameter on each objective can give insight into how parameter values should be selected. For example, the two plots in Figure 4.8 show how the length of a caudal fin affects each of the two objectives. For length, all values are between 6 and 12 cm, and higher values for length within this range produce higher speed, but

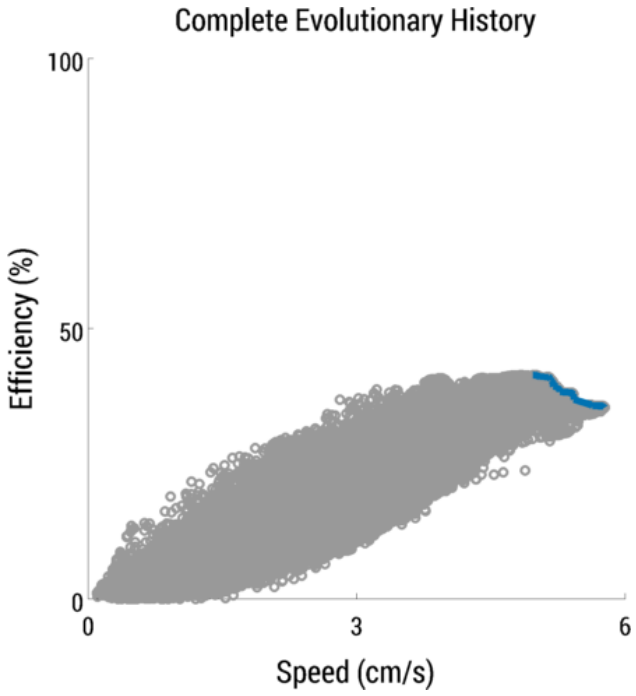


Figure 4.7: The complete evolutionary history for each of the 20 replicate experiments. Every feasible, evolved solution is plotted, and the Pareto-front is highlighted in dark blue.

decreased efficiency. A length in this range corresponds to roughly half the length of the body (14cm), which is a common ratio for biological fish. Although these values for length depend on the other parameters, intuitively increasing the length of the fin should increase thrust, and therefore speed, at the cost of increased power usage.

Figure 4.9 plots parameter values for the solutions in the combined Pareto (see Table 4.1 for the range of each parameter). In the figure, each parameter is scaled between 0 and 1 for an easier comparison. The lines between parameters connect values that belong to the same genome (evolved solution). The figure illustrates two important points. First, each parameter converges to a relatively small range of values, particularly fin flexibility (fins with a scaled value near 0 are very flexible). The evolution of relatively flexible fins is not unexpected, as they produce higher thrust for lower values of control frequency and amplitude [23]. Essentially, a more flexible fin is inherently more effective and efficient. Second, most of the variation in the final population is in the length and height of the

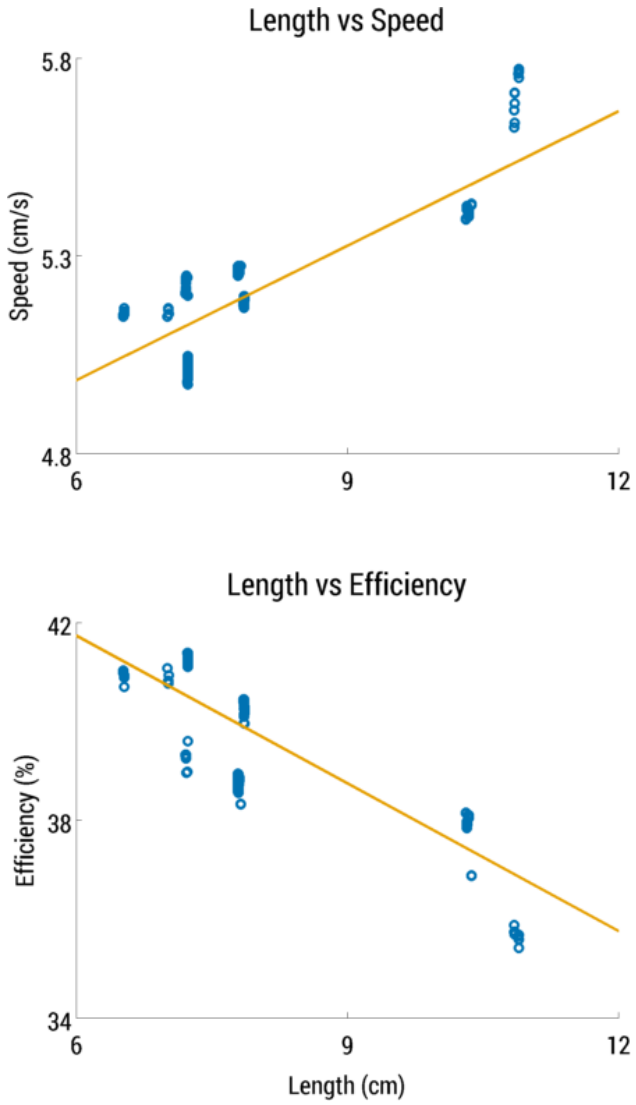


Figure 4.8: Plots of evolved solutions showing relationships between fin length (x-axis) and the two objectives, speed (top) and efficiency (bottom). All Pareto-optimal solutions have a fin length between 6 to 12 cm, thus the x-axis does not include the entire evolvable range (3 to 15 cm). The straight, orange lines indicate the best-fit for each set of values.

caudal fin, meaning that the control parameters and fin flexibility converge to values that are useful for fins of different shapes.

Inspecting the relationship between evolved solutions and constraints (Equations (4.1) and (4.2)) can provide further insight into what constitutes a *good* design. For example, the constraint on fin dimensions sets a minimum length-height ratio of 3:1, and although the height and length parameters exhibit a relatively high variation in the Pareto-optimal set,

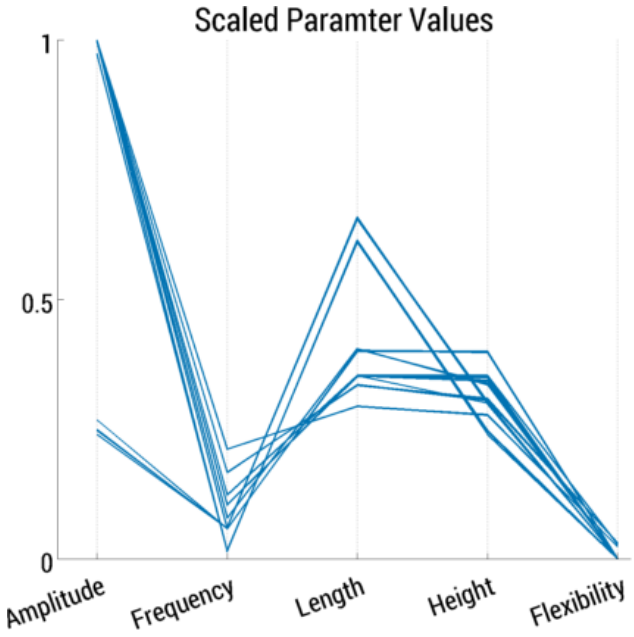


Figure 4.9: The evolved parameters scaled between 0 and 1 for the combined Pareto front solutions.

most converge to a similar ratio. More specifically, the best performing fins have a ratio less than roughly 3.5:1 (mean of 4.1 and median of 3.2), which signifies that while changing the height and length of the fin can alter the speed and efficiency, it is best to keep the ratio near 3:1 for better performance. Since simulation environment does not allow for fins with a ratio of less than 3:1, physical testing will have to be conducted to determine whether lower values are more or less beneficial. Likewise, inspecting the constraint on maximum power shows that the best solutions utilize as much power as is allowed, which is important for generating the most thrust.

As for control of the caudal fin, Figure 4.9 indicates that it is better to increase the amplitude and decrease frequency of the sinusoidal motion. This can best be explained by examining the equations of motion for the caudal fin:

$$\theta(t) = Asin(2\pi Ft), \tag{4.6}$$

$$\dot{\theta}(t) = 2\pi AFcos(2\pi Ft), \tag{4.7}$$

$$\ddot{\theta}(t) = -4\pi^2 AF^2 sin(2\pi Ft), \tag{4.8}$$

where A and F are the amplitude and frequency of the sinusoidal motion, and θ is the angle of the caudal fin with respect to the body. Equation (4.8) demonstrates that caudal fin acceleration is proportional to the square of frequency, but only linearly proportional to amplitude. As shown by Equations (3.1) and (3.2), the resulting thrust, and thus the total power, is proportional to this same acceleration. Thus, the evolutionary preference for high amplitudes and low frequencies suggests that higher frequencies require too much power, which violates the second constraint, and that it is more effective to increase speed by increasing amplitude.

Our interpretation of Figure 4.9, along with the calculated caudal fin aspect ratio, provides the following general guidelines on how to produce an effective, efficient robotic fish. Specifically, the caudal fin should: (1) be relatively flexible (have a low Young's modulus), (2) have a fin length-height ratio close to 3:1, (3) have a fin length of roughly one-half the length of the body, and (4) increase thrust and speed by increasing the amplitude of motion rather than frequency.

Since every solution on the Pareto front is *nondominated* (i.e., optimal in some respect), it is difficult to claim that one solution is better than another based on their fitness values. To compare solutions, however, we can devise a metric that captures what is deemed important. For example, we may consider how much total power is needed for the robotic fish to travel 1 meter, as in Equation 4.9:

$$P_{total} = t_{1meter} * P_{average} \tag{4.9}$$

where P_{total} is the total power accumulated, t_{1meter} is the time that it takes to travel 1 meter, and $P_{average}$ is the average power exerted by the evolved control pattern.

This metric includes aspects of both objectives: it is beneficial to decrease time spent traveling, t_{1meter}, by increasing speed, and beneficial to decrease $P_{average}$ by being efficient. With respect to Equation 4.9, we find that solutions near the middle of the Pareto front, as opposed to the two tails, perform best. The worst performing solutions are those that are faster but less efficient. While this is a simple test, a similar process can be followed to select a final evolved solution in more complex scenarios. The chosen metric need only relate to behaviors anticipated for a required task. For example, if maneuverability is important for avoiding obstacles, the metric can include turning efficiency.

4.4 Physical Validation

To verify that solutions evolved in simulation are physically meaningful, we selected four solutions (indicated by the labeled "+" symbols in Figure 4.6) from the combined Pareto front to fabricate and test. Making use of the best fit line from Figure 4.5 enables the 3D-printing of composite caudal fins with a specific evolved flexibility. For each chosen solution, a t_{inner} value is calculated and a fin of the correct flexibility, length and height is printed. To compare the speed of virtual and physical fins, we attached the printed composite fin to the robotic fish prototype shown in Figure 4.1. This robotic fish was placed in a water tank and speed was measured and averaged over 5 trials, where each trial was conducted in the same manner as fitness is calculated for the virtual robot. Results of these physical experiments are summarized in Table 4.2, where labels match those found in Figure 4.6. For three of the tested fins, these results show a good correspondence between simulation and reality. $Fin1$, however, is notably faster in reality given the same parameters as simulation, likely due to the nature of flexible fins where performance can change drastically for slight changes in flexibility.

Table 4.2: Simulation-Reality Speed Comparison

	Simulation	**Reality**
$Fin1$	5.17 (cm/s)	7.43 (cm/s)
$Fin2$	5.39 (cm/s)	4.00 (cm/s)
$Fin3$	5.62 (cm/s)	5.00 (cm/s)
$Fin4$	4.97 (cm/s)	4.90 (cm/s)

We note that, while the physical measurements roughly correspond with the simulation results, as shown in Figure 4.7, the Pareto-solutions are clustered within a relatively small range of values for speed (1 cm/s difference between minimum and maximum values). Effectively, all the fabricated fins produce good performance. As a consequence of this tight clustering, however, the trends among results for the physical experiments differ from those of the simulations. Additionally, imperfections introduced by the 3D printing process will amplify any disparity between simulation and reality.

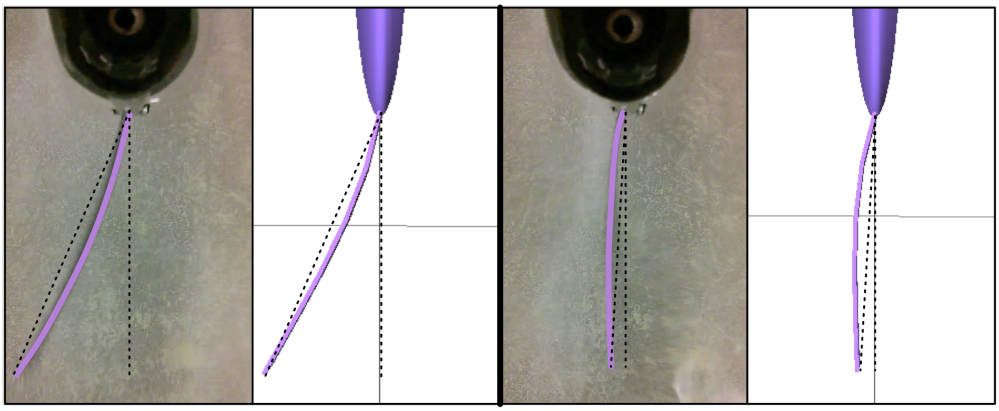


Figure 4.10: Comparison between the motion of a caudal fin for physical and virtual experiments. To increase visibility, a purple line traces the length of the caudal fin for the physical device. The dashed, black reference lines provide a common angle with which the side-by-side images can be compared.

When comparing virtual and physical results it is simple to compare their speeds, yet it is not as straightforward to validate power efficiency. However, since we are using mechanical power efficiency, the total power (denominator of power efficiency) of virtual and physical

results will be approximately equal when both are controlled with the same frequency and amplitude and the caudal fin motions match.

As supported by Figure 4.10, the dynamic model utilized in this study is accurate with respect to the motion of the caudal fin. The figure demonstrates that for similar conditions (i.e., the same time point, control pattern, and fin morphology) the virtual and physical robotic caudal fins have the same motion. Thus, it can be expected that total power between virtual and physical will be consistent with one another. To compare on the basis of power efficiency, we also require effective power (numerator of power efficiency). Since effective power is proportional to speed, the efficiency of virtual and physical trials can be compared by inspecting their average speeds. Accordingly, in Table 4.2 an increase in speed corresponds to an increase in efficiency, and vice versa.

4.5 Conclusion

In this Chapter, we presented our first study in which we applied evolutionary multiobjective optimization to the problem of balancing swimming performance and power efficiency in a robotic fish. Swimming performance is considered as maximizing the average speed, and power efficiency is calculated as the ratio between effective power and total power resulting from caudal fin actuation.

Results from NSGA-II evolutionary experiments provide insight into how a robotic fish should be designed for increased performance and efficiency. First, the control parameters (amplitude and frequency) should be set such that frequency is on the lower end of acceptable values and amplitude is on the higher end of its range, while still providing enough thrust to reach the desired speed. Setting these parameters in such a way will lead to increased efficiency compared to using higher values for frequency with lower amplitudes. Second, the length-height ratio of the caudal fin should be approximately 3:1, and the length of the fin should be roughly half of the length of the body. Finally, caudal fins should be have

a flexibility similar to rubber-like materials, as opposed to hard plastics. The conclusions drawn from these results are expected to generalize to any robotic fish of similar design, regardless of scale.

To verify that results from evolutionary simulation are physically meaningful, we fabricated several evolved fins and attached them to a robotic fish prototype. We utilized a multi-material 3D printer to fabricate fins made of composite materials, and based the evolutionary range of flexibilities on the capabilities of this printer. Experiments conducted in a water tank confirm that the evolved speeds correspond reasonably well with physical results. Further, through a visual comparison, we conclude that the total power of evolved solutions closely match reality, and thus, the solutions generated by NSGA-II constitute valid robotic fish designs. However, due to the clustering of evolved solutions, and the issues caused by the reality gap, trends among evolved solutions were lost when transfered to the physical device. In Chapters 6 and 7, we discuss our method of addressing this concern by combining evolutionary computation with adaptive control. In the next chapter, we continue this research with a more complex system and a different mechanism for calculating efficiency.

Chapter 5

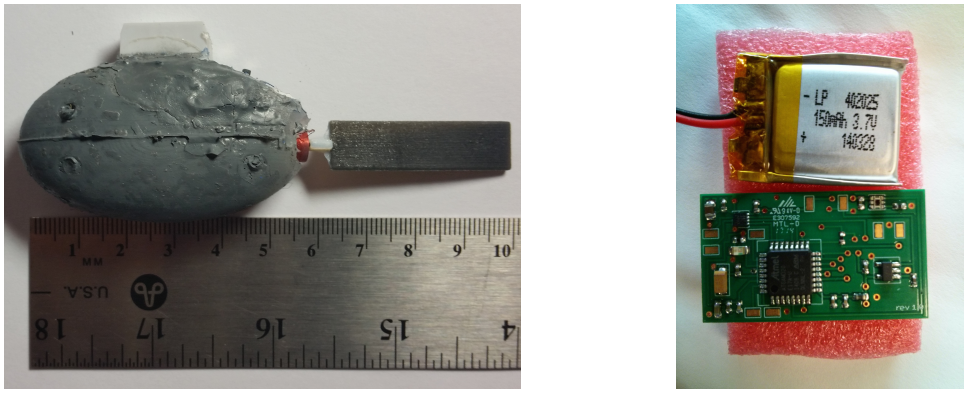
Evolving Swimming Performance with Electrical Efficiency

In this chapter we discuss our second study involving our evolutionary multiobjective optimization approach to the design and control of flexible fins for robotic fish. For the purpose of demonstration, we again focus on the caudal fin and two competing objectives: speed and power consumption. In particular, we explore the interactions between the stiffness, size, and the control pattern for the flexible fin, and we investigate how the speed performance and energy usage can be balanced. Some situations may require speed be sacrificed for efficiency (e.g., the system may need to return to port due to a low battery), while in other situations speed may be paramount (e.g., the system may need to escape a hazard).

The study presented in this chapter expands on the work described in Chapters 3 and 4. In Chapter 3, we explored how a conventional genetic algorithm could be applied to optimize morphological characteristics, including caudal fin flexibility, and control patterns for a robotic fish prototype. However, that system had limited capabilities (i.e., no sensory feedback, no communication abilities, and a low-power micro-controller), and optimization did

Some of the results and descriptions in this chapter were published in [27].

not address multiple objectives. In Chapter 4, we described an initial approach to applying EMO methods to robotic fish. However, that investigation used a platform with limited sensing capabilities. The electromagnetically driven robotic fish in this chapter, shown in Figure 5.1, is smaller, has more computational power and sensing capabilities, and enables a direct calculation of energy usage, as discussed in Section 5.1. We have also refined and enhanced the simulation modeling for flexible fin dynamics, and we have conducted more physical validation trials with the new robotic fish prototype. Finally, whereas in Chapter 4 we focused on useful mechanical power, here we are able to optimize for both the robot's speed and the average **electrical** power expended. As a result, the analysis of Pareto-optimal solutions yields results not apparent in the earlier studies.



(a) (b)

Figure 5.1: (a) Small robotic fish cast from liquid rubber; the body has been painted gray and the printed caudal fins are detachable. (b) The device's custom PCB and rechargeable battery.

5.1 Design of a Small Robotic Fish

In this section we provide details of the target robotic fish, the simulation model, and the evolutionary algorithm.

5.1.1 Robotic Fish.

Figure 5.1 shows the robotic fish used to test and validate the methods proposed in this chapter. The caudal fin is detachable, enabling us to test many fin designs, and the robotic fish is intended to operate either autonomously or via remote control for up to three hours under normal conditions (i.e., non-continuous communication, average actuator usage). The device is powered by a 150 mAh lithium-ion polymer battery, which provides over two hours of continuous operation under maximum load (including wireless communication, sensing, and actuation).

5.1.2 Body and Fin Fabrication.

The body of the robotic fish is designed to be as small as possible, while incorporating all components necessary for untethered operation. The body is cast from liquid rubber (Smooth-On Ecoflex®00-30), which results in a soft, "stretchy" form. All electrical and mechanical components are placed in a 3D-printed mold and the liquid rubber is poured around them. The mold (see Figure 5.2) was produced with an Objet350 Connex printer. A photograph of several 3D-printed caudal fins can be seen in Figure 5.3. Details regarding fin design and fabrication process are the same as in Chapter 4.

Figure 5.2: (a) A Solidworks model of the robotic fish mold. (b,c) Two images of the 3D-printed, clear plastic mold used during the casting process.

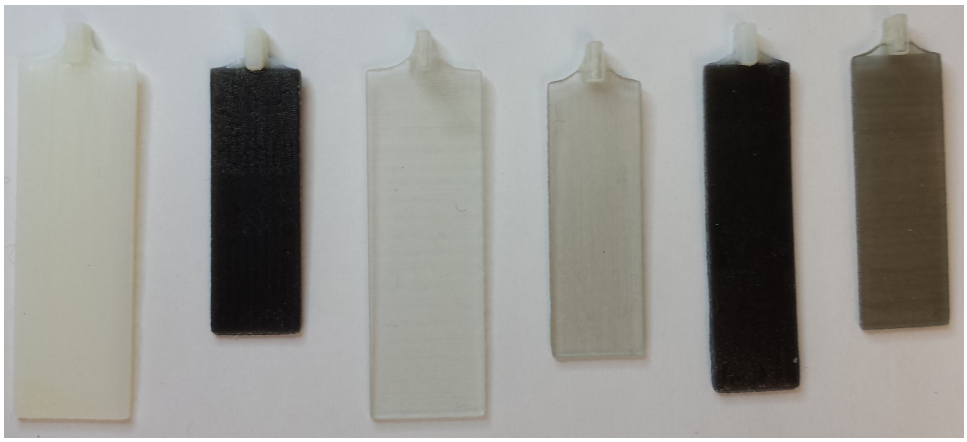


Figure 5.3: A photograph of several 3D-printed caudal fins. Each fin has different morphological characteristics: length, height, and flexibility.

5.1.3 Custom PCB.

To control the robotic fish, we designed a custom printed circuit board (PCB), pictured in Figure 5.1(b). The PCB includes a 32-bit ARM microcontroller (Atmel SAM D20), a 6-axis inertial measurement unit (IMU) (InvenSense MPU-6050), two light sensors (Intersil ISL29101), and wireless communication (Nordic Semiconductor nRF24L01+). The microcontroller is capable of executing complex adaptive control algorithms while filtering sensory data. The IMU enables the device to measure its linear and angular accelerations, which can be filtered to provide estimates of velocity. Wireless communication allows the device to be controlled and have its software updated remotely, as well as deliver sensed information to a base computer.

5.1.4 Electromagnetic Actuator.

The electromagnetic actuator is depicted in Figure 5.4. The actuator comprises a coil of magnet wire (9.5 mm outside diameter), a neodymium permanent magnet, an external centering magnet, and an attachment point for a caudal fin. To operate the actuator, a voltage is applied across the coil's terminals. The coil creates an electromagnetic field that exerts a torque on the permanent magnet, causing its poles to align with the magnetic field.

The actuator used in this study has a fixed maximum amplitude of 38° and a centering magnet that causes the actuator to return to its center when no voltage is applied to the coil. Equation (5.1) is used to calculate the angular acceleration of the permanent magnet resulting from an applied voltage:

$$\alpha = \frac{V\mu_0 NlR}{I_{effective}}, \tag{5.1}$$

where α is the angular acceleration, V is the voltage applied across the coil, μ_0 is the magnetic constant (or the permeability of free space), N, l, and R are the number of turns, length, and resistance of the coil, and $I_{effective}$ is the moment of inertia for the permanent magnet with an attached fin. This device exhibits a torque of roughly 500 μN·m with a volume of 35 mm^3, which yields a torque-per-volume of 15 μN per mm^3. In contrast, a typical commodity micro servomotor provides a torque of approximately 75 mN·m with a volume of 2200 mm^3 resulting in 35 μN per mm^3 torque-per-volume. For our purposes, alternatives to the electromagnetic actuator are either too large (commodity servomotors), require higher voltages (piezoelectric motors), or generate less torque (electropolymers and shape memory alloys) [4]. Likewise, systems that need external magnetic fields, such as the microrobots developed for medical applications [99], are too limiting in terms of the types of environments in which a robot could be deployed.

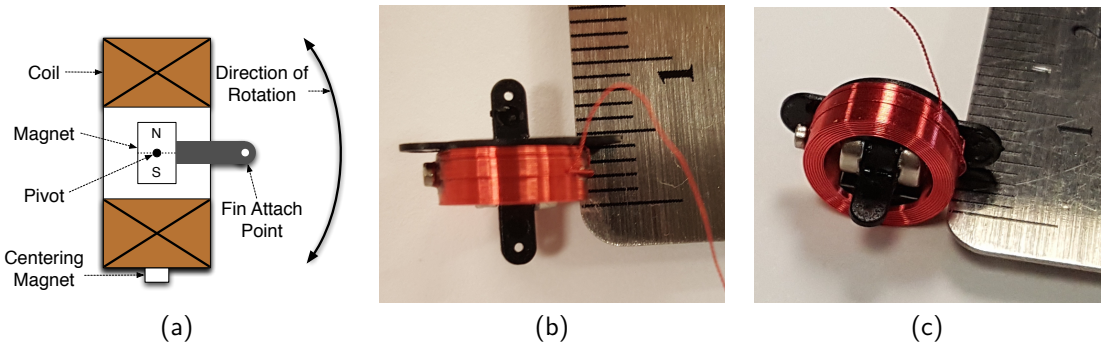


Figure 5.4: (a) Top-view diagram of the actuator, and (b,c) photographs of the electromagnetic actuator.

Electromagnetic actuators have been used as motors for robotic fish by a few other research groups. To support their fuel cell studies, Takada et al. [121] used a similar actuator for a robotic fish; this device was 10 cm in length, had diving capabilities, but did not include any sensing. Shin et al. [113] used a comparable electromagnetic mechanism for robotic tadpoles, which were less than 3 cm in length but did not include any sensing or complex control capabilities.

5.1.5 Control Signal.

The actuator is controlled by supplying a positive, negative, or zero voltage to the coil. For this study, we do not consider voltages other than 3.3 or -3.3 volts. This leaves two control parameters for forward thrust: an oscillating frequency and a pulse-width-ratio (PWR). Similar to a duty-cycle, PWR defines the fraction of time the control signal is active during a given period. We note, however, that PWR has negative components, as depicted in Figure 5.5. For the control signal example shown, PWR is set to 0.4, which results in an applied voltage that is active only 40 percent of the period. With this setup, adjusting PWR is the only way to alter energy consumption. For example, a PWR of 0.8 will result in twice the amount of energy consumed when compared to the example signal.

PWR can range from 0 to 1 and the resulting control signal will be a constant zero or a square wave, respectively. How to choose these two parameters (frequency and PWR) depends on the dimensions of the caudal fin and the desired balance between speed and energy consumption. For instance, it is rarely useful to set the PWR to 1, as doing so will result in wasted energy. Specifically, the actuator will consume energy while actively "pinning" the fin to one side, which does not generate any additional thrust. Moreover, the time required for the fin to reach its maximum amplitude depends on caudal fin dimensions. Since the actuator will always generate the same torque, using a larger fin will result in lower angular acceleration compared to fins with less surface area because the fin is "pushing" on

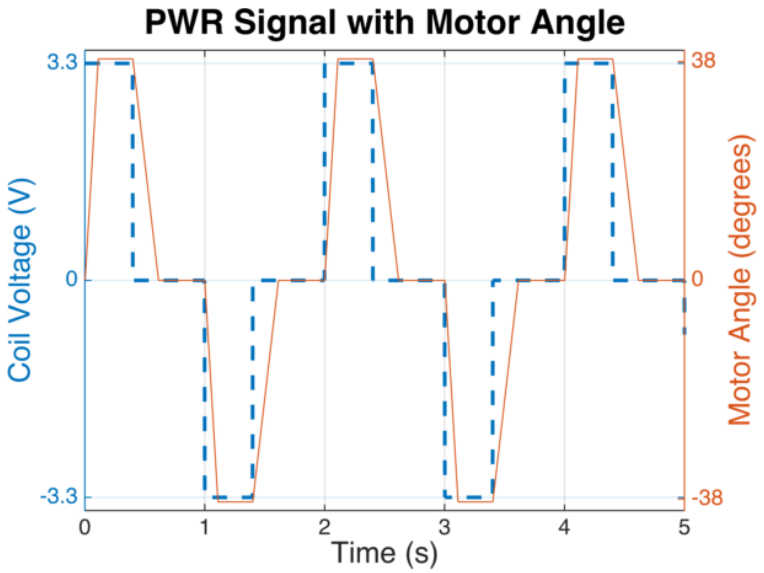


Figure 5.5: An example of the control voltage signal with the resulting simulated motor angle. The dashed, blue line depicts the control signal with a frequency of 0.5 Hz and a PWR of 0.4 (PWR indicates the fraction of time spent at either 3.3 or -3.3 V). The solid, orange line is the angle of the motor, which has a rise time related to the applied torque, and a fall time relating to the centering torque, due to the centering permanent magnet.

a larger volume of water. While conducting initial tests of the robotic fish, we found that the actuator was effective only for fins with a surface area less than 5 cm^2.

5.1.6 Evolutionary Optimization.

As done for the study presented in Chapter 4, we are again using the Simulink environment to perform numerical simulations. The dynamic simulation model takes several parameters, including the frequency and PWR of the control signal and the length, height, and flexibility (which dictates the spring and damping coefficients for fin segments) of the caudal fin. Evolved genomes comprise values for these parameters, and the allowable range for each is listed in Table 5.1. Values in the table were determined by testing the physical limitations of the physical robotic fish. For instance, if the frequency is near or above 5 Hz, the caudal fin will not have a sufficient amount of time to rotate before the actuator reverses direction. In effect, the caudal fin simply vibrates, producing very little thrust. The

limits on spring coefficients allow for caudal fins to behave similarly to materials as flexible as rubber or as stiff as hard plastic.

Table 5.1: Range of Evolved Parameters.

	Min	**Max**
Frequency (Hz)	0.1	5.0
Pulse-Width-Ratio	0.1	1.0
Fin Length (cm)	1.0	4.0
Fin Height (cm)	0.3	1.3
Spring Constant (Nm/rad)	5e-5	1e-1

5.1.7 NSGA-II Configuration.

As mentioned previously, the NSGA-II algorithm [37] was chosen to conduct the evolutionary multiobjective design. NSGA-II efficiently sorts a combined population of parent and children (created using genetic operators) into different ranks of nondominated Pareto fronts. Selection proceeds by accepting individuals from each rank in sequence until N individuals are selected in total. Crowding distance is used as a tie-breaker when an entire rank cannot be selected (because it would require accepting more than N individuals). Crowding distance is calculated as the Euclidean distance between the fitness vectors of two individuals. This method ensures that the elite (best) individuals are retained because all individuals in the first rank are Pareto-optimal. For evolving genomes comprising real values, NSGA-II requires the user to set the following four parameters (values chosen for our experiments are in parentheses): the probabilities of crossover (90%) and mutation (20%), and the distribution index for both simulated binary crossover (20) and polynomial mutation (20). Starting from values recommended by Deb et al. [37], these values were determined experimentally and cause the populations to converge in relatively few generations (i.e., within 100 generations, which corresponds to approximately 6400 fitness evaluations).

5.1.8 Fin Constraint.

Along with evolving the above parameters as real-valued numbers, NSGA-II also accommodates constraints. In our study, the dynamic model is only valid for an elongated fin in which fin length is at least three times the fin height:

$$length - 3\,height \geq 0, \tag{5.2}$$

where *length* and *height* refer to the evolved dimensions of the robotic fish caudal fin. In NSGA-II this limitation is configured as a constraint, which enables the algorithm to smoothly follow a gradient from *infeasible* (i.e., a solution that violates a constraint) to *feasible* solutions.

5.1.9 Fitness Evaluation.

Each individual in the population is evaluated for 10 seconds of simulation time; however, only the second half of this period determines fitness. This setup allows the robotic fish to reach a cruising speed and final heading, with average speed calculated over the final 5 seconds. In our preliminary study [28] (Chapter 4), efficiency was defined as the ratio between useful and total power, where useful power was calculated using the product of the total propulsive force projected onto the trajectory of travel and the travel speed, and the total power was calculated using the sum of all mechanical power exerted by the caudal fin. These calculations require instantaneous power to be calculated at every simulation time step. In contrast, since PWR of the control signal is known, in the current study average electrical power is directly calculated using the following equation:

$$P_{avg} = P_{MAX} * PWR, \tag{5.3}$$

where P_{avg} and P_{MAX} refer to the average and maximum instantaneous electrical power delivered to the motor, respectively. Since P_{MAX} and PWR are known values, a control signal's energy consumption can be calculated directly.

5.2 EMO Simulation Results

We conducted 25 replicate EMO simulations trials, each with populations of 64 individuals evolving for 100 generations. Simulation parameters for the robotic fish dynamics, other than those related to the flexible fin and the control signal, are based on the robotic fish prototype (see Section 5.1.1) to be used for physical validation. Most replicates (23 of 25) converged to nearly identical Pareto fronts. Figure 5.6 displays the evolutionary history of one replicate simulation. The population is randomly initialized (Figure 5.6(a)), and then evolves toward the final, optimal Pareto front (Figure 5.6(f)).

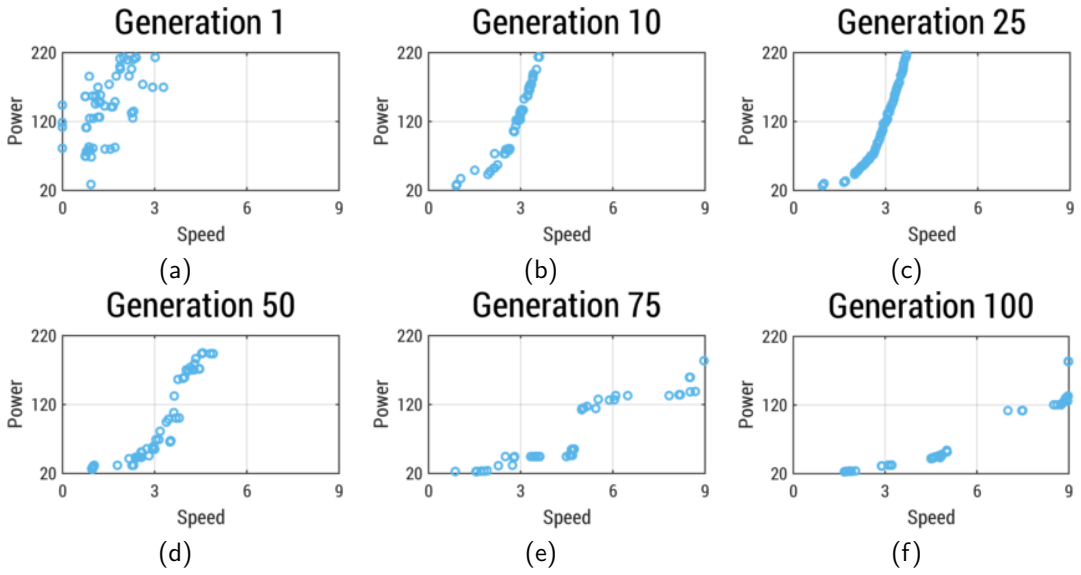


Figure 5.6: The evolutionary history of a single replicate simulation run. Every individual (including dominated individuals) in the population at a given generation is plotted in each figure. For all figures, the units for speed and power are cm/s and mW, respectively.

Figure 5.7(a) plots the average power and speed of every evolved (feasible) individual from all replicate simulations, with the Pareto-optimal individuals appearing in colors other than light gray. Jitter appears in the final Pareto fronts (some non-gray points appear to

be dominated) because replicate simulations do not find identical sets of Pareto-optimal solutions. The results are grouped into three clusters. First is the *Energy Cluster*, which includes individuals that on average consume low amounts of power. These individuals appear in the lower, left-hand section of the figure, and have speeds less than 5 cm/s and consume less than 60 mW of power on average. Second, the *Speed Cluster* includes individuals that are in the middle, right-hand section of the figure, and which consume roughly twice the amount of power but also swim twice as fast. The *Local-Optimum Cluster* is the set of individuals that have converged to a local optimum and appear in the upper, center section of the plot. The replicate run depicted in Figure 5.6 appears to have encountered the same local optimum (shown in plot 5.6(c)) but was able to find an evolutionary path toward the final Pareto front.

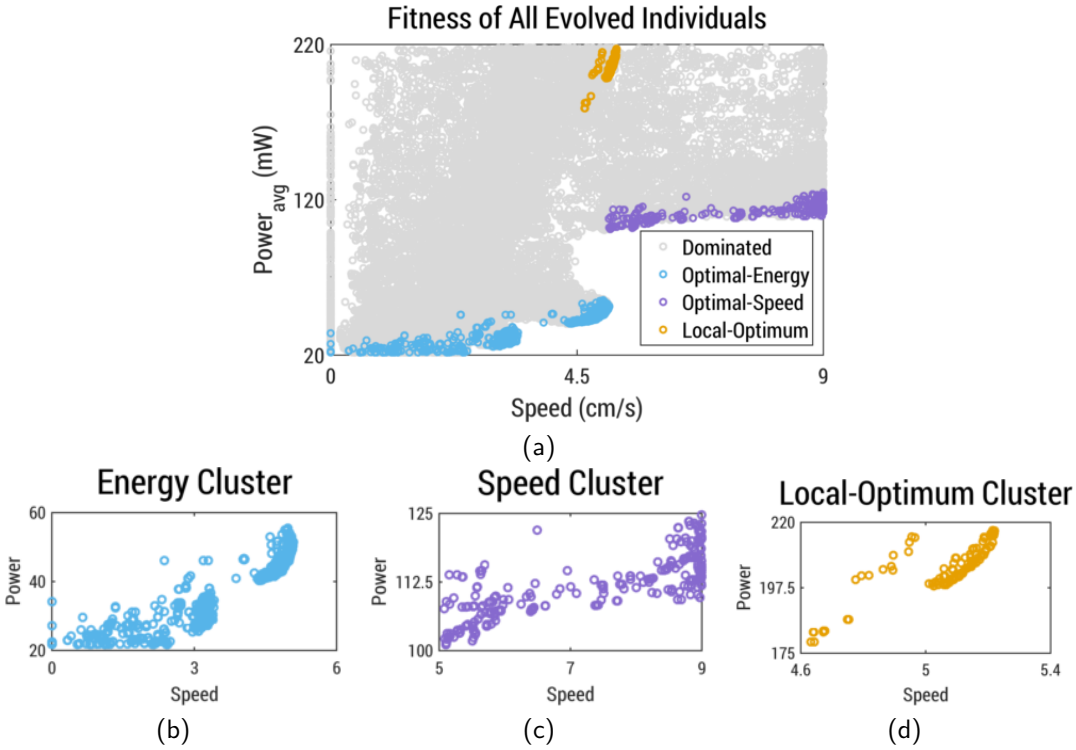


Figure 5.7: (a) Every evolved individual among all 25 replicate simulations (excluding infeasible solutions). Gray markers denote "dominated" individuals and non-gray markers denote individuals that were Pareto-optimal for a given replicate experiment. Three clusters have been identified and given different colors. The remaining plots display individuals belonging to the energy cluster (b), speed cluster (c), and local-optimum cluster (d). Please note that the cluster plots do not share the same ranges for their axes.

These three clusters are highlighted in Figure 5.7(b-d). The closer views show that within each cluster a smaller Pareto front is formed (maximizing speed and minimizing power usage). Inspecting how each of the parameters evolved gives insight into what created these distinct clusters. For example, Figure 5.8 shows that caudal fin morphology (length, height, and flexibility) remains fairly consistent for all Pareto-optimal individuals even across clusters; the data shown in this figure is scaled between 0 and 1 so that each parameter can be plotted on the same axis.

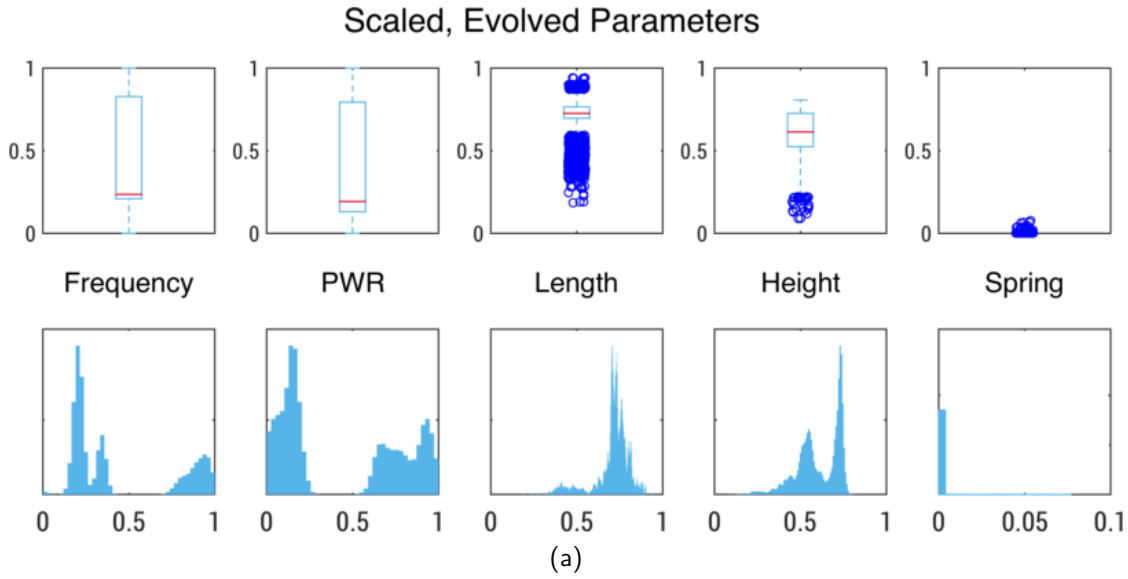


Figure 5.8: Box-plots (top row) and histograms (bottom row) of the distributions of each evolved parameter, scaled between 0 and 1, for all of the Pareto-optimal individuals. For each box-plot, the central red line indicates the median, the light blue box outlines the 25th through 75th percentiles, the dashed light blue vertical lines denote the box whiskers (non-outliers), and the dark blue circles denote outliers. In each histogram, the horizontal axes are scaled parameter values and the height of each bin indicates the density around a given value.

Convergence of morphology parameters indicates that for this particular robotic fish an ideal caudal fin is approximately 3.4 cm in length, 1 cm in height, and has the minimum allowed spring constant (i.e., a flexibility resembling a rubber material). Since the fins differ only slightly among solutions, we can infer that the control parameters account for most of the diversity in the final populations. A pairwise comparison of the parameters' variances using the Brown-Forsythe equality of variances test (and the Bonferroni multiplicity

correction resulting in a P value of 0.005) shows that the variances in the control parameters (frequency and PWR) are significantly higher than those of the morphological parameters (length, height, and flexibility). Referring again to Figure 5.7(b-d), we conclude that three distinct control strategies have been evolved, two of which are practical. Specifically, the energy-based control strategy, which has a maximum speed of roughly 5 cm/s, and the speed-based strategy, which has a minimum average power exerted of approximately 100 mW. This result suggests that, at least for the robot used in this study, morphology can be fixed and the trade-offs between the two objectives can be adjusted online by choosing a different set of parameters from the Pareto front.

Figure 5.9 compares parameters among the three different control strategies. The PWR of evolved control patterns is lower for lower power-consuming individuals (median value of 0.2), and higher for faster individuals (median value of 0.5). This result is expected, as the only way to minimize energy usage is to reduce the amount of *active* time for the actuator. Active time refers to the duration of time that a voltage is applied to the electromagnetic coil of the robot's actuator. Additionally, faster swimming individuals exhibit higher frequencies (median value of 4.0) than low power-consuming individuals (median value of 1.5).

The frequencies for lower power-consuming individuals suggest that they are sacrificing speed by evolving lower frequencies (i.e., since higher frequencies generally lead to higher speeds but increasing frequency does not affect power consumption). However, as will be discussed later, increasing frequency of these individuals actually reduces speed. As for the local optimum cluster, it appears that NSGA-II located a region in the search space where speed was maximized by increasing the PWR and sacrificing power consumption. Likely due to its exploitative nature, the algorithm was unable to escape this region of the search space in some replicates. That is, NSGA-II expands its current Pareto front by searching the neighborhood around current Pareto-optimal individuals. Therefore, it seems that it is difficult to find an evolutionary trajectory from this local optimum to the actual optimal Pareto-front.

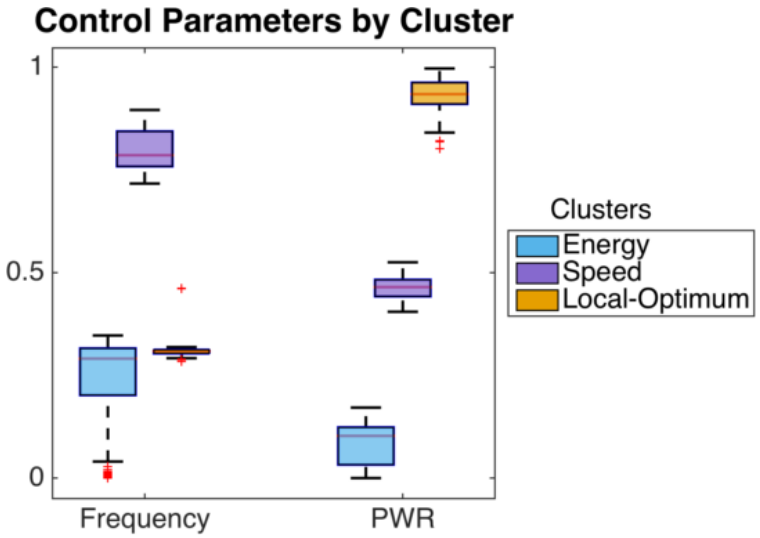


Figure 5.9: A comparison of control parameters among the three Pareto-optimal clusters. Frequency (right) and the PWR (left) are both increased to achieve faster swimming speeds and decreased to attain lower power consumption. For each box-plot, a central red line indicates the median, a box outlines the 25th through 75th percentiles, a vertical, dashed line denote the box whiskers (non-outliers), and red markers denote outliers.

.

Additionally, Pareto-optimal solutions tend to have either high frequency and high PWR or relatively low values for both parameters. This trend occurs because a high frequency with a low PWR will result in very little fin motion, due to the duration of time that the actuator would remain active. For example, with a frequency of 5 Hz (the maximum allowed value) and a PWR of 0.4, the actuator will remain active for only 40 ms at a time, which is not enough time to rotate the caudal fin for generating thrust. For comparison, the median value for activation duration for all Pareto-optimal solutions (excluding the local optima) is 64 ms with a standard deviation of 10 ms, and the minimum and maximum possible values are 10 and 1000 ms, respectively. Thus, although most solutions require the actuator to be active for relatively short durations, there appears to be a lower limit of approximately 50 ms.

To explore the generality of evolved solutions, we selected three evolved individuals and a solution with hand-chosen parameters to examine further. The values for these parameters are listed in Table 5.2. These parameter sets include morphologies near the optimal values

listed previously, and exhibit a range of control parameter values. The *Best Speed* and *Best Energy* parameters refer to evolved individuals near the end points of the Pareto front (the highest speed and lowest power consumption), the *Dominated* parameters refer to a randomly selected individual from an early generation that is not near the Pareto front, and *Hand Chosen* represents parameter values that we have chosen using expert knowledge.

Table 5.2: Parameter Sets Selected for Further Investigation.

	Frequency	**PWR**	**Length**	**Height**	**Spring Constant**
Best Speed	3.8 Hz	0.50	3.8 cm	1.0 cm	50 uNm/rad
Best Energy	1.1 Hz	0.12	3.0 cm	1.0 cm	94 uNm/rad
Dominated	3.0 Hz	0.64	3.9 cm	1.3 cm	48 uNm/rad
Hand Chosen	1.0 Hz	1.00	3.3 cm	1.1 cm	100 uNm/rad

For each fin design shown in Table 5.2, we first conducted a parameter sweep over the control parameters. The purpose of these control parameter sweeps is to evaluate individuals under a range of operating conditions, because for many practical applications it will be useful to dynamically adjust the relative importance of the two objectives. In such a situation, the morphology is fixed and only the control patterns can be adjusted. For example, if a robotic fish is able to charge its batteries via a solar cell, it will be more important to conserve energy when operating in limited sunlight. However, under ideal conditions the robotic fish may be able to sacrifice power consumption for better performance (e.g., swimming speed). Furthermore, even under ideal conditions there may be valid reasons to swim at speeds lower than the maximum value (e.g., if the robotic fish is tracking another object). Thus, during these sweeps all parameters relating to fin morphology were fixed. Results from two experiments (*Best Speed* and *Best Energy*) are plotted in Figure 5.10. Plots for *Dominated* and *Hand Chosen* are not shown as they display characteristics similar to those demonstrated by *Best Energy*. We have not provided data for the parameter sweeps against average power, as average power is directly proportional to PWR.

Considering the parameter sweep plots, a desirable trait is the ability to adjust speed to specific values. The only set of parameters to demonstrate this trait is the *Best Speed*

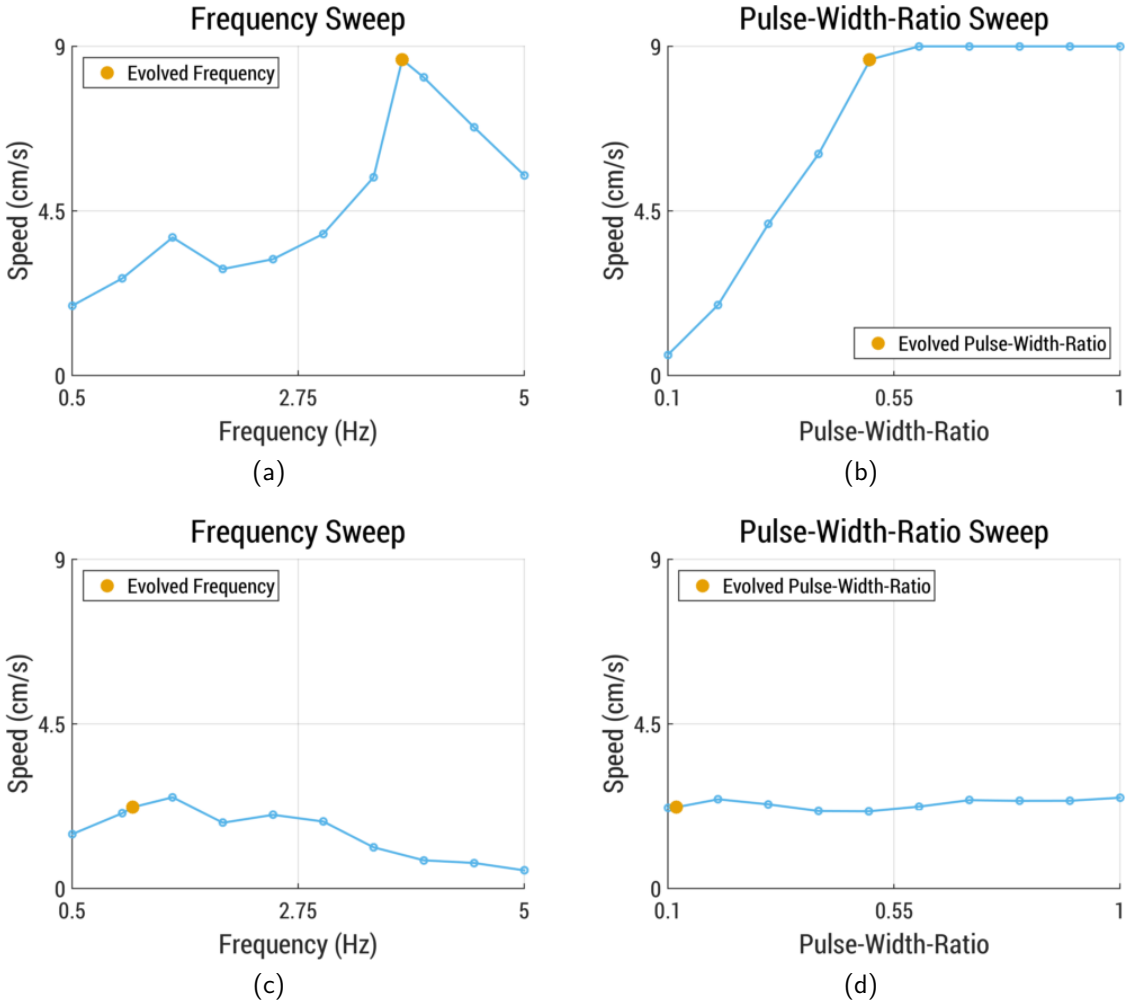


Figure 5.10: Control parameter sweeps for two sets of parameters: evolved individuals with high speed (a,b) and with low energy consumption (c,d).

individual, as shown in Figure 5.10(a,b). This configuration allows the robotic fish to swim at speeds ranging from 1 to 9 cm/s by adjusting the frequency or PWR. Additionally, as depicted in Figure 5.10(b), the robotic fish is can swim at a range of speeds by adjusting the PWR value, which means that the robotic fish can effectively adjust its average power consumption from 20 to 220 mW and its average speed from 0.2 cm/s to 9 cm/s. Moreover, the apparent linear relationship between PWR and speed allows us to consider the trade-offs between power consumption and swimming speed without worrying about drastic performance changes that can occur under certain conditions. Specifically, we do not need to avoid certain values for PWR due to a nonlinear relationship over a smaller interval.

However, any values over 120 mW of average delivered power (corresponding to a PWR of 0.5) appears to waste energy, as average power increases but speed does not. The remaining three sets of parameters (*Best Energy*, *Dominated*, and *Hand Chosen*) do not exhibit the same ability to adjust speed. However, different speeds may be achieved by these individuals by adjusting both frequency and PWR simultaneously.

We also conducted similar sweeps over the morphological parameters: fin length, height, and flexibility. In these sweeps, the control parameters were fixed. Results are again plotted for the *Best Speed* and *Best Energy* parameter values; this data can be found in Figure 5.11. For the best speed individual (Figure 5.11(a-c)) it is apparent that the morphology is optimized for speed. Specifically, any change in morphology results in a speed reduction. However, speeds attained by parameter values that lead to lower energy consumption (Figure 5.11(d-f)) are less affected by the morphological parameters. That is, changing the morphology results in smaller changes to speed.

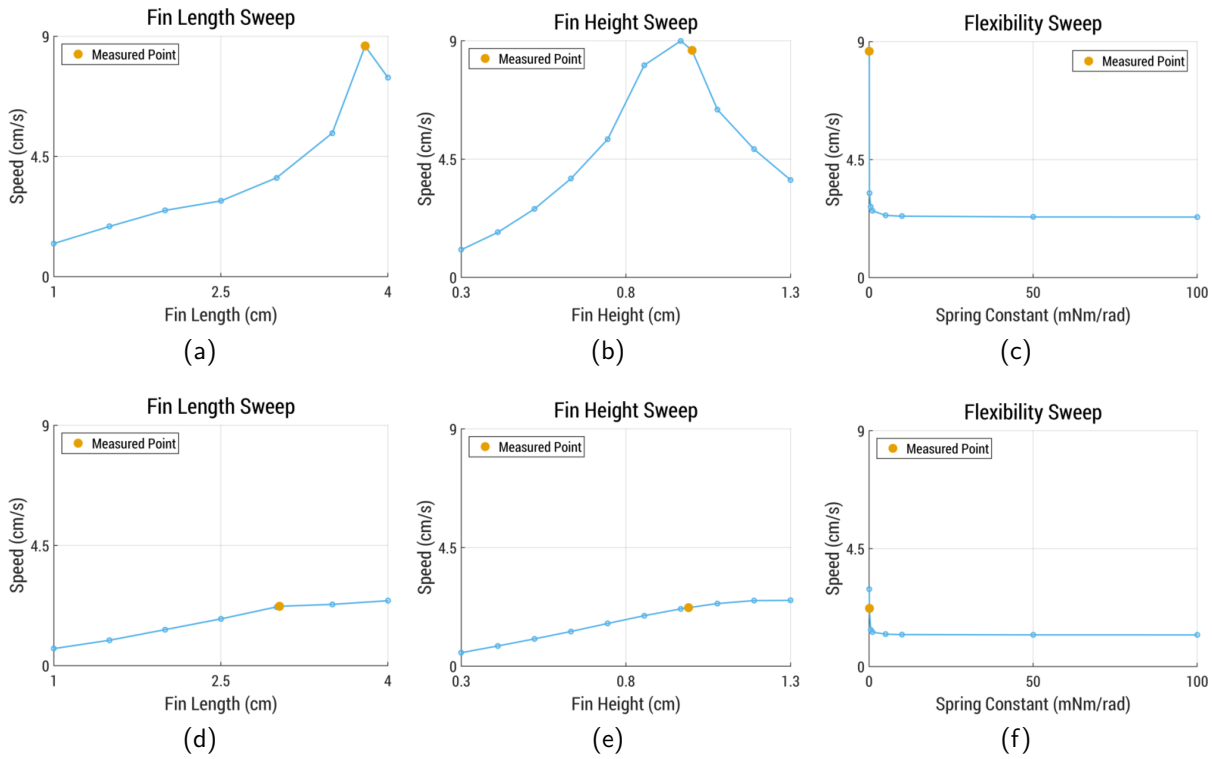


Figure 5.11: Morphological parameter sweeps for two sets of parameters: (a,b) an evolved individual with high speed, and (c,d) an evolved individual with low energy consumption.

5.3 Physical Validation

One goal of this study is to evolve solutions that can be applied to a physical robot. To confirm what we see in simulation carries into reality we selected three evolved designs and one hand-chosen parameter set to validate experimentally. The process used to design and fabricate caudal fins is that same as that discussed in Section 4.2. Specifically, we designed a composite material that allows us to 3D print caudal fins with specified characteristics (i.e., dimensions and flexibility).

The physical experiments use the same control and morphological parameters as their simulated counterparts (these values are listed in Table 5.2). Making use of the composite fins, described previously, enables 3D printing of caudal fins with a specified morphology (fin dimensions and flexibility). The fabricated fins are attached to the small robotic fish pictured in Figure 5.1. The robotic fish is then placed in a large water tank, and speed is measured using a method similar to that described in the methods section regarding the evolutionary fitness evaluation. Specifically, the robotic fish speed is measured after it reaches a steady maximum speed. For physical experiments, each reported speed value is the result of averaging data from five trials. Results from these experiments can be found in the Table 5.3.

Table 5.3: Comparison Between Speeds Acquired in Simulation and Attained by the Robot in a Water Tank.

	Simulation Speed	**Measured Speed**	**Absolute Error**
Best Speed	8.6 cm/s	3.8 cm/s	4.8 cm/s
Best Energy	2.2 cm/s	1.9 cm/s	0.3 cm/s
Dominated	3.2 cm/s	3.5 cm/s	0.2 cm/s
Hand Chosen	2.7 cm/s	2.9 cm/s	0.2 cm/s

For three of the four validated parameter sets, the results show a close relationship between simulation and reality. And for all four sets of parameters, the achieved speeds demonstrate a consistent order between the simulation and the experimental measurement; in other words, ranking the designs by their achieved speeds will result in the same ordering in

simulation as in experiments. These observations provide support for the effectiveness of the simulation model and the design approach. On the other hand, we note that the *Best Speed* evolved parameters result in a considerably faster speed in simulation than in experiments. Analyzing high-speed video of the robotic fish reveals that the physical caudal fin cannot reach as large an amplitude as in simulation. This behavior can likely be attributed to the high frequency at which the device is operating. As listed in Table 5.2, the frequency for this individual is 3.8 Hz, which is near the upper limit we found to be physically possible during our initial prototyping. Modeling of the electromagnetic actuator will need to be refined to ensure that the simulation environment cannot be exploited by the evolutionary algorithm, as occurs for the *Best Speed* experiment.

Even with a higher accuracy simulation the reality gap can still be an issue. For this reason, many research groups are investigating explicit methods for countering the reality gap. Two of the most prominent are the "intelligent trial-and-error" algorithm [34] and self-modeling [10]. As an alternative to these methods, in the next chapter we discuss our investigations regarding adaptive control as a method for mitigating effects of the reality gap [25]. In essence, the controller would *adapt* to the differences between simulation and reality.

5.4 Conclusion

In this chapter, we investigated an evolutionary multiobjective optimization approach to the design of morphology and control for a robotic fish with a flexible caudal. As a way of illustration, we chose two common objectives (maximizing swimming speed and minimizing energy consumption), although our approach would work with any combination of objectives (for example, agility, speed, and energy efficiency). The robotic fish used during validation experiments utilizes an electromagnetic actuator, which enables a direct calculation of energy consumption. Simulation-based optimization allowed us to explore trade-offs between the

two competing objectives. Certain evolved and hand-chosen sets of parameters were then selected for physical validation. With the aid of a 3D printer, simulated caudal fins were fabricated to match specifications, and simulation results were validated in a large water tank. Aside from individuals exhibiting the fastest speeds, physical trials support the effectiveness of the proposed evolutionary design approach.

Surprisingly, control parameters (frequency of oscillation and pulse-width-ratio) were found to account for most of the variation among Pareto-optimal individuals. This discovery is intriguing, as it indicates that at least for the robotic fish and objectives employed in this study, most of the engineering effort can be invested in controller design, while less time needs to be spent on morphology. Specifically, morphology can be fixed and control strategies can be designed such that most of the Pareto-optimal solution space can be reached. As a result, the robotic fish can move from high speed, high energy states to low speed, lower energy states that are near the final Pareto-front. Fixing the morphological characteristics and performing parameters sweeps on the control parameters enabled the discovery of individuals that exhibit such generality.

The purpose of this study was to investigate how compliant caudal fins could be matched with control parameters (as in Chapter 3) while considering performance and energy consumption. More broadly, the techniques presented in this study are intended to be a step toward our goal of producing optimization methods that can be used for any robots incorporating soft/flexible components. Although physical tests yielded results similar to those found in simulation, due to the reality gap there were some discrepancies. Furthermore, the discrepancy between simulation and reality tends to increase as the robot operates for longer periods of time; primarily due to battery depletion and changes to the 3D printed caudal fin dynamics caused by aging. In the next chapter, we introduce our research aimed at mitigating these modeling errors and runtime changes. Namely, we use adaptive controllers to automatically adapt to unexpected variations that occur during operation.

Chapter 6

Evolving Adaptive Control

The previous chapters demonstrated that evolutionary computation methods can be used to discover combinations of morphological and control pattern values that result in both high performance and energy efficiency. However, those methods are applied offline, prior to deployment. Yet, many robotic systems experience fluctuating dynamics during their lifetime. Variations can be attributed in part to material degradation and decay of mechanical hardware. One approach to mitigating these problems, as well as help to cross the reality gap, is to utilize an adaptive controller. For example, in model-free adaptive control (MFAC) a controller learns how to drive a system by continually updating link weights of an artificial neural network (ANN). However, determining the optimal control parameters for MFAC, including the structure of the underlying ANN, is a challenging process. In this chapter we investigate how to enhance the online adaptability of MFAC-based systems through computational evolution.

Some of the results and descriptions in this chapter were published in [24] and [25].

6.1 Model-free Adaptive Control

As described in Chapter 2, the MFAC used in our research is based upon an *adaptive* ANN, whose weights are dynamically updated. A diagram of the MFAC ANN is shown in Figure 6.1. As an input, the ANN receives a continuous error signal e, which is discretized at a sampling rate T_s. Sampling the error signal is necessary for computer systems, which are unable to handle continuous data. The discretized error signal is normalized between -1 and 1 using a configurable error bound e_b. This normalized, discretized error signal, E, is passed to the first input neuron, I_1, and then propagated to each subsequent input neuron at successive sampling times. This process is repeated such that the N_I input neurons store the N most recent error signals ($E_1..E_{N_I}$). By storing these values and using them as additional inputs to the ANN, the MFAC takes advantage of state information and can be called a *dynamic* system. Additionally, at each sampling time, the input neuron values ($E_1..E_{N_I}$) are fed forward from the input neurons ($I_1..I_{N_I}$) to the N_H ANN hidden neurons ($H_1..H_{N_H}$), which in turn feed their values to the output neuron (V). The final output of the MFAC, u, is a summation of the output neuron value and the current error signal amplified by the controller gain K_c.

The number of input neurons, N_I, the number of hidden neurons, N_H, as well as the rate at which link weights are updated (called the adaptive learning rate, η), are configurable. Responsiveness of an MFAC is also adjustable through a controller gain value K_c. An MFAC ANN activates similarly to a typical feed-forward neural network as follows:

$$p_j(n) = \sum_{i=1}^{N_I} w_{ij}(n) E_i(n) + bias_j, \tag{6.1}$$

$$q_j(n) = \sigma(p_j(n)), \tag{6.2}$$

where (n) denotes the sample time, p_j is an intermediate output of the jth hidden neuron, w_{ij} and E_i are the weight and value of a connection from the ith input neuron to the jth

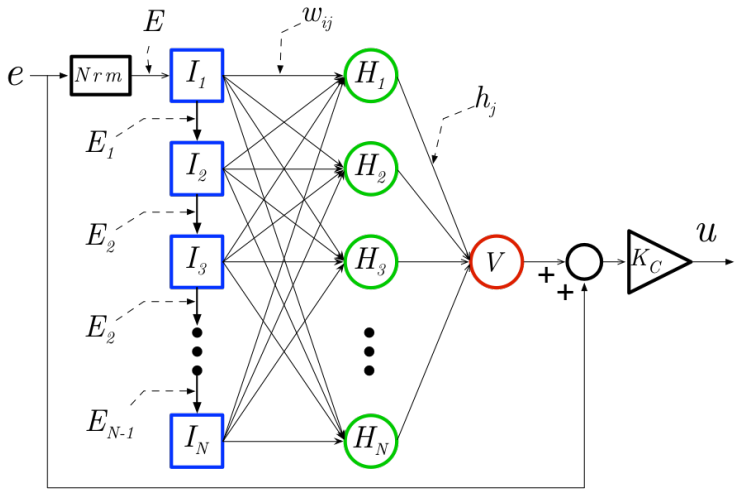


Figure 6.1: A graphical representation of the MFAC ANN. A continuous time error signal e is first normalized (Nrm block) and then propagated through the input neurons I_i. The ANN is then activated as a feed-forward network to produce an output V. The final controller output u is an amplified summation of V and e.

hidden neuron, respectively, $bias_j$ is the bias value for the jth hidden neuron, q_j is the output of the jth hidden neuron, and $\sigma(\cdot)$ is the sigmoid function. The sigmoid function and its derivative are as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{6.3}$$

$$\dot{\sigma}(x) = \sigma(x) \cdot (1 - \sigma(x)). \tag{6.4}$$

The simple nature of the sigmoid's derivative is one of the primary reasons that sigmoids are used for neural networks. The output of the neural network, and output of the entire MFAC are as follows:

$$o(n) = \sum_{j=1}^{N_H} h_j q_j + bias_o, \tag{6.5}$$

$$u(t) = K_c(o(t) + e(t)), \tag{6.6}$$

where o is the output of the output neuron, h_j is the value of a connection from the jth hidden neuron, $bias_o$ is the bias of the output neuron, u is the output of the MFAC, K_c is the controller gain, and e is the continuous error value (between the setpoint r and the output y).

What sets an MFA apart from typical feed-forward neural networks is how the weights (w_{ij} and h_j) are updated. The connection weight update equations are derived by minimizing error. The objective function is:

$$E_s(t) = \frac{1}{2}(r(t) - y(t))^2 = \frac{1}{2}e(t)^2, \tag{6.7}$$

To minimize the objective function with respect to the weights between the input and hidden layers (w_{ij}), we take the partial derivative as follows:

$$\begin{aligned}
\Delta w_{ij}(n) &\propto \frac{\partial E_s}{\partial w_{ij}}, && (6.8)\\
&= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial w_{ij}}, && (6.9)\\
&= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial w_{ij}}, && (6.10)\\
&= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial o}\frac{\partial o}{\partial w_{ij}}, && (6.11)\\
&= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial o}\frac{\partial o}{\partial q}\frac{\partial q}{\partial w_{ij}}, && (6.12)\\
&= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial o}\frac{\partial o}{\partial q}\frac{\partial q}{\partial p}\frac{\partial p}{\partial w_{ij}}. && (6.13)
\end{aligned}$$

Similarly, for the weights found between the hidden and output layers (h_j) we end up with the following partial derivatives:

$$\Delta h_j(n) \propto \frac{\partial E_s}{\partial h_j}, \tag{6.14}$$

$$= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial h_j}, \tag{6.15}$$

$$= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial h_j}, \tag{6.16}$$

$$= \frac{\partial E_s}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial o}\frac{\partial o}{\partial h_j}. \tag{6.17}$$

The solution to these partial fractions are as follows:

$$\frac{\partial E_s}{\partial y} = -e, \tag{6.18}$$

$$\frac{\partial y}{\partial u} = S_f, \tag{6.19}$$

$$\frac{\partial u}{\partial o} = K_c, \tag{6.20}$$

$$\frac{\partial o}{\partial q} = \sum_{j=1}^{N} h_j, \tag{6.21}$$

$$\frac{\partial q}{\partial p} = q_j(1 - q_j), \tag{6.22}$$

$$\frac{\partial p}{\partial w_{ij}} = E_i. \tag{6.23}$$

where S_f is a problem specific sensitivity function. Given Equations 6.13, 6.17, and the above solutions to the partial derivatives, we find the following update equations for the neural network weights:

$$\Delta w_{ij}(n) = -\eta\, K_c\, S_f(n)\, e(n)\, q_j(n)\, (1 - q_j(n))\, E_i(n) \sum_{k=1}^{N} h_k(n), \tag{6.24}$$

$$\Delta h_j(n) = -\eta\, K_c\, S_f(n)\, e(n)\, q_j. \tag{6.25}$$

Despite the benefits of the MFAC approach, determining the optimal values of the parameters (N_I, N_H, S_f, η, K_c, T_s, and a bound on the error e_b) is challenging and depends on the application domain.

6.2 Methods

In studying animal behavior, zoologists recognize the importance of the *evolved* relationships among morphology, perception, action, and environment [38]. This concept is also relevant to cyber-physical systems, where embedded computer systems need to interpret sensed information and respond accordingly through actuators. Control theory focuses on how to modify the behavior of dynamic systems using feedback. The control of a cyber-physical system typically involves monitoring the output of a system (e.g., speed, temperature, flow-rate, etc.) and adjusting the system input accordingly. Often the goal of a control system can be referred to as *tracking*. The objective for a tracking controller is to generate a signal that drives the controlled system to behave in a similar manner to an input reference signal. Two examples of controlling the speed of a robotic fish are shown in Figure 6.2. The system input reference signal (the black, dashed line) and the system output (blue line) are the desired and measured speeds, respectively, of the robot. The error between these two values, which acts as an input to the controller, is shown in red. The goal of a tracking controller is to minimize the error. In Figure 6.2(a), the controller exhibits poor tracking, that is, the system output does not closely track the reference signal (notice that the output speed oscillates around the desired speed). In contrast, the Figure 6.2(b) shows a controller that is effective in tracking.

6.2.1 MFAC for Robotic Fish.

Unlike controllers that adapt by updating parameters specific to a reference model of the system, an MFAC "relearns" how to control the system by continually updating link

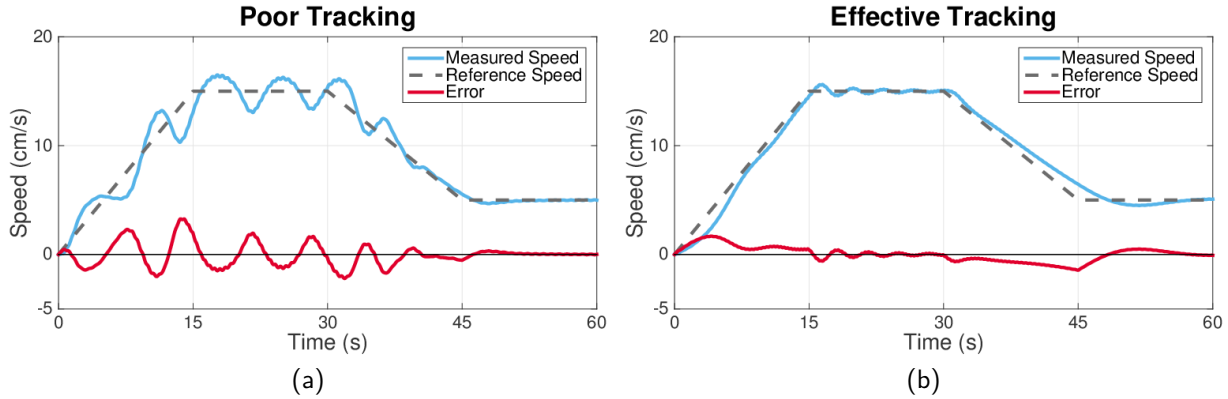


Figure 6.2: Examples of a robotic fish controller tracking an input reference signal representing desired speed. In each plot, the dashed, black line denote the input reference signal (r), the blue line represents the measured systems output (y), and the red line represents the error (e) between these two signals. Plot (a) is an example of relatively poor tracking, whereas plot (b) exhibits effective tracking.

weights in an adaptive ANN. As an input to the neural network, MFAC takes a continuous error signal and discretizes it based on a configured sampling rate. Using an error signal as input to the control system is a common approach to attaining a tracking behavior. For example, PID controllers also use a discretized error value; however, our experience with such systems show that, while they are able to adequately control robotic fish, they begin to exhibit poorer performance over time as characteristics of the physical device begin to change. By saving recent error signals and using them as additional inputs to the ANN (labeled I_1 to I_{NI} in Figure 6.3), the MFAC can take advantage of state information, or so-called neural network memory [52].

Figure 6.3 shows a block diagram of the MFAC controller with the robotic fish. The input to the entire system is a reference signal r, which can be any physical signal relating to the robotic fish. For this study, r refers to a *desired* speed, and the output of the robotic fish y is the *actual* (measured) speed. For physical experiments, speed can be measured by filtering and integrating accelerometer data. Generally, reference signals are generated by a higher-level module.

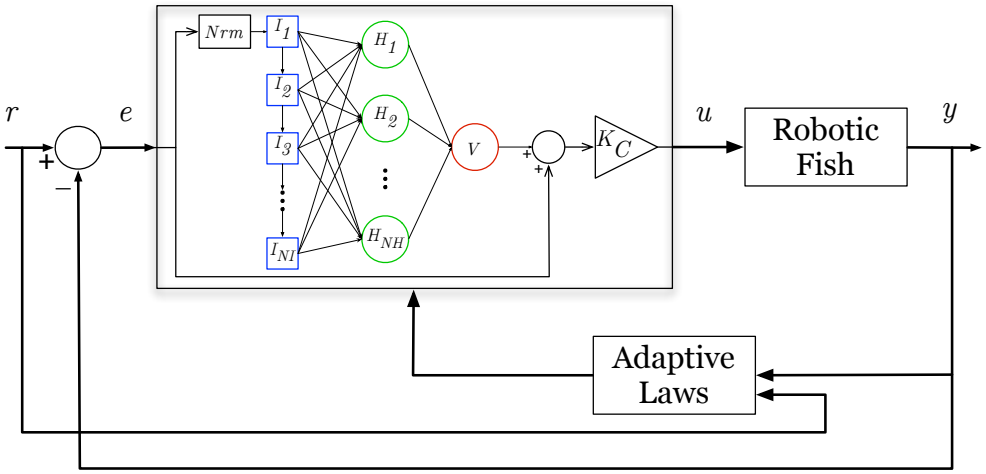


Figure 6.3: A block diagram of the MFAC controller and the robotic fish. Signals r and y denote the reference and measured speeds, respectively, e is the difference between reference and measured speeds, and u is the controller output.

The controller's objective is to produce a control signal u such that y closely tracks r. That is, an effective controller will force the robotic fish to closely match the desired speed, and have little error e between y and r. For the robotic fish, u is a frequency of oscillation for the caudal fin motor, and a numerically controlled oscillator (NCO) generates a sinusoidal pattern at the given frequency. For this study, we have fixed the sinusoid's amplitude to 20°.

Figure 6.4 shows the difference between a lack of adaptive ability and a controller that is able to adapt. In both figures the fin suffers damage at the 30 second point (denoted by the vertical orange line); specifically, the fin is shortened from 8 to 5.2 cm and the flexibility is changed from 3.0 to 2.1 GPa. Figure 6.4(a) shows the resulting behavior when the system cannot handle the change; the system starts oscillating around the target speed, but does not achieve good tracking. Figure 6.4(b) shows the results for an adaptive controller that is able to regain tracking after damage.

The motivating problem for this study is how to specify the MFAC parameters in such a way that they allow the controller to adapt to variations in caudal fin behavior. As a robotic fish's caudal fin undergoes regular wear and degradation it will begin to respond differently to motor commands, particularly if the fin is fabricated from flexible materi-

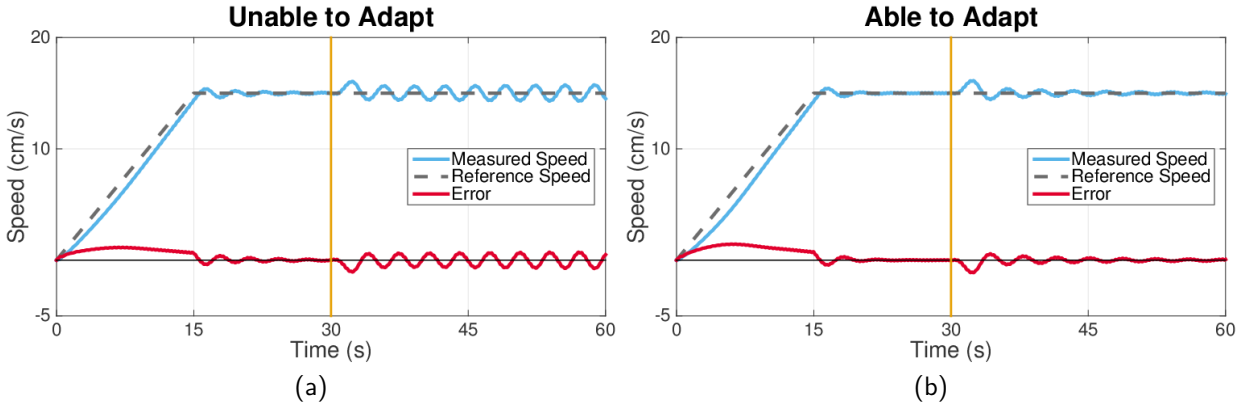


Figure 6.4: In both (a) and (b), the fin suffers damage at t=30 s. In (a) the robotic fish is controlled by a non-adaptive controller, and in (b) the robot is controlled by an adaptive controller. The adaptive controller is able to successfully regain the ability to track.

als. Consequently, a static, non-adaptive feedback controller will begin to *detune*, leading to deteriorated performance. Additionally, an MFAC controller should be better able to handle noisy accelerometer data, although we do not consider noisy measurements in our simulations.

MFAC parameters are typically set based on expert knowledge. For example, the robotic fish that this study is based on has a maximum tail frequency of roughly 3.5 Hz. Cheng et al. [20] recommend for the sampling rate to be less than one-third of the period of the controlled system, or roughly 0.1 seconds for the robotic fish. The error bound used for normalization e_b can be set to near the maximum speed at which the robotic fish is expected to travel (i.e., 15 cm/s for the modeled robotic fish). A good starting point for the number of input and hidden neurons N is 3, as the ANN can interpret inputs as the current error and its first and second derivatives. This setup would roughly correspond to a PID algorithm, a widely used general-purpose feedback controller. Typical initial values for the gain K_c and the learning rate η are 1.0 and 0.8, respectively. In Section 6.3, we compare the performance of an MFAC controller implemented with these values to that of one with evolved values.

6.2.2 Simulation Dynamics.

Simulation of the robotic fish is conducted in Simulink [115], enabling a straightforward translation of dynamic equations into simulation. More details regarding the simulation environment can be found in Chapters 2 and 4. The robotic fish prototype upon which this simulation is based (similar to that in Figure 7.1) is approximately 20 cm in length, including a 7.6 cm long tail fin of moderate flexibility made from a 3D-printed ABS plastic.

6.2.3 Differential Evolution.

Other studies have investigated how EC can be used to *enhance* more traditional engineering methods. For example, Coelho et al. [30] used evolutionary methods to improve the performance of an adaptive controller by evolving a neural compensator. For this work, we used differential evolution (DE) [120], a global optimization algorithm that operates in a similar manner to other evolutionary algorithms, but which has been shown to converge faster than real-valued genetic algorithms for certain classes of problems [100].

DE progresses in a fashion similar to other evolutionary algorithms. First, a population is randomly initialized. For this study, the population size is set to 50, which is the recommended value for a DE experiment with 5 evolving parameters (T_s, e_b, N, K_c, and η). Next, each individual is evaluated with a problem-specific fitness function. In this study, we employ two different fitness schemes. In the first, individuals are simulated for only one set of conditions, and fitness is assigned as the mean absolute error (MAE) (i.e., the average error between r and y). In the second, each individual is simulated under a variety of different conditions (varying caudal fin characteristics). Fitness is then assigned as the sum of the MAE for each set of conditions. Once each individual has been assigned fitness, the DE algorithm produces a new generation of individuals.

Mutation and crossover operators are where DE differs from conventional real-valued genetic algorithms. DE focuses on creating new individuals near the best member of the parent population. Each child is initialized as a linear combination of the best and at least

two other randomly selected parents. During this recombination, the relative weight of the *best* parent to the random parents is referred to as the mutation factor and is configurable (0.8 in this study). The child is then crossed with the *base* parent (each parent is taken as the base in turn) using a configurable crossover rate (0.7 in this study). DE algorithm specifics, as well as a comparison with other evolutionary optimization algorithms, can be found in [100]. Using Storn's DE notation, the algorithm utilized for this study is denoted as DE/best/2/bin, where best signifies that all children are created around the previous generations best individual, 2 denotes that mutation is based on two individuals, and bin refers to a binary crossover operation.

MFAC parameters are allowed to evolve only within a certain range. The range for each parameter is listed in Table 6.1. These ranges are based on the typical MFAC parameters discussed earlier.

Table 6.1: Evolutionary Range of MFAC Parameters, as Well as the Typical Values.

	Minimum	**Maximum**	**Typical**
T_s (s)	0.0	0.17	0.1
e_b (cm/s)	5.0	50.0	15.0
N	1	8	3
K_c	0.1	4.0	1.0
η	0.1	4.0	0.8

6.3 Single-Evaluation Experiments and Results

In this set of evolutionary experiments we evolve MFAC parameters under a single set of conditions. However, to provide a set of baseline results we first conduct a simulation of the robotic fish incorporating typical MFAC parameters (as listed in Table 6.1). Figure 6.5 shows results from this simulation. The task for the MFAC controller is to track a reference speed r (the orange, dashed line in Figure 6.5), which varies over time according to a predefined pattern. This reference speed, utilized during evolution and most test cases, is designed to contain periods requiring acceleration, deceleration, and sustaining a constant speed.

Despite choosing parameters based on expert knowledge, the controller struggles to track the reference speed. Ideally, in Figure 6.5 (and all similar figures) the solid blue line (y) would match the dashed orange line (r), and the error line (e) would remain at zero.

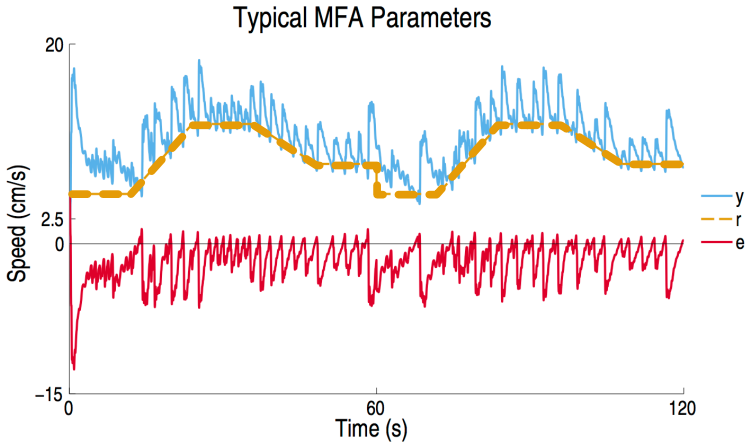


Figure 6.5: Results for an MFAC controller with typical parameters controlling a robotic fish. The dashed orange line denotes the reference speed r, the actual speed y of the robotic fish is the blue line, and the error e between these signals is red.

After the initial simulation using typical parameters, we conducted 20 replicate DE experiments. Replicates are seeded with a unique number, and DE algorithm parameters are configured as described in Section 6.1. Each set of MFAC parameters (i.e., individual solutions) is evaluated under identical circumstances: it is simulated for 60 seconds with the same reference speed signal, and fitness is measured as the mean absolute error (MAE). All replicate experiments converge to similar fitness values within 150 generations.

As shown in Figure 6.6, solutions from the single-evaluation experiments perform the evolutionary task well (i.e., tracking the reference speed encountered during evolution for 60 seconds). However, even a slight change to this task, such as doubling the simulation to 120 seconds, causes a large change in performance. This behavior can be seen during the final 60 seconds of Figure 6.6, where simply repeating the reference signal results in poorer tracking and increased error. This experiment demonstrates that evolved solutions are incapable of adapting to new conditions while maintaining the same level of performance.

More specifically, the evolved parameters appear to be *overfit*. The best solutions only work for the conditions encountered during evolution.

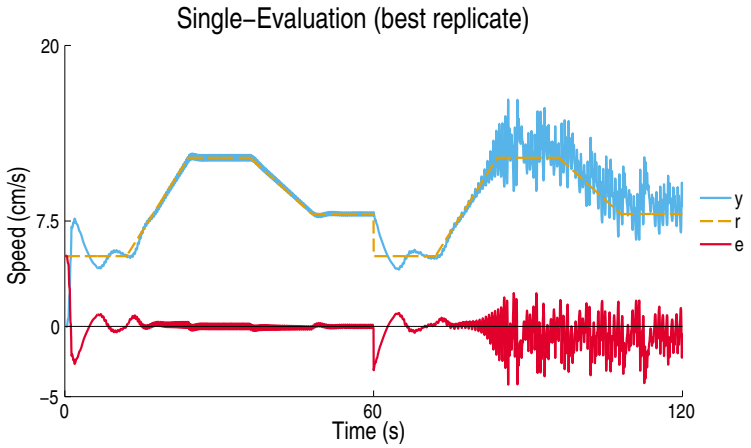


Figure 6.6: Results for the overall best (across all replicate experiments) single-evaluation solution simulated with default fin characteristics and the same reference signal utilized during evolution. The controller shows poor performance starting at the 80 second mark, and similar results were found in all replicate experiments.

6.4 Multi-Evaluation Experiments and Results

Given the results from Section 6.3, we conducted a second set of experiments in which fitness of each individual is based on its performance under *multiple* different conditions. The settings for these simulations, referred to as the 9-evaluation experiments, are listed in Table 6.2. In the table, *sim1* corresponds to the conditions used in the previous experiments. For each simulation, fin flexibility is set to: 100%, increased to 200%, or decreased to 50% of the default value. Likewise, the caudal fin length is set to: 100%, lengthened to 110%, or contracted to 90% of the default value.

Evaluating individuals under a variety of conditions is intended to eliminate the tendency of evolving overfit solutions. Experiencing multiple conditions also simulates how fins may change once deployed. For example, fin dynamics can change if the fin is damaged (e.g., cut) or encumbered by environmental entities (e.g., seaweed). Fitness is calculated as the summation of the MAE from each of the 9 60-second simulations. Evolving with this fitness

Table 6.2: Fin Characteristics for the 9-Evaluations Experiment.

	Flexibility	**Length**
sim1	100%	100%
sim2	200%	100%
sim3	50%	100%
sim4	100%	110%
sim5	200%	110%
sim6	50%	110%
sim7	100%	90%
sim8	200%	90%
sim9	50%	90%

function is meant to add an implicit objective to the fitness function: better solutions must be more adaptable.

Figure 6.7 shows the best solution from 20 replicates of the 9-evaluation experiments. Here, the controller continues to closely track the reference for 120 seconds, even though evolutionary evaluations are only 60 seconds in length. Figure 6.8 shows that tracking is accomplished by adjusting the fin's oscillating frequency in a pattern roughly matching that of the reference signal. Although this test indicates improvement over the single-evaluation experiment, it does not address the issue of adapting to different conditions.

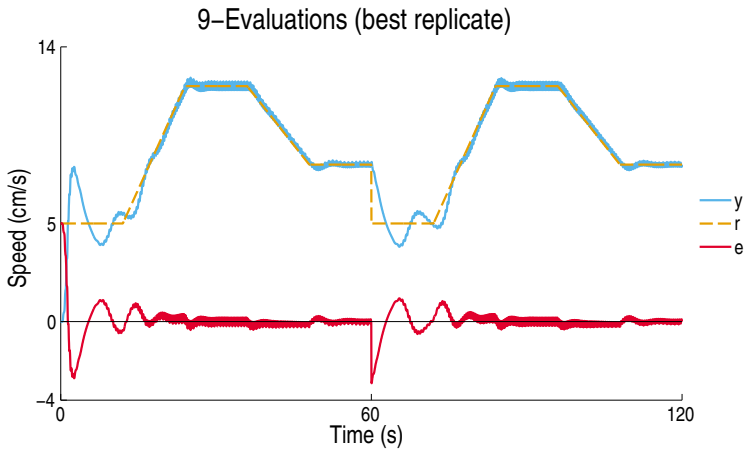


Figure 6.7: The overall best 9-evaluation solution evaluated on *sim1*. The MFAC controller is able to drive the robotic fish at the desired reference speed (r).

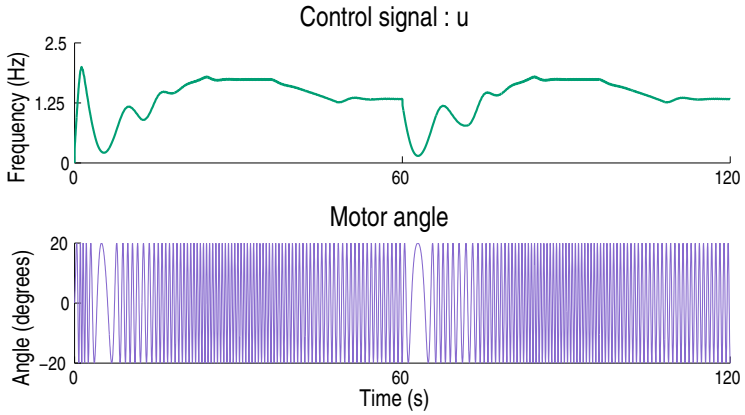


Figure 6.8: Control signal u and the resulting motor angle for the overall best 9-evaluation solution evaluated on *sim1*. The control signal trajectory roughly follows the reference signal.

Figure 6.9 depicts performance of the same solution when confronted with conditions that were *not* encountered during evolution. Notably, the controller is able to adapt to the novel fin lengths. Significantly, this evolved MFAC controller should allow a robotic fish to maintain a certain level of performance even if the fin length changes during operation.

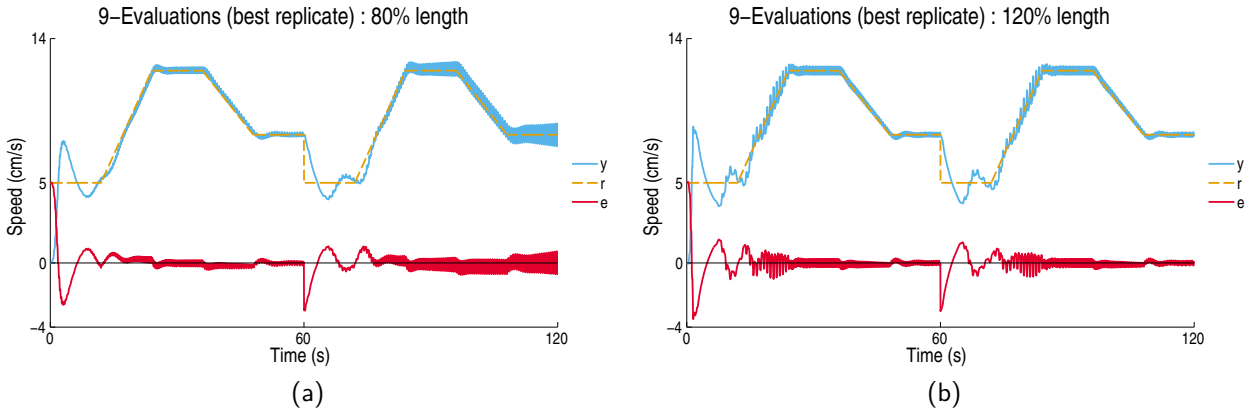


Figure 6.9: The overall best 9-evaluation solution tested against fin lengths that were not encountered during any of the evolutionary simulations. In (a) fin length is shortened to 80% of the default length, and in (b) fin length is lengthened to 120% of the default length. In both cases, the evolved controller is able to adapt to a novel fin length.

The fin can, however, reach lengths that cause the controller to lose its tracking ability. While fixing fin flexibility and the reference signal, we performed a sweep over a wide range of different fin lengths and found that the controller can maintain performance while caudal fin length is within a range of 60% to 137% of the default value. Effectively the caudal fin

can be cut from 7.6 to 4.5 cm (or lengthened to 10.4 cm) without the controller losing its ability to drive the robotic fish at a desired speed. Values outside of this range cause a noticeable increase in the error signal.

Figure 6.10 shows that evolved controllers are also able to adapt to changes in fin flexibility. Similar to the fin length parameter sweep, we found upper and lower limits for fin flexibility changes. While keeping all other factors constant, the evolved MFAC controllers can maintain performance as long as flexibility remains within a range from 90% to 160% of the default value.

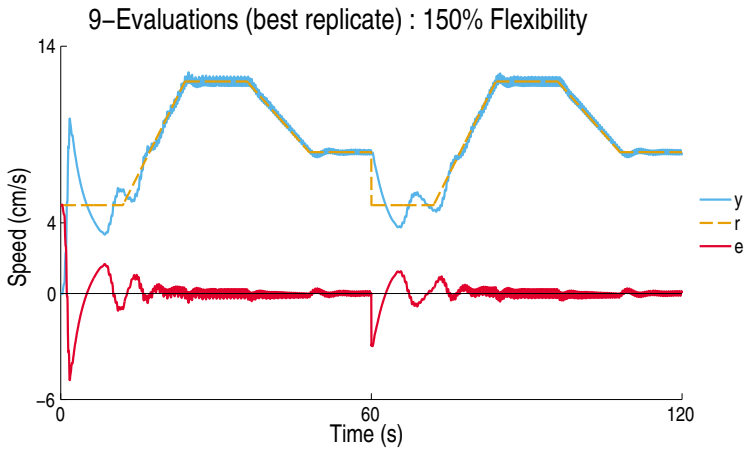


Figure 6.10: The overall best 9-evaluation solution evaluated with a fin that is 150% of the default fin flexibility. Evolved controllers were able to adapt to this novel value for flexibility.

In addition to changes in fin characteristics, evolved controllers have the ability to adapt to different reference signals. Figure 6.11 demonstrates that an evolved controller is capable of tracking a novel pattern for the reference speed, in this case alternating periods of fast acceleration and deceleration. Additional test results (not shown) demonstrate that limits on the reference signal depend only on the limits of the robotic fish. Specifically, the adaptive controller will remain effective as long as the reference signal does not require speeds, accelerations, or decelerations that are impossible for the robotic fish. For example, if the reference signal changes too quickly, the robotic fish may not be physically capable of accelerating fast enough.

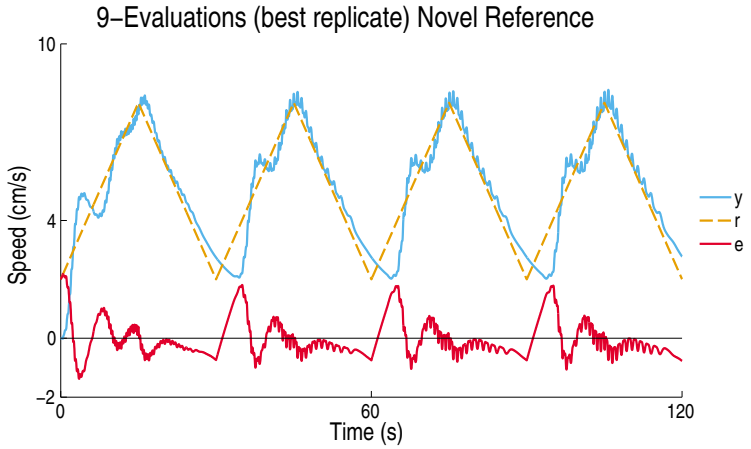


Figure 6.11: The overall best 9-evaluation solution simulated with default fin characteristics and a reference signal not encountered during evolution.

Figure 6.12 shows how the evolved controller handles simultaneous changes to both fin length and fin flexibility. For this test, fin length is set to values outside of the range encountered during all evolutionary simulations. For the test in 6.4, increasing the fin's length actually allows the evolved controller to adapt to a flexibility (80%) that would otherwise cause performance degradation. Specifically, a flexibility of 80% (of the default value) is beyond the lower limit found when flexibility was altered in isolation (i.e., the range of 90% to 160% mentioned previously). This is indicative of the complex interactions among material properties (e.g., flexibility and dimensions). Such interactions cause difficulties when designing a simple feedback controller, such as a PID controller, or a model-based controller that must account for all of the necessary dynamics. An evolved adaptive controller can automatically handle these complex interactions.

To further increase adaptability of an evolved MFAC controller (i.e., increase the range of fin characteristic variation while maintaining the same performance levels), the 9-evaluations experiments were repeated with larger variations from the default values. For instance, in *sim5* (refer to Table 6.2) the flexibility is set to 1000% of the default value, and length is increased to 200% of the default value. Likewise, in *sim9* flexibility is set to 10% of the default value, while length is decreased to 67% of the default value.

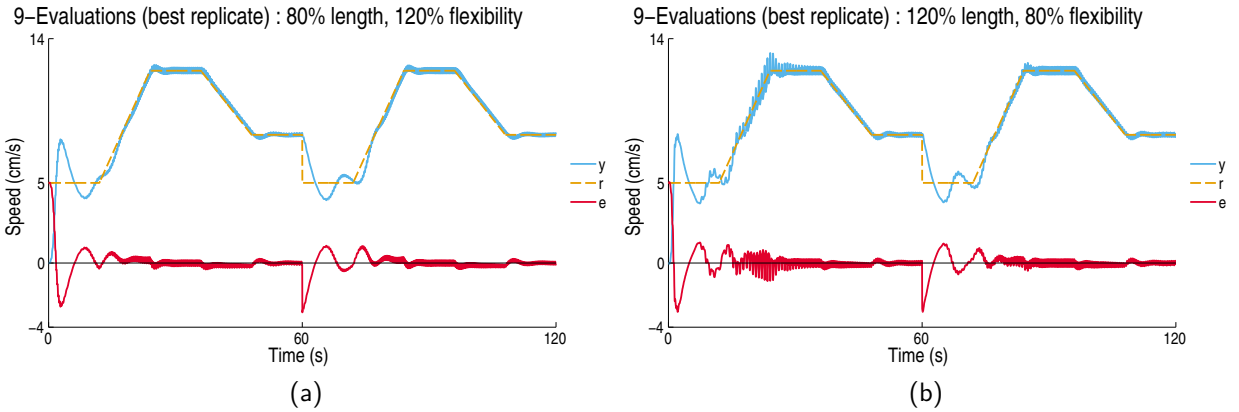


Figure 6.12: The overall best 9-evaluation solution tested against fin lengths and fin flexibilities that were not encountered during evolutionary simulations. In (a) fin length is shortened to 80% of the default length and the flexibility is increased to 120% of the default, and in (b) fin length is lengthened to 120% of the default length and fin flexibility is reduced to 80% of the default value.

Although the evolved controller is generally still able to track r, the best solutions from all replicates performed worse, on all test cases, than previous solutions. Figure 6.13 shows an individual from the altered experiments.

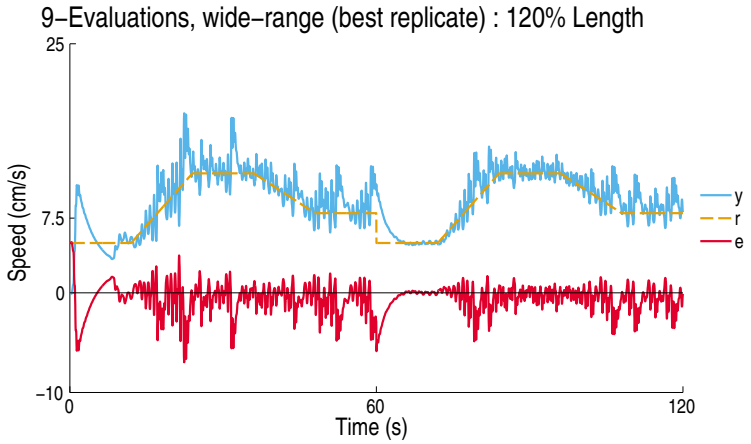


Figure 6.13: Performance of the best evolved solution from the altered 9-evaluations experiments tested with a fin length 120% of the default.

The primary reason for the MFAC's inability to adapt to large variations lies in how the dynamics change. Certain fin characteristics or combinations of characteristics cause the physical system to behave fundamentally differently. The basic MFAC presented in this study relies on the robotic system to be *direct-acting*. That is, as the MFAC controller output

increases, the output from the controlled device must monotonically increase. Figure 6.14 depicts how changes to fin characteristics can alter the robotic fish's response to commands from the controller. Essentially, if fin characteristics vary beyond a certain threshold, the robotic fish may no longer behave as a direct-acting robotic system. This issue is most clearly demonstrated by the green, dotted curve (200%), which changes from direct to reverse-acting near 1.5 Hz, and then back to direct-acting near 3.5 Hz. A similar effect can be seen, to a lesser extent, for each of the curves. Even an optimized MFAC controller will be unable to cope with these highly varied dynamics. Such limits, or *boundaries*, define an execution mode. That is, an adaptive controller can handle conditions within the mode, but is likely to fail if conditions go outside the mode.

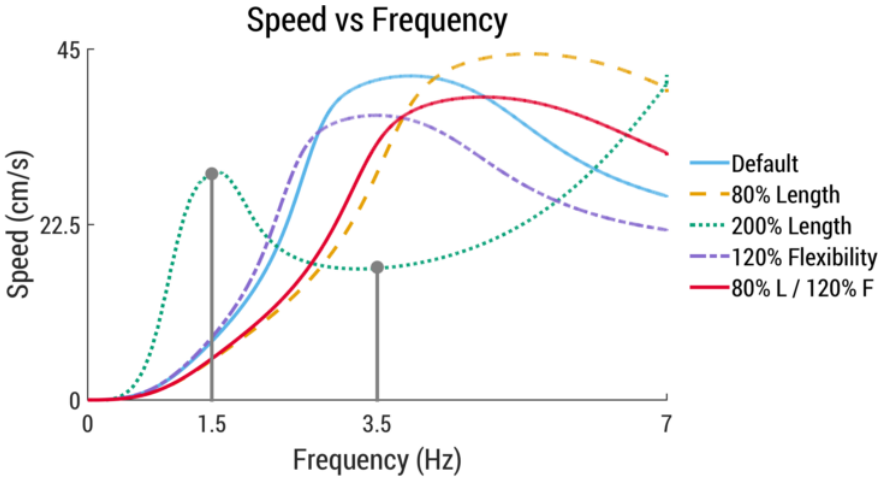


Figure 6.14: Speed vs. oscillating frequency for several different fin characteristics. For certain conditions increasing the frequency results in slower speeds.

6.5 Conclusions

In this chapter, we explored the integration of evolutionary computation and adaptive control. Specifically, we applied differential evolution to optimize the parameters of a model-free adaptive controller. The goal of evolution is to find a set of parameters that enable an MFAC controller to adapt to changes in fin characteristics (i.e., length and flexibility)

and changes to the reference signal (e.g., faster/slower accelerations). Additionally, evolved MFAC controllers should be able to handle changes to the reference signal (i.e., different desired reference speeds).

Results show that evolving MFAC parameters against a single set of fin characteristics can produce a controller capable of achieving *good* fitness (i.e., specifically, low mean absolute error). However, these solutions do not produce a controller capable of adapting to changing fin characteristics. Next, the fitness function was modified to include a variety of different fin characteristics. The newly evolved controllers were tested under several different conditions, including scenarios in which the fin characteristics were outside the range experienced during evolution. This method succeeded in generating more adaptable controllers. Specifically, the best MFAC controllers were able to maintain close tracking as long as fin length remained within 60% to 137% of the default and fin flexibility remained within 90% to 160% of the default.

To explore the limits of this approach, the fitness function was again modified to include a larger variety of different characteristics. However, these experiments resulted in poorer performing controllers. Drastically varying the fin characteristics essentially creates a fundamentally different set of governing dynamics. Evolved controllers require the system to be either direct- or reverse-acting, which is not always the case when, for example, the fin is made to be too flexible; in such cases, increasing the control frequency results in slower speeds rather than faster. Even the most fit individuals were incapable of successful adaptation when subjected to such conditions. In the next chapter, we discuss a method for automatically discovering limits for a given adaptive controller.

Chapter 7

Discovering Adaptation Boundaries

In the control theory domain, adaptation refers to a controller's ability to automatically adjust control parameters during operation in response to sensed feedback. A controller's environment includes any variables *not directly controlled by the system*. As we have seen in the preceding chapter, from the perspective of a robot's control software, the environment includes not only the physical surroundings in which the robot operates, but also characteristics of the body (morphology) of the robot. As with the physical environment, the morphology is subject to change. In this chapter, we use the term *scenario* to describe a specific set of these environmental parameters and their respective values. Compared to a static (i.e., non-adaptive) controller, the ability to adapt ensures that a single controller will remain effective for many scenarios. An important task for the designer of the robot, or any cyber-physical system, is to specify the limits of adaptation for the controller. That is, by how much can any given parameter change before a controller fails and the system needs to switch to a different mode of execution driven by a different controller?

In the previous chapter, we used DE to evolve the parameters of an MFAC in order to increase the controller's adaptability. Specifically, we demonstrated that exposing the MFAC to different scenarios during evolution produced a more resilient controller, even capable of

Some of the results and descriptions in this chapter were published in [110] and [7].

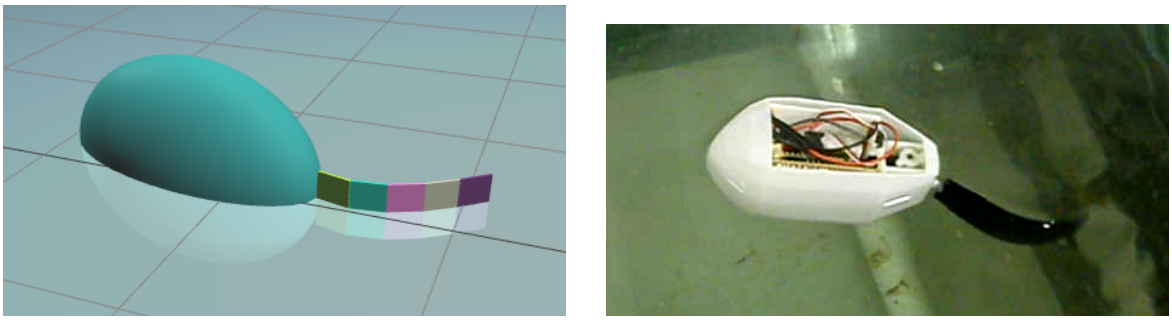
adapting to conditions beyond those to which the system was exposed during evolution. However, the focus of that work was on adaptability, and the scenarios were designed by hand. In addition, for some scenarios (such as extensive damage to the fin), the controller was simply unable to effectively adapt.

In this chapter, we explore the role of EC in discovering the boundaries of adaptive controllers. We again employ differential evolution, and combine it with a numerical simulation in order to evolve adaptive controller parameters. Over the course of evolution we expose controllers to different scenarios not only to enhance the adaptive capabilities of the controller (as in Chapter 6), but also to identify conditions under which the controller will fail. This information can be used to define the boundaries of an *execution mode*, enabling the system designer to implement higher-order strategies for switching among execution modes, typically involving a switch in controllers, when the system detects that a change is necessary [64]. But which method of generating scenarios is most effective in exploring the "adaptability boundaries" of the controller? The main contribution chapter is an approach that addresses uncertainty for autonomous robots at two levels. First, we evolve adaptive control to handle increasingly more adverse (and diverse) environmental conditions. And second, we automate the process of discovering the boundaries/limitations of adaptation, which will facilitate the development of decision logic for switching between adaptive controllers and their correspondingly mode of behavior.

7.1 Evolving a Base Morphology

As in the preceding chapters, the morphology of the caudal fin is described by three parameters: the flexibility, the depth (or height), and the length (simulated and 3D printed robotic fish are shown in Figure 7.1). We begin by evolving the three fin parameters for a simple task (maximum average speed), while using a sinusoidal signal as the "controller." The adaptive controllers evolved in the next sections will be required to respond to changes

in this "base" morphology. In Chapters 4 and 5, we validated the results of this process by 3D-printing fins and testing them on physical robots.



(a) (b)

Figure 7.1: (a) Rendering of the simulated robotic fish. Individual fin segments appear in different colors, and the fin appears to extend above the surface of water for visualization purposes only. (b) A prototype 3D-printed robotic fish with a flexible caudal fin (top cover removed for illustration).

We conducted 30 replicate differential evolution experiments in which the caudal fin morphology and the frequency and amplitude of the sinusoidal control signal are subject to evolution. Figure 7.2(a) plots the average evolutionary trajectory of these replicates and shows that they converged quickly (within 10 generations) to a final fitness value of approximately 22 cm/s. The box-plots in Figure 7.2(b) demonstrate that each of the five evolved parameters converged (the maximum value for a fin's Young's modulus is dictated by our fabrication process [27]). Only the first four parameter values (the three morphological parameters and control amplitude) are used to represent our base system. In subsequent experiments, the fifth parameter (frequency of the caudal fin) will be governed by an adaptive controller tasked with matching the speed of the robot to an input reference signal, despite changes to the base morphology.

7.2 Mode Discovery Algorithm

With this base morphology in hand, we can apply evolutionary search in order to discover and explicitly define the boundaries of adaptability for an MFAC that controls the caudal

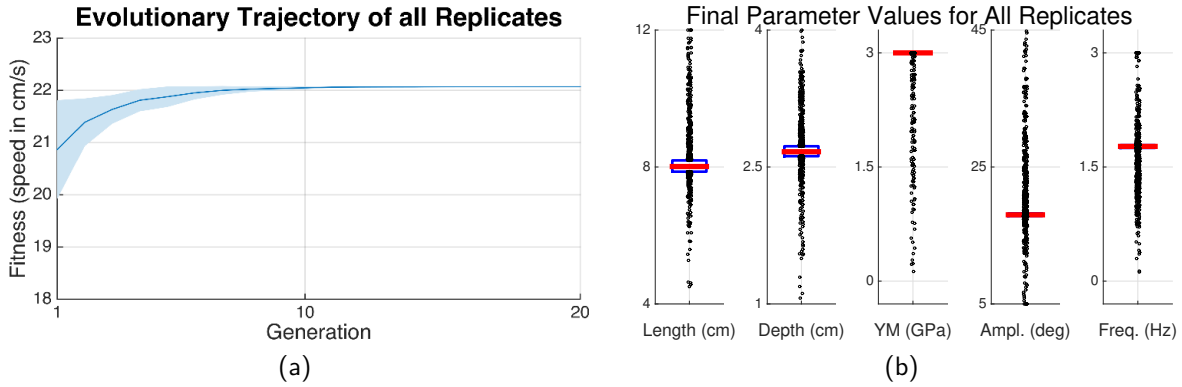


Figure 7.2: Evolution of base morphology: (a) Mean values for the best performers across all 30 replicate evolutionary runs. The shaded region represents the 95% confidence interval. (b) Box-plots of distributions of the final evolved parameters, across the replicate runs. The median values are represented by a horizontal red line, blue boxes represent one standard deviation either side of the mean (the blue boxes are sometimes covered by the median line), and black circles denote outliers.

fin. Notably, this approach should also produce an MFAC that can operate effectively within those boundaries. In general, the environment for a robotic fish controller includes the morphology of the physical system and the external aquatic environment (water eddies, turbulence, etc.). Here, we focus only on changes to morphology, which typically occur due to uncertainty and unpredictable circumstances such as damage or when the device becomes entangled. Our basic approach is to expose the MFAC to multiple scenarios during the evolutionary process. Next, we define precisely what constitutes a scenario and how we determine whether a scenario is feasible, followed by a description of the developed algorithm.

7.2.1 Scenario Parameters.

Each scenario includes the three morphological parameters discussed earlier: fin length, fin depth, and fin flexibility. Because the MFAC will be required to track a variety of reference signals (i.e., desired behaviors), each scenario also includes a reference input whose parameters are generated randomly. For this study, references describe only dynamics associated with the speed and accelerations of the robot, but the same approach could address other more complex behaviors and maneuvers. As depicted in Figure 7.3, each reference sig-

nal involves an interval of acceleration from zero cm/s up to a constant speed S_1, followed by either another acceleration or a deceleration to a second constant speed S_2. The durations of the four intervals are defined by parameters t_1, t_2 and t_3. This approach enables generation of novel reference signals that contain a rich set of dynamics (i.e., different maximum and minimum speeds and accelerations/decelerations).

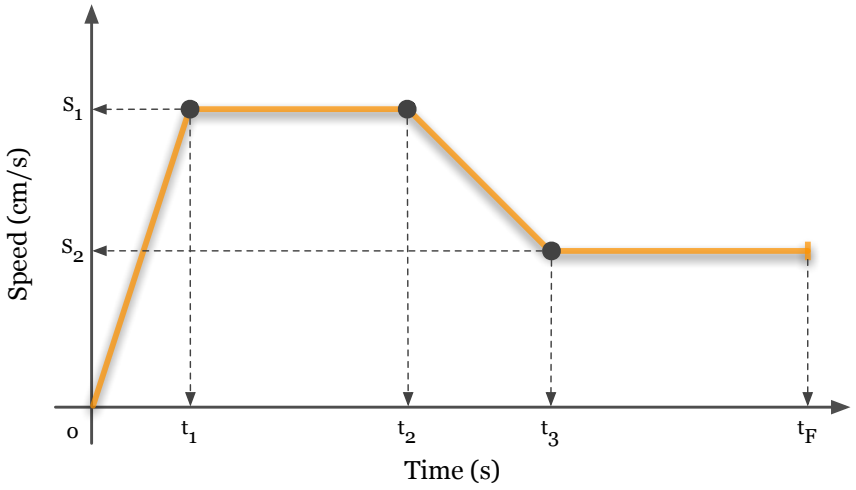


Figure 7.3: Reference signal scenario parameters include values to describe four time segments. In the first segment (from t=0 to t=t_1) speed ramps from 0 to S_1, in the second segment the reference speed remains steady at S_1. In the third segment (from t=t_2 to t=t_3) the speed ramps (up or down) to the final speed, which is held steady during the final time segment.

7.2.2 Determining Scenario Feasibility.

Of course, some combinations of morphological parameters may produce a robot that simply cannot be controlled effectively. For example, if the fin is severely damaged or too flexible, then it might be impossible to generate sufficient thrust to reach a specified reference speed, much less maintain it. Such a scenario is deemed infeasible, or *invalid*. We implemented an automated procedure for identifying such scenarios, so that they can be excluded from the resulting execution mode (as well as from the evolutionary search process). The feasibility procedure also considers the fact that the MFAC used in this study cannot be applied to systems that switch between direct and reverse acting. Direct (or reverse) acting

signifies that the system output, speed in this case, will always increase (or decrease) as the system input, frequency, increases. For example, for very flexible fins, speed can increase at low frequencies, but start to decrease at higher frequencies [27]. These behaviors are shown in Figure 9.

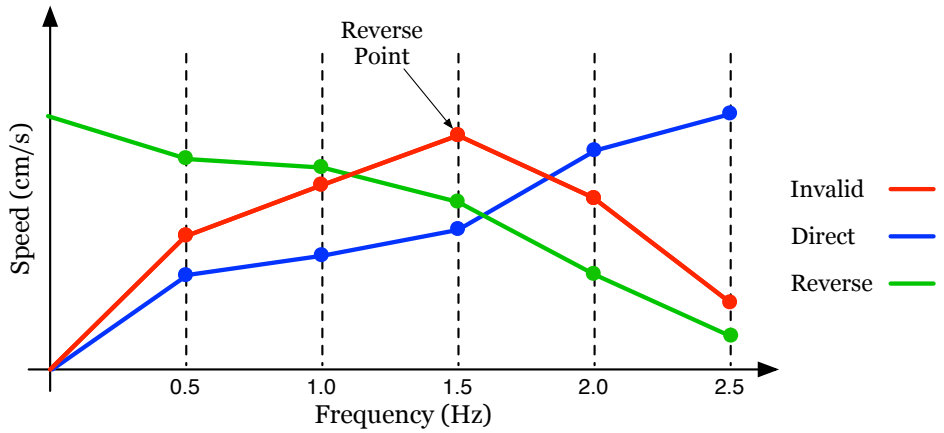


Figure 7.4: Three behaviors are shown. The blue and green lines denote systems that are direct and reverse acting, respectively. The red line shows a system that switches acting modes and is deemed infeasible.

Before considering a randomly generated scenario for integration in the evolutionary algorithm, it is first tested for feasibility as follows. A robot with the specified fin characteristics is simulated for 15 seconds with fin frequencies of 0.5, 1.5, and 2.5 Hertz. For each frequency, the robot is first allowed to reach a steady-state speed, and then an average speed is measured over the final 5 seconds. If the results show that the behavior changes from direct to reverse acting, or if the robot fails to reach a speed of 15 cm/s, then the scenario is considered to be infeasible. We chose 15 cm/s because the reference signal values are allowed to include a maximum speed of 20 cm/s, which is still lower than the maximum speed of the device. We note that no evolution (or feedback control) is involved in these evaluations; rather, they are intended only to evaluate the behavior of the parameterized robot at increasing control frequencies.

7.2.3 Evolvable Parameters.

Although an MFAC adapts by updating ANN link weights during execution, other parameters of the MFAC determine its *ability* to adapt. Table 7.1 lists the nine such parameters targeted in this study. Collectively, these parameters define the responsiveness and sensitivity of the controller, the structure and processing capabilities of the ANN, and the update periods for the ANN weights and the controller output. These parameters are configurable at design time, but typically do not change after deployment. Determining their optimal values is challenging and depends on the application domain. Traditionally, MFAC parameters are either chosen based on expert knowledge or tuned using proprietary software specific to the application. Here we *evolve* these parameters in order to enhance the adaptability of the controller.

Table 7.1: Evolvable MFAC Parameters

K_c	controller gain to amplify/reduce output
N_I	number of ANN input nodes
N_H	number of ANN hidden nodes
E_B	value used to normalize MFAC error inputs
η	learning rate for ANN edge weights
T_{out}	update period for generating MFAC output
T_{wt}	update period for MFAC weights
α	sensitivity parameter, reactivity of MFAC
β	sensitivity parameter numerator

7.2.4 Algorithm.

Figure 7.5 shows a flow chart of the algorithm developed for this study, which we refer to as the *Mode Discovery Algorithm*. We begin by evolving the base morphology. as described earlier. The base fin parameters are combined with a hand-designed reference signal to produce the *base scenario*, which is placed in the set of scenarios S applied in evolution. The base reference signal starts at 0, ramps up to 15 cm/s, and then ramps down to 5 cm/s. The algorithm then alternates between evolving the MFAC against scenarios in

S for a fixed number of generations (by default, 10) and generating a new scenario to add to S. Specifically, we evaluate the robot/MFAC for each scenario in S (taking the mean-absolute-value of the error signal), and fitness is calculated as the average of these results. The number of iterations through this basic loop is configurable (also 10 by default). The population size for the DE algorithm is 90. Evolving initially against only the base scenario is intended to bootstrap MFAC evolution, enabling the optimization process to start with an easier objective before adding a pressure for adaptability. We implemented and evaluated two methods for generating and selecting scenarios, termed *boundary selection* and *volume selection*. These methods, along with the corresponding results, are described in the next section.

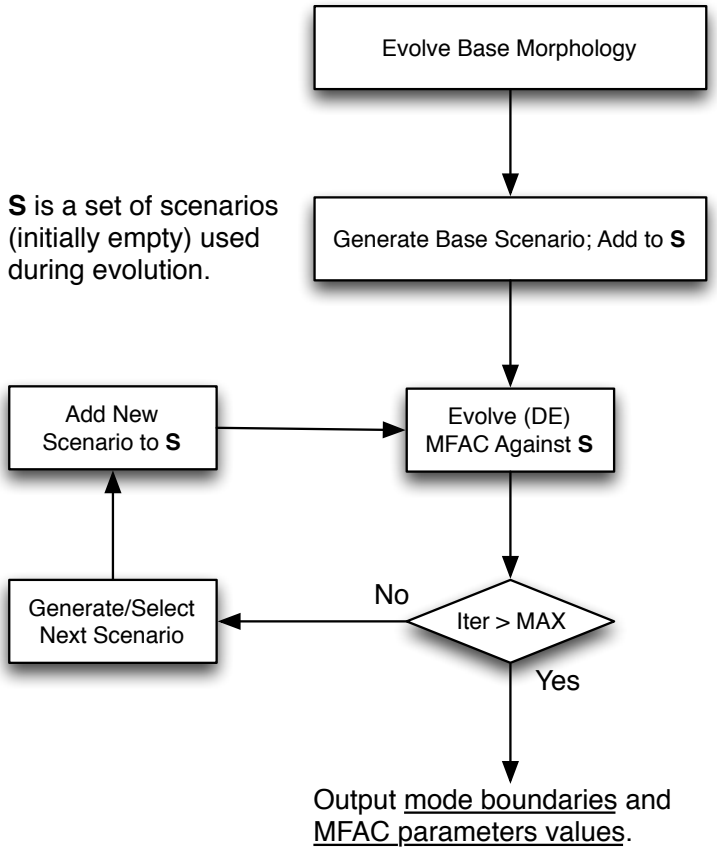


Figure 7.5: Flowchart of the *Mode Discovery Algorithm* used to discover execution mode boundaries and produce a MFAC parameter values for that mode.

7.3 Experiments and Results

In this section, we describe results for the two scenario selection methods, as well as for a parameter sweep experiment that provides a "ground truth" for the execution mode. Both approaches aim to enhance adaptability while at the same time providing information regarding execution mode boundaries.

7.3.1 Boundary Scenario Selection.

The first method starts by identifying the limits of each morphological parameter (i.e., fin length, depth, and flexibility), using the feasibility test described above. Specifically, we start at the base value for each parameter and increase/decease the value until the system becomes infeasible. During this process, we consider one parameter at a time, while the other parameters are fixed at their base values. The increment/decrement values were 1 mm for length and depth and 0.5 MPa for flexibility. The results of this process are depicted in Figure 7.6. We note that the maximum value for flexibility is equal to the base value of 3.0 GPa; our 3D printer cannot produce fins more rigid than 3.0 GPa, and the base value evolved for fin flexibility reached this maximum.

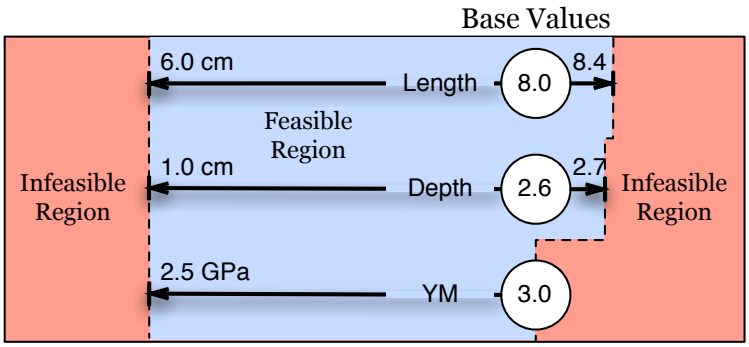


Figure 7.6: Boundary scenario values for the three fin parameters.

Collectively, these feasibility tests produce the six boundary scenarios listed in Table 7.2. These are integrated into the Mode Discovery Algorithm as follows. During the first 10 generations, the MFAC evolves using only the base scenario in fitness evaluation. At the

start of each subsequent iteration, one of the remaining scenarios is randomly selected and added to set S. Adding one scenario at a time gradually increases the difficulty of the task. At the start of the fifth iteration (and the beginning of the 60th generation overall) all six scenarios are in use. Since the order in which the scenarios are added is random, the orderings will differ across replicate experiments. Examination of 40 replicate experiments shows that the order has little if any effect.

Table 7.2: Boundary Scenarios (Fin Parameters)

	Length	**Depth**	**YM**
$scenario_{Base}$	8.0 cm	2.6 cm	3.0 GPa
$scenario_{length_{min}}$	6.0 cm	2.0 cm	3.0 GPa
$scenario_{length_{max}}$	8.4 cm	2.6 cm	3.0 GPa
$scenario_{depth_{min}}$	8.0 cm	1.0 cm	3.0 GPa
$scenario_{depth_{max}}$	8.0 cm	2.7 cm	3.0 GPa
$scenario_{YM_{min}}$	8.0 cm	2.6 cm	2.5 GPa

Consistent with the results in Chapter 4, adaptive controllers resulting from this process exhibit enhanced adaptability when compared to those evolved against only the base scenario. However, this approach has a limitation: it does not consider the interactions among morphological parameters. In earlier experiments [25], we observed combinations of parameters that were feasible, despite lying outside the feasible region depicted in Figure 7.6. For example, when a caudal fin is shortened it will generally continue to work well if it is sufficiently flexible. For example, when we test the feasibility of a fin with a length of 6.4 cm and a Young's modulus of 2.1 GPa (below the 2.5 GPa threshold) we find that it is, in fact, feasible.

These observations led us to explore the execution mode boundaries in two additional ways. The first is a brute force approach that checks for feasibility as before, but considers all combinations of the three fin parameters simultaneously. This method enables us to find the "ground truth" for the execution mode. While applicable to this relatively small problem, we emphasize that such an approach would likely be impractical on a more complex system due

to computational requirements. The second approach is to determine if we can find similar boundaries using the Mode Discovery Algorithm. Each approach is discussed in turn.

7.3.2 Simultaneous Parameter Sweep.

We performed a parameter sweep simultaneously across all morphological parameters, using the same granularity as before. Results for these simulations, 62,068 in total, are shown in Figure 7.7. Each blue dot represents a feasible scenario (and which theoretically an MFAC can handle), while each gray dot represents infeasible scenario. Collectively, the regions of blue dots define a possible execution mode for an adaptive controller. The red boxes represent the areas found by the boundary selection method, and the diagonal line in Figure 7.7 is due to a constraint in the simulation model of the flexible fin, which requires that the length must be at least three times the depth [129].

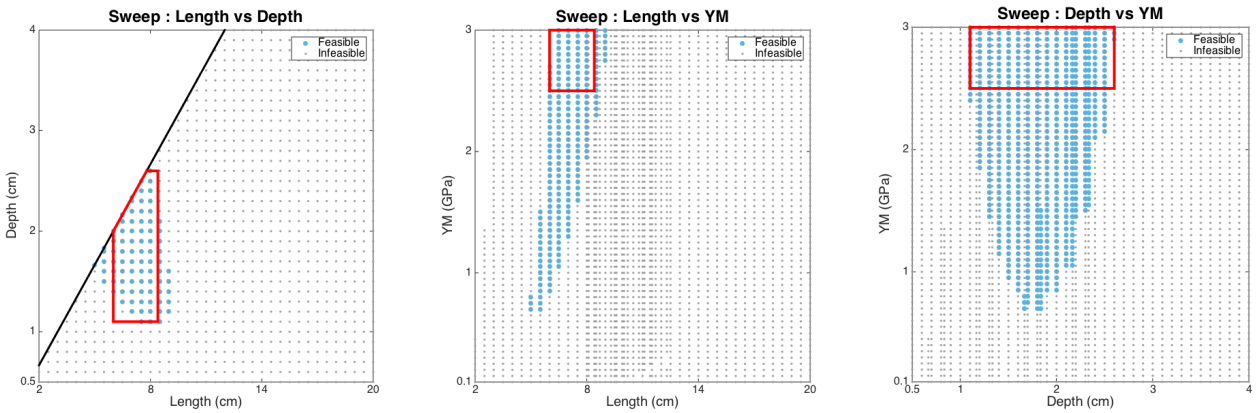


Figure 7.7: Parameter sweep plots: (a) length vs. depth; (b) length vs. flexibility (c) depth vs. flexibility. The red boxes denote the limitations of adaptability found by the boundary selection method, and the black line in (a) corresponds to the length-width limitation of the simulation model.

As shown in Figure 7.7, these simultaneous parameter sweeps reveal much larger and more complex feasible regions than the boundary experiments. However, this approach is computationally expensive and likely impractical for high-dimensional scenarios. Moreover, it does not produce an adaptive controller for this execution mode.

7.3.3 Volume-Based Scenario Selection.

Given these results, we developed a second scenario selection method that takes into account interactions among parameters. Since it is woven into the evolutionary process, it produces both the execution mode boundaries (at lower cost than sweeps) as well as an adaptive controller for that mode. Table 7.3 describes the limits for each scenario parameter. Unlike the boundary selection method, where the reference signal was always the same base pattern, here we randomize the reference signal values for each scenario, which adds an additional challenge for the adaptive controllers (i.e., they must adapt to different control requirements in addition to different caudal fin dynamics). Each evaluation takes 60s of simulation time. The time values (t_1, t_2, and t_3) shown in Figure 7.3 were produced by generating three random numbers such that $t_1 = t_{rand_1}$, $t_2 = t_1 + t_{rand_2}$, and $t_3 = t_2 + t_{rand_3}$. Each of the first three time segments is in the range [5, 25] s, and if t_3 is less than 60 seconds then the final segment includes all time remaining.

Table 7.3: Volume Scenario Limits

	Minimum	**Maximum**	**Base**
Fin Length	2.0 cm	20.0 cm	8.0 cm
Fin Depth	0.5 cm	4.0 cm	2.6 cm
Fin YM	0.1 GPa	3.0 GPa	3.0 GPa
S_i	0.0 cm/s	20.0 cm/s	15.0 cm/s
t_{rand_i}	5.0 s	25.0 s	15.0 s

As with the boundary approach, this method starts with the base scenario and adds one new scenario at the beginning of each iteration of the Mode Discovery Algorithm. However, since randomly generated scenarios are not guaranteed to be feasible we generate multiple (25) candidate scenarios per iteration. From these candidate scenarios, we choose the feasible scenario that produces the lowest fitness score when evaluated with the current best (highest fitness) MFAC controller. Essentially, we seek scenarios that are the most difficult for the adaptation process. At the end of the experiment, the set S contains 11 total scenarios. However, for comparison purposes, we cap the volume-based selection method to have a

comparable number of evaluations as the boundary selection method. So, for each iteration, if $|S|$ is greater than five, only five scenarios are randomly selected for use in evolution (each replicate experiment will have a different set $|S|$). By considering all feasible scenarios (including across replicate experiments) we can define the execution mode for the evolved controller. The 40 replicate experiments, 10 iterations, and testing of 25 random scenarios per iteration, yields 10,000 tested scenarios in total.

Figure 7.8 shows that with far fewer simulations (10K vs approximately 62K), the volume scenario method defines the same boundaries as the full parameter sweeps shown in Figure 7.7. The advantage in efficiency is expected to increase with the number of scenario parameters, which will be the case for many cyber-physical systems. In fact, parameter sweeps may be completely infeasible under some circumstances. However, generating a *good* spread of scenarios in higher dimensional space may be a complex process as has been shown by the evolutionary multiobjective community [8]. More specifically, it is desirable to have a set of scenarios that behaviorally different from one another, and thus placing an appropriate evolutionary pressure on adaptive controllers to handle the widest range of possible conditions.

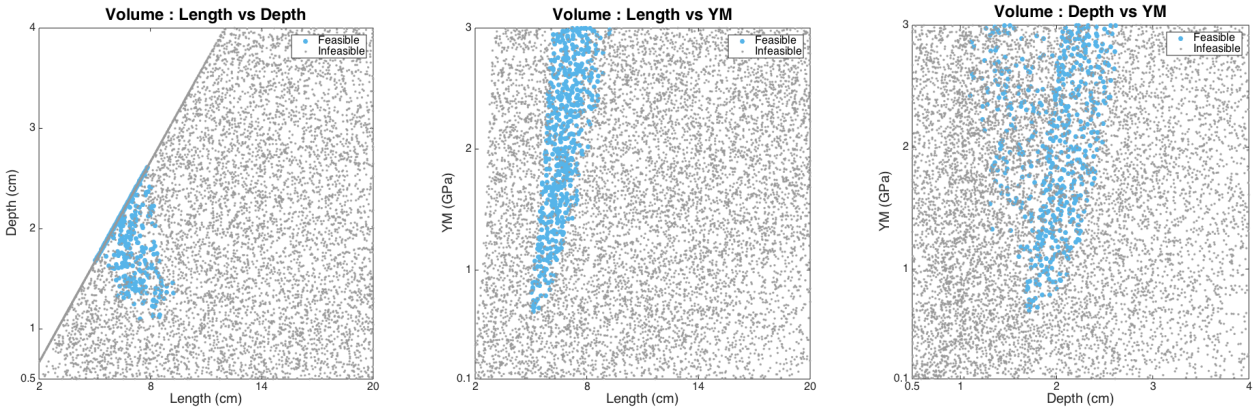


Figure 7.8: Plots for mode discovery using volume selection: (a) length vs. depth; (b) length vs. flexibility; (c) depth vs. flexibility.

The reader will notice that in Figure 7.8, some gray dots (representing infeasible scenarios) seem to fall within the discovered mode boundaries. This phenomenon is particularly evident in the depth vs. YM plot. We note that this appearance is simply a side effect of the pairwise plotting of the three parameters. Specifically those scenarios are infeasible because of the value of the third parameter (length in the case of the depth vs. YM plot). This relationship is also present in the data plotted in Figure 7.7, but does not appear in those plots because in the parameter sweeps, the values of parameters vary at regular intervals, and blue dots cover all the gray ones.

In addition to defining mode boundaries, the volume method generated effective adaptive controllers by sequentially integrating "difficult" scenarios into the evolutionary process. Figure 7.9 shows an example behavior of a simulated robotic fish that experiences damage halfway through operation. At 60 seconds, the fin length is reduced from 8.0 to 6.4 cm, depth is reduced from 2.6 to 2.1 cm, and flexibility is changed from 3.0 to 2.1 GPa. Despite these rather severe changes, the controller is able to quickly adapt and re-establish tracking.

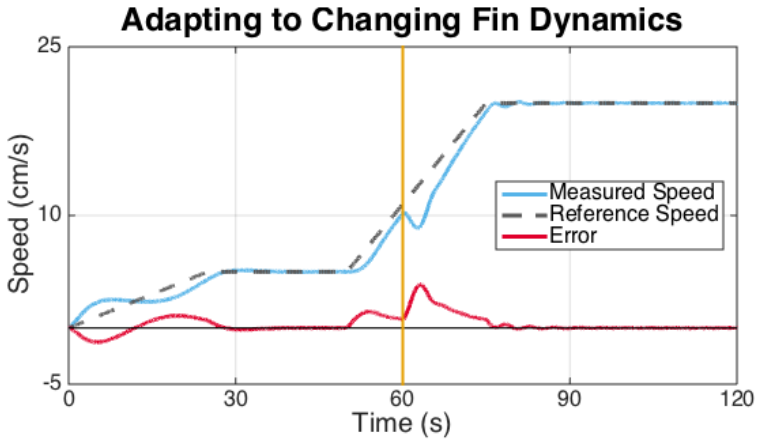


Figure 7.9: An example of an evolved MFAC adapting to sudden damage. At 60 seconds, each of the fin morphology parameters are abruptly changed.

When comparing the two scenario selection methods, we found that both lead to similar algorithm convergence rates and similar MFAC behaviors. Table 7.4 compares the fitness scores (the average mean-absolute-error recorded for each scenario used during fitness evaluation) for the overall best set of MFAC parameters from each experiment (lower values are

Table 7.4: MFAC Performance Comparison

	Boundary	**Volume**
$Base$	2.76%	2.60%
$length_{min}$	9.30%	7.63%
$length_{max}$	2.74%	2.73%
$depth_{min}$	6.23%	4.87%
$depth_{max}$	3.12%	2.92%
YM_{min}	2.98%	2.93%
$rand_{boundary}$	4.70%	4.54%
$rand_{volume}$	3.19%	3.14%

better). The $rand_{boundary}$ and $rand_{volume}$ rows represents the mean fitness score for 100 randomly generated, feasible scenarios that are within the modes found by the boundary and volume selection methods, respectively. The table shows that the volume-based scenario generation approach does an equal or better job in every test case.

7.4 Self-Modeling

The *Mode Discovery Algorithm* developed in this chapter will automatically discover the limitations of a given adaptive controller. With the ability to define execution modes for a set of controllers, a system can continue to operate effectively despite unexpected changes during runtime. As part of this process, however, the system must decide to which mode it currently belongs. Thus, the relevant question is: can the system discover an updated model for itself, at runtime, that accounts for the damage? In this section, we present an approach to achieving such runtime adaptability base on a modified version of the self-modeling process developed by Bongard et al. [12], where the system maintains an internal "mental image" of itself. If the system has an accurate simulation model of its own morphology, including sensors and actuators, it can derive compensatory behaviors dynamically using the model rather than its physical system. The Estimation-Exploration Algorithm (EEA) [12], is a general purpose self-modeling algorithm for reverse engineering complex, nonlinear systems, that has been applied to the self-repair of terrestrial robots and other applications.

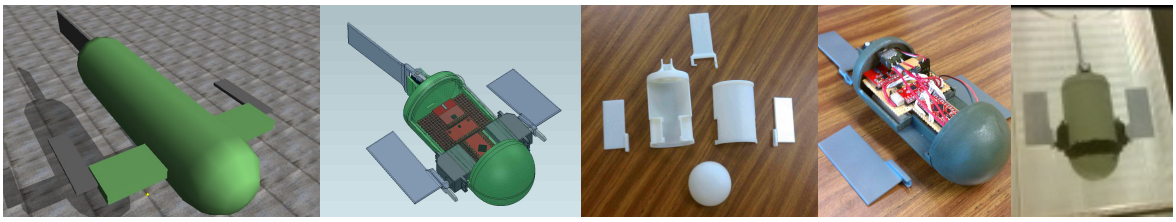


Figure 7.10: Modeling and fabrication of an aquatic robot [92]: (a) simulation model in Open Dynamics Engine (ODE); (b) corresponding SolidWorks model for fabricating prototype; (c) 3D-printed passive components of prototype; (d) integration of electronic components and battery into the prototype; (e) assembled, painted and waterproofed prototype in an elliptical flow tank. The main body of the physical prototype is 13cm long and 8cm in diameter with pectoral flippers that are 8cm long and 2cm wide.

7.4.1 The Aquatic Robot.

The device used in this section is shown in Figure 7.10. This aquatic device comprises a capsule-shaped main body, two pectoral flippers, and a single caudal fin. The main body of the physical robot contains an Arduino microcontroller board, two lithium polymer batteries, three servomotors, and a 6-axis inertial measurement unit (IMU). The IMU includes an integrated gyroscope and accelerometer, enabling the robot to compute its current position, orientation and velocity in three-dimensional space. The robot is designed to be configurable, in that plastic sleeves enable 3D-printed flippers and fins with different characteristics (size, shape, flexibility) to be swapped in and out for different experiments. The servomotors powering the flippers are a continuous rotation type that offer only a variable speed in either direction with no position feedback, a feature typical of servomotors of this size.

We chose this platform instead of a robotic fish as it has several properties that make the self-modeling process more difficult. Specifically, the lack of position information complicates the self-modeling process, since the behavior of the robot is highly coupled to its initial configuration. Similar to the work presented in Chapter 3, we implemented the simulated robot using the Open Dynamics Engine (ODE) [116]. Each simulation is parameterized to allow for changes in the robot morphology, specifically, the dimensions of each of the three

flippers can be altered (we do not consider fin flexibility in experiments described in this section).

7.4.2 Self-Modeling Algorithm.

As depicted in Figure 7.11, the EEA is a co-evolutionary process that alternates between gaining information about the target system (*exploration*) and integrating that information into its model hypothesis (*estimation*). The combination of an exploration phase and an estimation phase is referred to as a *round*. For this study these two phases are iterated until until a computation time limit has been exceeded.

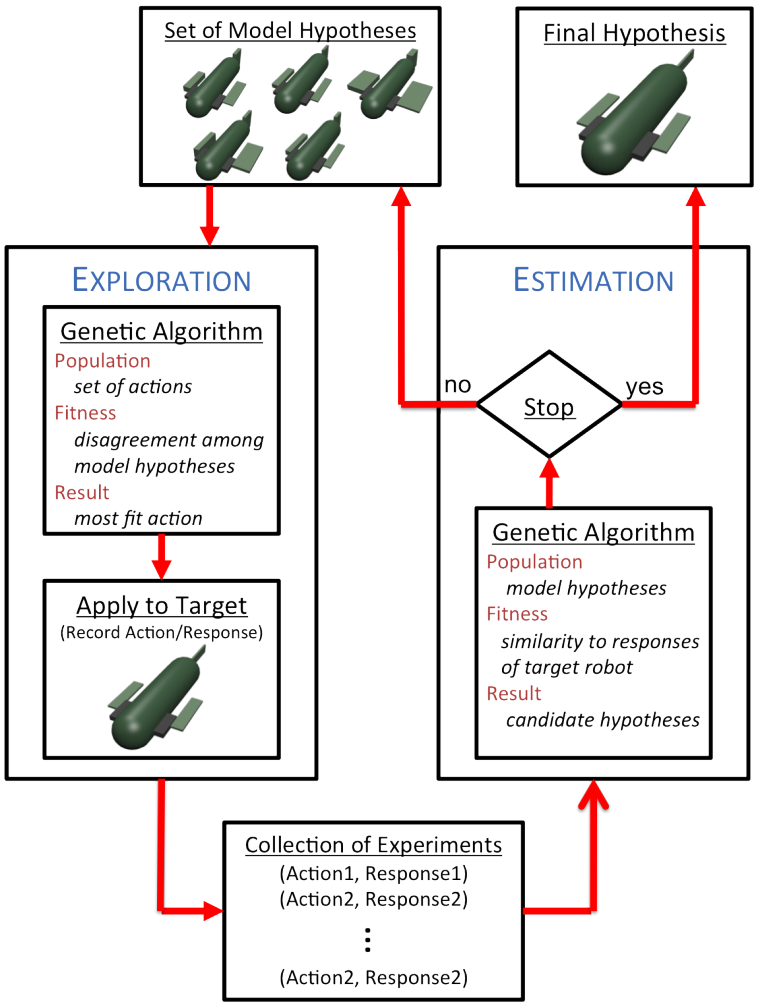


Figure 7.11: Basic operation of the Estimation-Exploration Algorithm [11].

The **Exploration Phase** is responsible for collecting behavioral information from the target system for use in the estimation phase. Specifically, the algorithm attempts to identify an action (in our case a particular movement of the robot's flippers) that provides maximal information regarding the current hypotheses of the robot morphology. A genetic algorithm is used to search for such an action; a population of individuals, each an encoding of a candidate action, executes for a predetermined number of generations. The fitness of a candidate action is based on the amount of disagreement it generates among the current model hypotheses, as discussed below. The action with the highest fitness in the final generation is selected and applied to the target system. The response from the target is collected from the system's sensors and recorded, along with the action that generated it (collectively known as an *experiment*), for use in the estimation phase.

The **Estimation Phase** is responsible for refining the hypothesis of the robot's morphology. To this end, it generates a set of candidate hypotheses that best explain the experiments performed in prior exploration phases. As in the exploration phase, a genetic algorithm is used for this search process, with candidate models encoded as binary strings. In this study, each model encoding comprises two sets of pectoral fin dimensions and one set of caudal fin dimensions. For each experiment found during the exploration phase, the corresponding action is applied to the candidate model, and the response is compared to that of the target system. The fitness of a candidate model is based on how well it reproduces the response of the target.

7.4.3 Inference Technique.

The main difficulty in applying the above methods directly is that the response of the robot to a given set of commands is highly dependent on the initial positions of the flippers. Without direct knowledge of this information, we need to infer the positions as part of the evolutionary process. To perform flipper position inference, the EEA is paused every 35 generations, and initial configurations are evolved for use during the model discovery process.

The fitness of candidate initial configurations reflects their ability to help the candidate models reproduce the target behavior. That is, a set of initial flipper positions that enable the candidate models to accurately reproduce the current experiment will receive a high fitness. For comparison, we performed three experiments: (1) an ideal baseline experiment for the unmodified EEA with *known* initial flipper/fin positions, (2) an uncertainty baseline experiment for the unmodified EEA in the presence of unknown starting positions, and (3) an inference experiment in which we employ the described inference technique.

7.4.4 Experimental Results.

Each experiment was executed as a set of 25 replicate runs, each of which was seeded with a different random number seed. The individual runs were executed as a single threaded process on late-model, commodity hardware and given 48 hours of CPU time to complete 25 rounds of the EEA process before being terminated. The orientation of the target robot was randomized between every action, reflecting what would likely be the case in a real-world situation. The genetic algorithm in the exploration phase used a population of 25 actions and executed for 25 generations. The genetic algorithm in the estimation phase used a population of 25 model hypotheses and executed for 100 generations. Both genetic algorithms used tournament selection with a tournament size of two, a crossover rate of 0.75 and a mean mutation rate of one gene per genome.

For the ideal baseline configuration, the target aquatic robot has random flipper orientations at the start of each EEA experiment, but those initial flipper positions are noted and passed back to the modeling process for use by candidate models. This case represents what the modeling process could achieve if the flipper positions were a measurable quantity (e.g., via a motor encoder) rather than a contextual uncertainty. In the second baseline, we removed the assumption of known initial flipper positions, introducing uncertainty into the modeling process. This case represents the performance of the unmodified EEA modeling process when faced with uncertainty. In the ideal baseline experiment, the EEA algorithm is

able to achieve maximum fitness across all replicate experiments, in that it produces an accurate model of the damaged robot. However, when flipper positions are unknown performance of the algorithm degrades to approximately 70% of the maximum score. With the proposed inference method fitness reached approximately 90% of the perfect score. These experiments affirm our assumption that the behavior of models, and therefore the modeling performance, is tightly coupled to the initial configuration of the aquatic robot. In Figure 7.12, we show results for the case of a damaged pectoral fin. As shown in the figure, the unmodified (b) and inference-based EEA (c) approaches arrived at very different morphologies, with the inference-based EEA result closely resembling the damaged robot (a).

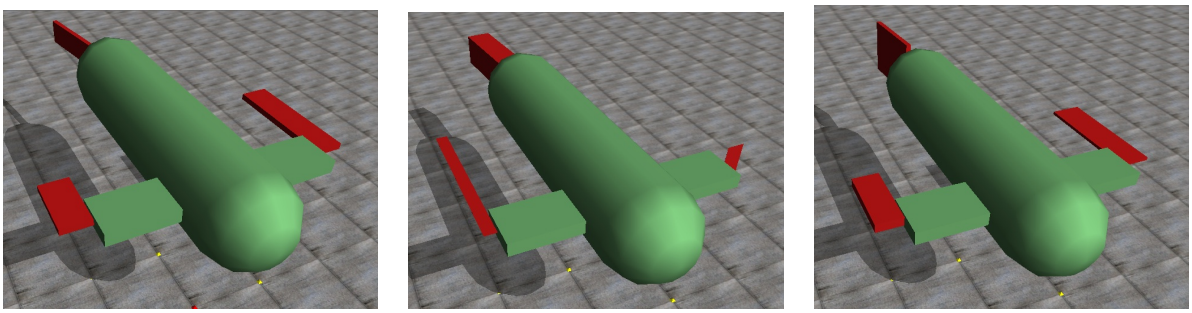


Figure 7.12: Simulated robots from the case study: (a) damaged target robot with a pectoral fin half the length of a normal one; (b) morphological model produced using the unmodified EEA approach; (c) morphological model produced by the *Inference-based* EEA approach.

Results of these experiments demonstrate the importance of the EEA approach and the need to reduce uncertainty as part of the process. While the unmodified EEA approach was unable to arrive at a good internal representation to facilitate transfer into the damaged robot, the *Inference* approach was able to build a fair internal representation to produce effective swimming in the damaged individual.

7.5 Discussion

In this chapter, we demonstrated an approach to automatically discover the boundaries of adaptability for a cyber-physical system. Specifically, this approach characterizes the range of operation (i.e., *mode*) for a robotic fish and its adaptive control software. From the software's point-of-view, the morphological properties of the robot are aspects of an uncertain environment. The approach to discovering mode boundaries involves generating scenarios uniformly within the range of the conditions that the system is expected to encounter, and then evolving an adaptive controller for those scenarios. The resulting information could aid in the design of mode-switching logic, where each mode includes an appropriate adaptive controller. Through a series of experiments, we found the volume-based scenario selection method to be both effective and computationally efficient. Although the approach was able to discover mode boundaries, for higher dimensional spaces, more strategic methods of generating and selecting scenarios might be warranted. Additionally, once the boundaries of a mode are discovered, it will be useful to find control strategies for adjacent modes. In this way, the system can have predefined controllers that are effective for multiple scenarios, including fail-safe operation (e.g., when the system is damaged it can still be controlled to return to a base station). Next, we presented results from our self-modeling research where an aquatic robot was able to discover a new model for itself. With the ability to map out modes and then decide which mode best matches sensed conditions, autonomous robotic systems will be able to both characterize damage and select/generate new behaviors (controllers).

Chapter 8

Conclusions

The overarching goal of the research presented in this dissertation is to discover a holistic approach for improving the performance, adaptability, and survivability of autonomous robotic systems. The methods and techniques developed as part of this research have used aquatic robots, primarily biomimetic robotic fish, as a test platform. These devices provided several advantages for this line of research. Specifically, they operate in difficult, nonlinear environments and require few actuators enabling, us to focus on developing general techniques rather than focusing on robot manufacturing details. Although we have discussed only aquatic robots in this document, the techniques described are applicable to a broad range of cyber-physical systems.

8.1 Offline Optimization of Morphology and Control.

Our initial work focused on improving the morphology (i.e., physical designs) and control patterns of robotic systems. One advantage of evolutionary robotics is that is can produce a coupling between body (morphology) and brain (control). In the case of robotic fish with flexible fins, our results show that it is essential for control patterns to match fin characteristics. Since the elongated fins behave as springs, improper control frequencies can actually work "against" the system, resulting in lower speeds. In fact, for a given caudal

fin morphology (dimensions and flexibility) a single optimal control frequency can be found. Furthermore, we discovered that pure sinusoids were as (if not more) effective than other possible control patterns.

8.2 Multiobjective Optimization.

In practical settings, most robotic systems must balance two or more competing objectives. For example, the primary mission of most autonomous robots is to the completion of a specific task. However, while performing the requisite actions, such systems must also be wary of depleting their power source (i.e., batteries). Thus, it is also important for these devices to perform efficiently. Intuitively, these two objectives (quickly completing tasks and power efficiency) are often at odds with one another—as speed and/or maneuverability increase efficiency is likely to decrease. This optimization process is further complicated when incorporating complex material characteristics, such as the flexibility of a robotic fish caudal fin. In this work we demonstrated that evolutionary multiobjective algorithms can provide a designer with additional information that can enhance the operation of a robotic fish. A set of rules can be derived from the Pareto front that gives insight into which parameters are most important in balancing objectives, while at the same time finding novel solutions with regards to the use of flexible materials. With this information, a final morphology can be chosen that is considered the best compromise of all possible Pareto-optimal choices. Furthermore, by examining the control parameters only (while ignoring morphological parameters) the final Pareto front, we can provide insight into how objectives can be balanced online by adjusting the control strategy.

8.3 Enhancing Adaptive Control.

Once deployed, an autonomous robot will likely experience changes due to the unavoidable wear and degradation of electromechanical components. For example, servomotor

performance will change as the device experiences different torques/loads and as it ages. Likewise, the performance characteristics of flexible materials are likely to change during their lifetime due to the nature of the fabrication process. Our 3D-printed caudal fins, for instance, will exhibit different characteristics depending on water absorption, water temperature, and whether or not they have been coated with a silicon sealant. The field of control theory has developed a set of techniques specifically targeted at alleviating these gradual changes: adaptive control. In this dissertation, we demonstrated that evolutionary algorithms can improve the performance of model-free adaptive controllers such that they are capable of handling a broader range of conditions. Specifically, subjecting adaptive controllers to a variety of conditions during evolution produced controllers capable of adapting to conditions even beyond those encountered during evolution.

8.4 Mode Discovery.

Of course, even an adaptive controller has limits. When an adaptive controller begins to fail due to drastic changes to the system or its environment, a robotic device may become unable to complete its task. However, this situation does not mean that the device itself cannot continue to operate. In fact, it may be possible for the system to switch to a new adaptive controller, and carry on, albeit in a lower-performance manner. Our final area of research addressed the discovery of the limits on a given adaptive controller. The approach to discovering mode boundaries involves generating scenarios uniformly at random within the range of the conditions that the system is expected to encounter, and then evolving an adaptive controller for those scenarios. With this knowledge regarding controller limitations, we can define execution modes, where each mode is associated with a different adaptive controller. The system can then switch to an appropriate controller based on the discovered execution modes. The switching process can be facilitated by self-modeling to determine which execution mode is appropriate. More specifically, when a robotic device suffers damage

it can precisely update its own self-model to reflect the damage, and then switch to the correct controller. Given a new model, the system can do one of two things: (1) select a new controller based on a database of modes and their accompanying controllers, or (2) generate a new behavior using an on-board evolutionary algorithm. In either case, the system will be able to continue operation despite significant damage.

8.5 Potential for Future Investigations.

The results presented in this dissertation provide a foundation for several potential research projects. The first area is the on-board application of EMO Pareto front data. In our studies (Chapters 4 and 5), once the evolutionary experiments have completed we select a single Pareto-optimal solution and discard the others. However, it might be helpful to retain the additional information and develop an algorithm that operates online to automatically adjust the relative importance of multiple objectives depending on the situation. For example, under certain conditions it may be more useful to focus on speed (such as when a robot is near a recharging station), and at other times it will be important to be more energy-efficient.

Second, we note that the MFAC controllers presented in this work are designed to handle a single-input, single-output (SISO) system. However, more complex MFAC controllers have been designed to accommodate multiple inputs and multiple outputs. Thus, a potential line of investigation is to extend the approach presented in this dissertation to include tracking multiple reference signals simultaneously. For example, our current work could be extended such that a robotic fish might track both a desired reference speed and a desired reference heading. Resulting controllers would be capable of more complex behaviors involving both turning and forward locomotion. Increasing the number of controller inputs and outputs makes the optimization process significantly more challenging because, as we have shown in

Chapter 3, there exist trade-offs between achieving the maximum possible speed and high maneuverability.

Another potential area of interest, is to investigate how increasingly complex tasks affect the proposed design process. Currently, our evaluation functions consider only simple tasks (i.e., swimming speed and efficiency). A more complex behavior might include way-point following or station-keeping. We anticipate that with increased task complexity the proposed methods will provide an an even greater value when compared to more traditional optimization methods. Indeed, evolutionary algorithms are known for discovering novel, and sometime unintuitive, solutions to difficult problems.

Finally, for this dissertation we relied solely on aquatic robots. It will be important to apply the similar experiments to different categories of autonomous robots. Ground vehicles and unmanned aerial vehicles share many dynamics with aquatic robots, however, they present their own challenges. For example, ground vehicles are more likely to encounter obstacles while carrying out a task, and aerial vehicles require greater care with respect to safety. These new domains will necessitate extending our work to incorporate new algorithms for collision avoidance and new controller verification techniques in order to ensure that evolved controllers avoid dangerous failures. We anticipate that the methods described in this dissertation will be widely applicable to such devices and many other cyber-physical systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Awards for human-competitive results produced by genetic and evolutionary computation. Competition held as part of the annual Genetic and Evolutionary Computation Conference (GECCO), sponsored by ACM SIGEVO. Results available at http://www.human-competitive.org.

[2] *Dynamics and Control of Turning Maneuver for Biomimetic Robotic Fish*. IEEE, October 2006.

[3] Xu Aidong, Zheng Yangbo, Song Yan, and Liu Mingzhe. An improved model free adaptive control algorithm. In *Proceedings of the Fifth International Conference on Natural Computation (ICNC)*, volume 1, pages 70–74. IEEE, 2009.

[4] Alessio Alessi, Angelo Sudano, Dino Accoto, and Eugenio Guglielmelli. Development of an autonomous robotic fish. In *Proceedings of the 2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1032–1037. IEEE, 2012.

[5] P. V. Alvarado and K. Youcef-Toumi. Design of machines with compliant bodies for biomimetic locomotion in liquid environments. *Journal of Dynamic Systems, Measurement, and Control*, 128:3–13, 2006.

[6] J. M. Anderson and N. K. Chhabra. Maneuvering and stability performance of a robotic tuna. *Integr. Comp. Biol.*, 42:118–126, 2002.

[7] Jared M. Moore Betty H. C. Cheng Anthony J. Clark, Byron DeVries and Philip K. McKinley. (submitted) automated discovery of adaptability boundaries through computational evolution. In *10th IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, 2016.

[8] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103, 2012.

[9] Atılım Güneş Baydin. Evolution of central pattern generators for the control of a five-link bipedal walking mechanism. *Paladyn, Journal of Behavioral Robotics*, 3(1):45–53, 2012.

[10] Josh Bongard and Hod Lipson. Evolved machines shed light on robustness and resilience. *Proceedings of the IEEE*, 102(5):899–914, May 2014.

[11] Josh Bongard, Victor Zykov, and Hod Lipson. Automated synthesis of body schema using multiple sensor modalities. In *In Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems (ALIFEX*, pages 220–226, 2006.

[12] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[13] Josh C Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[14] Josh C Bongard and Hod Lipson. Once more unto the breach: Co-evolving a robot and its simulator. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, pages 57–62, 2004.

[15] D.J. Braun, F. Petit, F. Huber, S. Haddadin, P. van der Smagt, A. Albu-Schaffer, and S. Vijayakumar. Robots driven by compliant actuators: Optimal control under actuation constraints. *IEEE Transactions on Robotics*, PP(99):1–17, 2013.

[16] Nicolas Bredeche, Evert Haasdijk, and AE Eiben. On-line, on-board evolution of robot controllers. In *Artifical Evolution*, pages 110–121. Springer, 2010.

[17] Rodney A. Brooks. Artificial life and real robots. In *Proceedings of the First European Conference on Artificial Life*, pages 3–10. MIT Press, Cambridge, MA, 1992.

[18] Z. Chen, S. Shatara, and X. Tan. Modeling of biomimetic robotic fish propelled by an ionic polymer-metal composite caudal fin. *IEEE/ASME Transactions on Mechatronics*, 15(3):448–459, 2010.

[19] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 167–174, Amsterdam, The Netherlands, 2013. ACM.

[20] George S Cheng. Model-free adaptive (MFA) control. *Computing and Control Engineering*, 15(3):28–33, 2004.

[21] Hillel J. Chiel and Randall D. Beer. The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*, 20(12):553 – 557, 1997.

[22] Project Chrono. Chrono::Engine, http://www.chronoengine.info/, 2014.

[23] Anthony J. Clark and Philip K. McKinley. Evolutionary optimization of robotic fish control and morphology. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 21–22, New York, NY, USA, 2013. ACM.

[24] Anthony J. Clark, Philip K. McKinley, and Xiaobo Tan. On-board evolution of a model-free adaptive controller for a robotic fish. In *Proceedings of Evolution of Physical Systems Workshop, held in conjunction with the 14th International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, New York, New York, July 2014.

[25] Anthony J. Clark, Philip K. McKinley, and Xiaobo Tan. Enhancing a model-free adaptive controller through evolutionary computation. Technical report, Michigan State University, New York, NY, USA, July 2015.

[26] Anthony J. Clark, Jared M. Moore, Jianxun Wang, Xiaobo Tan, and Philip K. McKinley. Evolutionary design and experimental validation of a flexible caudal fin for robotic fish. In *Proceedings of the Thirteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 325–332, East Lansing, Michigan, USA, July 2012. Best Paper Award.

[27] Anthony J. Clark, Xiaobo Tan, and Philip K. McKinley. Evolutionary multiobjective design of a flexible caudal fin for robotic fish. *Bioinspiration & Biomimetics, special issue on Bioinspired Soft Robotics*, 10(6):065006, November 2015.

[28] Anthony J. Clark, Jianxun Wang, Xiaobo Tan, and Philip K. McKinley. Balancing performance and efficiency in a robotic fish with evolutionary multiobjective optimization. In *Proceedings of the IEEE International Conference on Evolvable Systems, held in conjunction with the 2014 IEEE Symposium on Computational Intelligence (SSCI)*, pages 227–234, Orlando, Florida, Dec 2014.

[29] Jeff Clune, Benjamin E Beckmann, Charles Ofria, and Robert T Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2764–2771, Trondheim, Norway, 2009. IEEE.

[30] L. S. Coelho, Marcelo Wicthoff Pessôa, Rodrigo Rodrigues Sumar, and Antonio Augusto Rodrigues Coelho. Model-free adaptive control design using evolutionary-neural compensator. *Expert Systems with Applications*, 37(1):499–508, 2010.

[31] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002.

[32] Erwin Coumans. Bullet Physics Library, http://www.bulletphysics.org/, 2014.

[33] Alessandro Crespi, Daisy Lachat, Ariane Pasquier, and Auke Jan Ijspeert. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*, 25(1-2):3–13, December 2007.

[34] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[35] Kenneth A. De Jong. *Evolutionary Computation: A Unified Approach.* MIT Press, 2002.

[36] Kenneth A De Jong. *Evolutionary computation: a unified approach.* MIT press, 2006.

[37] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[38] F. C. Dyer and H. J. Brockmann. Biology of the Umwelt. In L. D. Houck and L. C. Drickamer, editors, *Foundations of Animal Behavior*, pages 529–538. University of Chicago Press, 1996.

[39] Matthias Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research*, 147(1):343–360, 2006.

[40] Hadi El Daou, Taavi Salumäe, Lily D Chambers, William M Megill, and Maarja Kruusmaa. Modelling of a biologically inspired robotic fish driven by compliant parts. *Bioinspiration & Biomimetics*, 9(1):016010, 2014.

[41] M. Epstein, J. E. Colgate, and M. A. MacIver. Generating thrust with a biologically-inspired robotic ribbon fin. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2412–2417, Beijing, China, 2006.

[42] Christopher J Esposito, James L Tangorra, Brooke E Flammang, and George V Lauder. A robotic fish caudal fin: effects of stiffness and motor program on locomotor performance. *The Journal of experimental biology*, 215(1):56–67, 2012.

[43] Jolyon J. Faria, John R. G. Dyer, Romain O. Clément, Iain D. Couzin, Natalie Holt, Ashley J. W. Ward, Dean Waters, and Jens Krause. A novel method for investigating the collective behaviour of fish: introducing 'robofish'. *Behavioral Ecology and Sociobiology*, 64(8):1211–1218, 2010.

[44] Kara L Feilich and George V Lauder. Passive mechanical models of fish caudal fins: effects of shape and stiffness on self-propulsion. *Bioinspiration & Biomimetics*, 10(3):036002, 2015.

[45] Dario Floreano, Phil Husbands, and Stefano Nolfi. Evolutionary Robotics. In *Handbook of Robotics*. Springer Verlag, Berlin, 2008.

[46] Dario Floreano and Francesco Mondada. Evolutionary neurocontrollers for autonomous mobile robots. *Neural networks*, 11(7):1461–1478, 1998.

[47] Dario Floreano and Joseba Urzelai. Evolution of plastic control networks. *Autonomous robots*, 11(3):311–317, 2001.

[48] Stephanie Forrest, ThanhVu Nguyen, Westley Weimer, and Claire Le Goues. A genetic programming approach to automated software repair. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 947–954. ACM, 2009.

[49] Minyue Fu and B. Barmish. Adaptive stabilization of linear systems via switching control. *IEEE Transactions on Automatic Control*, 31(12):1097–1103, Dec 1986.

[50] David Edward Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[51] Faustino J Gomez and Risto Miikkulainen. Active guidance for a finless rocket using neuroevolution. In *Genetic and Evolutionary Computation—GECCO 2003*, pages 2084–2095, San Francisco, 2003. Springer, Morgan Kaufmann.

[52] Simon Haykin. *Neural networks and learning machines*. Prentice-Hall, 2009.

[53] João Hespanha, Daniel Liberzon, A. Stephen Morse, Brian D. O. Anderson, Thomas S. Brinsmead, and Franky De Bruyne. Multiple model adaptive control. part 2: switching. *International Journal of Robust and Nonlinear Control*, 11(5):479–496, 2001.

[54] João P. Hespanha, Daniel Liberzon, and A.Stephen Morse. Overcoming the limitations of adaptive control by means of logic-based switching. *Systems & Control Letters*, 49(1):49 – 65, 2003. Adaptive Control.

[55] João P. Hespanha and A.Stephen Morse. Switching between stabilizing controllers. *Automatica*, 38(11):1905 – 1917, 2002.

[56] Jonathan D. Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2012.

[57] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[58] Gregory S Hornby, Jordan B Pollack, et al. Body-brain co-evolution using l-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 868–875, 2001.

[59] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3 – 35, 2013.

[60] Zhongsheng Hou and Wenhu Huang. The model-free learning adaptive control of a class of SISO nonlinear systems. In *Proceedings of the 1997 American Control Conference*, volume 1, pages 343–344, 1997.

[61] Franz S. Hover, Ryan M. Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J. Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *International Journal of Robotics Research*, 31(12):1445–1464, 2012.

[62] Qingsong Hu, D.R. Hedgepeth, Lihong Xu, and Xiaobo Tan. A framework for modeling steady turning of robotic fish. In *IEEE International Conference on Robotics and Automation (IRCA)*, pages 2669 –2674, may 2009.

[63] P. Ioannou and B. Fidan. *Adaptive Control Tutorial*. Advances in Design and Control. Society for Industrial and Applied Mathematics, 2006.

[64] Petros A Ioannou and Jing Sun. *Robust Adaptive Control*. Dover Books on Electrical Engineering. Dover Publications, 2012.

[65] Julio Jerez and Alain Suero. Newton Game Dynamics, http://newtondynamics.com/forum/newton.php, 2012.

[66] S. Jiang, J. Zhang, Y.-S. Ong, A.N. Zhang, and P.S. Tan. A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm. *IEEE Transactions on Cybernetics*, PP(99):1–1, 2014.

[67] Rahul Kala. Multi-robot path planning using co-evolutionary genetic programming. *Expert Systems with Applications*, 39(3):3817–3831, 2012.

[68] Igor Karoń. Modified model-free adaptive controller for a nonlinear rotor system. In *Artificial Intelligence and Soft Computing*, pages 458–465. Springer, 2012.

[69] N. Kato. Control performance in the horizontal plane of a fish robot with mechanical pectoral fins. *IEEE Journal of Oceanic Engineering*, 25(1):121–129, 2000.

[70] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in Biotechnology*, 31(5):287–294, 2013.

[71] Eric Klavins, Haldun Komsuoglu, and J Robert. The Role of Reflexes versus Central Pattern Generators in Dynamical Legged Locomotion. *for biomimetic robots*, 2002.

[72] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

[73] Sylvain Koos, Antoine Cully, and Jean-Baptiste Mouret. Fast damage recovery in robotics with the t-resilience algorithm. *The International Journal of Robotics Research*, 32(14):1700–1723, 2013.

[74] Sylvain Koos, J Mouret, and Stéphane Doncieux. Automatic system identification based on coevolution of models and tests. In *IEEE Congress on Evolutionary Computation.*, pages 560–567. IEEE, 2009.

[75] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 119–126. ACM, 2010.

[76] John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Je ssen Yu, and Guido Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Genetic Programming. Springer, 1st edition, 2005.

[77] John R. Koza and James P. Rice. Automatic programming of robots using genetic programming. In *AAAI*, pages 194–207, 1992.

[78] G. V. Lauder, E. J. Anderson, J. Tangorra, and P. G. A. Madden. Fish biorobotics: Kinematics and hydrodynamics of self-propulsion. *Journal of Experimental Biology*, 210:2767–2780, 2007.

[79] Ljung Lennart. System identification: theory for the user, 1999.

[80] M. J. Lighthill. Large-Amplitude Elongated-Body Theory of Fish Locomotion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 179(1055):125–138, November 1971.

[81] Hod Lipson. Evolutionary robotics and open-ended design automation. In Bar Cohen, editor, *Biomimetics*, pages 129–155. CRC Press, 2005.

[82] Hod Lipson. Challenges and opportunities for design, simulation, and fabrication of soft robots. *Soft Robotics*, 1(1):21–27, 2014.

[83] Hod Lipson and Josh Bongard. An exploration-estimation algorithm for synthesis and analysis of engineering systems using minimal physical testing. In *Proceedings of the 2004 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2004)*, pages 1087–1093, Salt Lake City, Utah, 2004.

[84] J. H. Long, Jr., T. J. Koob, K. Irving, K. Combie, V. Engel, H. Livingston, A. Lammert, and J. Schumacher. Biomimetic evolutionary analysis: Testing the adaptive value of vertebrate tail stiffness in autonomous swimming robots. *Journal of Experimental Biology*, 209(23):4732–4746, 2006.

[85] K. H. Low and C. W. Chong. Parametric study of the swimming performance of a fish robot propelled by a flexible caudal fin. *Bioinspiration & Biomimetics*, 5(4):046002, 2010.

[86] Carmel Majidi. Soft robotics: A perspective – current trends and prospects for the future. *Soft Robotics*, 1:5–11, July 2013.

[87] Richard Mason and Joel W Burdick. Experiments in carangiform robotic fish locomotion. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 428–435. IEEE, 2000.

[88] K Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 376:367–376, 1985.

[89] Kiyotoshi Matsuoka. Analysis of a neural oscillator. *Biological Cybernetics*, 104(4-5):297–304, May 2011.

[90] Craig Mautner and Richard Belew. Evolving robot morphology and control. *Artificial Life and Robotics*, 4:130–136, 2000.

[91] Orazio Miglino, Henrik Hautop Lund, and Stefano Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1996.

[92] Jared M. Moore and Philip K. McKinley. Evolution of an amphibious robot with passive joints. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pages 1443–1450, Cancun, Mexico, 2013.

[93] Jared M. Moore and Philip K. McKinley. Investigating modular coupling of morphology and control with digital muscles. In *Proceedings of the 14th International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, pages 148–155, New York, New York, July 2014.

[94] Jean-Baptiste Mouret and Paul Tonelli. *Artificial Evolution of Plastic Neural Networks: A Few Key Concepts*, pages 251–261. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[95] K. S. Narendra, J. Balakrishnan, and M. K. Ciliz. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems*, 15(3):37–51, Jun 1995.

[96] Kumpati S Narendra. *Applications of adaptive control*. Elsevier, 2012.

[97] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[98] Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The Biology, Intelligence, and Technology of Self-organizing Machines*, volume 150. MIT press, Cambridge, MA, USA, 2000.

[99] Takuya Okada, Shuxiang Guo, Nan Xiao, Fu Qiang, and Yasuhiro Yamauchi. Control of the wireless microrobot with multi-dofs locomotion for medical applications. In *Proceedings of the 2012 International Conference on Mechatronics and Automation (ICMA)*, pages 2405–2410. IEEE, 2012.

[100] Nikhil Padhye, Piyush Bhardawaj, and Kalyanmoy Deb. Improving differential evolution through a unified approach. *Journal of Global Optimization*, pages 1–29, 2013.

[101] Daniel Paluska and Hugh Herr. The effect of series elasticity on actuator power and work output: Implications for robotic and prosthetic joint design. *Robotics and Autonomous Systems*, 54(8):667–673, 2006.

[102] Y-J Park, Useok Jeong, Jeongsu Lee, S-R Kwon, H-Y Kim, and K-J Cho. Kinematic condition for maximizing the thrust of a robotic fish using a compliant caudal fin. *IEEE Transactions on Robotics*, 28:1216–1227, 2012.

[103] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. Self-Organization, Embodiment, and Biologically Inspired Robotics. *Science*, 318(5853):1088–1093, 2007.

[104] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. The challenges ahead for bio-inspired 'soft' robotics. *Communications of the ACM*, 55(11):76–87, 2012.

[105] P. Phamduy, G. Polverino, R. C. Fuller, and M. Porfiri. Fish and robot dancing together: bluefin killifish females respond differently to the courtship of a robot with varying color morphs. *Bioinspiration & Biomimetics*, 9(3):036021, 2014.

[106] Jordan B Pollack, Hod Lipson, Sevan Ficici, Pablo Funes, Greg Hornby, and Richard A Watson. Evolutionary techniques in physical robotics. In *Evolvable Systems: from biology to hardware*, pages 175–186. Springer, 2000.

[107] Torsten Reil and Phil Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168, 2002.

[108] Charles Richter and Hod Lipson. Untethered hovering flapping flight of a 3d-printed mechanical insect. *Artificial life*, 17(2):73–86, 2011.

[109] John Rieffel, Davis Knox, Schuyler Smith, and Barry Trimmer. Growing and evolving soft robots. *Artificial Life*, 20(1):143–162, 2014.

[110] Matthew J. Rose, Anthony J. Clark, Jared M. Moore, and Philip K. McKinley. Just keep swimming: Accounting for uncertainty in self-modeling aquatic robots. In *Proceedings of the 6th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems*, Taormina, Italy, September 2013. Best Paper Award.

[111] Wilson J. Rugh and Jeff S. Shamma. Research on gain scheduling. *Automatica*, 36(10):1401 – 1425, 2000.

[112] Haitham Seada and Kalyanmoy Deb. U-NSGA-III: A unified evolutionary optimization procedure for single, multiple, and many objectives proof-of-principle results. In *Proceedings of Eighth Conference on Evolutionary Multi-Criterion Optimization (EMO-2015)*, pages 34–49. Springer, 2015.

[113] Bu Hyun Shin, Kyung-Min Lee, and Seung-Yop Lee. A miniaturized tadpole robot using an electromagnetic oscillatory actuator. *Journal of Bionic Engineering*, 12(1):29–36, 2015.

[114] Karl Sims. Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[115] Simulink: Dynamic system simulation for MATLAB, User's Guide. The MathWorks Inc., Natick, Massachusetts, USA, 1990-2013.

[116] Russell Smith. Open Dynamics Engine, http://www.ode.org/, 2014.

[117] Lee Spector and Jon Klein. Trivial geography in genetic programming. *Genetic programming theory and practice III*, 2006.

[118] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[119] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[120] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[121] Yogo Takada, Ryosuke Araki, Yukinobu Nakanishi, Motohiro Nonogaki, Kazuaki Ebita, and Tomoyuki Wakisaka. Development of small fish robots powered by small and ultra-light passive-type polymer electrolyte fuel cells. *Journal of Robotics and Mechatronics*, 22(2):150, 2010.

[122] Yogo Takada, Keisuke Koyama, and Takahiro Usami. Position estimation of small robotic fish based on camera information and gyro sensors. *Robotics*, 3(2):149–162, 2014.

[123] Xiaobo Tan. Autonomous robotic fish as mobile sensor platforms: Challenges and potential solutions. *Marine Technology Society Journal*, 45(4):31–40, 2011.

[124] Paul Tonelli and Jean-Baptiste Mouret. On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PloS ONE*, 8(11), 2013.

[125] Michael S. Triantafyllou and George S. Triantafyllou. An efficient swimming machine. *Scientific American*, 272:64, 1995.

[126] R. Van Ham, T. G. Sugar, B. Vanderborght, K. W. Hollander, and D. Lefeber. Compliant actuator designs. *IEEE Robotics & Automation Magazine*, pages 81–94, September 2009.

[127] Barthélémy von Haller, Auke Ijspeert, and Dario Floreano. Co-evolution of structures and controllers for neubot underwater modular robots. In *Advances in Artificial Life*, volume 3630 of *Lecture Notes in Computer Science*, pages 189–199. Springer Berlin Heidelberg, 2005.

[128] J. Wang, P. K. McKinley, and X. Tan. Dynamic modeling of robotic fish with a flexible caudal fin. In *Proceedings of the ASME 2012 5th Annual Dynamic Systems and Control Conference, joint with the JSME 2012 11th Motion and Vibration Conference, Ft. Lauderdale, Florida, USA*, Ft. Lauderdale, Florida, USA, October 2012. Paper DSCC2012-MOVIC2012-8695.

[129] Jianxun Wang, Philip K. McKinley, and Xiaobo Tan. Dynamic modeling of robotic fish with a base-actuated flexible tail. *Journal of Dynamic Systems, Measurement and Control*, 137(1), August 2014.

[130] Yu Wang, Rui Tan, Guoliang Xing, Jianxun Wang, and Xiaobo Tan. Accuracy-aware aquatic diffusion process profiling using robotic sensor networks. In *Proceedings of the 11th ACM/IEEE Conference on Information Processing in Sensor Networks*, pages 281–292, Beijing, China, 2012.

[131] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[132] Junzhi Yu, Rui Ding, Qinghai Yang, Min Tan, Weibing Wang, and Jianwei Zhang. On a bio-inspired amphibious robot capable of multimodal motion. *IEEE/ASME Transactions on Mechatronics*, PP(99):1–10, 2011.

[133] Junzhi Yu and Lizhong Liu. Dynamic modeling and experimental validation of biomimetic robotic fish. In *Proceedings of the 2006 American Control Conference*, pages 4129–4134, 2006.

[134] Juan Cristóbal Zagal and Javier Ruiz-Del-Solar. Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems*, 50(1):19–39, 2007.

[135] Ji Zhang and Betty H. C. Cheng. Model-based develpment of dynamically adaptive software. In *IEEE International Conference on Software Engineering (ICSE06)*, Shanghai, China, May 2006. IEEE. Distinguished Paper Award.

[136] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[137] Chunlin Zhou, C.W. Chong, Yu Zhong, and K.H. Low. Robust gait control for steady swimming of a carangiform fish robot. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 100–105, July 2009.

[138] Chunlin Zhou and K. H. Low. Locomotion planning of biomimetic robotic fish with multi-joint actuation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2132–2137, 2009.

[139] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer, 2004.

[140] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland, 2001.