Motor Learning on a Heaving Plate via Improved-SNR Algorithms

by

John W. Roberts

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2009

Author . . . .
Department of Mechanical Engineering
January 16, 2009

Certified by . . . .
Russ Tedrake
X Consortium Assistant Professor of EECS
Thesis Supervisor

Certified by . . . .
Michael Triantafyllou
Professor of Mechanical and Ocean Engineering
ME Faculty Reader

Accepted by . . . .
David E. Hardt
Chairman, Committee on Graduate Students

Motor Learning on a Heaving Plate via Improved-SNR Algorithms

by

John W. Roberts

Submitted to the Department of Mechanical Engineering
on January 16, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Creatures in nature have subtle and complicated interactions with their surrounding fluids, achieving levels of performance as yet unmatched by engineered solutions. Model-free reinforcement learning (MFRL) holds the promise of allowing man-made controllers to take advantage of the subtlety of fluid-body interactions solely using data gathered on the actual system to be controlled. In this thesis, improved MFRL algorithms, motivated by a novel Signal-to-Noise Ratio for policy gradient algorithms, are developed, and shown to provide more efficient learning in noisy environments. These algorithms are then demonstrated on a heaving foil, where it is shown to learn a flapping gait on an experimental system orders of magnitude faster than the dynamics can be simulated, suggesting broad applications both in controlling robots with complex dynamics and in the study of controlled fluid systems.

Thesis Supervisor: Russ Tedrake
Title: X Consortium Assistant Professor of EECS

Acknowledgments

I would like to thank my advisor, Russ Tedrake, for making life as a graduate student more interesting, exciting, educational and fun than I had dared hope; Professor Jun Zhang, for a great deal of assistance with all the experimental components of this work and for many enjoyable visits to New York City; the other students in the Robot Locomotion Group (Alec, Elena, Katie, Rick and the oft-teased UROPs) for helping to make lab so lively, and my friends in CSAIL and Course 2 for offering plentiful distractions. Finally, I'd like to thank my mom, who took it in stride when I said I didn't have room for her in the acknowledgments because I needed to thank the NSF. Based upon her pestering I suspect she worried about my thesis at least as much as I did.

John Roberts was partially supported by an NSF GRFP fellowship.

Contents

Chapter 1

Introduction

The interactions between birds, insects, fish and the fluids surrounding them are exceptionally subtle, complex and critical to survival. This complexity arises both from the large number of degrees-of-freedom (DOF) and from the sophistication of the governing dynamics. While the robotics community has addressed the problem of high-DOF systems well, neither the motor learning community nor the controls community has a solution to the problem of complexity of dynamics. However, animals possess an amazing ability to perform well in environments of great dynamic complexity. There is evidence that these creatures learn the intricacies of their controllers through experience, a practice in sharp contrast with the model-centric approach used by humans. Birds do not solve Navier-Stokes when they make use of the richness of fluid dynamics, but rather learn controllers which, in the case of locomotion, we believe to be much simpler and more compact than the systems they control.

The task of learning, however, is fraught with its own set of difficulties. In the case of problems involving fluid-body interactions, modeling the system is a difficult task in and of itself, and can be computationally prohibitive or limited to a small regime in which the model is valid. A possible answer to this is to use **model-free reinforcement learning** (MFRL) algorithms applied to actual experimental systems. These algorithms, however, tend to be **data limited**, and are thus forced to make as much use as possible of every evaluation performed on the system they are trying to control. As one wishes to take this data on an actual robot, these evaluations can be highly stochastic and extremely expensive. Finally, if one wishes to make an argument for the biological plausibility of the learning methodology, it must be possible to implement the algorithm on a robot during operation without artificial stops-and-starts between trials. This requirement of **online learning** imposes many additional demands on the learning algorithm used, but can in turn offer many benefits. With online learning policies can converge much faster, and if the robot is used to accomplish some functional purpose, it need not stop performing its task to learn, but rather can simply adapt to its (possibly changing) surroundings as it does its job. However, it must also continue to function reasonably well even as it explores different policies, and be capable of connecting a result with the action or actions that caused it, a non-trivial task in complicated systems.

The goal of this work, then, is to present the methodology by which the difficulties inherent in online MFRL were overcome in the context of a rich and sophisticated fluid system that acts as a model of flapping flight. To this end, this thesis is divided into seven parts, with the intent that the significance of the fluid system be shown to exceed that of a test bed for MFRL algorithms, and that the utility of the newly developed Improved **Signal-to-Noise Ratio** (ISNR) algorithms is shown to extend to a broad class of problems of which this fluid system is simply one instance. The purpose of each section is as follows:

- **Introduction** - To present the problem and anticipate some of the key points in the sections that follow, in addition to providing a place for this handy list.

- **Previous Work** - To discuss the extensive literature relating to both aspects of this work: the control and analysis of flapping flight and the improvement of the performance of MFRL algorithms.

- **Experimental Setup** - To introduce the system (a simple model of forward flapping flight) on which learning was performed, and shed some light on the richness of this fluid system's dynamics.

- **Learning Algorithms** - To develop the MFRL algorithms which allowed for learning to be performed efficiently on a noisy, nonlinear physical system, and discuss the theoretical foundations which suggest further algorithmic improvements.

- **Learning on the Flapping Plate** - To demonstrate the successful application of these algorithms to the experimental flapping system, and show that these powerful and theoretically appealing techniques work in the real world.

- **Conclusion** - To reflect on the significance of these results, and what they may say about how animals learn to swim and fly.

- **Derivations** - To contain some of the more involved of the nitty-gritty mathematical details that underpin this work.

Chapter 2

Previous Work

Both fluid dynamical studies of flapping flight and reinforcement learning possess extensive literatures while remaining dynamic fields of continuing research. The present work, which follows several other examples of applying reinforcement learning to physical robotic systems [35, 24, 28], exists in both fields, and thus must be placed in context in both realms. To this end, the relevant aspects of the two fields will be briefly reviewed, with special attention paid to stroke optimization in flapping flight and variance reduction measures in MFRL.

2.1 Flapping Flight in Fluid Dynamics

The study of flapping flight in fluid dynamics has a long history [19, 14, 37, 13], and continues to be a vibrant field of investigation [41]. It will not be attempted to summarize the extensive literature here, but rather the work most relevant to this thesis - that on simple experimental flapping models and stroke optimization - will be briefly discussed.

2.1.1 Simple experimental flapping models

Dynamically scaled experimental flapping models have greatly contributed to the understanding of flapping flight. The fluid dynamics of fish locomotion have been examined closely using swimming foils and robotic fish [38, 2, 5]. The long-running question of the means by which insects generated enough lift to fly has been greatly assisted by experiments on dynamically scaled systems [15, 10, 31, 11, 1]. Finally, the dynamics of flapping flight at the scale of birds has also been illuminated by very simple experimental models, with thrust-producing mechanisms examined on simple rigid wings [40, 39]. This last system is of particular importance to this thesis as it was on this system that learning was implemented.

2.1.2 Stroke optimization

Existing work on optimizing strokeforms for flapping flight has focused on theory and simulation, with much of the difficulty arising from obtaining models rich enough to describe the dynamics in the relevant regime, and simple enough to be computationally or analytically tractable. Hall and Pigott [17] developed and optimized a model of a flapping wing using a specialized method tied to the model itself. This method converges quickly, but requires a mathematically tractable description of the dynamics. Berman and Wang [9] optimized stroke kinematics for a simulated model of a hovering insect using genetic algorithms (GA) and Powell's method. These optimization techniques have broad application, but are not well-suited to being run in real-time (due to the need to cycle through possibly very different members of a GA's population). While both of these works provide insight into the mechanics of flapping, both depend upon efficient and trustworthy models, and neither offers a plausible mechanism by which animals obtain a mastery of control in fluids over the course of their lifetime, although it does offer a plausible mechanism by which creatures could improve their control over generations.

Barrett et al. [6, 4] did perform an experimental optimization of a swimming gait on a robot fish in a tow-tank using a GA, a process which avoids the problems introduced by optimizing a modeled system. This optimization used a seven parameter policy representation (reduced from fourteen using a model of the swimming kinematics), and achieved convergence in appoximatly 100 evaluations (500 minutes) [6]. However, this technique required that each policy be evaluated on a separate experimental run, thus losing the improved learning performance offered by online learning. Also, while a GA can be an effective optimization tool, it is not a biologically plausible means for an individual animal to learn.

2.2 Reinforcement Learning

The fundamental task of reinforcement learning (RL) is to obtain a **policy** π that minimizes a given **cost function** (or, equivalently, maximizes a reward function) on a certain system over a certain period of time. The policy is generally assumed to be determined by a parameterization θ. It is by changing this θ that the behavior of the policy is controlled.

Minimizing this cost function can be done for a specified finite period of time (**finite horizon** problems) or for all time (**infinite horizon** problems). There are two basic approaches, **value function methods** and **policy gradient methods**. Both of these methods can suffer from poor performance as the problem's dimension increases, and in particular value function methods struggle with large state spaces (some of the dimensionality issues can be dealt with using function approximation, but this introduces seprate issues regarding the approximator's stability). Because of this, in many applications policy gradient methods are used in conjunction with **variance reduction techniques**, which can greatly speed convergence. Attempts to reduce variance and the time taken during learning resulted in a combination of value function and policy gradient methods, creating **actor-critic methods**, a set of techniques that can provide many benefits over either technique individually.

2.2.1 Value function methods

Value function methods learn either a **value function** $V^{\pi}(s)$ which assigns a cost to a given state s when executing the policy π, or an **action-value function** $Q^{\pi}(s, a)$, which assigns a value to taking a specific action a from state s. This value depends on both the immediate cost (given by the cost function) and the future cost (resulting from following the policy π until the end of the trial, whether it be the finite horizon or to infinity). Each algorithm attempts to estimate this future cost, and thus obtain an accurate picture of how favorable (or unfavorable) certain states are. Value function methods can be very useful, as they provide a feedback policy via the value function. Their utility is limited, however, by the difficulty of scaling them to large state spaces. Those algorithms with guaranteed convergence do not scale well to high-dimension state spaces due to computational costs, while those using function approximation to scale more efficiently have no guarantees of convergence and several trivial examples where they diverge. Four of the most significant value function methods will be briefly discussed here: Value Iteration, SARSA, Q-Learning and TD(λ).

2.2.1.1 Value iteration

Value Iteration(see [34] Section 4.4)is an extremely powerful algorithm that can be usefully applied to low-dimensional problems that have accurate models. It begins with an estimate of the value function (often all zeros will suffice) over a discrete state space (continuous state spaces must be discretized), and evaluates the expected value of every action at every state (except for certain special cases, the available actions must also be discretized, if they are not already). This expectation is clearly dependent on the current value function estimate of other states, and thus multiple iterations must be performed for it to converge to the true value function. The benefit to this technique is that globally optimal feedback policies may be found, as the value function allows the best action in each state to be determined. It must be remembered, however, this the value function is for the discretized system. While interpolation (e.g., barycentric as in [27] or volumetric) allows application to continuous state spaces, in practice value functions and the associated optimal policies can be somewhat off from the actual optima, a result of the discretization, and interpolaters inability to resolve cusps and discontinuities sufficiently well.

2.2.1.2 SARSA

SARSA (State-Action-Reward-State-Action, see [30] for its introduction, and [34] Section 6.4 for a more unified view) is designed to perform **on-policy learning** of a 'Q function', which is a function of both state and action (note that this Q function can be much larger than the same problem's value function, and thus take longer to learn). It is then possible to update the policy greedily (or ϵ-greedily) based upon this Q function, which will in turn cause a new Q function to be learned, until ultimately the Q function converges. It is important to note that the Q function learned will be tied to the policy that developed it, and thus as the policy changes, the Q-function being learned also changes. The optimality of this Q function depends upon, in the limit, visiting all states and trying all actions an infinite number of times, and the policy converging to the greedy policy. This can be achieved by choosing ϵ-greedy actions with the ϵ approaching zero as t approaches infinity, and assuming that the dynamics are **ergodic**.

2.2.1.3 Q-Learning

Q-Learning (see [34] Section 6.5 and [42]) offers the exciting ability to perform **off-policy learning**. The goal of the algorithm is (like SARSA) to learn the Q function, but as it is an off-policy method, it is capable of using information learned from one policy to help determine the Q function of another policy. This allows the estimated Q function to converge toward the true optimal Q function regardless of the currently executing policy. A policy can then be derived from this Q function, and as the Q function converges the derived policy will become optimal. An issue is that exploration requires ϵ-greedy actions to be chosen, and thus the policy derived from the Q function may not actually be optimal given this non-deterministic action selection (causing a situation in which SARSA may perform significantly better, as the Q function it learns takes into account the ϵ-greedy selection of actions). This can be resolved by decreasing ϵ with time, reducing exploration and exploiting more of what has already been learned.

2.2.1.4 TD(λ)

TD(λ) (see [34] Chapter 7 and [33]), an algorithm again learning a value function $V(s)$ rather than an action-value function Q, introduces the concept of an **eligibility trace**. The eligibility trace is an idea with broad application, and can be used in the context of Q-Learning and SARSA as well. The fundamental idea is to assign credit or blame for the current performance of the system to states or actions in the past. In the case of TD(λ) this takes the form of storing an exponentially decreasing eligibility for each state visited in the past, with the eligibility very high for recent states, and very low for distal states. The eligibility decreases as λ^n where n is the number of steps since a state was visited. This mechanism is the causal functional equivalent of having the cost at a state be an exponentially decreasing sum of the costs at all future states. This algorithm can allow for much more efficient learning of an approximate value function (approximate because of the discounting caused by $\lambda < 1$), and has been successfully applied to a number of problems (perhaps most famously to backgammon in [36]). The idea of eligibility traces, however, extends even further and is of even greater utility.

2.2.2 Policy gradient methods

A very different approach to the reinforcement learning problem is the use of policy gradient methods. These methods do not attempt to learn a value function, instead focusing on learning the policy π directly. This methodology offers many advantages, most significantly the opportunity to scale to much larger state spaces, as the policy may be represented very compactly even if the state space is extremely large (e.g., a linear controller in an n-dimensional state space where each dimension is divided into d bins requires n^2 parameters, while the number of parameters required for the value function grows as d^n). The disadvantage is that a feedback policy is not automatically provided through finding a value function, and the possibility of getting stuck in highly sub-optimal local minima arises. Four policy gradient algorithms will briefly be discussed here: Gradient Descent via Back-Propagation Through Time (i.e., 'True Gradient Descent'), REINFORCE, GPOMDP and Weight Perturbation (WP). It is worth noting that there are many other optimizers such as Sequential Quadratic Programming and Conjugate Gradient, but as the Policy Gradient methods used here are based only on pure Gradient Descent, it may be considered "ground truth" with regards to these algorithms.

2.2.2.1 Back-prop through time

Gradient descent using back-prop through time (see [43]) is a particularly effective means of optimizing a policy for a well-modeled system. Parameterizing the controller π by a vector θ, it is possible to take advantage of one's knowledge of the system's dynamics and the cost function, and compute the true gradient of the cost function with respect to each element of θ. θ can then be updated based upon this gradient, and the process repeated. Eventually this will converge to local minimum in cost (assuming the cost function cannot be pushed indefinitely toward negative infinity). Computing this gradient requires that the system be simulated forward, a process which can be computationally intensive for complicated systems. Furthermore, the gradient of θ is tied to an initial condition, and thus the policy that is found is optimized with respect a trajectory starting at that initial condition (a distribution of initial conditions can also be used by computing each update using an IC drawn from that distribution, however it may not then be relatively ill-suited to a given specific initial conditions). This trajectory can often be stabilized using a Linear-Time Varying feedback policy around the trajectory, but this is only a local solution. If one wishes to cover a larger region of the state space with a feedback policy gradient descent must be solved again for different initial conditions such that the relevant regions of state space are all near trajectories. Despite these issues, this technique is ideally suited to certain problems, and has been used for high-dimensional motor control [12] and gait-synthesis in bipeds [20]. Its primary practical disadvantage is the requirement that the system be well-modeled and capable of being simulated efficiently.

2.2.2.2 REINFORCE

REINFORCE (see [45]) is a completely model-free policy gradient algorithm that can be shown to have many desirable theoretical properties. It operates through the use of a stochastic policy, in which the action taken at every step is selected from a random distribution. Because of this randomness, it only makes sense to speak of 'expected reward', as even if the system is deterministic, the reward obtained will likely not be. The action chosen is used as a sample to estimate the local gradient of the cost function with respect to the policy parameters θ, and the parameters are updated accordingly. While this update to the parameters is not guaranteed to be in the direction of the true gradient, what makes REINFORCE algorithms special is the ability to prove that the true gradient of expected reward is followed in expectation. The analytical behavior is related to the forms of admissible stochastic policies. Many popular distributions exist within this class, including Gaussians, but certain distributions which offer improved performance are not.

2.2.2.3 GPOMDP

GPOMDP (introduced in [7] and treated further in [8]) is an algorithm designed to trade some bias in the direction of its update for reduced variance in that update. By using a discounted estimate of the current cost, GPOMDP is capable of greatly reducing the variance of an update, at the cost of increasing bias. There is a parameter (generally termed β) that regulates this trade-off, giving the user the ability to tune it as desired. This algorithm has recieved a great deal of theoretical treatment, but has not been applied to physical experimental systems, being validated instead of simulations (see, for example, [16] section 8.3, where GPOMDP is applied to a simulated puck).

2.2.2.4 Weight perturbation

Weight Perturbation is in many ways similar to REINFORCE, but allows for a broader class of sampling distributions at the price of certain aspects of analytical tractability (i.e., the actual mechanics of the updates are identical, but REINFORCE assumes a certain class of sampling stratagies, while WP is not restricted by this assumption). However, WP is a very useful practical algorithm, and it is in fact WP which was used as the basis for the algorithmic developments presented later in this thesis. Like REINFORCE, WP uses a stochastic policy to obtain samples of the local gradient and update its policy parameters θ accordingly. WP is not guaranteed to follow the true gradient of the expected cost, but it can be shown that for a small sampling magnitude[1] allows WP to follow the true point gradient of the cost function. What WP gains is the freedom to use a broader class of distributions, and by analyzing its behavior through linearization (via a first-order Taylor expansion) a number of practically useful theoretical results may be obtained.

[1]The required "smallness" is that the magnitude of a sample is small with respect to the higher-order terms of the cost function's local Taylor expansion. In other words, the sampling should occur over a region in which the cost function is approximately linear

2.2.3 Variance reduction in policy gradient

The primary contributions of this thesis to the RL literature is the development of an signal-to-noise ratio (SNR) for policy gradient, and the development of novel algorithms using this SNR to obtain improved performance. While maximizing the SNR is not equivalent to minimizing variance, this idea of SNR maximization fits within the literature of variance reduction methods. To this end, the state of variance reduction techniques will be reviewed more closely.

2.2.3.1 State-independent baselines

The simplest means of reducing variance in policy gradient RL is the use of a state-independent baseline (also called a constant baseline). Such a baseline is an integral aspect of REINFORCE [45], and has been proposed by a number of others(e.g., [26, 22]). The appropriate selection of this baseline has been the subject of study, with the most popular form - the current expected value of the reward - being argued to be suboptimal, with other forms proposed (see [16]). In practice, often times a simple decaying exponential sum of previous evaluations of the cost function is used, and significantly improves performance.

2.2.3.2 Actor-Critic methods

Due to their improved performance and broad application, actor-critic methods have long been the subject of intense interest (see, for example, [25, 23, 3]). Combining the two threads of reinforcement learning, actor-critic methods both learn the policy directly (the actor) and develop an estimated value function (the critic) which is used to reduce variance and improve convergence time. The form and behavior of the critic is a subject of ongoing research with wide variation between the structure of the various proposed algorithms (e.g., while [23] has a structure relatively similar to that of the policy gradient methods used in this thesis, [25] has quite a different form). Selection of appropriate value functions [16], importance sampling techniques [44], and means of feature selection [29] are also all the focus of current attention.

Chapter 3

Experimental Setup

For the work performed in this thesis, motor learning was executed in real-time on an experimental system developed for the purpose of studying flapping flight by the Applied Math Lab (AML) of the Courant Institute of Mathematical Sciences at New York University (see Figure 3-1). As the objective of our control was to execute the forward flapping flight more efficiently via changes in the heaving motion of the wing, modifications were necessary to allow the vertical motion of the wing to be controlled more finely (previously it moved only in sinusoidal motion). In the original setup, a horizontal heaving wing was driven through a vertical kinematic profile and allowed to spin freely about its vertical axis. This system acts as a simple model for forward flapping flight, possessing a Reynold's number of approximately 16,000[1] during these experiments, placing it in the same regime as a large dragon fly flying forward. The original experimental system built by the AML was designed to study the hydrodynamics of this system and as such did not have the required flexibility for our experiments.

This chapter will discuss the details of the original system designed to perform fluid experiments, followed by the improvements to the experimental setup (particularly

[1]This Reynolds number is determined using the forward spinning motion of the wing, as opposed to the heaving Reynolds number related to the vertical driven motion of the wing

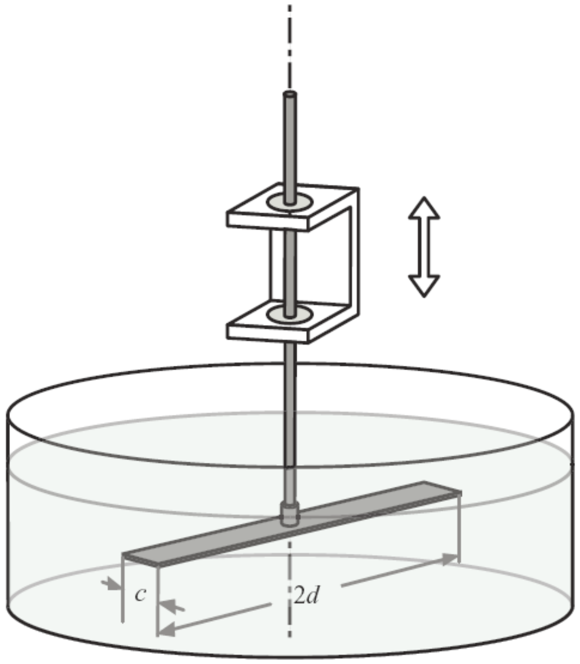


Figure 3-1: Schematic of experimental flapping system. Figure from [39].

with regards to sensing) that allowed for control to be implemented. Finally, it will discuss the structure of the controller, the control policy, and the quantitative formulation of the control objective which the learning was tasked to solve.

3.1 Original Experimental System

In this section, the original experimental system as built by the AML will be discussed. The wing was attached to the end of a 1/4" steel shaft and submerged in a water-filled tank. The shaft was held by shaft collars running against ball bearings, allowing the wing to rotate with very little friction. The angular position of the wing about the vertical axis was measured by a rotational encoder (US-Digital EM-1-1250) attached to the shaft. The vertical motion of the wing was produced by a DC electric motor (48V, 19A custom ElectroCraft motor (P/N 1843622004) with stall torque of 200 oz-in and no load speed of 3000RPM) passing through a reducing gearbox with a ratio of 22:1 (Servo System Co. P/N 25EP022-1063). The output of the gearbox drove a scotch yoke, which transformed the rotational motion of the motor into the vertical linear motion of the wing. As the original experiments used only sinusoidal driving profiles, the motor could simply be driven at a constant speed and a constant frequency sinusoidal motion would be produced by the scotch yoke. The motor control was performed by a servo amplifier (Advanced Motion Controls BE12A6), which possesses the functionality of performing direct speed control via an optical encoder (Servo System Co., SSC DA15-1000-SVLD) attached to the back of the motor.

Two of the wings used in learning (the rubber-edged and the pitching) were also developed for these experiments. The rubber wing had a 1.5" rigid metal section, with a 1.5" rubber trailing edge attached. The pitching wing used two rigid 3" plastic sections allowed to rotate about their leading edges. These plastic sections were coupled through three miter gears, forcing the pitch angles of the two sections to be the same. A restoring force was applied to the pitching wings via a torsional spring, with the rest length of the spring set such that the plastic plates would be approximately horizontal when submerged.

3.2 Improvements to Sensing

The application of reinforcement learning to the system required additional sensors to be integrated into the system. As the ability to execute non-sinusoidal waveforms was required, it was decided the the vertical position of the wing would be measured by a linear encoder (Renishaw RGH41T). This allowed for a general class of waveforms to be executed, as rather than executing full rotations of the scotch yoke the pin of the yoke was moved back and forth about the horizontal. Using a simple linear proportional feedback controller the vertical wing position was commanded to match the reference trajectory (this is described in greater detail in §3.4). This allowed for the reference trajectory to be learned, which is a more compact and natural

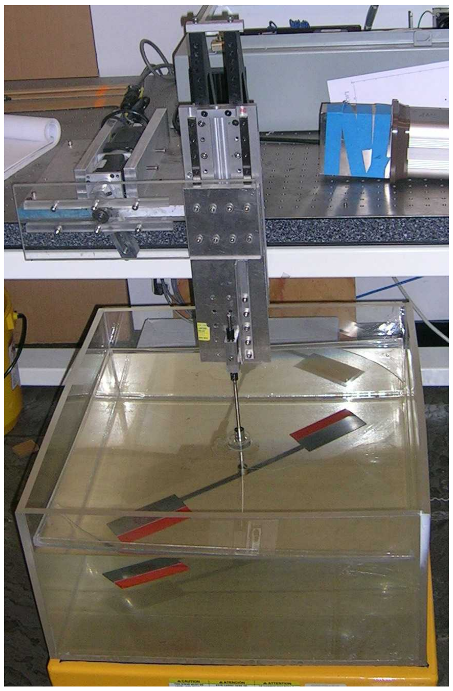


Figure 3-2: Original experimental flapping system.

representation of the flapping behavior than learning torques with feedback terms directly.

It was also desired that the energy input into the system be sensed. As the losses in the motor and gearbox could be very high, and the carriage to which the wing was mounted was relatively massive (thus large energy expenditures were required to accelerate it independent of fluid forces), it was decided that measuring the electrical energy input to the system would not be representative of the work done on the fluid. To solve this problem, a tension-compression load cell (Honeywell Sensotec Model 41 50lb P/N 060-0571-04) was placed just before the wing itself, with only the forces resulting from interaction with the fluid and acceleration of the wing itself being measured. This load cell signal was passed through a hollow shaft to a slip ring (Airflyte CAY-1666) placed immediately above the rotational encoder, allowing the signal from the rotating load cell to be passed to the fixed laboratory frame. As the load cell signal was relatively weak, it was passed through an inline amplifier

(Honeywell Sensotec Inline Amp P/N 060-6827-02) driven with a 20V potential.

Both of the encoders and the load-cell were then passed through a Sensoray 526 I/O board into a PC-104 computer running the XPC toolbox for Matlab's Simulink. The data was logged at 1kHz where it was used for the computation of energy input, rotational speed and control. This PC-104 was then linked via an ethernet connection to a Microsoft Windows PC running Matlab. This PC controlled the learning aspects of the trial, collated the data collected by the sensors and presented them to the user to ensure the proper operation of the system.

3.3 Improvements to Rigid Wing

The introduction of the slip ring (discussed in the previous section) unfortunately resulted in increased friction on the spinning system, as the slip ring's contacts produced much more friction than the ball bearings. In the case of the rubber-edged and pitching wings, the forces produced by the wing were large enough that this did not produce a significant effect. However, the rigid wing used for previous experiments did not always produce enough thrust to begin moving in the presence of friction, and thus a larger rigid wing was made. This larger wing experienced significantly greater fluid forces, and thus was able to operate well despite the added friction of the slip ring. The fact that the forces resulting from interaction with the fluid dominated bearing friction can be seen in Figure 3-3.

Introducing this larger rigid wing required that a larger tank be used (a Chem-Tainer cylindrical tank 36" in diameter and 24" high was chosen). A new rectangular rigid wing, with dimensions of 72cm x 5cm x 1/8", was made from type 2 titanium (chosen for its high stiffness, low density and high corrosion resistance). This was fastened to the 1/4" rotating shaft by a stainless steel mount. It was desired that this wing be effectively rigid in its interaction with the fluid, thus the expected deflection of the tip was calculated, with one half of the wing being treated as a cantilevered

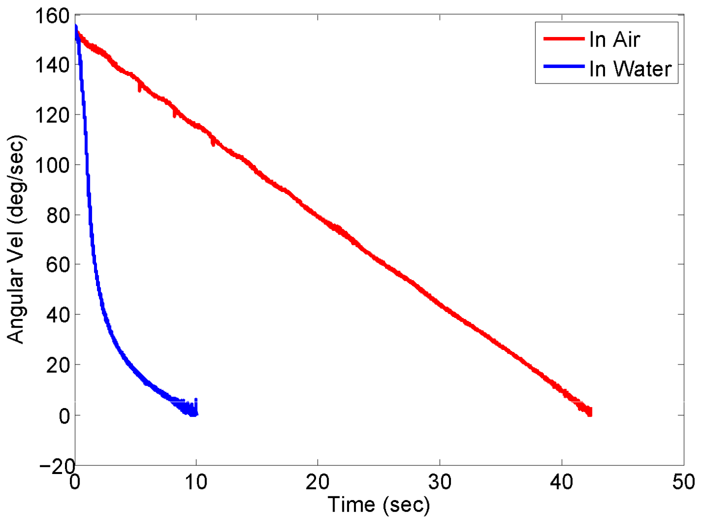


Figure 3-3: Comparison of fluid and bearing friction for large rigid wing. The foil (not undergoing a heaving motion) was spun by hand in both air and water, with the angular velocity measured as a function of time. Five curves for both cases were truncated to beginning at the same speed then averaged and filtered with a zero-phase low-pass filter to generate these plots. The quick deceleration of the wing in water as compared to air indicates that fluid viscous losses are much greater than bearing losses. At the speeds achieved at the convergence of the learning, the frictional forces in water were over six times that seen in air.

beam. A uniform load from fluid drag was assumed, and the line pressure calculated as follows, with p the line load on the beam, v the vertical velocity of the wing (determined by using the maximum velocity seen in learning), A the area of the beam, l the length of the beam and ρ the density of the fluid.:

$$p = \frac{1}{2l}\rho v^2 A = 9.87\,\mathrm{N/m} \tag{3.1}$$

The force due to gravity in the presence of buoyancy was then calculated. The density of titanium is approximately 4500 kg/m^3, thus the gravitational force when submerged is 1.96 N, or 5.44 N/m. The sum of these forces (15.31 N/m) was used as the distributed load on a rectangular beam, with the tip displacement computed using cantilevered beam theory. The deflection was computed as follows, with w_{max} the deflection at the tip, p the distributed load found above, E the Young's Modulus of titanium (116 GPa), and I the moment of inertia of the rectangular cross section of the beam:

$$w_{max} = \frac{pL^4}{8EI} = .519\,\mathrm{mm} \tag{3.2}$$

Based upon this calculation it was clear the titanium beam would be effectively rigid in its interactions with the fluid, as the fluid structures have characteristic dimensions on the centimeter scale.

3.4 Simulink Controls

The PC-104 computer which was used to log sensing data also executed the controller for the flapping plate. The controller's task was the accurate tracking of a given kinematic strokeform, accomplished by executing torque control, with the commanded torque determined by a linear feedback law. This controller consisted of several major components, outlined below:

- **Centering Controller** - Executed at the beginning of every learning run, this controller turns the motor through a full rotation, obtaining the full span of encoder measurements and ensuring that the yoke is appropriately centered.

- **Policy Look-Up Tables** - Two look-up tables store consecutive policies to be executed. These tables serve as reference signals to the controller. Two tables are used to allow time for a new policy to be generated and loaded into one table over ethernet from the PC while the policy stored in the other table is executed.

- **Linear Feedback Law** - The motor used to drive the yoke was extremely powerful, and thus a simple linear feedback law was sufficient to produce good tracking of the policy. PD (proportional-differential) policies were also attempted, but tracking was not much improved and noisy measurements of velocity resulted in chattering at higher gains.

- **Reward Calculator** - As the data required to compute reward values was not easily available to the PC orchestrating the learning aspects of the trial, the reward was computed during the execution of the policy, then set to a readable "pin" in the Simulink code (in actuality an addressable signal). This allowed the learning PC to simply take the final value of the reward, rather than the data required for computation of this value. The determination of an appropriate reward function is a difficult task, and a discussion of the reward function used in this work, and the reasoning behind it, is given in §3.6.

- **Director** - This component was distributed over several Simulink blocks, and was tasked with ensuring that reward calculations, policy executions and timings were synchronized.

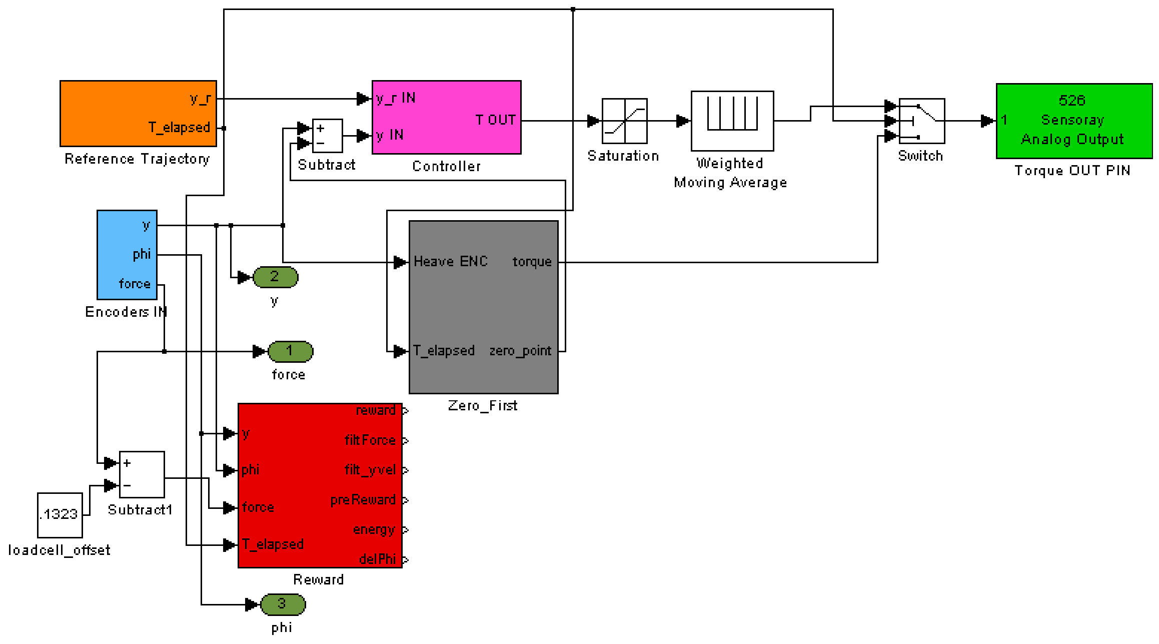


Figure 3-4: The simulink block structure used to encode the controller.

3.5 Policy Representation

The parameterization of the policy is a critical aspect in the performance of learning. Finding a representation with a small number of parameters, each well correlated with reward, can greatly improve convergence rates. Furthermore, certain parameterizations can be found with fewer local minima and smoother gradients than others, although determining these properties a priori is often impractical. Ultimately several parameterizations were tried, with all taking the form of encoding the z height of the wing as a function of time t over one period T, with the ends of the waveform constrained to be periodic (i.e., the beginning and end have identical values and first derivatives).

Parameterizing the policy in the seemingly natural fashion of a finite Fourier series was ruled out due to the difficulty in representing many intuitively useful waveforms (e.g., square waves) with a reasonable number of parameters. A parameterization using the sum of base waveforms (i.e., a smoothed square wave, a sine wave and a smoothed triangle wave) was used and shown to learn well, but was deemed a too restrictive class which predetermined many features of the waveform. Learning the base period T and the amplitude A of the waveform was also tried, and shown to perform well without significant difficulty. However, it was discovered that periods as long as possible and amplitudes as small as possible were selected (this result was useful in determining how the system's dynamics related to the reward function, and is discussed in detail in §5.2).

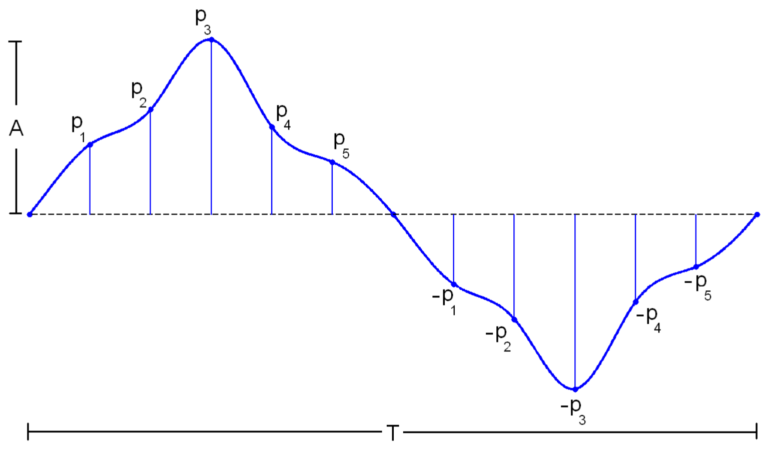


Figure 3-5: A schematic of the parameterization of policies used in this work. Note the symmetry of the up and down strokes, and the fact that five independent parameters are used to encode the shape of the waveform.

The parameterization ultimately chosen took the following form: the vertical heaving motion of the wing was represented by a 13-point periodic cubic spline with fixed amplitude and frequency, giving height z as a function of time t. There were five independent parameters, as the half-strokes up and down were constrained to be symmetric about the t axis (i.e., the first, seventh and last point were fixed at zero, while points 2 and 8, 3 and 9 etc. were set to equal and opposite values which were determined by the control parameters). This parameterization represented an interesting class of waveforms, had reasonably strong correlation between each parameter and reward, and was seen to converge reasonably quickly and robustly.

3.6 Reward Function

To precisely formulate the control task of achieving efficient forward flight, a reward function had to be chosen. Deciding upon an appropriate reward function is one of the most critical stages of formulating a reinforcement learning problem. The function must appropriately encapsulate the goals of the system, while being computable from the available data. Furthermore, functions that vary smoothly as the policy changes, have uniform magnitudes of gradient and—perhaps most importantly—differentiate between policies, can greatly improve the performance of learning. Choosing a function which satisfies these requirements can be difficult, particularly as there is often little guidance beyond these qualitative goals and intuition.

The function ultimately chosen was the inverse cost-of-transport over one period T, given as:

$$c_{mt} = \frac{mg \int_T \dot{x}(t)dt}{\int_T |F_z(t)\dot{z}(t)|dt}. \tag{3.3}$$

where F_z is the vertical force, m is the mass of the body and g is gravitational acceleration. Maximizing this quantity is equivalent to minimizing the energy cost of traveling a given distance, which is a good metric for the desired task of maximizing energy efficiency. Its integral form performs smoothing on the collected data, which is a desirable property, and intuitively is should differentiate meaningfully between different policies.

There are several reasons for selecting this form as the reward function for a system; primarily the fact that this form is the standard means of measuring transport cost for walking and running creatures, and that the non-dimensionilization comes in as a simple scalar, and thus does not change as the policy changes. While alternatives such as using a mass ratio instead of the mass were debated, changes such as these would affect the magnitude of the cost, but not the optima and learning behavior, as these are invariant to a scaling. Therefore, implementing this change would not effect the found optimal policy. Dividing by expected energy to travel through the fluid would depend upon the speed of travel, and thus would have a more complicated form.

While a cost with a speed-dependent non-dimensionilizing term could be used, and new behavior and optima may be found, rewarding very fast flapping gaits strongly (as this would tend to do) was undesireable simply because the experimental setup struggled mechanically with the violent motions found when attempting to maximize speed. The cost function selected often produced relatively gentle motions, and as such put less strain on the setup.

Chapter 4

Learning Algorithm

Due to the relatively high cost of evaluating a policy on an experimental system, it is critical to obtain as much information as possible from each sample. To maximize the amount of improvement obtained through each sample, we developed a signal-to-noise ratio (SNR) for policy gradient algorithms. In this chapter we show that a high SNR is predictive of good long-term learning performance, and show that optimizing this SNR can lead to substantial improvements to the performance of our algorithms. The application of this SNR to Weight Perturbation (a simple, easy to implement and biologically plausible policy gradient algorithm, see §2.2.2.4) was investigated in depth, and attempts to maximize the SNR for this case suggested several improvements. This chapter will concern itself with the development of the SNR, its application to improving reinforcement learning algorithms, and a demonstration that it has a meaningful improvement on learning performance, in particular on the heaving plate system with which we are concerned.

4.1 The Weight Perturbation Update

Consider minimizing a scalar function $J(\vec{w})$ with respect to the parameters $\vec{w}$ (note that it is possible that $J(\vec{w})$ is a long-term cost and results from running a system with the parameters $\vec{w}$ until conclusion). The weight perturbation algorithm [18] performs this minimization with the update:

$$\Delta\vec{w} = -\eta \left(J(\vec{w} + \vec{z}) - J(\vec{w}) \right) \vec{z}, \tag{4.1}$$

where the components of the "perturbation", $\vec{z}$, are drawn independently from a mean-zero distribution, and η is a positive scalar controlling the magnitude of the update (the "learning rate"). Performing a first-order Taylor expansion of $J(\vec{w} + \vec{z})$ yields:

$$\Delta\vec{w} = -\eta \left(J(\vec{w}) + \sum_i \frac{\partial J}{\partial \vec{w}_i} z_i - J(\vec{w}) \right) \vec{z} = -\eta \sum_i \frac{\partial J}{\partial \vec{w}_i} z_i \cdot \vec{z}. \tag{4.2}$$

In expectation, this update becomes the gradient times a (diagonal) covariance matrix, and reduces to

$$E[\Delta\vec{w}] = -\eta\sigma^2 \frac{\partial J}{\partial \vec{w}}, \tag{4.3}$$

an unbiased estimate of the gradient, scaled by the learning rate η, and σ^2(the variance of the perturbation)[1]. However, this unbiasedness comes with a very high variance, as the direction of an update is uniformly distributed. It is only the fact that updates near the direction of the true gradient have a larger magnitude than do those nearly perpendicular to the gradient that allows for the true gradient to be achieved in expectation. Note also that all samples parallel to the gradient are equally useful, whether they be in the same or opposite direction, as the sign does not affect the resulting update.

The WP algorithm is one of the simplest examples of a policy gradient reinforcement learning algorithm, and has been a popular method for treating model-free learning problems. It has been shown to be a powerful learning technique with broad applications, and its simplicity makes it amenable to analytical treatment. In the special case when $\vec{z}$ is drawn from a Gaussian distribution, weight perturbation can

[1]It is important to note that this linearization is not required to show unbiasedness. However, by linearizing a number of other avenues of analysis are offered.

be interpreted as a REINFORCE update[45]. Furthermore, update behavior of this form has been proposed as a biologically plausible mechanism by which neurons learn (see [32]). While biological plausibility is not necessarily important in engineering tasks, demonstrating the power and scalability of this simple learning model provides evidence in support of the hypothesis that biological systems behave along the same lines.

4.2 SNR for Policy Gradient Algorithms

The SNR is the expected power of the signal (update in the direction of the true gradient) divided by the expected power of the noise (update perpendicular to the true gradient). Taking care to ensure that the magnitude of the true gradient does not effect the SNR, we have:

$$\mathrm{SNR} = \frac{E\left[\Delta\vec{w}_{\parallel}^T \Delta\vec{w}_{\parallel}\right]}{E\left[\Delta\vec{w}_{\perp}^T \Delta\vec{w}_{\perp}\right]}, \tag{4.4}$$

with

$$\Delta\vec{w}_{\parallel} = \left(\Delta\vec{w}^T \frac{\vec{J}_w}{\left\|\vec{J}_w\right\|}\right) \frac{\vec{J}_w}{\left\|\vec{J}_w\right\|}, \quad \Delta\vec{w}_{\perp} = \Delta\vec{w} - \vec{w}_{\parallel}, \tag{4.5}$$

and using $\vec{J}_w(\vec{w}_0) = \frac{\partial J(\vec{w})}{\partial \vec{w}}\Big|_{(\vec{w}=\vec{w}_0)}$ for convenience.

Intuitively, this expression measures how large a proportion of the update is "useful". If the update is purely in the direction of the gradient the SNR would be infinite, while if the update moved perpendicular to the true gradient, it would be zero. As such, all else being equal, a higher SNR should generally perform as well or better than a lower SNR, and result in less violent swings in cost and policy for the same improvement in performance.

4.2.1 Weight perturbation with Gaussian distributions

Evaluating the SNR for the WP update in Equation 4.1 with a deterministic $J(\vec{w})$ and $\vec{z}$ drawn from a Gaussian distribution yields a surprisingly simple result. If one first considers the numerator:

$$\begin{aligned} E\left[\Delta\vec{w}_{\parallel}^{T}\Delta\vec{w}_{\parallel}\right] &= E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^4}\left(\sum_{i,j} J_{w_i}J_{w_j}z_iz_j\right)\vec{J_w}^{\,T}\cdot\left(\sum_{k,p} J_{w_k}J_{w_p}z_kz_p\right)\vec{J_w}\right] \\ &= E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^2}\sum_{i,j,k,p} J_{w_i}J_{w_j}J_{w_k}J_{w_p}z_iz_jz_kz_p\right] = Q, \end{aligned} \tag{4.6}$$

where we have named this term Q for convenience as it occurs several times in the expansion of the SNR. We now expand the denominator as follows:

$$\begin{aligned} E\left[\Delta\vec{w}_{\perp}^{T}\Delta\vec{w}_{\perp}\right] &= E\left[\Delta\vec{w}^{T}\Delta\vec{w} - 2\Delta\vec{w}_{\parallel}^{T}(\Delta\vec{w}_{\parallel}+\Delta\vec{w}_{\perp}) + \Delta\vec{w}_{\parallel}^{T}\Delta\vec{w}_{\parallel}\right] \\ &= E\left[\Delta\vec{w}^{T}\Delta\vec{w}\right] - 2Q + Q \end{aligned} \tag{4.7}$$

Substituting Equation (4.1) into Equation (4.7) and simplifying results in:

$$E\left[\Delta\vec{w}_{\perp}^{T}\Delta\vec{w}_{\perp}\right] = \frac{\eta^2}{\left\|\vec{J_w}\right\|^2}E\left[\sum_{i,j,k} J_{w_i}J_{w_j}z_iz_jz_k^2\right] - Q. \tag{4.8}$$

We now assume that each component z_i is drawn from a Gaussian distribution with variance σ^2. Taking the expected value, it may be further simplified to:

$$Q = \frac{\eta^2}{\left\|\vec{J_w}\right\|^4}\left(3\sigma^4\sum_i {J_{w_i}}^4 + 3\sigma^4\sum_i {J_{w_i}}^2\sum_{j\neq i}{J_{w_j}}^2\right) = \frac{3\sigma^4}{\left\|\vec{J_w}\right\|^4}\sum_{i,j}{J_{w_i}}^2{J_{w_j}}^2 = 3\sigma^4, \tag{4.9}$$

$$E\left[\Delta\vec{w}_{\perp}^{T}\Delta\vec{w}_{\perp}\right] = \frac{\eta^2\sigma^4}{\left\|\vec{J_w}\right\|^2}\left(2\sum_i {J_{w_i}}^2 + \sum_{i,j}{J_{w_i}}^2\right) - Q = \sigma^4(2+N) - 3\sigma^4 = \sigma^4(N-1), \tag{4.10}$$

where N is the number of parameters. Canceling σ results in:

$$\text{SNR} = \frac{3}{N-1}. \tag{4.11}$$

Thus, for small noises (where the linearization assumption holds) and constant σ the SNR and the number of parameters have a simple inverse relationship. This is a particularly concise model for performance scaling in PG algorithms.

4.2.2 Relationship of the SNR to learning performance

To evaluate the degree to which the SNR is correlated with actual learning performance, we ran a number of experiments on a simple quadratic bowl cost function, which may be written as:

$$J(\vec{w}) = \vec{w}^T A \vec{w}, \tag{4.12}$$

where the optimal is always at the point $\vec{0}$. The SNR suggests a simple inverse relationship between the number of parameters and the learning performance. To evaluate this claim we performed three tests: 1) true gradient descent on the identity cost function (A set to the identity matrix) as a benchmark, 2) WP on the identity cost function and 3) WP on 150 randomly generated cost functions (each component drawn from a Gaussian distribution), all of the form given in Equation (4.12), and for values of N between 2 and 10. For each trial $\vec{w}$ was initially set to be $\vec{1}$. As can be seen in Figure 4-1a, both the SNR and the reduction in cost after running WP for 100 iterations decrease monotonically as the number of parameters N increases. The fact that this occurs in the case of randomly generated cost functions demonstrates that this effect is not related to the simple form of the identity cost function, but is in fact related to the number of dimensions.

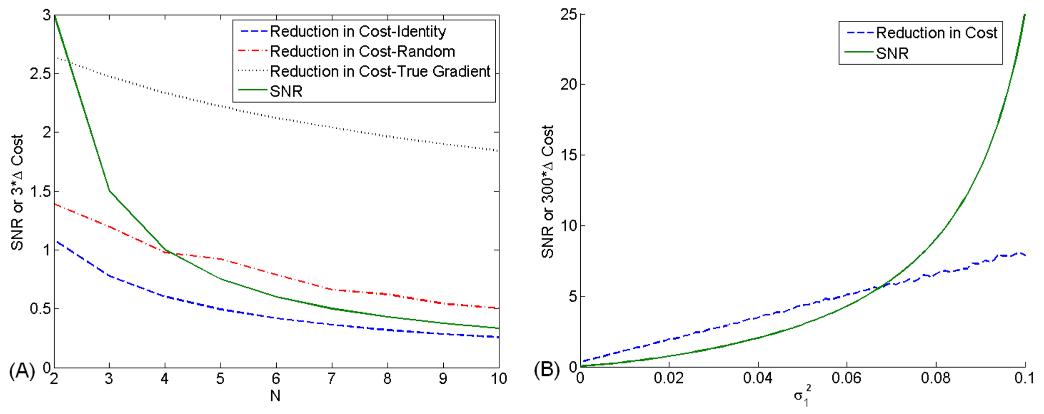


Figure 4-1: Two comparisons of SNR and learning performance: (A) Relationship as dimension N is increased (Section 4.2.2). The curves result from averaging 15,000 runs each, each run lasting 100 iterations. In the case of randomly generated cost functions, 150 different A matrices were tested. True gradient descent was run on the identity cost function. The SNR for each case was computed in accordance with Equation (4.11). (B) Relationship as Gaussian is reshaped by changing variances for case of 2D anisotropic (gradient in one direction 5 times larger than in the other) cost function (Section 4.3.1.1). The constraint $\sigma_1^2 + \sigma_2^2 = 0.1$ is imposed, while σ_1^2 is varied between 0 and .1. For each value of σ_1 15,000 updates were run and averaged to produce the curve plotted. As is clear from the plot, variances which increase the SNR also improve the performance of the update.

4.2.3 SNR with parameter-independent additive noise

In many real world systems, the evaluation of the cost $J(\vec{w})$ is not deterministic, a property which can significantly affect learning performance. In this section we investigate how additive 'noise' in the function evaluation affects the analytical expression for the SNR. We demonstrate that for very high noise WP begins to behave like a random walk, and we find in the SNR the motivation for an improvement in the WP algorithm that will be examined in Section 4.3.2.

Consider modifying the update seen in Equation (4.1) to allow for a parameter-independent additive noise term v and a more general baseline $b(\vec{w})$, and again perform the Taylor expansion. Writing the update with these terms gives:

$$\Delta \vec{w} = -\eta \left(J(\vec{w}) + \sum_i J_{w_i} z_i - b(\vec{w}) + v \right) \vec{z} = -\eta \left(\sum_i J_{w_i} z_i + \xi(\vec{w}) \right) \vec{z}. \tag{4.13}$$

where we have combined the terms $J(\vec{w})$, $b(\vec{w})$ and v into a single random variable $\xi(\vec{w})$. The new variable $\xi(\vec{w})$ has two important properties: its mean can be controlled through the value of $b(\vec{w})$, and its distribution is independent of parameters $\vec{w}$, thus $\xi(\vec{w})$ is independent of all the z_i.

We now essentially repeat the calculation seen in Section 4.2.1, with the small modification of including the noise term. When we again assume independent z_i, each drawn from identical Gaussian distributions with standard deviation σ, we obtain the expression:

$$\text{SNR} = \frac{\phi + 3}{(N-1)(\phi+1)}, \quad \phi = \frac{(J(\vec{w}) - b(\vec{w}))^2 + \sigma_v^2}{\sigma^2 \|\vec{J_w}\|^2} \tag{4.14}$$

where σ_v is the standard deviation of the noise v and we have termed the error component ϕ. This expression depends upon the fact that the noise v is mean-zero and independent of the parameters, although as stated earlier, the assumption that v is mean-zero is not limiting. It is clear that in the limit of small ϕ the expression reduces to that seen in Equation (4.11), while in the limit of very large ϕ it becomes the expression for the SNR of a random walk (see Section 4.2.4). This expression makes it clear that minimizing ϕ is desirable, a result that suggests two things: (1) the optimal baseline (from the perspective of the SNR) is the value function (i.e. $b^*(\vec{w}) = J(\vec{w})$) and (2) higher values of σ are desirable, as they reduce ϕ by increasing the size of its denominator. However, there is clearly a limit on the size of σ due to higher order terms in the Taylor expansion; very large σ will result in samples which do not represent the local gradient. Thus, in the case of noisy measurements, there is some optimal sampling distance that is as large as possible without resulting in poor sampling of the local gradient. This is explored in Section 4.3.2.1.

4.2.4 SNR of a random walk

Due to the fact that the update is squared in the SNR, only its degree of parallelity to the true gradient is relevant, not its direction. In the case of WP on a deterministic function, this is not a concern as the update is always within 90° of the gradient, and thus the parallel component is always in the correct direction. For a system with noise, however, components of the update parallel to the gradient can in fact be in the incorrect direction, contributing to the SNR even though they do not actually result in learning. This effect only becomes significant when the noise is particularly large, and reaches its extreme in the case of a true random walk (a strong bias in the "wrong" direction is in fact a good update with an incorrect sign). If one considers moving by a vector drawn from a multivariate Gaussian distribution without any correlation to the cost function, the SNR is particularly easy to compute, taking the form:

$$\begin{aligned}\text{SNR} &= \frac{\frac{1}{\|\vec{J}_w\|^4}\sum_i J_{w_i} z_i \vec{J}_w^{\,T} \sum_j J_{w_j} z_j \vec{J}_w}{(\vec{z} - \frac{1}{\|\vec{J}_w\|^2}\sum_i J_{w_i} z_i \vec{J}_w)^T (\vec{z} - \frac{1}{\|\vec{J}_w\|^2}\sum_i J_{w_i} z_i \vec{J}_w)} \\ &= \frac{\sigma^2}{N\sigma^2 - 2\sigma^2 + \sigma^2} = \frac{1}{N-1} \qquad (4.15)\end{aligned}$$

As was discussed in Section 4.2.3, this value of the SNR is the limiting case of very high measurement noise, a situation which will in fact produce a random walk.

4.3 Applications of SNR

This analysis of SNR for the WP algorithm suggests two distinct improvements that can be made to the algorithm. We explore them both in this section.

4.3.1 Reshaping the Gaussian distribution

Consider a generalized weight-perturbation algorithm, in which we allow each component z_i to be drawn independently from separate mean-zero distributions. Returning to the derivation in Section 4.2.1, we no longer assume each z_i is drawn from an identical distribution, but rather associate each with its own σ_i (the vector of the σ_i will be referred to as $\vec{\sigma}$). Removing this assumption results in the following expression of the SNR (see Appendix A.1 for a complete derivation):

$$\mathrm{SNR}(\vec{\sigma}, \vec{J_w}) = \left[\frac{\left\| \vec{J_w} \right\|^2 \left(2 \sum_i {J_{w_i}}^2 \sigma_i^4 + \sum_{i,j} {J_{w_i}}^2 \sigma_i^2 \sigma_j^2 \right)}{3 \sum_{i,j} {J_{w_i}}^2 \sigma_i^2 {J_{w_j}}^2 \sigma_j^2} - 1 \right]^{-1} . \tag{4.16}$$

An important property of this SNR is that it depends only upon the direction of $\vec{J_w}$ and the relative magnitude of the σ_i (as opposed to parameters such as the learning rate η and the absolute magnitudes $\|\vec{\sigma}\|$ and $\|\vec{J_w}\|$).

4.3.1.1 Effect of reshaping on performance

While the absolute magnitudes of the variance and true gradient do not affect the SNR given in Equation (4.16), the relative magnitudes of the different σ_i and their relationship to the true gradient *can* affect it. To study this property, we investigate a cost function with a significant degree of anisotropy. Using a cost function of the form given in Equation (4.12) and $N = 2$, we choose an A matrix whose first diagonal component is five times that of the second. We then investigate a series of possible variances σ_1^2 and σ_2^2 constrained such that their sum is a constant ($\sigma_1^2 + \sigma_2^2 = C$). We observe the performance of the first update (as opposed to running a full trial to completion), as the correct reshaped distribution is dependent upon the current gradient, and thus can change significantly over the course of a trial[2]. As is clear in Figure 4-1b, as the SNR is increased through the choice of variances the performance of this update is improved. The variation of the SNR is much more significant than the change in performance, however this is not surprising as the SNR is infinite if the update is exactly along the correct direction, while the improvement from this update will eventually saturate.

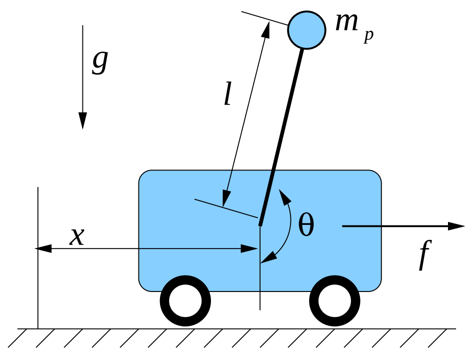
‘

4.3.1.2 Demonstration in simulation

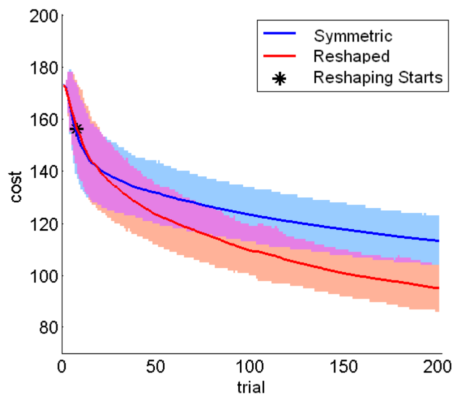
The improved performance of the previous section suggests the possibility of a modification to the WP algorithm in which an estimate of the true gradient is used before each update to select new variances which are more likely to learn effectively. Changing the shape of the distribution does add a bias to the update direction, but the resulting biased update is in fact descending the natural gradient of the cost function. To make use of this opportunity, some knowledge of the likely gradient direction is required. This knowledge can be provided via a momentum estimate (an average of previous updates) or through an inaccurate model that is able to capture some facets of the geometry of the cost function. With this estimated gradient the expression given in Equation (4.16) can be optimized over the σ_i numerically using a method such as Sequential Quadratic Programming (SQP). Care must be taken to avoid converging to very narrow distributions (e.g. placing some small minimum noise on all parameters regardless of the optimization), but ultimately this reshaping of the Gaussian can provide real performance benefits.

To demonstrate the improvement in convergence time this reshaping can achieve, weight perturbation was used to develop a barycentric feedback policy for the cart-

[2]Over the course of running a trial the true gradient's direction can, in general, take on many values, and thus the appropriate reshaped distribution and the improvement this distribution offers to SNR can change dramatically. While a new reshaped distribution can be found before every update (see §4.3.1.2 for an exploration of this idea), to simplify this examination a single update was used.



(a)



(b)

Figure 4-2: (a) The cart-pole system. The task is to apply a horizontal force f to the cart such that the pole swings to the vertical position. (b) The average of 200 curves showing reduction in cost versus trial number for both a symmetric Gaussian distribution and a distribution reshaped using the SNR. The blue shaded region marks the area within one standard deviation for a symmetric Gaussian distribution, the red region marks one standard deviation for the reshaped distribution and the purple is within one standard deviation of both. The reshaping began on the eighth trial to give time for the momentum-based gradient estimate to stabilize.

pole swingup task, where the cost was defined as a weighted sum of the actuation used and the squared distance from the upright position. A gradient estimate was obtained through averaging previous updates, and SQP was used to optimize the SNR prior to each trial. As the SQP solution can become expensive as the dimensionality gets large, means of reducing the dimensionality of the problem solved via SQP are important. Because the SNR is dominated by the most significant directions (i.e., those with the largest value of expected gradient), the SQP problem can simply be solved on the largest twenty or so dimensions without the result being significantly different, resulting in much faster evalutations. Figure 4-2 demonstrates the superior performance of the reshaped distribution over a symmetric Gaussian using the same total variance (i.e. the traces of the covariance matrices for both distributions were the same).

4.3.1.3 WP with Gaussian distributions follow the natural gradient

The natural gradient for a policy that samples with a mean-zero Gaussian of covariance Σ may be written (see [21]):

$$\tilde{\vec{J}}_w = F^{-1}\vec{J}_w, \quad F = E_{\pi(\vec{\xi};\vec{w})}\left[\frac{\partial \log \pi(\vec{\xi};\vec{w})}{\partial w_i}\frac{\partial \log \pi(\vec{\xi};\vec{w})}{\partial w_j}\right]. \tag{4.17}$$

where F is the Fisher Information matrix, π is the sampling distribution, and $\vec{\xi} = \vec{w} + \vec{z}$. Using the Gaussian form of the sampling, F may be evaluated easily, and becomes as Σ^{-1}, thus:

$$\tilde{\vec{J}}_w = \Sigma\ \vec{J}_w. \tag{4.18}$$

This is true for all mean-zero multivariate Gaussian distributions, thus the biased update, while no longer following the local point gradient, does follow the natural gradient. It is important to note that the natural gradient is a function of the shape of the sampling distribution, and it is because of this that all sampling distributions of this form can follow the natural gradient.

4.3.2 Non-Gaussian distributions

The analysis in Section 4.2.3 suggests that for optimizing a function with noisy measurements there is an optimal sampling distance which depends upon the local noise and gradient as well as the strength of higher-order terms in that region. For a simple two-dimensional cost function of the form given in Equation (4.12), Figure 4-3 shows how the SNR varies depending upon the radius of the shell distribution (i.e. the magnitude of the sampling). For various levels of additive mean-zero noise the SNR was computed for a distribution uniform in angle and fixed in its distance from the mean (this distance is the "sampling magnitude"). The fact that there is a unique maximum for each case suggests the possibility of sampling *only* at that maximal magnitude, rather than over all magnitudes as is done with a Gaussian, and thus im-

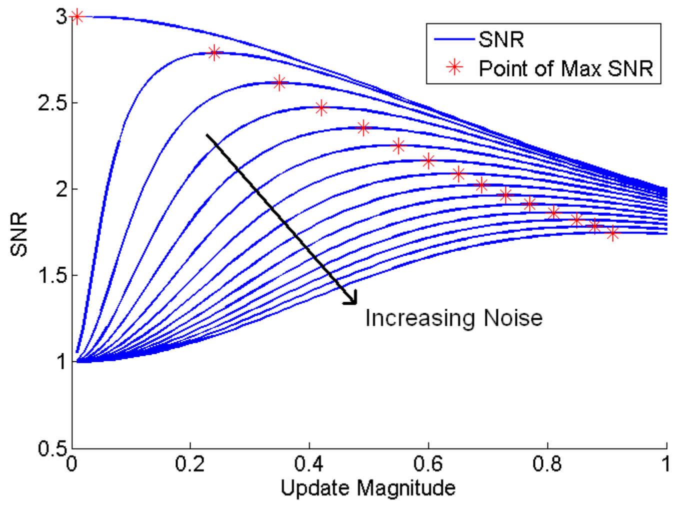


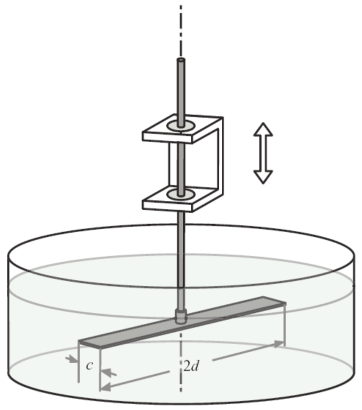
Figure 4-3: SNR as a function of update magnitude for a 2D quadratic cost function. Mean-zero measurement noise is included with variances ranging from 0 to .65. As the noise is increased, the sampling magnitude producing the maximum SNR is larger and the SNR achieved is lower. It is interesting to note that the highest SNR achieved is for the smallest sampling magnitude with no noise, where it approaches the theoretical value for the 2D case of 3. Also note that for small sampling magnitudes and larger noises the SNR approaches the random walk value of 1.

proving SNR and performance. While determining the exact magnitude of maximum SNR may be impractical, choosing a distribution with uniformly distributed direction and a constant magnitude close to this optimal value, performance can be improved.

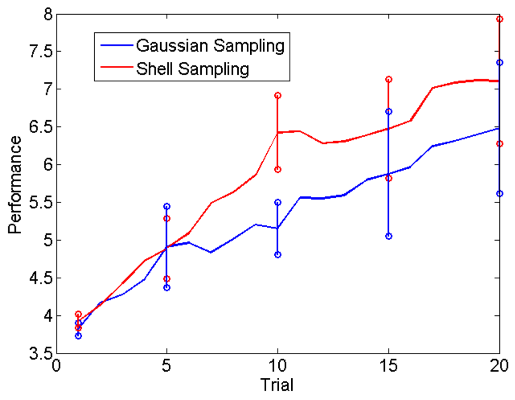
4.3.2.1 Experimental demonstration

The concept of the shell distribution is ideally suited to the experimental system outlined in §3. This flapping system is noisy, evaluations are expensive and the complexity of its dynamics makes model-free techniques of prime importance. Therefore, we decided to evaluate the performance of this modified distribution on the flapping system itself.

Beginning with a smoothed square wave, weight-perturbation was run for 20 updates using shell distributions and Gaussians. Rather than performing online learning as is discussed in §5, longer trials were used, with approximatly 30 flaps averaged to obtain the reward for a single trial, and approximatly 15 seconds given for the system to reach steady state before reward calculation began. This prevented interferance



(a)



(b)

Figure 4-4: (a) Schematic of the flapping setup on which non-Gaussian noise distributions were tested. The plate may rotate freely about its vertical axis, while the vertical motion is prescribed by the learnt policy. This vertical motion is coupled with the rotation of the plate through hydrodynamic effects, with the task being to maximize the ratio of rotational displacement to energy input. (b) 5 averaged runs on the flapping plate system using Gaussian or Shell distributions for sampling. The error bars represent one standard deviation in the performance of different runs at that trial.

between consecutively run policies and produced more accurate reward estimates at the cost of increased run time (full runs taking about an hour). This was done to make the flapping system's behavior closer to the idealized noisy system presented in §4.2.3.

Both forms of distributions were run 5 times and averaged to produce the curves seen in Figure 4-4. The sampling magnitude of the shell distribution was set to be the expected value of the length of a sample from the Gaussian distribution, while all other parameters were set as equal. As is clear from the figure, the shell distribution outperformed Gaussians in a very practical task on a very complicated system, and indeed the results presented in §5 were obtained using this modification, as it was seen to improve convergence robustly and effectively.

Chapter 5

Learning on the Flapping Plate

The ultimate goal of the work presented in this thesis was the the control of a rich fluid system through reinforcement learning techniques. This goal both stimulated the algorithmic developments of §4 and resulted in an improved understanding of the fluid dynamical system presented in §3. In this section, the success of one of these new techniques at performing online learning on this complicated system will be presented, and the insights into the dynamical properties of this controlled system that the learning provided will be discussed.

5.1 Learning in 15 Minutes

The fact that the reinforcement learning techniques presented here successfully learned the strokeform for a system of such complexity in only 15 minutes is the most significant success presented this thesis. In this section the learning performance on this problem will be shown in §5.1.1, and the issue which seemed the greatest barrier to learning online at such a fast rate—inter-trial correlation of cost—will be examined in §5.1.2.

5.1.1 Convergence of learning

On each type of wing (rigid, rubber-edged and pitching), learning was performed starting from several different initial waveforms. The rigid and rubber-edged wings were both run with a base period of $T = 1s$, while the pitching wing used a period of $T = 2$s. Learning was completed and the solution well-stabilized after 1000 flapping cycles in all cases (Fig. 5-1), with convergence seen in under fifteen minutes $T = 1$s. Both the rigid wing and the wing with the rubber trailing edge converged to a solution very similar to a triangle wave (see Figure 5-2), albeit with different achieved values of reward (the rubber-edged wing outperforming the rigid). When we tried learning on a wing with very different dynamical properties (one free to pitch about its leading edge) we found a different optimal waveform, demonstrating that the learning algorithm was capable of converging to appropriate optima for different dynamical systems possessing different optimal controllers. For all trials, the parameters of the learning algorithm were identical.

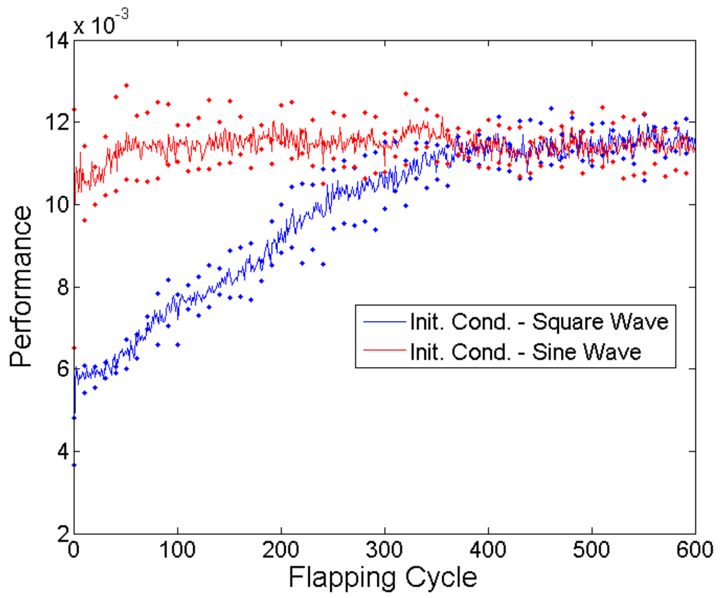


Figure 5-1: Five averaged learning curves for rigid plate starting from sine and square waves. Dots mark +/- one standard deviation from mean.

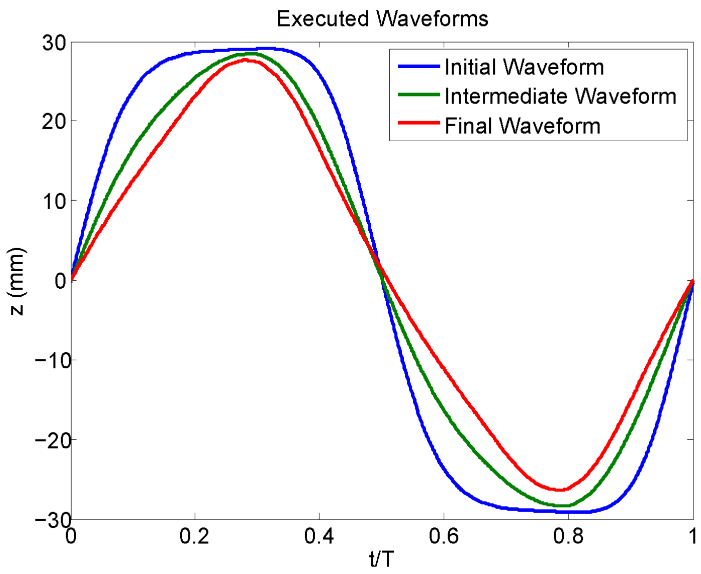


Figure 5-2: A series of waveforms (initial, intermediate and final) seen during learning on the rigid plate.

The success of learning when applied to these three different wing forms, with different rewards, initial conditions and ultimately solutions demonstrates the flexibility of the technique, and its ability to learn effectively and efficiently without modification when presented with a variety of systems (i.e., rigid, rubber-edged and pitching wings) each exhibiting different behaviors.

5.1.2 Inter-trial correlation

The speed of convergence achived by learning was dependent upon the ability to run the algorithms online with very little averaging. Ultimately, one flapping cycle was used as the length of a trial, which allowed updates to be performed every second (or every two seconds for the pitching wing). A significant worry regarding the efficacy of this method was the possibility that it took several flaps for an even subtly altered policy to converge to its true value. Eligibility traces (see §2.2.1.4) offer a possible solution to this problem, but only if the interaction between constantly changing policies does not make the measurements meaningless (e.g., the transient fluid effects *always* perform significantly worse than similar steady state behavior, and not in a manner that represents the steady state behavior of that same policy). Investigating this behavior by alternating between two policies (the difference between policies is representative of the change seen between learning trials), it became clear that convergence to steady state behavior was very fast, but there was a 'one-flap transient' (see Figure 5-3). However, this transient did not disrupt learning, as the direction of reward change was correct even though the value had yet to converge. Because of this, gradient descent was still successful, and online learning was successfully implemented.

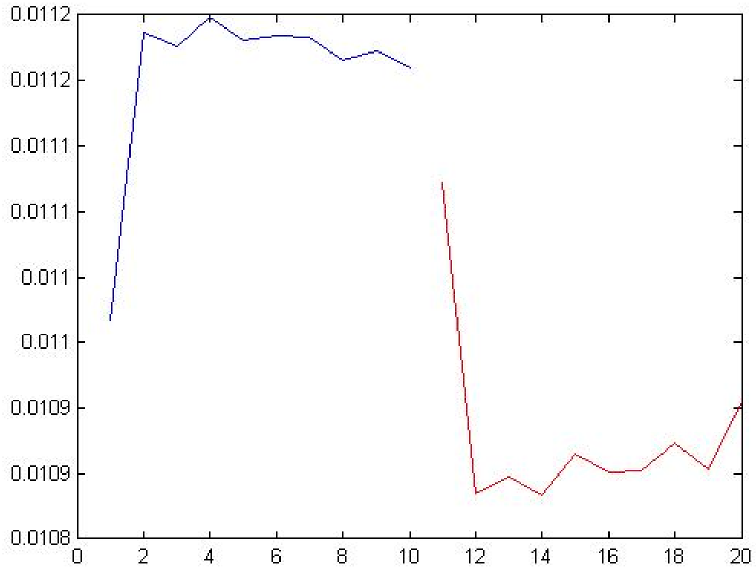


Figure 5-3: Reward versus trial number, averaged over 150 cycles. The policy is changed half-way through, and again at the end (after which it wraps around). It is clear that it takes two flaps to converge to the correct cost for a policy after a change, but the direction of change is correct, even if the magnitude is not. This allows gradient descent operate effectively without eligability traces or longer trial lengths.

5.2 Revealed Dynamical Properties

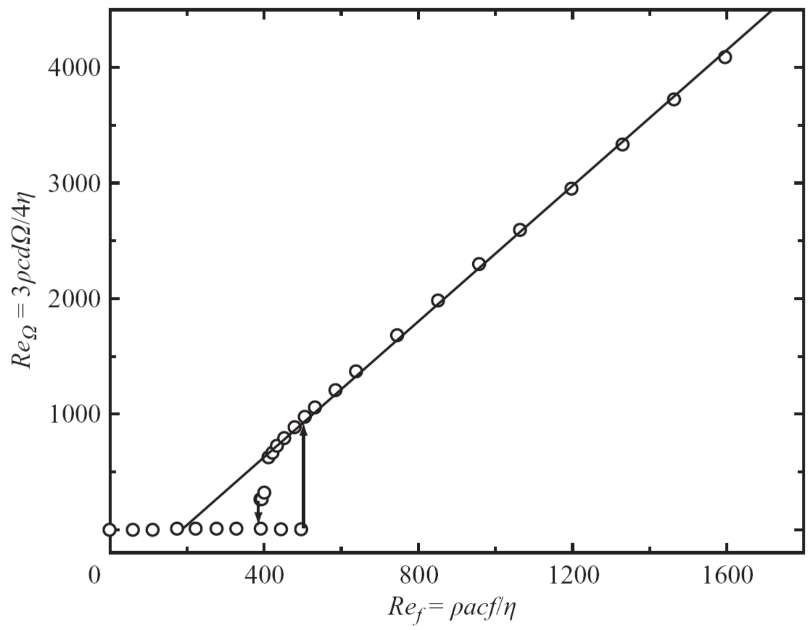


Figure 5-4: Linear growth of forward speed with flapping frequency. The axes of this curve have been non-dimensionalized as shown, and the data was taken for a sine wave. Figure from [40].

The optimal waveform found for the rigid and rubber-edged wings is consistent with results from experiments using sinusoidal strokeforms. If one considers the reward function used in this work (see 3.6), the basis of this behavior's optimality becomes more clear. As drag force is approximately quadratic with speed in this regime, the denominator behaves approximately as:

$$\int_T |F_z(t)\dot{z}(t)|dt \sim \frac{1}{2}\rho C_d \langle V^2\rangle T, \tag{5.1}$$

where C_d is the coefficient of drag and $\langle V^2\rangle$ is the mean squared heaving speed. However, forward rotational speed was found to grow linearly with flapping frequency (see [40] and Figure 5-4), thus the numerator can be written approximatly as:

$$mg \int_T \dot{x}(t)dt \sim C_f \langle V \rangle T, \tag{5.2}$$

where C_f is a constant relating vertical speed to forward speed, and $\langle V \rangle$ is the mean vertical speed. Therefore, higher speeds result in quadratic growth of the denominator of the reward function and linear growth of the numerator. This can be seen as reward having the approximate form:

$$c_{mt} \sim C \frac{\langle V \rangle}{\langle V^2 \rangle}, \tag{5.3}$$

with C a constant. This results in lower speeds being more efficient, causing lower frequencies and amplitudes to be preferred. If period and amplitude are fixed, however, the average speed is fixed (assuming no new extrema of the strokeform are produced during learning, a valid assumption in practice). A triangle wave, then, is the means of achieving this average speed with the minimum average *squared* speed. The fact that the learned optimal waveform suggested this line of reasoning demonstrates its utility in studying controlled fluid systems, where learned successful behaviors can present new insights into how the system behaves.

Chapter 6

Conclusion

This thesis has presented the steps taken to produce efficient, online learning on a complicated fluid system. The techniques used here were shown to be extremely effective, with convergence being achieved on the flapping plate in approximately fifteen minutes. The algorithms developed here have additional applications to many other systems, and by succeeding on a problem possessing great dynamic complexity, a reasonably large dimensionality and noisy evaluations, they have been shown to be robust and useful. The fact that they learn quickly on a physical experimental system demonstrates that they can be applied to practical problems, and indeed outperform many state-of-the-art algorithms.

Beyond the algorithmic improvements developed in this thesis, the fact that a simple and biologically plausible learning rule was able to learn is the complicated fluid environment presented here provides further evidence for the biological plausibility of the learning rule discussed in §4.1. Any learning rule which hopes to offer a convincing biological mechanism for neuronal learning must scale well to complicated dynamical systems and reasonable dimensionalities. Clearly, the representation used was critical to the success of the learning, with the form of the policy encoding prior knowledge of the system. However, this does not dent the argument for biological plausibility, as biological systems are often partially hard-wired to perform certain tasks, and perhaps to make the learning of them quicker and easier.

Finally, there are a number of future directions suggested by this work, with further improvements to reinforcement learning algorithms and further applications to fluid systems both promising potential next steps. However, what may be the most interesting direction is the application of these techniques to the feedback control of fluid systems. Current feedback control of high-performance systems in fluid environments, such as fighter aircraft and helicopters, is dependent upon numerous linear control laws painstakingly developed to cover the state space, with the controller changing depending upon the state of the system. Learning techniques offer the possibility of producing more general controllers, and perhaps much more quickly. Flight regimes currently avoided, such as high-angle-of-attack landing maneuvers (i.e., perching) could be exploited, and controllers could continuously adapt to the system, offering better performance in unexpected situations. It is in this direction that the greatest opportunities lie, and it may not be long before learning becomes the favored method for controlling these most challanging systems.

Appendix A

Derivations

A.1 SNR for Non-Symmetric Gaussians

Consider the update rule of, where each z_i is independantly normally distributed with mean zero:

$$\Delta\vec{w} = -\eta(g(\vec{w}+\vec{z}) - g(\vec{w}))\vec{z} \tag{A.1}$$

Let $\vec{J_w}(\vec{w}_0) = \frac{\partial J(\vec{w})}{\partial\vec{w}}\Big|_{(\vec{w}=\vec{w}_0)}$ and σ_i be the standard deviation of noise in the i direction. The expected value of the update (assuming g is locally linear around $\vec{w}$) is then given as:

$$E(\Delta\vec{w}) = -\eta\begin{pmatrix} \\ {\sigma_1}^2(J_w)_1 \\ {\sigma_2}^2(J_w)_2 \\ \vdots \end{pmatrix} \tag{A.2}$$

Now consider the signal-to-noise ratio, defined below:

$$\text{SNR} = \frac{E\left[\left(\left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)^T \left(\left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\right]}{E\left[\left(\Delta\vec{w} - \left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)^T \left(\Delta\vec{w} - \left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\right]} \tag{A.3}$$

The numerator is the expected power of the signal (the update in the direction of the true gradient) while the denominator is the expected power of the noise (the update perpendicular to the gradient). As $\Delta\vec{w} = \eta \sum_i (J_w)_i z_i \vec{z}$, the numerator may be rewritten as:

$$E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^2} \sum_i (J_w)_i z_i \sum_j (J_w)_j z_j \sum_k (J_w)_k z_k \sum_p (J_w)_p z_p\right] = \\ E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^2} \sum_{i,j,k,p} (J_w)_i (J_w)_j (J_w)_k (J_w)_p z_i z_j z_k z_p\right] \tag{A.4}$$

As each z_i is independant, this results in two non-zero cases. One where $i = j = k = p$, and three identical cases where two pairs of z are the same (e.g. $i = j$ and $k = p$ but $i \neq k$). Thus, the numerator is equal to:

$$\frac{\eta^2}{\left\|\vec{J_w}\right\|^2}\left(\sum_i (J_w)_i^4 \mu_{4_i} + 3\sum_i (J_w)_i^2 \sigma_i^2 \sum_{j\neq i} (J_w)_j^2 \sigma_j^2\right) = Q \tag{A.5}$$

Where we have named this term Q as it will occur several times in this expression. The denominator then has the form:

$$\eta^2 \left(\sum_i (J_w)_i z_i \vec{z} - \frac{1}{\left\| \vec{J_w} \right\|^2} \sum_{i,j} (J_w)_i (J_w)_j z_i z_j \vec{J_w} \right)^T \left(\sum_i (J_w)_i z_i \vec{z} - \frac{1}{\left\| \vec{J_w} \right\|^2} \sum_{i,j} (J_w)_i (J_w)_j z_i z_j \vec{J_w} \right) \tag{A.6}$$

Distributing results in three terms, the first of which is:

$$E\left[\sum_{i,j,k} (J_w)_i (J_w)_j z_i z_j z_k^2 \right] = \eta^2 \left(\sum_i (J_w)_i^2 \mu_{4_i} + \sum_i (J_w)_i^2 \sigma_i^2 \sum_{j \neq i} \sigma_j^2 \right) \tag{A.7}$$

The second term is given as:

$$-2E\left[\frac{\eta^2}{\left\| \vec{J_w} \right\|^2} \sum_{i,j,k,p} (J_w)_i (J_w)_j (J_w)_k (J_w)_p z_i z_j z_k z_p \right] = -2Q \tag{A.8}$$

The final term is then:

$$\begin{aligned} E\left[\frac{\eta^2}{\left\| \vec{J_w} \right\|^4} \sum_{i,j,k,p} (J_w)_i (J_w)_j (J_w)_k (J_w)_p z_i z_j z_k z_p \sum_q (J_w)_q^2 \right] &= \\ E\left[\frac{\eta^2}{\left\| \vec{J_w} \right\|^4} \sum_{i,j,k,p} (J_w)_i (J_w)_j (J_w)_k (J_w)_p z_i z_j z_k z_p \left\| \vec{J_w} \right\|^2 \right] &= Q \end{aligned} \tag{A.9}$$

Substituting these back into the SNR expression gives:

$$\text{SNR} = \frac{1}{\dfrac{\sum_i (J_w)_i^2 \mu_{4_i} + \sum_i (J_w)_i^2 \sigma_i^2 \sum_{j \neq i} \sigma_j^2}{\dfrac{1}{\left\| \vec{J_w} \right\|^2} \left(\sum_i (J_w)_i^4 \mu_{4_i} + 3 \sum_i (J_w)_i^2 \sigma_i^2 \sum_{j \neq i} (J_w)_j^2 \sigma_j^2 \right)} - 1} \tag{A.10}$$

From this it can be useful to consider the case where all the σ_i are equal. Making this assumption, and substituting in $\mu_{4_i} = 3\sigma^4$ the SNR simplifies as follows:

$$\text{SNR} = \frac{1}{\dfrac{\left\|\vec{J_w}\right\|^2 \left(3\sigma^4 \left\|\vec{J_w}\right\|^2 + \sigma^4(N-1)\left\|\vec{J_w}\right\|^2\right)}{3\sigma^4 \sum_i (J_w)_i^4 + 3\sigma^4 \sum_i (J_w)_i^2 \left(\left\|\vec{J_w}\right\|^2 - (J_w)_i^2\right)} - 1} \tag{A.11}$$

Simplifying this expression and canceling out the σ terms gives:

$$\text{SNR} = \frac{3}{\dfrac{\left\|\vec{J_w}\right\|^4 (N+2)}{\sum_i (J_w)_i^4 + \sum_i (J_w)_i^2 \left(\left\|\vec{J_w}\right\|^2 - (J_w)_i^2\right)} - 3} = \frac{3}{N-1} \tag{A.12}$$

A.2 SNR for Parameter-Independent Cost Function Evaluation Noise

Consider the update rule of, where each z_i is identically and independantly normally distributed with mean zero:

$$\Delta\vec{w} = -\eta(g(\vec{w} + \vec{z}) + \zeta - b(\vec{w}))\vec{z} \tag{A.13}$$

Let $\vec{J_w}(\vec{w}_0) = \left.\frac{\partial J(\vec{w})}{\partial \vec{w}}\right|_{(\vec{w}=\vec{w}_0)}$, σ be the standard deviation of noise for each parameter z_i, $b(\vec{w})$ be a baseline function of the parameters $\vec{w}$ and ζ a mean zero noise term which is independent of $\vec{w}$. The expected value of the update (assuming g is locally linear around $\vec{w}$) is then given as:

$$E(\Delta\vec{w}) = -\eta \begin{pmatrix} \\ {\sigma_1}^2 (J_w)_1 \\ {\sigma_2}^2 (J_w)_2 \\ \vdots \end{pmatrix} \tag{A.14}$$

It will be useful to define a term $\phi(\vec{w}) = g(\vec{w}) - b(\vec{w}) + \zeta$. Now consider the signal-to-noise ratio, defined below:

$$\mathrm{SNR} = \frac{E\left[\left(\left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)^T \left(\left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\right]}{E\left[\left(\Delta\vec{w} - \left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)^T \left(\Delta\vec{w} - \left(\Delta\vec{w}^T \frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\frac{\vec{J_w}}{\|\vec{J_w}\|}\right)\right]} \tag{A.15}$$

The numerator is the expected power of the signal (the update in the direction of the true gradient) while the denominator is the expected power of the noise (the update perpendicular to the gradient). As $\Delta\vec{w} = \left(\sum_i (J_w)_i z_i + \phi\right)\vec{z}$, the numerator may be rewritten as:

$$E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^2}\left(\sum_i (J_w)_i z_i + \phi\right)\left(\sum_i (J_w)_i z_i + \phi\right)\sum_k (J_w)_k z_k \sum_p (J_w)_p z_p\right] =$$
$$E\left[\frac{\eta^2}{\left\|\vec{J_w}\right\|^2}\left(\sum_{i,j,k,p} (J_w)_i (J_w)_j (J_w)_k (J_w)_p z_i z_j z_k z_p + \phi^2 \sum_{k,p} (J_w)_k (J_w)_p z_k z_p\right)\right] \tag{A.16}$$

Looking first at the fourth-order term, as each z_i is independant, this results in two non-zero cases. One where $i = j = k = p$, and three identical cases where two pairs of z are the same (e.g. $i = j$ and $k = p$ but $i \neq k$). The second-order term is simpler, requiring that $k = p$. Using $\mu_{4_i} = 3\sigma^4$ (from the Gaussian character of the noise), the numerator is equal to:

$$\frac{\eta^2}{\left\|\vec{J_w}\right\|^2}\left(3\sigma^4 \sum_i (J_w)_i^4 + 3\sigma^4 \sum_i (J_w)_i^2 \sum_{j\neq i} (J_w)_j^2\right) + \eta^2\sigma^2\psi = Q \tag{A.17}$$

Where have named this term Q as it will occur several times in this expression, and replaced $E[\phi^2] = (g(\vec{w}) - b(\vec{w}))^2 + \sigma_\zeta^2$ with ψ. The denominator then has the form:

$$\eta^2 \left(\left(\sum_i (J_w)_i z_i + \phi \right) \vec{z} - \frac{1}{\left\| \vec{J_w} \right\|^2} \left(\sum_i (J_w)_i z_i + \phi \right) \sum_j (J_w)_j z_j \vec{J_w} \right)^T \times$$
$$\left(\left(\sum_i (J_w)_i z_i + \phi \right) \vec{z} - \frac{1}{\left\| \vec{J_w} \right\|^2} \left(\sum_i (J_w)_i z_i + \phi \right) \sum_j (J_w)_j z_j \vec{J_w} \right) \tag{A.18}$$

Distributing results in three terms, again using $\mu_{4_i} = 3\sigma^4$ (from the Gaussian character of the noise) the first of these is:

$$E\left[\sum_{i,j,k} (J_w)_i (J_w)_j z_i z_j z_k^2 + \phi^2 \sum_k z_k^2 \right] = 3\sigma^4 \left\| \vec{J_w} \right\|^2 + \sigma^4 (N-1) \left\| \vec{J_w} \right\|^2 + N\psi\sigma^2 \tag{A.19}$$

The second term is given as:

$$-2E\left[\frac{\eta^2}{\left\| \vec{J_w} \right\|^2} \left(\sum_i (J_w)_i z_i + \phi \right) \left(\sum_i (J_w)_i z_i + \phi \right) \sum_k (J_w)_k z_k \sum_p (J_w)_p z_p \right] = -2Q \tag{A.20}$$

The final term is then:

$$E\left[\frac{\eta^2}{\left\| \vec{J_w} \right\|^2} \left(\sum_i (J_w)_i z_i + \phi \right) \left(\sum_i (J_w)_i z_i + \phi \right) \sum_k (J_w)_k z_k \sum_p (J_w)_p z_p \right] = Q \tag{A.21}$$

Substituting these back into the SNR results in the expression that follows:

$$\mathrm{SNR} = \cfrac{1}{\cfrac{3\sigma^4 \left\| \vec{J_w} \right\|^2 + \sigma^4 (N-1) \left\| \vec{J_w} \right\|^2 + N\psi\sigma^2}{\cfrac{1}{\left\| \vec{J_w} \right\|^2} \left(3\sigma^4 \sum_i (J_w)_i^4 + 3\sigma^4 \sum_i (J_w)_i^2 \sum_{j \neq i} (J_w)_j^2 \right) + \sigma^2 \psi} - 1} \tag{A.22}$$

Simplifying this expression and canceling out the σ terms gives:

$$\begin{aligned}
\mathrm{SNR} &= \cfrac{1}{\cfrac{\left\| \vec{J_w} \right\|^4 (N+2) + \left\| \vec{J_w} \right\|^2 \cfrac{N\psi}{\sigma^2}}{3 \sum_i (J_w)_i^4 + 3 \sum_i (J_w)_i^2 \left(\left\| \vec{J_w} \right\|^2 - (J_w)_i^2 \right) + \left\| \vec{J_w} \right\|^2 \cfrac{\psi}{\sigma^2}} - 3} = \\
&\cfrac{3 \left\| \vec{J_w} \right\|^4 + \left\| \vec{J_w} \right\|^2 \cfrac{\psi}{\sigma^2}}{\left\| \vec{J_w} \right\|^4 (N-1) + \left\| \vec{J_w} \right\|^2 \cfrac{(N-1)\psi}{\sigma^2}} = \\
&\cfrac{3 + \cfrac{\psi}{\sigma^2 \left\| \vec{J_w} \right\|^2}}{(N-1) \left(\cfrac{\psi}{\sigma^2 \left\| \vec{J_w} \right\|^2} + 1 \right)}.
\end{aligned} \tag{A.23}$$

Introducing the parameter $\chi = \frac{(g(\vec{w}) - b(\vec{w}))^2 + \sigma_\zeta^2}{\sigma^2 \left\| \vec{J_w} \right\|^2}$ makes this somewhat clearer:

$$\mathrm{SNR} = \frac{\chi + 3}{(N-1)(\chi + 1)}. \tag{A.24}$$

Glossary

Action-Value Function

Also called a Q-function, this function assigns a value not simply to a state s (as the value function $V(s)$ does) but to a state-action pair $Q(s, a)$. Like the value function, $Q(s, a)$ is partially determined by the policy used. 16

Actor-Critic Methods

Reinforcement learning methods that learn a policy directly (the 'Actor'), while maintaining an estimate of the value function to assist in the learning and reduce variance (the 'Critic'). 15

Cost Function

A function of state, action and time that gives the cost associated with taking a certain action from a certain state at a certain time. Minimizing the value of this function through the optimization of a policy is the goal of reinforcement learning algorithms. 15

Data Limited

In its use here, this refers to the property of many reinforcement learning algorithms that the temporal cost of running the algorithm is dominated by the evaluation of the cost function. 9

Eligibility Trace

A vector storing the 'eligibility' of the value function $V(s)$ or policy $\pi(s)$ at different states to be updated based upon the current incurred cost of the system. The eligibility is generally related to how long ago the state was visited. 18

Ergodic

In its use here it roughly means that, when run for a long time, the system 'forgets' its initial state and can reach all states. 17

Finite Horizon

A class of problems in which the end time is specified, and thus the policy must minimize the cost over some finite interval. 15

Infinite Horizon

Problems which are run to infinity. Generally, to ensure the cost is well behaved, either the rate at which cost is accrued must approach zero resulting in finite cost over infinite time, a discount rate must be included which reduces to importance of costs far in the future, or the 'average cost' must be computed, which looks at the ratio of cost to time of trial. 15

Model-Free Reinforcement Learning (MFRL)

A class of reinforcement learning that require no model of the system on which learning is to be performed. In general, all that is required is a means of evaluating the performance of a policy on the system. 9

Off-Policy Learning

The property that what an algorithm learns while executing one policy can be generalized to other policies. 17

On-Policy Learning

The property that what an algorithm learns while executing one policy cannot necessarily be generalized to other policies. 17

Online Learning

A style of learning in which the robot does not cease to operate during learning trials, but rather adapts simultaneously with the performance of its task. 9

Policy

A function whose output is the action to be applied to a controlled system. It can be a function of state (a 'feedback' policy), a function only of time (an 'open-loop' policy) or probabilistic (a 'stochastic' policy). 15

Policy Gradient Methods

Reinforcement learning methods that directly learn a policy, without attempting to find or store a value function. 15

Signal-to-Noise Ratio (SNR)

A ratio between the energy in the signal (the useful information) and the noise (the useless, distracting information). 10

Value Function

A function ascribing an expected long-term cost (or reward) to a given state. Because it attempts to value a state based upon the subsequent evolution of the system, different policies will produce different value functions on the same system with the same cost function. 16

Value Function Methods

Reinforcement learning methods that attempt to assign a 'value' to each state (and possibly to each state-action pair), and derive a policy from this function. 15

Variance Reduction Techniques

Methods in policy gradient reinforcement learning which attempt to reduce the noise present in updates to the policy, resulting in faster learning and policy convergence 15

Bibliography

[1] Douglas L. Altshuler, William B. Dickson, Jason T. Vance, Stephen P. Roberts, and Michael H. Dickinson. Short-amplitude high-frequency wing strokes determine the aerodynamics of honeybee flight. *Proceedings of the National Academy of Sciences*, 102(50):1821318218, December 2005.

[2] Anderson, J. M. Streitlien, K. Barrett, D.S. Triantafyllou, and M.S. Oscillating foils of high propulsive efficiency. *Journal of Fluid Mechanics*, 360:41–72, 1998.

[3] Leemon C. Baird and Andrew W. Moore. Gradient Descent for General Reinforcement Learning. Advances in Neural Information Processing Systems, 1999.

[4] Barrett, D., Grosenbaugh, M., Triantafyllou, and M. The optimal control of a flexible hull robotic undersea vehicle propelled by an oscillating foil. *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, pages 1–9, Jun 1996.

[5] Barrett, D.S. Triantafyllou, M.S. Yue, D.K.P. Grosenbaugh, M.A. Wolfgang, and M.J. Drag reduction in fish-like locomotion. *Journal of Fluid Mechanics*, 392:183–212, 1999.

[6] David Scott Barrett. *Propulsive Efficiency of a Flexible Hull Underwater Vehicle.* PhD thesis, Massachusetts Institute of Technology, 1996.

[7] J. Baxter and P.l. Bartlett. Infinite-Horizon Policy-Gradient Estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 11 2001.

[8] J. Baxter, P.L. Bartlett, and L. Weaver. Experiments with Infinite-Horizon, Policy-Gradient Estimation. *Journal of Artificial Intelligence Research*, 15:351–381, November 2001.

[9] Berman, G.J., Wang, and Z.J. Energy-minimizing kinematics in hovering insect flight. *Journal of Fluid Mechanics*, 582:153–+, jun 2007.

[10] James M. Birch and Michael H. Dickinson. Spanwise flow and the attachment of the leading-edge vortex on insect wings. *Nature*, 412:729–733, August 2001.

[11] James M. Birch and Michael H. Dickinson. The influence of wingwake interactions on the production of aerodynamic forces in flapping flight. *The Journal of Experimental Biology*, 206(13):2257–2272, July 2003.

[12] Remi Coulom. Feedforward Neural Networks in Reinforcement Learning Applied to High-dimensional Motor Control. ALT2002, 2002.

[13] Michael H. Dickinson. Unsteady Mechanisms of Force Generation in Aquatic and Aerial Locomotion. *Amer. Zool.*, 36:537–554, 1996.

[14] Michael H. Dickinson and Karl G. Gtz. UNSTEADY AERODYNAMIC PERFORMANCE OF MODEL WINGS AT LOW REYNOLDS NUMBERS. *J. exp. Biol.*, 174:4564, August 1993.

[15] Michael H. Dickinson, Fritz-Olaf Lehmann, and Sanjay P. Sane. Wing Rotation and the Aerodynamic Basis of Insect Flight. *Science*, 284(5422):1954–60, June 1999.

[16] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, December 2004.

[17] Kenneth C. Hall and Steven A. Pigott. Power Requirements for Large-Amplitude Flapping Flight. *Journal of Aircraft*, 35(3), May 1998.

[18] M. Jabri and B. Flower. Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks. *IEEE Trans. Neural Netw.*, 3(1):154–157, January 1992.

[19] Martin Jensen and T. Weis-Fogh. Biology and Physics of Locust Flight. V. Strength and Elasticity of Locust Cuticle. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 245(721):137–169, Oct 1962.

[20] Jih-Gau Juang and Chun-Shin Lin. Gait synthesis of a biped robot using backpropagation through time algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1710–1715, 1996.

[21] Sham Kakade. A Natural Policy Gradient. In *Advances in Neural Information Processing Systems (NIPS14)*, 2002.

[22] Kimura, H., , Kobayashi, and S. Reinforcement learning for continuous action using stochastic gradient ascent. *In Intelligent Autonomous Systems*, pages pp. 288–295, 1998.

[23] Hajime Kimura and Shigenobu Kobayashi. An Analysis of Actor/Critic Algorithms using Eligibility Traces: Reinforcement Learning with Imperfect Value Functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 278–286, 1998.

[24] Nate Kohl and Peter Stone. Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[25] V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[26] P. Marbach and J. N. Tsitsiklis. Simulation-Based Optimization of Markov Reward Processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, February 2001.

[27] Remi Munos and Andrew Moore. Barycentric Interpolators for Continuous Space and Time Reinforcement Learning. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 1024–1030. NIPS, MIT Press, 1998.

[28] Andrew Y. Ng, H. Jin Kim, Michael I. Jordan, and Shankar Sastry. Autonomous helicopter flight via Reinforcement Learning. *Advances in Neural Information Processing Systems (NIPS)*, 16, 2003.

[29] Khashayar Rohanimanesh, Nicholas Roy, and Russ Tedrake. Towards Feature Selection In Actor-Critic Algorithms. Technical report, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2007.

[30] Rummery and Niranjan. Online Q-Learning using Connectionist Systems. Technical report, Cambridge University Engineering Department, September 1994.

[31] Sanjay P. Sane and Michael H. Dickinson. The control of flight by a flapping wing: lift and drag production. *Journal of experimental biology*, 204:2607–2626, 2001.

[32] H.S. Seung. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron.*, 40(6):1063–73., Dec 18 2003.

[33] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9–44, 1988.

[34] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[35] Russell L Tedrake. *Applied Optimal Control for Dynamically Stable Legged Locomotion.* PhD thesis, Massachusetts Institute of Technology, 2004.

[36] Gerald Tesauro. Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, March 1995.

[37] G. S. Triantafyllou, M. S. Triantafyllou, and M. A. Grosenbaugh. Optimal Thrust Development in Oscillating Foils with Application to Fish Propulsion. *Journal of Fluids and Structures*, 7(2):205–224, February 1993.

[38] Michael S. Triantafyllou and George S. Triantafyllou. An efficient swimming machine. *Scientific American*, 272(3):64, March 1995.

[39] Nicolas Vandenberghe, Stephen Childress, and Jun Zhang. On unidirectional flight of a free flapping wing. *Physics of Fluids*, 18, 2006.

[40] Nicolas Vandenberghe, Jun Zhang, and Stephen Childress. Symmetry breaking leads to forward flapping flight. *Journal of Fluid Mechanics*, 506:147–155, 2004.

[41] Z. Jane Wang. Aerodynamic efficiency of flapping flight: analysis of a two-stroke model. *The Journal of Experimental Biology*, 211:234–238, October 2007.

[42] C J C H Watkins and P Dayan. Q-Learning. *Machine Learning*, 8(3-4):279–292, May 1992.

[43] Werbos and P.J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990.

[44] Jason L. Williams, John W. Fisher III, and Alan S. Willsky. Importance Sampling Actor-Critic Algorithms. In *Proceedings of the 2006 American Control Conference*, June 14-16 2006.

[45] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.