Enhancing a Model-Free Adaptive Controller through Evolutionary Computation

Anthony J. Clark
Computer Science and Engineering
Michigan State University
East Lansing, MI, USA
ajc@msu.edu

Philip K. McKinley
Computer Science and Engineering
Michigan State University
East Lansing, MI, USA
mckinle3@msu.edu

Xiaobo Tan
Electrical and Computer Engineering
Michigan State University
East Lansing, MI, USA
xbtan@egr.msu.edu

ABSTRACT
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 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 paper we investigate how to enhance the online adaptability of MFAC-based systems through computational evolution. We apply the proposed methods to a simulated robotic fish propelled by a flexible caudal fin. Results demonstrate that the robot is able to effectively respond to changing fin characteristics and varying control signals when using an evolved MFAC controller. Notably, the system is able to adapt to characteristics not encountered during evolution. The proposed technique is general and can be applied to improve the adaptability of other cyber-physical systems.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—Control theory

Keywords
adaptive control; model-free control; robotic fish; flexible materials; differential evolution

1. INTRODUCTION
Increasingly, robots are being deployed in complex and uncertain environments, where reaction and physical agility are essential to the completion of tasks. The adaptability and robustness exhibited by natural organisms has led to many bio-inspired approaches to robot design. For example, integration of soft and/or flexible materials into the morphology of a robot can partially compensate for actuation capabilities that are primitive relative to those of biological organisms [10]. Examples include the “whipping” action of a flexible fin on a robotic fish [3], shock absorption by flexible joints on a legged robot [19], and thrust generation by flexible wings on a robotic flying insect [23]. However, integrating flexible materials into a robot poses numerous challenges to controlling the system, since the flexibility of a structure affects the resulting forces and torques experienced during interactions with the environment [8]. Moreover, the properties of such components are likely to change over time due to degradation of materials and changing environmental conditions such as temperature. To produce effective locomotion, on-board controllers must account for these uncertainties.

One promising approach is model-free adaptive control (MFAC) [2, 20], intended for “gray box” situations where only partial, and possibly inaccurate, information is known. Like traditional adaptive control [6], this method attempts to minimize the error between desired and actual outcomes. However, instead of requiring a precise model of the system, an MFAC controller learns how to control a device by continually updating the link weights of an artificial neural network (ANN). Moreover, by saving recent error signals and using them as additional inputs to the ANN, MFAC controllers take advantage of state information, or so-called neural network memory. Although this is a general purpose control paradigm, the values of several parameters, as well as the structure of the ANN, need to be specified a priori, potentially limiting adaptability of the system. This raises the following questions. Is it possible to optimize an MFAC controller in order to enhance adaptability after deployment? And how can this be done such that the controller can account for changes in the robot morphology?

In this study we investigate the coupling of evolutionary computation with model-free adaptive control. Specifically, the differential evolution (DE) algorithm [14] is used to find good MFAC controller parameters that enable a robot to adapt during its lifetime to changes such as mechanical wear and material degradation. We apply this method to a simulated robotic fish that swims by means of a flexible caudal fin; an example of a target robot is shown in Figure 1. Equipping a robotic fish with a flexible caudal fin has been shown to improve efficiency with respect to thrust and power [3, 12]. However, the increase in performance introduces difficulty in controlling the system, particularly in a complex and highly nonlinear aquatic environment. Here, the ob-
Adaptive control is a well-established field of study for dealing with uncertainty in cyber-physical systems [6]. For example, in model reference adaptive control (MRAC), a reference model defines the desired response of the system and control laws are designed such that the controlled system is forced to behave as this reference model. However, reliance on a model of the target system makes it difficult to accommodate unexpected conditions for which the underlying reference model does not apply. In addition, the complexity associated with many physical systems, such as robots with flexible components, often renders the design of a model-based controller intractable or inconvenient.

Hou and Huang [4] first proposed the idea of model-free adaptive control (MFAC), based on the concept of a pseudo-partial-derivative and reliance on only system inputs and outputs. The MFAC approach we use in this study, proposed by Cheng [2], combines a traditional proportional controller with an adaptive ANN, as shown in Figure 2. As an input, the ANN receives a continuous error signal $e$, calculated as the difference between a desired reference input and the actual state of the robot (see Figure 3). The error signal is discretized at a sampling rate $T_s$, and then normalized between -1 and 1 using an error bound $e_0$. This normalized, discretized error signal, denoted $E$, is passed to the first input neuron, $I_1$, and then propagated to each subsequent input neuron at successive discrete sampling times. This process is repeated such that the $N$ input neurons store the $N$ most recent error signals $E$. By storing these values and using them as inputs to the ANN, MFAC controllers take advantage of state information. Additionally, at each sampling time, the input neuron values ($E_1...E_N$) are fed forward to the ANN hidden neurons ($H_1...H_N$) which in turn feed their values to the output neuron ($V$). The final output of the controller, $u$, is the sum of the value from a single output neuron and the current error signal, amplified by the controller gain $K_c$. Adding the current error signal to the network output improves the responsiveness of the controller.

Specifically, any change in error will result in an immediate change in the controller’s output. Hidden and output neurons are activated with a sigmoidal activation function. The weights of links connecting the input layer to the hidden layer ($w_{ij}$) and from the hidden layer to the output neuron ($h_j$) are updated at each sample time. Learning rules were derived by minimizing the error signal [2]: specifically, the partial derivative of an objective function, based on the error signal, is taken with respect to the link weights. Formally, these rules are described by Equations 1 and 2:

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

$$\Delta h_j(n) = \eta K_c e(n) q_j(n), \quad (2)$$

where $(n)$ denotes the sample time and $q_j$ refers to the output of the $j$th hidden neuron. The number of recent error signals, $N$, as well as the rate at which link weights are updated (called the adaptive learning rate, $\eta$), are configurable.
such that by filtering and integrating accelerometer data. Generally, and the output of the robotic fish to the robotic fish. For this study, reference signal MFAC for Robotic Fish.

For example, a plastic ANN may be used to dynamically the adaptive controller is regulating a desired control signal. Specifically, a well-known method for the adaptive control process discussed in this paper from other forms of adaptive control. Particularly if the fin is fabricated from flexible materials.

The magnitude of an MFAC controller output is adjustable through the controller gain value \( K_c \).

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°.

The magnitude of an MFAC controller output is adjustable through the controller gain value \( K_c \).

The controller’s objective is to produce a control signal \( u \) such that \( y \) closely tracks \( r \). That is, an effective controller...
created around the previous generation's best individual, as DE/best/2/bin, where best signifies that all children are made from the previous best individual. Therefore, mutation is based on two individuals, and bin denotes that mutation is based on two individuals, and bin refers to a binary crossover operation.

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 \(\eta\)). 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 \(\dot{r}\) and \(\ddot{y}\)). 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 all forces acting on the fin segments.

Differential evolution.

To enhance the adaptability of MFAC controllers we apply differential evolution (DE), a global optimization algorithm developed by Storn et al. [14]. DE was chosen because studies have shown that it will converge faster than real-valued genetic algorithms for problems similar to ours [11].

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 \(\eta\)). 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 \(\dot{r}\) and \(\ddot{y}\)). 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 all forces acting on the fin segments.

The robotic fish prototype upon which this simulation is based (similar to that in Figure 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.

Figure 4: Graphical representation of the simulated hydrodynamics. Linear velocity \(v\) and angular velocity \(\omega\) 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.

4. SINGLE-EVALUATION RESULTS

In this set of evolutionary experiments we evolve MFAC parameters under a single set of conditions. However, we first conduct a simulation of the robotic fish incorporating typical MFAC parameters (as listed in Table 1). Figure 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 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 (i.e., closely match) the reference speed. Ideally, in Figure 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.

Table 1: Evolutionary range of MFAC parameters, as well as the typical values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Typical</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_s) (s)</td>
<td>0.0</td>
<td>0.17</td>
<td>0.1</td>
</tr>
<tr>
<td>(e_b) (cm/s)</td>
<td>5.0</td>
<td>50.0</td>
<td>15.0</td>
</tr>
<tr>
<td>(N)</td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>(K_c)</td>
<td>0.1</td>
<td>4.0</td>
<td>1.0</td>
</tr>
<tr>
<td>(\eta)</td>
<td>0.1</td>
<td>4.0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Figure 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 differential evolution (DE) experiments. Replicates are seeded with a unique number, and DE algorithm parameters are configured as described in Section 3. 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, 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, 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.

![Single-Evaluation (best replicate)](image)

Figure 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.

5. MULTI-EVALUATION RESULTS

Given the results from Section 4, 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 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.

![9-Evaluations (best replicate)](image)

Figure 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 (\(r\)).

<table>
<thead>
<tr>
<th>Name</th>
<th>Flexibility</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>sim1</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>sim2</td>
<td>200%</td>
<td>100%</td>
</tr>
<tr>
<td>sim3</td>
<td>50%</td>
<td>100%</td>
</tr>
<tr>
<td>sim4</td>
<td>100%</td>
<td>110%</td>
</tr>
<tr>
<td>sim5</td>
<td>200%</td>
<td>110%</td>
</tr>
<tr>
<td>sim6</td>
<td>50%</td>
<td>110%</td>
</tr>
<tr>
<td>sim7</td>
<td>100%</td>
<td>90%</td>
</tr>
<tr>
<td>sim8</td>
<td>200%</td>
<td>90%</td>
</tr>
<tr>
<td>sim9</td>
<td>50%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Table 2: Fin characteristics for the 9-evaluations experiment.

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

Figure 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 9 depicts performance of the same solution when confronted with conditions that were not encountered during evolution. As shown, the controller is able to adapt to the novel fin lengths. This evolved MFAC controller should allow a robotic fish to maintain a certain level of performance even if the fin length changes during operation.

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...
Evaluations (best replicate) : 80% length

Time (s)
Speed (cm/s)

(a)

Figure 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.

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 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.

In addition to changes in fin characteristics, evolved controllers have the ability to adapt to different reference signals. Figure 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 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 12(b), 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 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.

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 13 shows an individual from the altered experiments.
Evaluations (best replicate) : 80% length, 120% flexibility

Time (s)

Speed (cm/s)

y
a

(a)

Evaluations (best replicate) : 120% length, 80% flexibility

Time (s)

Speed (cm/s)

y
a

(b)

Figure 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.

Evaluations, wide-range (best replicate) : 120% Length

Time (s)

Speed (cm/s)

y
a

Figure 13: Performance of the best evolved solution from the altered 9-evaluations experiments tested with a fin length 120% of the default.

Figure 14: Speed vs. oscillating frequency for several different fin characteristics. For certain conditions increasing the frequency results in slower speeds.

6. CONCLUSION

In this study, 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.

We note that the MFAC controllers presented in this study 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. In the future, we plan to extend the approach presented in this study to include tracking a desired reference heading while simultaneously tracking a reference speed. Doing so will produce controllers capable of more complex behaviors that incorporate both turning and forward locomotion. Adaptive techniques may also improve an evolved solution’s ability to cross the reality gap, as all unmodeled and poorly modeled dynamics will be treated as variations and accounted for during adaptation. Although this study demonstrates adaptation in the aquatic domain, the proposed technique can be applied to similarly control the speeds of terrestrial robots, and more broadly to other cyber-physical system.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions and feedback on the work provided by Jared Moore, Jianxun Wang, and the BEACON Center at Michigan State University. This work was supported in part by National Science Foundation grants IIS-1319602, CCF-1331852, CNS-1059373, CNS-0915855, and DBI-0939454, and by Michigan State University. This work was supported in part by National Science Foundation grants IIS-1319602, CCF-1331852, CNS-1059373, CNS-0915855, and DBI-0939454, and by Michigan State University.

8. REFERENCES