Theory and Implementation of Biomimetic Motor Controllers

Thesis submitted for the degree of "Doctor of Philosophy"
by

Yuval Tassa

Submitted to
the Senate of the Hebrew University of Jerusalem
February 2011

This work was carried out under the supervision of

Prof. Naftali Tishby
and
Prof. Emanuel Todorov

Acknowledgements

This dissertation owes its existence to many people, but three stand out. First and foremost my friend and mentor Emo. His unquenchable curiosity and creativity and his incredible aptitude for simplifying the complex, have made him a brilliant scientist, and a funny one too. Tom Erez is my oldest and best collaborator. His industrious and intelligent enthusiasm has never failed me. Menachem Tassa is my first friend. It was during conversations with him, on long Saturday morning walks, that the world began to pulse with magical science.

Yuval Tassa

Jerusalem, Israel
February 2011

Abstract

Control is the process of acting on a dynamical system in order to achieve a goal. Biological controllers or *brains*, perform difficult control tasks such as locomotion and manipulation despite nonlinearities, internal noise and delays, and external perturbations.

This thesis is concerned with the synthesis of motor controllers that are as agile, versatile, and robust as biological motor systems. The underlying motivation is that by solving such problems in a principled, general way, we might learn something about brains, despite the different properties of neural and digital computers. An example and inspiration in this regard are the insights into biological *sensory* systems provided by the Bayesian and Information-Theoretic frameworks: though numerical and biological inference are algorithmically different, the same general principles apply.

The theoretical framework used here is *Optimal Control*, which describes the choice of actions that minimize future costs. Besides being very general and applicable to the problems we wish to solve, certain characteristics of biological movement have been explained by postulating this framework, providing further support for its use.

In the experiments detailed here, we solve progressively more difficult Optimal Control problems, and in the process gain deep insights into the requirements and constraints of motor control. The central themes which emerge are the power of *local* control strategies, and the importance of modeling dynamical *stochasticity*. Both these elements are shown to reduce the computational burden and make resulting controllers more robust.

In the first experiment, a neural-network is used to approximate the Value function over the entire state-space by solving the HJB equation in the least-squares sense. While this approach scales only up to 4 state dimensions, we discovered that by assuming stochastic rather than deterministic dynamics, the solution is considerably smoother and convergence is much improved.

In the second experiment, a local algorithm called Differential Dynamic Programming is used to construct a locally-linear trajectory-library controller for simulated swimming robots. This approach scales very well, allowing us to solve problems of up to 24 state-dimensions.

In the third experiment, we deal with the problem of contact. Efficient local methods assume differentiability of the dynamics, which ceases to be the case when contact and friction are involved. We use the insights gained in the first experiment to define a new model of contact and friction that is based on a stochastic representation of the state, effectively smoothing the discontinuities. We then use this new model to control a simulated robotic finger which spins an object by flicking at it. The controller is based on the trajectory-library approach developed in the second experiment.

In the fourth experiment, we describe a new local algorithm that explicitly solves for periodic behaviour. Unlike our previous approaches, Dynamic Programming cannot be used here since the limit-cycle solution is non-Markov. Instead, we make use of the duality that holds between Bayesian estimation and stochastic Optimal Control, and recast the problem as inference over a loopy graphical model. We use our new method to generate a controller for a walking robot with 23 state-dimensions. Foot contact dynamics are modeled using the friction model developed in the third experiment.

Contents

Chapter 1

Introduction

1.1 Prologue: the Analogy of the Eye

Say that we wish to understand the function of the eye. The **empirical** approach would be to examine, dissect, measure and compare eyes of different organisms, and to study ocular disorders. Once enough data has been collected, a hypothesis might be proposed, and experiments designed to strengthen or falsify it.

A different possibility, which might be termed the **synthetic** approach, is to design *optical instruments*. These may or may not be similar to biological eyes, but perhaps by learning about the principles of optics, we would glean general insights that are also applicable to vision.

The actual historical development of ocular research involved both approaches, with important breakthroughs made by both students of optics and of physiology. As described in (Wade and Finger, 2001), it was the astronomer Johannes Kepler (1571-1630) who first focused light through lenses and recognized that an inverted image was formed on the retina. This idea had been vaguely proposed earlier, notably by Leonardo da Vinci, but the unintuitive inversion of the retinal image was considered to disprove it. Kepler had no such compunctions, writing "I leave it to natural philosophers to discuss the way in which this image is put together by the spiritual principles of vision". Though physiologists like Christoph Scheiner (1571-1650) and Jan Evangelista Purkinje (1787-1869) made significant subsequent contributions, it was the physicist Thomas Young (1773-1829) who first hypothesized correctly that focal accommodation is the result of changes to the curvature of the lens. The modern synthesis of our understanding of the eye was made by the physicist and medical doctor Hermann von Helmholtz, in his seminal 1868 lecture “The eye as an optical instrument”.

In this thesis we propose to study neural motor functions by taking the *synthetic approach*. Rather than examine the ways in which biological brains control bodies, we study the general control problem, under similar conditions. The theoretical framework we use is Optimal Control, a choice we will now justify.

1.2 Motor Control and Optimization

The generation of *behaviour* is arguably the central function of nervous systems. Brains gather information from sensory (afferent) neurons and act on the world via motor (efferent) neurons. A simplified picture of this dynamic, sometimes called the *perception-action cycle*, is depicted in Figure (1.1). The parts of the nervous system which are involved in sending the efferent signals to the muscles (roughly, the bottom left of the figure) are collectively called the *motor system*, and their function *motor control*.

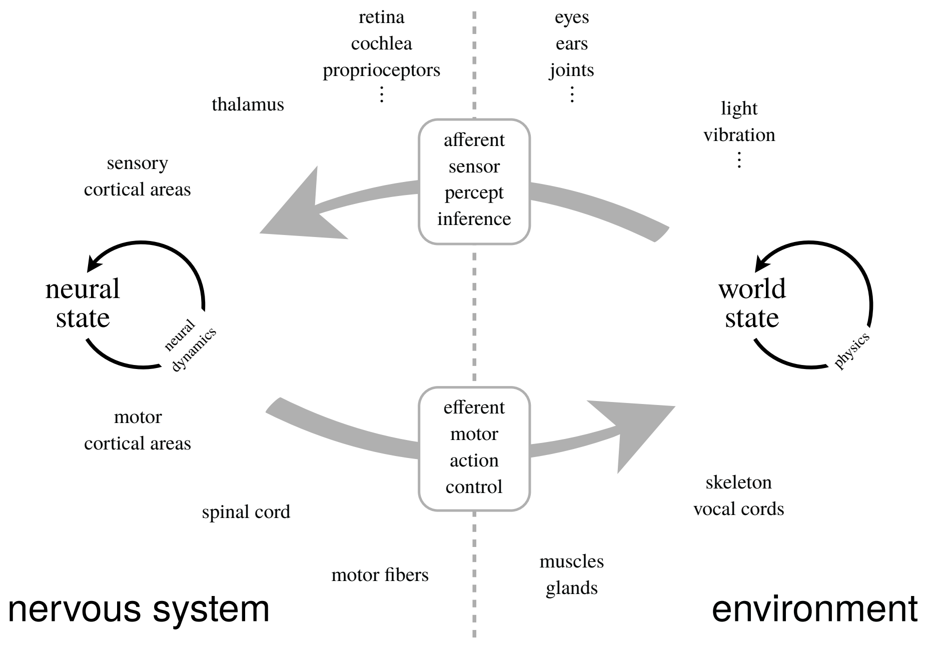


Figure 1.1: The perception-action cycle.

the plan-execute paradigm

Classic descriptions of the neural motor system usually involved two related ideas

- Planning and execution of movement are performed separately by "high-level" and "low-level" modules.
- Communication between these modules involves combinations of simple commands which activate motor "building blocks".

The central piece of evidence for these paradigms was the evident hierarchical organization of the motor system. As reflexes and other simple neural circuits were identified in the late 19th and early 20th century by Sherrington (1923), Babinski and others, it seemed reasonable to postulate that movement was generated by modulations and combinations of reflexes.

In the mid 20th century, as motor control and biomechanics became independent fields of research, the notion that there is a low-level controller with some specific structure, and a high-level planner which commands this controller, began to take the form of concrete hypotheses. Proposals for the low-level structure included the coordination patterns or "synergies" of Bernstein (1967), the related notions of "servo control" and "equilibrium-point hypothesis" of Merton (1953) and Feldman (1966) respectively, and the "motor schema" of Schmidt (1975). These ideas were reinforced by the nomenclature of control engineering, which was concurrently being developed for aerospace flight control and industrial robotics. In the engineering context, planning and control are inherently separate; the human engineer plans while the controller executes. Just as the engineer would like specify the plan using as few parameters as possible (e.g. a sequence of states that the system should pass through) and let the controller worry about the details, it seemed obvious that cortical motor areas should specify the desired movement in some compressed format, and then send this "motor program" to the spinal cord, which would deal with the details of the execution.

As biomechanical data accumulated in the second half of the 20th century regarding geometric-kinematic regularities of biological movement, these findings were interpreted as evidence for the postulated separation of planning and execution. A favorite laboratory setup for investigation was (and still is) hand-reaching. In humans this low degree-of-freedom behaviour is easily measurable and perturbable (with an actuated manipulandum), while in-vivo electrophysiology during reaching movements is possible with monkeys (Georgopoulos et al., 1982). Since the arm is highly over-actuated, many different combinations of muscle activations can achieve the same trajectory, and many different trajectories can reach the same end-point. The control problem was therefore framed as the selection of a particular trajectory by the high-level planner, which the low-level controller would then execute. Kinematic regularities like the "2/3 power-law", an empirical relationship between hand speed and path curvature, was interpreted as the result of a selection process of the appropriate trajectory. The "minimum jerk" selection criterion was proposed by Flash and Hogan (1985). While this criterion pertained to the selection of a motor plan, it was one of the first to explicitly make use of optimization, and in that sense remains an important precursor to the general optimization approach.

the optimization paradigm

Though the plan-execute paradigm and the various postulated structures of the planning-unit and execution-unit were useful for describing simple tasks like reaching, they fail to explain some important aspects of biological movements. First, movements tend to display large variability for all degrees-of-freedom, except those which are relevant to the task. This suggests that the low-level controller must have access to the goal of the movement. Second, many tasks are under-actuated, that is the motor system can control only some degrees of freedom. In under-actuated problems not every trajectory is physically realizable, and a hypothetical trajectory-planner would have to know about the dynamical constraints, making the separation conceptually inefficient. Finally, many tasks involve such a high degree of uncertainty that pre-planned trajectories are infeasible.

An different paradigm developed in the last two decades (Meyer et al., 1988; Loeb et al., 1990; Kuo, 1995), based on direct optimization of goal-related performance criteria. As summarized in (Todorov, 2004), the assumption that the motor system as a whole acts as optimal controller can explain the observed phenomena that confound the plan-execute paradigm. Rather than providing a mechanistic/algorithmic description of motor system, this theory describes the type of problem which is being solved, rather than actual solution. Since this idea is central to the work presented here, we elevate to a formal proposition:

proposition 1.1 (motor optimality hypothesis)**.** *Biological behaviour maximizes performance with respect to internally measured goals.*

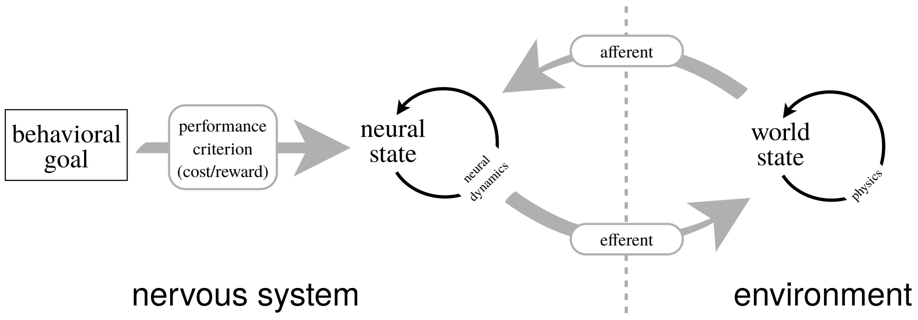


Figure 1.2: The optimization-based view of motor control.

It's important to stress that this proposition does not stand in opposition to earlier ideas, nor does it dispute the hierarchical structure of the motor system. The difference is best described using the notion of *levels of analysis*. The neuroscientist David Marr famously postulated (Marr, 1982) that information processing in the brain can be described on three levels:

- **computational**: what does the system do and why?
- **algorithmic/representational**: how does the system do what it does, what representations does it use, and what processes does it employ to manipulate them?
- **implementational**: how is the system physically realized?

Seen through this lens, earlier approaches can be seen to belong mostly to the algorithmic/representational level, while the optimization-based view lies on the computational level. A desirable development would be the agreement of results from both levels, a development which indeed has precedent in research of visual sensory processing.

The celebrated discovery of edge detectors in the primary visual cortex of cats by Hubel and Wiesel (1962), initially led to mechanistic, "bottom-up" models of progressively complex representations, as in the work of Marr (1982). As rigorous understanding of abstract inference was developed in the machine learning and machine vision communities, it was shown that edge detectors *emerge* as optimal features in unsupervised learning of natural scenes (Olshausen et al., 1996).

The parallel development in motor control would be to synthesize controllers for biologically plausible problems, and to observe the emergence of postulated internal hierarchical structures, without any a-priori assumptions. However for this to be possible, we must be able to efficiently solve complex control problems. The goal of this thesis is to advance our theoretical understanding and algorithmic competence of optimal control, to the point where such comparisons are feasible.

1.3 Outline

In Chapter 2 we provide essential mathematical background. Most of the chapter summarizes known results, but Section 2.2.3 describes novel improvements to the Differential Dynamic Programming algorithm.

In Chapter 3, previously published in (Tassa and Erez, 2007), a neural-network is used to approximate the Value function over the entire state-space by approximating a least-squares solution to the HJB equation. While this approach scales only up to 4 state dimensions, we discovered that by assuming stochastic rather than deterministic dynamics, the solution is considerably smoother and convergence is much improved.

In Chapter 4, previously published in (Tassa et al., 2008), Differential Dynamic Programming is used to construct a locally-linear trajectory-library controller for simulated swimming robots. This approach scales very well, allowing us to solve problems of up to 24 state-dimensions. The surprising robustness of the trajectory-library controller is a main result of this work.

In Chapter 5, previously published in (Tassa and Todorov, 2010), we deal with the problem of contact. Efficient local methods (like DDP) assume differentiability of the dynamics, which ceases to be the case when contact and friction are involved. We use the insights gained in Chapter 3 to define a new model of contact and friction that is based on a stochastic representation of the state, effectively smoothing the discontinuities. We then use this new model to control a simulated robotic finger which spins an object by flicking at it. The controller is based on the trajectory-library approach developed in Chapter 4.

In Chapter 6, we describe a new local algorithm that explicitly solves for periodic behaviour. Unlike our previous approaches, Dynamic Programming cannot be used here since the limit-cycle solution is non-Markov. Instead, we make use of the duality that holds between Bayesian estimation and stochastic Optimal Control, and recast the problem as inference over a loopy graphical model. We use our new method to generate a controller for a walking robot with 23 state-dimensions. Foot contact dynamics are modeled using the friction model developed in Chapter 5.

Chapter 7 contains conclusions for the entire body of work, and outlines possible future research directions.

Chapter 2

Background

2.1 Optimal Control – Theory

Optimal Control describes the choice of actions which minimize future costs. This section provides a brief outline of Optimal Control theory, first for discrete and then for continuous state and time. The results and definitions of this section are given without proof, and can be found in standard textbooks, e.g. Bertsekas (2000); Stengel (1994).

2.1.1 Discrete State

In the discrete-state case, Optimal Control reduces to the solution of a *Markov Decision Process* or MDP. The analytic tractability of this setting has made it a popular context for much of the research into optimal behaviour. In particular, the Reinforcement Learning subdiscipline of Machine Learning, has mostly restricted itself to discrete states. As we discuss in Section 2.2, solutions of discrete problems have some inherent computational drawbacks which make the setting unsuitable for real-world problems. However, the discrete-state setting remains ideal for presenting the concepts and notations of the theory.

definitions and notations

The state of the agent is $x \in \mathcal{X}$, one of $|\mathcal{X}|$ possible states. At each time-step the agent chooses an action $u \in \mathcal{U}$ out of a fixed set of $|\mathcal{U}|$ possible actions. The state is then propagated according to the *controlled transition*

notation	meaning
$x \in \mathcal{X}$	a state in the set of possible states
$u \in \mathcal{U}$	an action in the set of possible actions
$p(x'\|x,u))$	controlled transition probability
$\pi(x) \in \mathcal{U}$	policy function
$\ell(x,u)$	immediate cost
$\ell_T(x)$	terminal cost
$v^\pi(x)$	cost-to-go of starting at state x and acting according to the policy $\pi()$
$\pi^*(x)$	the optimal policy, which minimizes the cost-to-go
$v^*(x)$	the minimal cost-to-go, i.e. $v^{\pi^*}(x)$

Table 2.1: Common notation for Section 2.1.1.

probabilities

$$x' \sim p(x'|x,u)),$$

where $x' \in \mathcal{X}$ denotes state at the next time step. The choice which the controller makes, indeed the entire controller, is encapsulated in the *policy* function $\pi(x) \in \mathcal{U}$, which maps states to controls. For now our policies will be deterministic but we note that in some contexts it makes sense to consider stochastic policies $\pi = p(u|x)$.

The *immediate cost* is a function over the states and actions $\ell(x,u) \in \mathbb{R}$. A Markov Decision Process is formally the tuple $\{\mathcal{X},\mathcal{U},p(\cdot|\cdot,\cdot),\ell(\cdot,\cdot)\}$. The solution to the MDP is the policy $u = \pi(x)$ which minimize the expected sum of future costs, also known as *cost-to-go* or *Value*. There are several different formulations of this quantity.

finite-horizon formulation

In this case the quantities x, p, π, ℓ are indexed by a discrete time in some finite range $t = 0 \ldots T$, where 0 is the initial (current) time and T is the final time, also called the *horizon*. We allow the dynamics and cost to be time-dependent. The finite-horizon cost-to-go is defined as

$$v_t^\pi(x) = \mathop{\mathbb{E}}_{x' \sim p_s(\cdot|x,\pi_s(x))} \left[\ell_T(x_T) + \sum_{s=t}^{T-1} \ell_s(x_s, \pi_s(x_s)) \middle| x_t = x \right] \tag{1}$$

This cost-to-go is the expectation of the total costs incurred up to the horizon T, where the expectation is over the trajectories emanating from x, generated by the transition probabilities $p_s()$, when using the policies $\pi_s()$. Now, due to the nested nature of this definition, we can write

$$v_t^\pi(x) = \ell_t(x_t, \pi_t(x_t)) + \mathop{\mathbb{E}}_{x' \sim p_t(\cdot|x,\pi_t(x))} v_{t+1}^\pi(x')$$

In words – if we know the cost-to-go from time $t+1$, it is easy to compute the cost-to-go at time t. This is in fact the central insight of Dynamic Programming: due to the causal nature of the dynamics, the cost-to-go can best be described as a progression *backwards in time*, starting with the boundary condition $v_T^\pi(x) = \ell_T(x)$ at the horizon. While this is true for any policy, consider the smallest possible cost go, a minimization over the entire sequence of time-dependent policies $\{\pi_0(\cdot), \pi_1(\cdot) \dots \pi_{T-1}(\cdot)\}$:

$$v_0^*(x) = \min_{\{\pi_t(\cdot)\}_{t=0}^{T-1}} [v_0^\pi(x)] \,. \tag{2}$$

Exploiting the nested structure, we have that

$$v_t^*(x) = \min_{u \in \mathcal{U}} \left[\ell_t(x,u) + \mathop{\mathbb{E}}_{x' \sim p_t(\cdot|x,u)} v_{t+1}^*(x') \right] . \tag{3}$$

Thus, we have converted a minimization over an entire sequence of policies (2) to a sequence of minimizations over a single control (3), proceeding backwards in time. Eq. 3 is called the *Bellman* equation. The optimal policies are implicitly defined by the minimization

$$\pi_t^*(x) = \operatorname*{argmin}_{u \in \mathcal{U}} \left[\ell_t(x,u) + \mathop{\mathbb{E}}_{x' \sim p_t(\cdot|x,u)} v_{t+1}^*(x') \right] .$$

The quantity being minimized is called the *pseudo-Hamiltonian*

$$\bar{H}[x,u,v(\cdot),t] = \ell_t(x,u) + \mathop{\mathbb{E}}_{x' \sim p_t(\cdot|x,u)} v(x') \tag{4}$$

in correspondence with the continuous time formulation, an analogy which will become clear in Section 2.1.2.

pseudo-Hamiltonian	$\bar{H}[x, u, v(\cdot), t] = \ell_t(x, u) + \mathbb{E}_{x' \sim p_t(\cdot\|x,u)}\, v(x')$
Bellman equation	$v_t^*(x) = \min_{u \in \mathcal{U}} \bar{H}[x, u, v_{t+1}^*(\cdot), t]$
optimal policy	$\pi_t^*(x) = \operatorname{argmin}_{u \in \mathcal{U}} \bar{H}[x, u, v_{t+1}^*(\cdot), t]$
boundary condition	$v_{\mathcal{T}}^*(x) = \ell_{\mathcal{T}}(x)$

Table 2.2: Summary of finite-horizon problems.

first-exit formulation

It is often the case that we would like trajectories to end not at a pre-specified time, but to have the trajectory terminate when the state reaches some set of absorbing states $x \in \mathcal{X}_{\mathcal{T}}$. For the first-exit problem to be to be well-defined, trajectories from any initial state must reach the terminal set with non-zero probability. Defining the upper limit of the time index as that time when the trajectory reaches the terminal set $T : x_T \in \mathcal{X}_{\mathcal{T}}$, the cost-to-go is

$$v^\pi(x) = \mathop{\mathbb{E}}_{x' \sim p(\cdot|x,\pi(x))} \left[\ell_{\mathcal{T}}(x_T) + \sum_{t=0}^{T-1} \ell(x_t, \pi(x_t)) \,\middle|\, \begin{matrix} x_0 = x \\ x_T \in \mathcal{X}_{\mathcal{T}} \end{matrix} \right]$$

In this case the cost-to-go is time-independent and so are the pseudo-Hamiltonian and the optimal policy.

pseudo-Hamiltonian	$\bar{H}[x, u, v(\cdot)] = \ell(x, u) + \mathbb{E}_{x' \sim p(\cdot\|x,u)}\, v(x')$
Bellman equation	$v^*(x) = \min_{u \in \mathcal{U}} \bar{H}[x, u, v^*(\cdot)]$
optimal policy	$\pi^*(x) = \operatorname{argmin}_{u \in \mathcal{U}} \bar{H}[x, u, v^*(\cdot)]$
boundary condition	$v^*(x \in \mathcal{X}_{\mathcal{T}}) = \ell_{\mathcal{T}}(x)$

Table 2.3: Summary of first-exit problems.

The first-exit formulation is in fact more general than the finite-horizon setting. In order to embed a finite-horizon problem in a first-exit problem, we “unroll” the state-space $\mathcal{X}$ along the time-axis (i.e. duplicate it T times) and let all the states at the horizon T be absorbing states.

discounted-horizon formulation

In this case there are no absorbing states and trajectories are infinite, but the cost-to-go remains finite due to an exponential *discount factor* $0 < \alpha < 1$. The cost-to-go in this case is

$$v^\pi(x) = \frac{1}{1-\alpha} \mathop{\mathbb{E}}_{x' \sim p(\cdot|x,\pi(x))} \left[\sum_{t=0}^{\infty} \alpha^t \ell(x_t, \pi(x_t)) \middle| x_0 = x \right]$$

The coefficient $1/(1-\alpha)$ normalizes the discounting so that the cost-to-go v remains in the same units as the cost ℓ, but is not strictly necessary.

pseudo-Hamiltonian	$\bar{H}_\alpha[x, u, v(\cdot)] = (1-\alpha)\ell(x,u) + \alpha \, \mathbb{E}_{x' \sim p(\cdot|x,u)} \, v(x')$
Bellman equation	$v^*(x) = \min_{u \in \mathcal{U}} \bar{H}_\alpha[x, u, v^*(\cdot)]$
optimal policy	$\pi^*(x) = \operatorname{argmin}_{u \in \mathcal{U}} \bar{H}_\alpha[x, u, v^*(\cdot)]$

Table 2.4: Summary of discounted-horizon problems.

The discounted-horizon formulation is natural in Economics due to inflation and investment, hence the “discount” terminology. Discounted problems can also be embedded in first-exit problems by positing a universal terminal state of zero cost which can be reached from any other state with a probability of $1-\alpha$. This interpretation of the discount factor as the “probability of not dying” is relevant in understanding the empirical consistency of biological behaviour with discounted reward models. This formulation is also useful when constructing convergence proofs based on fixed-point contraction mappings.

average-cost formulation

The average-cost[1] setting also deals with infinite trajectories, but does not use any discounting. The way to make sure the cost-to-go does not diverge is the following. Say that we have computed the average cost per time-step

[1]The full name is “infinite-horizon average cost per stage”.

for a given policy

$$c^\pi = \lim_{T\to\infty}\left(\frac{1}{T}\mathop{\mathbb{E}}_{x'\sim p(\cdot|x,\pi(x))}\left[\sum_{t=0}^{T}\ell(x_t,\pi(x_t))\middle| x_0 = x\right]\right)$$

This quantity will be the same for all states if the dynamics are ergodic, i.e. given enough time any state can reach any other state with non-zero probability (for some policy). By subtracting the average cost, the following sum does not diverge:

$$\tilde{v}^\pi(x) = \lim_{T\to\infty}\left(-c^\pi T + \mathop{\mathbb{E}}_{x'\sim p(\cdot|x,\pi(x))}\left[\sum_{t=0}^{T}\ell(x_t,\pi(x_t))\middle| x_0 = x\right]\right) \tag{5}$$

The quantity $\tilde{v}(x)$ is called the "differential Value function" and encodes the difference between the state-specific and average cost-to-go. Summing over all states, the first term in the limit in (5) cancels the second one, resulting in the normalization constraint $\sum_x \tilde{v}^\pi(x) = 0$. Once again exploiting the nested structure, the differential Value function is seen to satisfy the recursion

$$\tilde{v}^\pi(x) = \ell(x,\pi(x)) - c^\pi + \mathop{\mathbb{E}}_{x'\sim p(\cdot|x,\pi(x))}\tilde{v}^\pi(x'). \tag{6}$$

For the optimal average cost $c^* = \min_{\pi(\cdot)} c^\pi$, we then have the results in Table 2.5. The average-cost formulation is different from the previous ones

pseudo-Hamiltonian	$\bar{H}[x,u,v(\cdot)] = \ell(x,u) + \mathbb{E}_{x'\sim p(\cdot|x,u)}\, v(x')$
Bellman equation	$\tilde{v}^*(x) + c^* = \min_{u\in\mathcal{U}} \bar{H}[x,u,\tilde{v}^*(\cdot)]$
optimal policy	$\pi^*(x) = \mathrm{argmin}_{u\in\mathcal{U}}\, \bar{H}[x,u,\tilde{v}^*(\cdot)]$
normalization constraint	$\sum_x \tilde{v}^*(x) = 0$

Table 2.5: Summary of average-cost problems.

in that both the function $\tilde{v}^*(x)$ and the scalar c^* need to be found simultaneously. We also note that though the limit in (5) is guaranteed not to diverge, it can oscillate indefinitely and is therefore not proper. However, (6) is exact, and can be taken to be the definition of $\tilde{v}^\pi(x)$.

2.1.2 Continuous State

In the continuous-state case, the Bellman equation of the previous section becomes the Hamilton Jacobi Bellman partial differential equation. While conceptually similar, continuous spaces can support

feature the notion of locality, which did not exist

definitions and notations

notation	meaning
$\mathbf{x}(t) \in \mathbb{R}^n$	n-dimensional state vector
$\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$	m-dimensional control vector
$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$	controlled deterministic dynamics
$\pi(\mathbf{x}) \in \mathcal{U}$	policy function
$\ell(\mathbf{x}, \mathbf{u})$	cost rate
$\ell_{\mathcal{T}}(\mathbf{x})$	terminal cost
$v^{\pi}(\mathbf{x})$	cost-to-go or Value of starting at state $\mathbf{x}$ and acting according to the policy $\pi()$
$\pi^*(\mathbf{x})$	the optimal policy, which minimizes the cost-to-go
$v^*(\mathbf{x})$	the minimal cost-to-go, i.e. $v^{\pi^*}(\mathbf{x})$

Table 2.6: Common notation for Section 2.1.2. Vector quantities are in bold letters unless they are greek (e.g. π). Gradients are denoted by subscripts $v_{\mathbf{x}}(\mathbf{x}) \equiv \nabla_{\mathbf{x}} v(\mathbf{x})$, and obvious dependencies sometimes dropped for readability $v_{\mathbf{x}} \equiv v_{\mathbf{x}}(\mathbf{x})$.

We now let the the state be a continuous-valued vector. If it is n-dimensional we write $\mathbf{x}(t) \in \mathbb{R}^n$, though some dimensions are often angular variables[2]. The m-dimensional control or action vector is $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$, where $\mathcal{U}$ is often either unbounded or box-bounded around the origin. In continuous time, the state evolves according to the differential equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}).$$

[2] Angles are embeddable in $\mathbb{R}$ with the appropriate periodic boundary conditions.

Most the equations of this section can also be written using discrete-time dynamics

$$\mathbf{x}(t+1) = \bar{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{x}(t) + \int_t^{t+1} \mathbf{f}(\mathbf{x}(s), \mathbf{u}(s))ds,$$

with no consequential differences. When implementing algorithms on a digital computer time has to be discretized. There are two reason for using continuous-time notation here. The first is that equation are usually more elegant. The second is that unlike deterministic dynamics which can be treated more-or-less equally in discrete and continuous time, the formalism of continuous-time *diffusions* is exact, while the corresponding discrete-time conditional gaussian dynamics is only approximate. The argument again shifts in the favor of discrete-time dynamics when contact and friction are considered, see Chapter 5.

The Hamilton-Jacobi-Bellman partial differential equation

The finite-horizon, continuous state-and-time analogue of Eq. 1 is now the integral

$$v^\pi(\mathbf{x}, t) = \ell_\mathcal{T}(\mathbf{x}(T)) + \int_t^T \ell(\mathbf{x}(s), \pi(\mathbf{x}(s)))ds$$

with $\mathbf{x}(t) = \mathbf{x}$ the initial condition. As before, the finite-horizon formulation can accommodate time-dependent running costs $\ell(\mathbf{x}, \mathbf{u}, t)$ and dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$, but we drop the time dependence for notational convenience. We now split the integral into two parts

$$\int_t^T \ell(s)ds = \int_t^{t+\Delta} \ell(s)ds + \int_{t+\Delta}^T \ell(s)ds$$

and take the first-order approximation of the first integral, while again exploiting the temporally nested definition:

$$v^\pi(\mathbf{x}(t), t) \approx \ell(\mathbf{x}(t), \pi(\mathbf{x}, t))\Delta + v^\pi(\mathbf{x}(t+\Delta), t+\Delta).$$

Substituting the first-order Taylor expansion of the dynamics

$$\mathbf{x}(t+\Delta) = \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))\Delta + O(\Delta^2)$$

into the first-order expansion of the cost-to-go

$$v^\pi(\mathbf{x}(t+\Delta), t+\Delta) = v^\pi(\mathbf{x}(t), t) + \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))^\mathsf{T} v^\pi_\mathbf{x}\Delta + v^\pi_t \Delta + O(\Delta^2)$$

and taking the limit $\Delta \downarrow 0$, we obtain

$$0 = \ell(\mathbf{x}, \mathbf{u}) + v^\pi_t(\mathbf{x}, t) + \mathbf{f}(\mathbf{x}, \mathbf{u})^\mathsf{T} v^\pi_\mathbf{x}(\mathbf{x}, t).$$

Repeating this procedure for the optimal Value function gives us the *Hamilton Jacobi Bellman* (HJB) partial differential equation:

$$-v^*_t(\mathbf{x}, t) = \min_\mathbf{u} \left[\ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^\mathsf{T} v^*_\mathbf{x}(\mathbf{x}, t)\right]. \tag{7}$$

It is not trivial that this equation has a unique solution. Even if the cost ℓ and the dynamics $\mathbf{f}$ are smooth, the cost-to-go can be non-differentiable, due to the discontinuity of the min() operator. Using the *viscosity solution* formalism, developed by Crandall and Lions (1983), a unique solution is guaranteed. A Viscosity solution is a possibly discontinuous function which is given as the limit of two smooth functions which tightly bound it from below and from above. Once again defining the Hamiltonian functional as the argument of the minimization

$$H[\mathbf{x}, \mathbf{u}, v(\cdot, t)] = \ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^\mathsf{T} v_\mathbf{x}(\mathbf{x}, t), \tag{8}$$

the finite-horizon HJB equation becomes

$$-v^*_t(\mathbf{x}, t) = \min_{\mathbf{u}\in\mathcal{U}} H[\mathbf{x}, \mathbf{u}, v^*(\cdot, t)].$$

The same limit can be taken for first-exit, discounted-horizon and average-cost problems. As in the discrete state case, trajectories in the first-exit setting terminate upon reaching the set $\mathcal{X}_\mathcal{T}$, and the cost-to-go and Hamiltonian are time-independent. The discounted-horizon case employs a normalized exponential weighting function[3] with a time-scale τ

$$v^\pi(\mathbf{x}) = \frac{1}{\tau}\int_0^\infty e^{-s/\tau}\ell(\mathbf{x}(s), \pi(\mathbf{x}(s)))ds,$$

[3]As before, the unit-preserving normalization is not strictly necessary.

and the corresponding Hamiltonian is

$$H_\tau[\mathbf{x}, \mathbf{u}, v(\cdot)] = \ell(\mathbf{x}, \mathbf{u}) + \tau \mathbf{f}(\mathbf{x}, \mathbf{u})^\mathsf{T} v_\mathbf{x}(\mathbf{x}).$$

A summary of HJB equations is given in Table 2.7.

formulation	**HJB equation**	**boundary conditions**
finite-horizon	$-v_t^*(\mathbf{x}, t) = \min_\mathbf{u} H[\mathbf{x}, \mathbf{u}, v^*(\cdot, t)]$	$v^*(\mathbf{x}, T) = \ell_\mathcal{T}(\mathbf{x})$
first-exit	$0 = \min_\mathbf{u} H[\mathbf{x}, \mathbf{u}, v^*(\cdot)]$	$v^*(\mathbf{x} \in \mathcal{X}_\mathcal{T}) = \ell_\mathcal{T}(\mathbf{x})$
discounted	$v^*(\cdot) = \min_\mathbf{u} H_\tau[\mathbf{x}, \mathbf{u}, v^*(\cdot)]$	
average-cost	$c^* = \min_\mathbf{u} H[\mathbf{x}, \mathbf{u}, \tilde{v}^*(\cdot)]$	

Table 2.7: Summary of HJB equations for various formulations of deterministic continuous space and time problems. Minimizations are over $\mathbf{u} \in \mathcal{U}$.

Computing the policy

As in the discrete case, the policy minimizes the Hamiltonian. For example for the undiscounted time-independent Hamiltonian

$$\pi^*(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{u} \in \mathcal{U}} \left[\ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^\mathsf{T} v_\mathbf{x}^*(\mathbf{x}) \right].$$

A necessary condition at the minimum is the vanishing of the **u**-gradient

$$0 = \ell_\mathbf{u}(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})_\mathbf{u}^\mathsf{T} v_\mathbf{x}^*(\mathbf{x}), \tag{9}$$

which often turns out to be sufficient. For many dynamical systems the control signal enters the dynamics linearly, i.e.

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{a}(\mathbf{x}) + B(\mathbf{x})\mathbf{u}$$

In particular for mechanical systems, if the control variables are forces or torques, linearity is enforced by Newton's second law. Additionally, it is often the case that the running cost is a sum of a state-cost and a control-cost

$$\ell(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + r(\mathbf{u}).$$

If $r(\mathbf{u})$ is a smooth and strictly convex function so that

$$r_{\mathbf{uu}}(\mathbf{u}) > 0 \quad \forall\, \mathbf{u}, \tag{10}$$

then $r_{\mathbf{u}}(\mathbf{u}^*) = 0$ implies $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}} r(\mathbf{u})$, Eq. 9 has a single solution, and the minimization can be performed analytically:

$$\pi^*(\mathbf{x}) = -r_{\mathbf{u}}^{-1}\left(B^{\mathsf{T}}(\mathbf{x}) v^*_{\mathbf{x}}(\mathbf{x})\right). \tag{11}$$

Expression (11) shows that a simple way of dealing with box-bounded control sets $\mathcal{U} \subset \mathbb{R}^m$, is to use a control cost with vertical asymptotes at the bounds. For example if $\mathcal{U} = [-1, 1]$, we can use the convex function[4] $r(u) = \int \operatorname{arctanh}(u)du$ so that $r_u^{-1}(\cdot) = \tanh(\cdot)$ has the appropriate bounds. However the most common control-cost is the classic quadratic $r(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathsf{T}} R\mathbf{u}$, for which

$$\pi^*(\mathbf{x}) = -R^{-1}B^{\mathsf{T}}(\mathbf{x}) v^*_{\mathbf{x}}(\mathbf{x}). \tag{12}$$

Linear-Quadratic problems

Of particular importance is the case when the dynamics are linear

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = A\mathbf{x} + B\mathbf{u}$$

and the cost is quadratic

$$\ell(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\mathbf{x}^{\mathsf{T}} Q\mathbf{x} + \frac{1}{2}\mathbf{u}^{\mathsf{T}} R\mathbf{u}.$$

Linear-quadratic problems can be solved analytically, and form the basis of many algorithms. Substituting the ansatz of a quadratic optimal cost-to-go $v^*(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathsf{T}} S\mathbf{x}$ in (12) we have that

$$\pi^*(\mathbf{x}) = -R^{-1}B^{\mathsf{T}} S\mathbf{x}. \tag{13}$$

In this case the controller drives the state to the origin exponentially, so in the average-cost case $c^* = 0$ and $\tilde{v}(\mathbf{x}) \equiv v(\mathbf{x})$. Substituting (13) into the average-cost HJB equation (last row of Table 2.7), and dropping the min()

[4]The precise formula is $\int \operatorname{arctanh}(u)du = \frac{1}{2}\big((1 + x)\ln(1 + x) + (1 - x)\ln(1 - x)\big)$.

operator (since the policy is already optimal) results in

$$\begin{aligned}
0 &= \frac{1}{2}\mathbf{x}^\mathsf{T} Q\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathsf{T} SBR^{-1}RR^{-1}B^\mathsf{T} S\mathbf{x} + (A\mathbf{x} - BR^{-1}B^\mathsf{T} S\mathbf{x})^\mathsf{T} S\mathbf{x} \\
&= \frac{1}{2}\left(\mathbf{x}^\mathsf{T} Q\mathbf{x} - \mathbf{x}^\mathsf{T} SBR^{-1}B^\mathsf{T} S\mathbf{x}\right) + \mathbf{x}^\mathsf{T} A^\mathsf{T} S\mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^\mathsf{T}\left(Q - SBR^{-1}B^\mathsf{T} S + A^\mathsf{T} S + SA\right)\mathbf{x},
\end{aligned}$$

which holds for all $\mathbf{x}$ and therefore implies the *Riccati* matrix equation:

$$0 = Q - SBR^{-1}B^\mathsf{T} S + A^\mathsf{T} S + SA.$$

The Maximum Principle boundary value problem

An important aspect of the continuous-state formulation is that it allows solutions to be defined locally, along an *extremal trajectory*. The set of differential equations which describe the local solutions are called "Pontryagin's Maximum Principle" and are related to variational principles in mechanics. The Maximum Principle is defined for the finite-horizon case, so we start with the time-dependent HJB equation (7), substitute the optimal policy into $\mathbf{u}$ and thence drop the min() operator (and asterisk superscripts):

$$0 = v_t(\mathbf{x}, t) + \ell(\mathbf{x}, \pi(\mathbf{x}, t)) + \mathbf{f}(\mathbf{x}, \pi(\mathbf{x}, t))^\mathsf{T} v_\mathbf{x}(\mathbf{x}, t).$$

Differentiating with respect to $\mathbf{x}$ (and omitting dependencies) produces

$$\begin{aligned}
0 &= v_{\mathbf{x}t} + \ell_\mathbf{x} + \pi_\mathbf{x}^\mathsf{T}\ell_\mathbf{u} + \mathbf{f}_\mathbf{x}^\mathsf{T} v_\mathbf{x} + \pi_\mathbf{x}^\mathsf{T}\mathbf{f}_\mathbf{u}^\mathsf{T} v_\mathbf{x} + v_{\mathbf{xx}}\mathbf{f} \\
&= \dot{v}_\mathbf{x} + \ell_\mathbf{x} + \mathbf{f}_\mathbf{x}^\mathsf{T} v_\mathbf{x} + \pi_\mathbf{x}^\mathsf{T}(\ell_\mathbf{u} + \mathbf{f}_\mathbf{u}^\mathsf{T} v_\mathbf{x}),
\end{aligned}$$

where we use the *total* derivative $\dot{v}_\mathbf{x} \equiv \frac{d}{dt}v_\mathbf{x} = v_{\mathbf{x}t} + v_{\mathbf{xx}}\mathbf{f}$. Now noting that (9) implies that the term in parentheses vanishes, we have

$$-\dot{v}_\mathbf{x} = \ell_\mathbf{x} + \mathbf{f}_\mathbf{x}^\mathsf{T} v_\mathbf{x}.$$

Defining a new variable λ to be the cost-to-go gradient along the optimal trajectory $\lambda(t) \equiv v_{\mathbf{x}}(t)$, the Hamiltonian (8) becomes

$$H[\mathbf{x}, \mathbf{u}, \lambda] = \ell(\mathbf{x}, \mathbf{u}) + \mathbf{f}(\mathbf{x}, \mathbf{u})^{\mathsf{T}} \lambda,$$

and the equations which constitute the Maximum Principle are

$$\dot{\mathbf{x}} = H_{\lambda}[\mathbf{x}, \mathbf{u}, \lambda] \tag{14a}$$

$$-\dot{\lambda} = H_{\mathbf{x}}[\mathbf{x}, \mathbf{u}, \lambda] \tag{14b}$$

$$\mathbf{u} = \underset{\tilde{\mathbf{u}}}{\operatorname{argmin}}\, H[\mathbf{x}, \tilde{\mathbf{u}}, \lambda], \tag{14c}$$

with the boundary conditions

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{14d}$$

$$\lambda(T) = \tfrac{\partial}{\partial \mathbf{x}} \ell_T(\mathbf{x}(T)), \tag{14e}$$

where $\mathbf{x}_0$ is some initial state. If additionally (10) holds (in this context known as the *Legendre-Clebsch* condition), the equations take the particularly elegant form

$$\begin{aligned} \dot{\mathbf{x}} &= H_{\lambda}[\mathbf{x}, \mathbf{u}, \lambda] \\ -\dot{\lambda} &= H_{\mathbf{x}}[\mathbf{x}, \mathbf{u}, \lambda] \\ 0 &= H_{\mathbf{u}}[\mathbf{x}, \mathbf{u}, \lambda]. \end{aligned}$$

This formulation clearly illustrates the relationship to classical mechanics and justifies the use of the term "Hamiltonian". However, unlike in mechanics, the Maximum Principle is not an ordinary differential equation but rather a two-point boundary value problem, due to conditions (14d),(14e).

Stochastic dynamics

Stochastic Calculus describes the algebraic properties of continuous random processes, and is a thorny discipline. Below we will only describe the relevant results and provide heuristic arguments to support them. We make use of controlled drift-diffusions, which are a stochastic process described by

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u})dt + G(\mathbf{x}, \mathbf{u})d\omega. \tag{16}$$

$\omega(t) \in \mathbb{R}^k$ is a unit covariance Brownian process and $G \in \mathbb{R}^{n\times k}$ is the noise gain matrix. The drift-diffusion (16) is most easily be understood as the small-Δ limit of the discrete process

$$\mathbf{x}(t+\Delta) - \mathbf{x}(t) = \mathbf{f}(\mathbf{x},\mathbf{u})\Delta + G(\mathbf{x},\mathbf{u})\sqrt{\Delta}\varepsilon.$$

where ε is drawn from a unit gaussian $\varepsilon \sim \mathcal{N}(0, I_k)$. The scaling of the noise term in proportion to the square-root of the timestep $d\omega \propto \sqrt{dt}$, is the central phenomenon of stochastic calculus and can be arrived at from simple random walk arguments[5]. The consequence is that expressions which are quadratic in $d\mathbf{x}$ pick up terms which are linear in dt. Specifically, if $\mathbf{x}$ evolves according to (16), and $v(\mathbf{x})$ is some twice differentiable function, Itō's Lemma, which is the analogue of the chain rule for diffusions, states that

$$\begin{aligned} dv &= v_{\mathbf{x}}^{\mathsf{T}} d\mathbf{x} + \tfrac{1}{2} d\mathbf{x}^{\mathsf{T}} v_{\mathbf{xx}} d\mathbf{x} \\ &= (v_{\mathbf{x}}^{\mathsf{T}}\mathbf{f} + \tfrac{1}{2}\operatorname{tr}(GG^{\mathsf{T}} v_{\mathbf{xx}}))dt + v_{\mathbf{x}}^{\mathsf{T}} G d\omega \end{aligned}$$

It follows that the expected change in $v(t)$ along a trajectory $\mathbf{x}(t)$ is

$$\mathcal{L}[v(\cdot)](\mathbf{x}) \equiv \lim_{\Delta\downarrow 0} \frac{\mathbb{E}_{\mathbf{x}(0)=\mathbf{x}}[v(\mathbf{x}(\Delta))] - v(\mathbf{x})}{\Delta} = v_{\mathbf{x}}^{\mathsf{T}}\mathbf{f} + \tfrac{1}{2}\operatorname{tr}(\Sigma v_{\mathbf{xx}})$$

where $\Sigma(\mathbf{x},\mathbf{u}) \equiv G(\mathbf{x},\mathbf{u})G(\mathbf{x},\mathbf{u})^{\mathsf{T}}$ is the covariance-rate of the diffusion. The operator $\mathcal{L}$ which computes the expected directional derivative is called the *generator* of the diffusion. We now recall the limit which took us from the pseudo-Hamiltonian (4) to the Hamiltonian (8): the expected Value term became the directional derivative in the continuous limit. Clearly the appropriate generalization for diffusions is

$$\begin{aligned} H[\mathbf{x},\mathbf{u},v(\cdot)] &= \ell(\mathbf{x},\mathbf{u}) + \mathcal{L}[v(\cdot)](\mathbf{x}) \\ &= \ell(\mathbf{x},\mathbf{u}) + \mathbf{f}(\mathbf{x},\mathbf{u})^{\mathsf{T}} v_{\mathbf{x}}(\mathbf{x}) + \tfrac{1}{2}\operatorname{tr}(\Sigma(\mathbf{x},\mathbf{u}) v_{\mathbf{xx}}(\mathbf{x})). \end{aligned}$$

[5] A discrete random walk has a binomial distribution with variance proportional to the number of steps. In the continuous limit it becomes a gaussian with variance proportional to time and therefore standard-deviation proportional to the square-root of time.

Linear-Quadratic-Gaussian problems

If the diffusion (16) is linear and the noise is independent of the state and the control $d\mathbf{x} = (A\mathbf{x} + B\mathbf{u})dt + Gd\omega$, and additionally the cost is quadratic $\ell(\mathbf{x}, \mathbf{u}) = \frac{1}{2}\mathbf{x}^\mathsf{T} Q\mathbf{x} + \frac{1}{2}\mathbf{u}^\mathsf{T} R\mathbf{u}$, the problem is known as Linear-Quadratic-Gaussian. It turns out that in this case the differential cost-go-go maintains the quadratic ansatz form $\tilde{v}^*(x) = \frac{1}{2}\mathbf{x}^\mathsf{T} S\mathbf{x}$, and therefore the Itō term is constant, and identifies with the average cost:

$$\tfrac{1}{2}\operatorname{tr}(\Sigma(\mathbf{x}, \mathbf{u}) v_{\mathbf{xx}}(\mathbf{x})) = \tfrac{1}{2}\operatorname{tr}(GG^\mathsf{T} S) = c^*.$$

Due to the noise, the controller can never drive the state to the origin, leading to a non-zero c^*. More importantly, this constant has no effect on the minimization of the Hamiltonian, allowing us to repeat the derivation of the policy and Riccati equation. In other words, the optimal controller in the linear-quadratic setting is *invariant* to Gaussian noise.

2.2 Optimal Control – Algorithms

2.2.1 Discrete State – Global Methods

Value and policy iteration

The two classic methods for the solution of discrete-state problems are an iterative application of the Bellman operator (minimization of the Hamiltonian) until a fixed-point is reached. If the object of the iteration is the Value function, the algorithm is:

Algorithm 1 Value Iteration

```
repeat
    for all $x \in \mathcal{X}$ do
        $v^{k+1}(x) \leftarrow \min_u \bar{H}[x, u, v^k(\cdot)]$
    end for
until convergence
```

Here written using the first-exist variant, it also converges for the others. For the finite-horizon case Value Iteration is a single sweep backwards from $v_N(\cdot)$ to $v_1(\cdot)$. In the discounted case it converges exponentially due to contraction properties related to the discount factor. In the first-exit and average-cost case it converges under mild assumptions (Bertsekas, 2000). If the object is the policy, the algorithm is:

Algorithm 2 Policy Iteration

```
repeat
    for all $x \in \mathcal{X}$ do
        $\pi^{k+1}(x) \leftarrow \text{argmin}_u \bar{H}[x, u, v^{\pi^k}(\cdot)]$
    end for
until convergence
```

Policy Iteration converges after a finite number of steps. Note that the policy-specific value function $v^{\pi^k}(\cdot)$ needs to be evaluated at each step.

Temporal-difference methods

Temporal difference methods (Sutton, 1998) are a group of algorithms that use the residual of the Bellman equation for supervised learning of the Value function, usually in an online, model-free setting. Though provably convergent in some settings (Tsitsiklis and Roy, 2002), like most online algorithms in the flavor of stochastic gradient descent, these methods are less efficient than their batch counterparts, and will therefore not be analyzed here. For a single state transition, the Bellman residual, also known as the *temporal-difference error*

$$\delta = \ell(x, u) - \big(v(x) - v(x')\big) \, ,$$

describes the degree of "cost-surprise" experienced by the agent, as measured by the difference between the measured and expected immediate cost. Outputs of dopaminergic neurons in the Basal Ganglia have been shown to have similar properties to this error signal (Schultz et al., 1997), and remain an important piece of supporting evidence for the use of optimal control models of biological behaviour.

2.2.2 Continuous State – Local Methods

Curse of dimensionality

It is possible to apply global methods to continuous problems by discretizing the state-space and solving the resulting MDP. A relatively efficient implementation replaces basic grid-based discretization with a smart variable-resolution grid as in (Munos and Moore, 2002). Similar attempts have been made to “featurize” large discrete state spaces (Tsitsiklis and Roy, 1996; Lagoudakis and Parr, 2003) using kernel-based and similar methods. Another alternative is to try to approximate the solution of the HJB equation, as described in Chapter 3. Unfortunately, all of these methods appear to suffer from exponential scaling with the state dimension. This phenomenon, known as the *curse of dimensionality*, makes global methods inapplicable to all but the simplest problems.

Discrete-time Maximum Principle

The Maximum Principle is a first-order algorithm in the sense that it requires only first derivatives of the dynamics, but also in the sense that convergence is linear (i.e. like gradient descent). The discrete time dynamics are[6]

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t), \quad t = 0 \dots T-1.$$

The finite-horizon cost-to-go (1), restricted along the trajectory $\{\mathbf{x}_t, \mathbf{x}_{t+1} \dots, \mathbf{x}_T\}$, for the controls $\mathbf{u}_{t..T-1} \equiv \{\mathbf{u}_t, \mathbf{u}_{t+1} \dots, \mathbf{u}_{T-1}\}$ is

$$v_t(\mathbf{x}_t, \mathbf{u}_{t..T-1}) = \sum_{s=t}^{T-1} \ell(\mathbf{x}_s, \mathbf{u}_s) + \ell_T(\mathbf{x}_T).$$

The Bellman equation (3) becomes

$$v_t^*(\mathbf{x}) = \min_{\mathbf{u}} [\ell(\mathbf{x}_t, \mathbf{u}_t) + v_{t+1}^*(\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t))] \tag{17}$$

and identifying $\lambda_t \equiv \frac{\partial}{\partial \mathbf{x}} v_t(\mathbf{x}_t)$, the (pseudo-)Hamiltonian is

$$H[\mathbf{x}_t, \mathbf{u}_t, \lambda_{t+1}] = \ell(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)^{\mathsf{T}} \lambda_{t+1}$$

and the Maximum Principle (14) becomes

$$\begin{aligned} \mathbf{x}_{t+1} &= H_\lambda[\mathbf{x}_t, \mathbf{u}_t, \lambda_{t+1}] = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) && \text{(18a)} \\ \lambda_t &= H_{\mathbf{x}}[\mathbf{x}_t, \mathbf{u}_t, \lambda_{t+1}] = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}(\mathbf{x}_t, \mathbf{u}_t)^{\mathsf{T}} \lambda_{t+1} && \text{(18b)} \\ \mathbf{u}_t &= \operatorname{argmin}_{\mathbf{u}} H[\mathbf{x}_t, \mathbf{u}, \lambda_{t+1}], && \text{(18c)} \end{aligned}$$

[6]In this section t and s subscripts denote time indexing when unambiguous, and primes denote the next time step ($v' \equiv v_{t+1} \equiv v(t+1)$). Subscripts of $\mathbf{x}$ and $\mathbf{u}$ denote differentiation ($v_{\mathbf{x}} \equiv \frac{\partial}{\partial \mathbf{x}} v$).

$$\mathbf{x}_0 = \mathbf{x}_{\text{initial}} \tag{18d}$$

$$\lambda_T = \frac{\partial}{\partial \mathbf{x}} \ell_T(\mathbf{x}_T), \tag{18e}$$

Algorithm 3 Discrete time Maximum Principle

```
initialize: $\mathbf{u}_{0..T-1}$
  repeat
    $\mathbf{x}_0 \leftarrow \mathbf{x}_{\text{initial}}$
    for $t = 0$ to $T - 1$ do
      $\mathbf{x}_{t+1} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$
    end for
    $\lambda_T \leftarrow \frac{\partial}{\partial \mathbf{x}} \ell_T(\mathbf{x}_T)$
    for $t = T - 1$ downto $0$ do
      $\mathbf{u}_t \leftarrow \operatorname{argmin}_{\mathbf{u}} \left[\ell(\mathbf{x}_t, \mathbf{u}) + \mathbf{f}(\mathbf{x}_t, \mathbf{u})^\mathsf{T} \lambda_{t+1}\right]$
      $\lambda_t \leftarrow \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}(\mathbf{x}_t, \mathbf{u}_t)^\mathsf{T} \lambda_{t+1}$
    end for
  until convergence
```

Differential Dynamic Programming

Differential Dynamic Programming (DDP) is a second-order trajectory optimizer introduced in (Mayne, 1966) that requires second derivatives of the dynamics, but also features quadratic convergence (like Newton's Method) near the minimum. We begin with the classic algorithm and then introduce our modifications in Section 2.2.3. Like the Maximum Principle, DDP involves iterative backward and forward passes along the current $(\mathbf{x}_t, \mathbf{u}_t)$ trajectory. In the backward pass a quadratic approximation to $v_t(\mathbf{x})$ and a linear approximation to $\mathbf{u}_t(\mathbf{x})$ is constructed, followed by a forward pass which applies the new control sequence to form a new trajectory. Let Q be the perturbed argument of the minimization in (17) around the t-th $(\mathbf{x}, \mathbf{u})$ pair

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) = {} & \ell(\mathbf{x}_t + \delta\mathbf{x}, \mathbf{u}_t + \delta\mathbf{u}) + v_{t+1}(\mathbf{f}(\mathbf{x}_t + \delta\mathbf{x}, \mathbf{u}_t + \delta\mathbf{u})) \\ & - \left(\ell(\mathbf{x}_t, \mathbf{u}_t) + v_{t+1}(\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t))\right), \end{aligned}$$

and expand to second order

$$\approx \frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}0 & Q_{\mathbf{x}}^{\mathsf{T}} & Q_{\mathbf{u}}^{\mathsf{T}}\\ Q_{\mathbf{x}} & Q_{\mathbf{xx}} & Q_{\mathbf{xu}}\\ Q_{\mathbf{u}} & Q_{\mathbf{ux}} & Q_{\mathbf{uu}}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}. \tag{19}$$

The expansion coefficients are

$$Q_{\mathbf{x}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} v'_{\mathbf{x}} \tag{20a}$$
$$Q_{\mathbf{u}} = \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} v'_{\mathbf{x}} \tag{20b}$$
$$Q_{\mathbf{xx}} = \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} v'_{\mathbf{xx}}\mathbf{f}_{\mathbf{x}} + v'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{xx}} \tag{20c}$$
$$Q_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} v'_{\mathbf{xx}}\mathbf{f}_{\mathbf{u}} + v'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{uu}} \tag{20d}$$
$$Q_{\mathbf{ux}} = \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} v'_{\mathbf{xx}}\mathbf{f}_{\mathbf{x}} + v'_{\mathbf{x}} \cdot \mathbf{f}_{\mathbf{ux}} \tag{20e}$$

Note that the last terms in (20c, 20d, 20e) denote contraction with a tensor. Minimizing (19) WRT $\delta\mathbf{u}$ we have

$$\delta\mathbf{u}^* = \operatorname*{argmin}_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = -Q_{\mathbf{uu}}^{-1}(Q_{\mathbf{u}} + Q_{\mathbf{ux}}\delta\mathbf{x}), \tag{21}$$

giving us an open-loop term $\mathbf{k} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}}$ and a feedback gain term $\mathbf{K} = -Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}$. Plugging the result back in (19), we have a quadratic model of the Value at time t:

$$\Delta v(t) = -\tfrac{1}{2}Q_{\mathbf{u}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{u}} \tag{22a}$$
$$v_{\mathbf{x}}(t) = Q_{\mathbf{x}} - Q_{\mathbf{u}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}} \tag{22b}$$
$$v_{\mathbf{xx}}(t) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}}Q_{\mathbf{uu}}^{-1}Q_{\mathbf{ux}}. \tag{22c}$$

Recursively computing the local quadratic models of v_t and the control modifications $\{\mathbf{k}_t, \mathbf{K}_t\}$, constitutes the backward pass. Once it is completed, a forward pass computes a new trajectory:

$$\hat{\mathbf{x}}_0 = \mathbf{x}_{\text{initial}} \tag{23a}$$
$$\hat{\mathbf{u}}_t = \mathbf{u}_t + \mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t) \tag{23b}$$
$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \tag{23c}$$

2.2.3 Improvements to Differential Dynamic Programming

This section describes small but significant improvements to the DDP algorithm which have been developed by the author in the last several years, but remain unpublished.

Improved Regularization

It has been shown by Liao and Shoemaker (1992) that the steps taken by DDP are comparable to or better than a full Newton step for the entire control sequence. As in Newton's method, care must be taken when the Hessian is not positive-definite or when the minimum is far and the quadratic model inaccurate. The standard regularization, proposed by Jacobson and Mayne (1970) and further explored by Liao and Shoemaker (1991), is to add a diagonal term to the local control-cost Hessian $\widetilde{Q}_{\mathbf{uu}} = Q_{\mathbf{uu}} + \mu\mathbf{I}_m$, where μ plays the role of a Levenberg-Marquardt parameter. This modification amounts to adding a quadratic cost around the current control-sequence, making the steps more conservative. The drawback to this regularization scheme is that the same control perturbation can have different effects at different times, depending on the control-transition matrix $\mathbf{f}_\mathbf{u}$. We therefore introduce a scheme that penalizes deviations from the states rather than controls:

$$\widetilde{Q}_{\mathbf{uu}} = \ell_{\mathbf{uu}} + \mathbf{f}_\mathbf{u}^\mathsf{T}(v'_{\mathbf{xx}} + \mu\mathbf{I}_n)\mathbf{f}_\mathbf{u} + v'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{uu}} \tag{24a}$$

$$\widetilde{Q}_{\mathbf{ux}} = \ell_{\mathbf{ux}} + \mathbf{f}_\mathbf{u}^\mathsf{T}(v'_{\mathbf{xx}} + \mu\mathbf{I}_n)\mathbf{f}_\mathbf{x} + v'_\mathbf{x} \cdot \mathbf{f}_{\mathbf{ux}} \tag{24b}$$

$$\mathbf{k} = -\widetilde{Q}_{\mathbf{uu}}^{-1}\widetilde{Q}_\mathbf{u} \tag{24c}$$

$$\mathbf{K} = -\widetilde{Q}_{\mathbf{uu}}^{-1}\widetilde{Q}_{\mathbf{ux}} \tag{24d}$$

This regularization amounts to placing a quadratic cost around the previous state-sequence. Unlike the standard control-based regularization, the feedback gains $\mathbf{K}$ do not vanish for large μ but rather force the new trajectory closer to the old one, significantly improving robustness.

Finally, we make use of the improved Value update proposed in Todorov and Li (2005). Examining (19, 21, 22), we see that several cancelations of $Q_{\mathbf{uu}}$ and its inverse have taken place, but since we are modifying this matrix in (24), making those cancelations would induce an error in our

Value approximation. The correct Value update is therefore

$$\begin{aligned} \Delta v(t) &= \quad +\tfrac{1}{2}\mathbf{k}^{\mathsf{T}} Q_{\mathbf{uu}}\mathbf{k} + \mathbf{k}^{\mathsf{T}} Q_{\mathbf{u}} && \text{(25a)} \\ v_{\mathbf{x}}(t) &= Q_{\mathbf{x}} \; + \mathbf{K}^{\mathsf{T}} Q_{\mathbf{uu}}\mathbf{k} \; + \mathbf{K}^{\mathsf{T}} Q_{\mathbf{u}} \; + Q_{\mathbf{ux}}^{\mathsf{T}}\mathbf{k} && \text{(25b)} \\ v_{\mathbf{xx}}(t) &= Q_{\mathbf{xx}} + \mathbf{K}^{\mathsf{T}} Q_{\mathbf{uu}}\mathbf{K} + \mathbf{K}^{\mathsf{T}} Q_{\mathbf{ux}} + Q_{\mathbf{ux}}^{\mathsf{T}}\mathbf{K}. && \text{(25c)} \end{aligned}$$

Improved Line Search

The forward pass of DDP, given by Eqs. (23) is the key to the algorithm's fast convergence. This is because the feedback gains in (23b) generate a new control sequence that takes into account the new states as they are being integrated. For example when applying DDP to a linear-quadratic system, even a time-varying one, an exact solution is obtained after a single iteration. The caveat is that for a general non-linear system, when the new trajectory strays too far from the model's region of validity, the cost may not decrease, divergence may occur. The solution is to introduce a backtracking line-search parameter $0 < \alpha \le 1$ and integrate using

$$\hat{\mathbf{u}}_t = \mathbf{u}_t + \alpha\mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t)$$

For $\alpha = 0$ the trajectory would be unchanged, but for intermediate values the resulting control step is not a simple scaled version of the full step, due to the presence of feedback. As advocated in Jacobson and Mayne (1970), we use the expected total-cost reduction in the line-search procedure, with two differences. The first is that we use the improved formula (25a) rather than (22a) for the expected reduction:

$$\Delta v(\alpha) = \alpha \sum_{t=1}^{T-1} \mathbf{k}_t^{\mathsf{T}} Q_{\mathbf{u}_t} + \frac{\alpha^2}{2} \sum_{t=1}^{T-1} \mathbf{k}_t^{\mathsf{T}} Q_{\mathbf{uu}t}\mathbf{k}_t.$$

The second is that when comparing the actual and expected reductions

$$z = [v(\mathbf{u}_{1..T-1}) - v(\hat{\mathbf{u}}_{1..T-1})]/\Delta v(\alpha),$$

we accept the iteration only if

$$0 < c_1 < z < c_2 < \infty.$$

The unintuitive upper limit c_2 is used because sometimes the improvement is due to the initial trajectory being particularly bad, and the new trajectory has jumped from one bad region of state-space to another slightly-less-bad region, often introducing undesired features and landing in a local minimum. This situation might be likened to encountering a vertical drop when trying to get down a mountain. Jumping off might decrease one's altitude quickly in short run, but might be a bad idea with regards to reaching the bottom.

Algorithm 4 Differential Dynamic Programming (improved)

```
initialize: nominal $\mathbf{u}_{0..T-1}$ and $\mathbf{x}_{0..T}$
  repeat
        $v(T) \leftarrow \ell_T(\mathbf{x}_T)$
        $v_{\mathbf{x}}(T) \leftarrow \frac{\partial}{\partial \mathbf{x}} \ell_T(\mathbf{x}_T)$
        $v_{\mathbf{xx}}(T) \leftarrow \frac{\partial^2}{\partial \mathbf{x}^2} \ell_T(\mathbf{x}_T)$
      for $t = T-1$ downto $0$ do      // backward pass
          $Q_{\mathbf{x}}, Q_{\mathbf{u}}, Q_{\mathbf{xx}}, Q_{\mathbf{uu}}, Q_{\mathbf{ux}} \leftarrow$ Eq. 20
          $\widetilde{Q}_{\mathbf{uu}}, \widetilde{Q}_{\mathbf{ux}}, \mathbf{k}, \mathbf{K} \leftarrow$ Eq. 24
          if $\widetilde{Q}_{\mathbf{uu}}$ is not positive-definite then
              increase $\mu$, restart backward pass
          end if
          $\Delta v(t), v_{\mathbf{x}}(t), v_{\mathbf{xx}}(t) \leftarrow$ Eq. 25
      end for
      decrease $\mu$
      $\alpha \leftarrow 1$
      $\hat{\mathbf{x}}_0 \leftarrow \mathbf{x}_{\text{initial}}$
      for $t = 0$ to $T-1$ do      // forward pass
          $\hat{\mathbf{u}}_t \leftarrow \mathbf{u}_t + \alpha \mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t)$
          $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t)$
      end for
      if $c_1 < z < c_2$ then      // accept changes
          $\mathbf{u}_{0..T-1} \leftarrow \hat{\mathbf{u}}_{0..T-1}$
          $\mathbf{x}_{0..T} \leftarrow \hat{\mathbf{x}}_{0..T}$
      else
          decrease $\alpha$, restart forward pass
      end if
  until convergence
```

Chapter 3

Neural-Network representation of the Value Function

Least Squares Solutions of the HJB Equation With Neural Network Value-Function Approximators

Yuval Tassa and Tom Erez

***Abstract*—In this paper, we present an empirical study of iterative least squares minimization of the Hamilton–Jacobi–Bellman (HJB) residual with a neural network (NN) approximation of the value function. Although the nonlinearities in the optimal control problem and NN approximator preclude theoretical guarantees and raise concerns of numerical instabilities, we present two simple methods for promoting convergence, the effectiveness of which is presented in a series of experiments. The first method involves the gradual increase of the horizon time scale, with a corresponding gradual increase in value function complexity. The second method involves the assumption of stochastic dynamics which introduces a regularizing second derivative term to the HJB equation. A gradual reduction of this term provides further stabilization of the convergence. We demonstrate the solution of several problems, including the 4-D inverted-pendulum system with bounded control. Our approach requires no initial stabilizing policy or any restrictive assumptions on the plant or cost function, only knowledge of the plant dynamics. In the Appendix, we provide the equations for first- and second-order differential backpropagation.**

***Index Terms*—Differential neural networks (NNs), dynamic programming, feedforward neural networks, Hamilton–Jacoby–Bellman (HJB) equation, optimal control, viscosity solution.**

I. Introduction

The field of optimal control is concerned with finding the control law which, applied to a given dynamical system, will minimize some performance index, usually the temporal integral of an incurred cost. One way of solving this problem involves the computation of the value function, a measurement of the performance index as a function of space and time. In the continuous case considered here, the value function satisfies a nonlinear partial differential equation called the Hamilton–Jacobi–Bellman (HJB) equation. In the simple case of linear dynamics and quadratic costs, this equation famously reduces to the matrix Riccati equation, which can be accurately solved by analytical or numerical methods.

In the general case, the HJB equation has proven difficult to solve. One reason for this is that value functions are frequently differentiable only almost everywhere, requiring the framework of nonsmooth analysis and viscosity solutions [1], so that a weak sense of solution can first be defined and then shown to exist. Even when a solution is guaranteed to exist, the task of finding it is not made any more tractable. When using approximation schemes, artifacts or hidden extrapolation introduced by an imperfect approximator can couple with the nonlinearity inherent in the minimization operator [see (3)] and produce divergent feedback phenomena [2]. Finally, the computational complexity required to describe the value function grows exponentially with the dimension of its domain (“curse of dimensionality”), limiting the feasibility of an effective approximation. These difficulties in solving the general case have led many research efforts towards finding those special cases for which a suitably designed technique can be proven to always work.

Feedforward neural networks (NNs) provide a generic framework for smooth function approximation, with a sound theoretical foundation [3] and a vast knowledge based on their implementation in terms of architectures and learning algorithms. Previous studies (e.g., [4] and [5]; see Section II) have shown the potential of NN approximation of value functions for special cases.

In this paper, we study a straightforward approach of performing least squares minimization of the HJB residual with an NN approximation of the value function. This approach is attractive because it is simple to implement, requires few *a priori* assumptions and can be applied to almost any problem, including nonlinear problems which frustrate solution by classical methods of control theory. Deemed prone to numerical instabilities and divergent phenomena, this course is rarely taken in practice. Instead of retreating to a limited set of cases where convergence can be guaranteed due to some restrictive property, we wish to propose here two methods to overcome the drawbacks of the unconstrained problem. Intended as a proof-of-concept, this paper provides no formal convergence analysis, but rather presents a series of successful experiments and argues that these methods are likely to promote convergence in most cases. In the rest of this section, we review related work and the required theoretical background. In Section II, we present the details of the framework and our proposed methods for promoting convergence and discuss their motivation. Section III describes the results of numerical experiments with one linear quadratic and two nonlinear optimal control problems, and finally, Section IV contains a conclusion of the presented work and some possible ramifications. The Appendix provides the equations for differential backpropagation in feedforward NNs.

A. Related Work

Research regarding value function approximation has been done in several academic disciplines. When the dynamics are

Manuscript received December 1, 2005; revised September 20, 2006; accepted February 5, 2007.

Y. Tassa is with The Interdisciplinary Center for Neural Computation, The Hebrew University, Jerusalem 91904, Israel (e-mail: tassa@alice.nc.huji.ac.il; yuval.tassa@gmail.com; tassa@pob.huji.ac.il).

T. Erez is with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: etom@cse.wustl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.899249

a discrete Markov decision process, the dynamic programming [6] framework provides a rigorous environment for the description and analysis of algorithms. Though extensions to continuous time and space have been made [7], these involve approximating the value function around a single optimal trajectory rather than in a volume of the state space.

In the reinforcement learning (RL) branch of computational learning, the optimal control problem is recast as interaction between an agent (the controller) and an environment (the plant), with the negative cost (usually denoted as $\mathcal{L}$) considered as a reward (denoted r) which the agent strives to maximize. This perspective induces several departures from the control theoretic approach, notably, a bias towards online methods and minimal assumed prior knowledge of the environmental dynamics, e.g., in temporal-difference (TD) methods [8]. This focus on online methods, which might be attributed to the biological plausibility of similar processes taking place in nervous systems [9], biases the application of gradient-based methods towards stochastic gradient descent. In this paper, we show how high condition numbers of the error Hessian give second-order batch methods a distinct computational advantage. Least squares temporal difference (LSTD) and related methods [10] comprise a class of RL algorithms which do use batch learning. The work presented in this paper can be considered an extension of these methods to the continuous case.

When solving continuous problems, most work in RL has concentrated on discretization of the state space and subsequent solution by dynamic programming or related methods [11], probably due to the availability of powerful analytical tools for the discrete case. An exception is Doya's generalization of TD-λ to continuous problems [12] using radial-basis functions. This approach was subsequently applied by Coulom [5] using sigmoidal NNs. In Coulom's work, though arguably a most impressive use of NNs for value function estimation, the learning is computationally intensive and rather unstable, with nonmonotonic convergence. Besides the work of Coulom, mixed results have been reported regarding the application of NNs to value function approximation. After the initial success of Tesauro [13], most papers reported unsatisfactory results [2], [14], [15].

In the control community, the general control problem is usually defined to be that of forcing the output of some dynamical system to follow a given desired signal. By considering the difference between the current and desired states, the problem is recast as bringing this difference to the origin. This assumption, which in optimal control manifests as constraints on the cost function to be zero at the origin and positive elsewhere, conceals the possibilities afforded by other types of problems. For example, Coulom [5] uses a cost function which is linear in the velocity to design "swimmers" whose controllers give rise to limit cycle, rather than stabilizing, dynamics. Another meta-constraint on control theoretic research, due to the applied nature of the field, is the legitimate emphasis on provably convergent algorithms.

Value function approximation has not been a popular method for nonlinear control design, perhaps due to the availability of other powerful methods. An important exception is [16], which uses a Galerkin expansion to construct the value function, but requires computationally intensive integrations. Other current state-of-the-art approximation approaches include adaptive meshing [11] and level sets [17].

Though extensive use of NNs has been performed in the control theory literature [18], they have been used rarely for value function approximation. In cases where they have been used, only special cases of the problem had been considered. Goh [19] used an NN to solve a nonlinear quadratic optimal regulator problem. Several assumptions were made, including a quadratic cost function and the origin being a fixed point of the dynamics, which allow the author to assume a sum-of-squares structure for the value function. Abu-Khalaf and Lewis [4] assumed a convex cost and approximated the value function using a linear function approximator with polynomial basis functions. By accepting these restrictions, they enjoyed the guarantees conferred by a unique minimum. This paper effectively demonstrates how the approaches of [19] and [4] can be extended to general feed-forward NNs and general nonlinear optimal control problems. Although provable convergence is lost, general applicability and *de facto* stability are gained.

A necessary step in the use of NNs as approximate solutions to differential equations is the computation of the derivatives of the output of the network with respect to (w.r.t.) its input variables, and the derivatives of these values w.r.t. the weights of the network. Although the equations for computing these quantities have appeared before, either implicitly [20] or explicitly [5], [21], [22], we feel that a clearer derivation is required to do justice to their great usefulness, and include them in the Appendix.

B. HJB Equation

Consider a state vector $\mathbf{x} \in X \subseteq \mathbb{R}^d$ of some dynamical system which evolves according to $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, with $\mathbf{u} \in U \subseteq \mathbb{R}^m$ being a control signal. Both X and U are assumed to be bounded compact sets, with U convex. Given some scalar cost $\mathcal{L}(\mathbf{x}, \mathbf{u})$, our goal is to find the control law or policy $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$, which will minimize the cost[1] incurred along the trajectory $\mathbf{x}(t)$ as measured by an integral performance index called the value function

$$V^u(\mathbf{x}) = \int_t^\infty \frac{1}{\tau} e^{-\frac{t'-t}{\tau}} \mathcal{L}\left(\mathbf{x}(t'), \mathbf{u}\left(\mathbf{x}(t')\right)\right) dt'. \tag{1}$$

Putting ∞ as the limit of the integral makes the value and policy independent of t. The exponential discounting function $e^{-(t'-t/\tau)}$ ensures convergence and determines the horizon time scale. The coefficient $1/\tau$ is not essential to the exposition but was added because it normalizes the exponential integrand and fixes V to be in the same units as $\mathcal{L}$. Specifically, for $\tau \to 0$, the normalized exponent shrinks to a δ-function and $V \to \mathcal{L}$. Our goal is to find a policy for which the value function is minimized. This value function, denoted V^*, is called the optimal value function

$$V^*(\mathbf{x}) = \min_{\mathbf{u}(\mathbf{x})} \left[V^u(\mathbf{x})\right].$$

[1]In the formally equivalent continuous reinforcement learning framework, a corresponding reward is maximized.

By direct differentiation w.r.t. t, any function which satisfies (1), including V^*, satisfies the linear differential relation

$$\tau \frac{d}{dt} V(\mathbf{x}) = V(\mathbf{x}) - \mathcal{L}(\mathbf{x}, \mathbf{u})$$

or

$$V(\mathbf{x}) = \tau \dot{V}(\mathbf{x}) + \mathcal{L}(\mathbf{x}, \mathbf{u}) = \tau \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathcal{L}(\mathbf{x}, \mathbf{u}). \quad (2)$$

Now, assume we have found V^* and invoke the minimum principle: Since V^* is already optimal w.r.t. $\mathbf{u}$, any perturbation to $\mathbf{u}$ must necessarily increase the $\mathbf{u}$-dependent right-hand side

$$V^*(\mathbf{x}) = \min_{\mathbf{u}} \left[\tau \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathcal{L}(\mathbf{x}, \mathbf{u}) \right]. \quad (3)$$

This nonlinear partial differential equation (PDE) is called the HJB equation.[2]

C. Solutions of the HJB Equation

For linear dynamics and quadratic costs, the HJB equation famously reduces to the Riccati matrix equation of the linear quadratic regulator. For general problems, however, V^* is not differentiable everywhere and the equation does not hold in the classical sense. The usual framework for analyzing nonsmooth solutions to HJB equations is the formalism of viscosity solutions. A way to avoid this is to introduce some stochasticity in the state dynamics $d\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u})dt + \mathbf{n}dt$ with $\mathbf{n}$ denoting a Brownian motion term of covariance $\mathbf{N}$. In this case, the modified HJB equation becomes

$$V^*(\mathbf{x}) - \frac{\tau}{2} \mathrm{tr} \left(\mathbf{N} \cdot \frac{\partial^2 V^*(\mathbf{x})}{\partial \mathbf{x}^2} \right) = \min_{\mathbf{u}} \left[\tau \frac{\partial V^*(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathcal{L}(\mathbf{x}, \mathbf{u}) \right] \quad (4)$$

which provides a regularizing effect that guarantees that the value function is differentiable everywhere [23].

When attempting to solve the HJB equation in some finite compact volume of space $X \subseteq \mathbb{R}^d$, boundary conditions on ∂X must be considered. For the integral (1) to be defined, the trajectory $\mathbf{x}(t)$ must either remain in X or, upon reaching the boundary, incur some terminal cost. When stabilizing controllers are sought in the control theoretic context, the concept of a Lyapunov function is used to ensure that trajectories are restricted to a domain. In the general nonlinear context, where such techniques are unavailable, boundary conditions must be dealt with explicitly.

D. Value Iteration

Given any value function $V(\mathbf{x})$, a policy which minimizes the RHS of (3)

$$\mathbf{u}^{\mathrm{G}}(\mathbf{x}) = \underset{\mathbf{u}}{\mathrm{argmin}} \left[\tau \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathcal{L}(\mathbf{x}, \mathbf{u}) \right] \quad (5)$$

is called a greedy policy. Assuming $\mathcal{L}(\mathbf{x}, \mathbf{u}) = \mathcal{L}_x(\mathbf{x}) + \mathcal{L}_u(\mathbf{u})$, if $\mathcal{L}_u(\mathbf{u})$ and $\mathbf{f}(\mathbf{u})$ are differentiable, and $\partial \mathcal{L}_u / \partial \mathbf{u}$ invertible, then the minimization in (3) can be reduced to differentiating w.r.t. $\mathbf{u}$, equating to zero and solving for $\mathbf{u}$. If the dynamics are affine in the control $\dot{\mathbf{x}} = \mathbf{f}_x(\mathbf{x}) + \mathbf{f}_u(\mathbf{x}) \cdot \mathbf{u}$,[3] then a sufficient condition for the invertibility of $\partial \mathcal{L}_u / \partial \mathbf{u}$ is the Legendre–Clebsch condition (convexity of $\mathcal{L}_u$)

$$\frac{\partial^2}{\partial \mathbf{u}^2} \left(\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathcal{L}(\mathbf{x}, \mathbf{u}) \right) = \frac{\partial^2}{\partial \mathbf{u}^2} \mathcal{L}_u(\mathbf{u}) > 0.$$

The greedy policy is then given in closed form by

$$\mathbf{u}^{\mathrm{G}} = - \frac{\partial \mathcal{L}_u}{\partial \mathbf{u}}^{-1} \left(\tau \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{u}} \right)^{\mathrm{T}}. \quad (6)$$

The process of iteratively forming a value function for a given policy, and then, deriving a new greedy policy with respect to the new value function, is called value iteration or method of approximations, and has been shown to converge when exact measurements are possible [24]. When the object of the iterative improvement is the policy rather than the value function, the process is called policy iteration. In cases such as this one, where the greedy policy is a deterministic closed-form function of the value, the value- and policy-iteration algorithms become nearly indistinguishable.

E. Differential NNs

The Pineda architecture [25] is a generalized topology for feedforward NNs which can generate layered and other topologies as special cases. Given a feedforward network V, an input vector $\mathbf{x}$, and a weight vector $\mathbf{w}$, the Appendix shows how to calculate the following quantities in the Pineda formalism:

$$\begin{array}{cc} V(\mathbf{x}, \mathbf{w}) & \frac{\partial}{\partial \mathbf{w}} V(\mathbf{x}, \mathbf{w}) \\ \frac{\partial}{\partial \mathbf{x}} V(\mathbf{x}, \mathbf{w}) & \frac{\partial}{\partial \mathbf{w}} \frac{\partial}{\partial \mathbf{x}} V(\mathbf{x}, \mathbf{w}) \\ \frac{\partial^2}{\partial \mathbf{x}^2} V(\mathbf{x}, \mathbf{w}) & \frac{\partial}{\partial \mathbf{w}} \frac{\partial^2}{\partial \mathbf{x}^2} V(\mathbf{x}, \mathbf{w}). \end{array}$$

The quantities on the left are computed by "forward" propagation while the values on the right are computed by "back" propagation, using intermediate values obtained in the forward pass. We used standard $\tanh$ sigmoids but the nonlinearity can belong to any class of smooth functions [radial basis function (RBF), polynomial, trigonometric, etc.].

F. Least Squares

The approach taken here to approximating a solution to (3) is to minimize the square of the left-hand side (LHS) minus the RHS, called the residual or error. More generally, given an error vector $\mathbf{e} \in \mathbb{R}^n$, which is a function of a weight vector $\mathbf{w} \in \mathbb{R}^q$, the object of least squares minimization is to find $\mathbf{w}_{\mathrm{LS}} = \arg\min_{\mathbf{w}} [\mathbf{e}^{\mathrm{T}} \mathbf{e}]$. The Gauss–Newton approximation takes the first-order expansion of $\mathbf{e}$ in $\mathbf{w}$

$$\mathbf{e}_{\mathbf{w}} = \mathbf{e} + \mathbf{J}\mathbf{w} + O(\mathbf{w}^2)$$

[2]This semiformal derivation is intended to provide an intuition of the problem. For rigorous results, see, e.g., [23].

[3]As is usually the case in mechanical systems.

where $\mathbf{J} = \partial\mathbf{e}/\partial\mathbf{w}$ is the $n \times q$ Jacobian matrix. The square error is then given by the quadratic

$$\mathbf{e}_\mathbf{w}^\mathrm{T}\mathbf{e}_\mathbf{w} = \mathbf{e}^\mathrm{T}\mathbf{e} + 2\mathbf{w}^\mathrm{T}\mathbf{J}^\mathrm{T}\mathbf{e} + \mathbf{w}^\mathrm{T}\mathbf{J}^\mathrm{T}\mathbf{J}\mathbf{w} + O(\mathbf{w}^3)$$

which is minimized by $\Delta\mathbf{w}_\mathrm{GN} = -(\mathbf{J}^\mathrm{T}\mathbf{J})^{-1}\mathbf{J}^\mathrm{T}\mathbf{e}$. In the Levenberg–Marquardt variation, the approximated Hessian $\mathbf{J}^\mathrm{T}\mathbf{J}$ is conditioned by a scaled identity matrix[4] with a positive factor μ

$$\Delta\mathbf{w}_\mathrm{LM} = -(\mathbf{J}^\mathrm{T}\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^\mathrm{T}\mathbf{e}. \tag{7}$$

Of the various heuristics for controlling μ during minimization, we use the one suggested in [26].

II. Proposed Method

In this section, we first present a straightforward implementation of the techniques presented previously. Then, we describe some common causes for approximation failure, and afterwards, suggest two methods that circumvent many of these causes.

A. Naive Implementation

The naive approach to least squares minimization of the HJB residual is the following. We take as an input n points $\mathbf{x}_{i=1\ldots n} \in X$, and an NN approximation of the value function $V(\mathbf{x}, \mathbf{w})$ with a weight vector $\mathbf{w}$ initialized using any standard weight initialization scheme (e.g., [27]). Then, the algorithm repeatedly performs Levenberg–Marquardt steps on the squared HJB residual for all points simultaneously until a local minimum is reached.

Algorithm: *Naive* $(\mathbf{x}_{i=1\ldots n} \in X, \mathbf{w}_\mathrm{initial})$

```
1) repeat
2)   for $i \leftarrow 1$ to $n$
3)     do $\mathbf{u}_i^\mathrm{G} \leftarrow -(\partial\mathcal{L}_u/\partial\mathbf{u})^{-1}(\tau(\partial V(\mathbf{x}_i,\mathbf{w})/\partial\mathbf{x}) \cdot(\partial\mathbf{f}(\mathbf{x}_i)/\partial\mathbf{u}))^\mathrm{T}$
4)        $\dot{\mathbf{x}}_i \leftarrow \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i^\mathrm{G})$
5)        $e_i \leftarrow V(\mathbf{x}_i,\mathbf{w}) - \tau(\partial V(\mathbf{x}_i,\mathbf{w})/\partial\mathbf{x}) \cdot \dot{\mathbf{x}}_i - \mathcal{L}(\mathbf{x}_i, \mathbf{u}_i^\mathrm{G})$
6)        $(\partial e_i/\partial\mathbf{w}) \leftarrow (\partial/\partial\mathbf{w})V(\mathbf{x}_i,\mathbf{w}) - \tau(\partial/\partial\mathbf{w})(\partial/\partial\mathbf{x}) \times V(\mathbf{x}_i,\mathbf{w}) \cdot \dot{\mathbf{x}}_i$
7)   $\mathbf{e} \leftarrow (e_1, e_2, \ldots e_n)^\mathrm{T}$
8)   $\mathbf{J} \leftarrow ((\partial e_1/\partial\mathbf{w}), (\partial e_2/\partial\mathbf{w}), \ldots (\partial e_n/\partial\mathbf{w}))^\mathrm{T}$
9)   $\mathbf{w} \leftarrow \mathbf{w} - (\mathbf{J}^\mathrm{T}\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^\mathrm{T}\mathbf{e}$
10)    $\mu \leftarrow \mu_\mathrm{new}$ as in [26]
11) until $\Delta(\mathbf{e}^\mathrm{T}\mathbf{e}) < \epsilon$
12) return the weight vector $\mathbf{w}$
```

[4]The variation which takes the diagonal of $\mathbf{J}^\mathrm{T}\mathbf{J}$ (à la Marquardt) rather than the identity (à la Levenberg) was tried and found to be less stable.

Note that we assume we can compute $(\partial\mathbf{f}(\mathbf{x},\mathbf{u})/\partial\mathbf{u})$ and $(\partial\mathcal{L}_u/\partial\mathbf{u})^{-1}$ [step 3)], $\mathbf{f}(\mathbf{x},\mathbf{u})$ [step 4)], and $\mathcal{L}(\mathbf{x},\mathbf{u})$ [step 5)]. Additionally, apart from the obvious requirement to calculate $V(\mathbf{x},\mathbf{w})$ and $\partial V(\mathbf{x},\mathbf{w})/\partial\mathbf{x}$ [steps 3) and 5)], we need the derivatives of these quantities w.r.t. $\mathbf{w}$ [step 6)]. The Appendix explains in detail how these quantities are calculated. Note that since we never actually simulate the plant dynamics, we do not have to determine a time step Δt, nor do we have to deal with the accuracy issues which arise in numerical integration.

B. Distribution of Points and Conditioning

One of our main results is the extremely wide range of sensitivities of the squared error $\mathbf{e}^\mathrm{T}\mathbf{e}$ to changes in $\mathbf{w}$. In all our experiments, the condition number of the approximated Hessian $\mathbf{J}^\mathrm{T}\mathbf{J} + \mu\mathbf{I}$ regularly exceeded 10^{10}. This fact is made even more dramatic when we realize that the lower eigenvalues of this matrix are dominated by the condition factor μ and can never approach 0. We feel that this explains why methods based on stochastic gradient descent do not perform well.

Another consequence of this sensitivity is the choice of a fixed set of points $\mathbf{x}_i$ during the learning. We initially experimented with resampling the point set before each iteration or collecting the training points along trajectories of a behaving system, but both approaches turned out to introduce too much noise and prevent good convergence. In all our experiments, the point set was drawn from the quasi-random Halton sequence [28], which provided consistently better results than sampling from a uniform distribution, though not by a large margin.

C. Boundary Conditions

The definition of the value function (1) as an integral into the "future" requires special attention at the boundaries of $X \subseteq \mathbb{R}^d$, the compact set over which we wish to approximate V. The points on ∂X, the boundary surface of $\mathbf{X}$, can be divided into the following three distinct categories

∂X_1	Regions where $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ points into X for all $\mathbf{u}$. These regions require no special attention since the integral (1) is well defined.
∂X_2	Regions where $\mathbf{f}(\mathbf{x},\mathbf{u})$ points out of X for all $\mathbf{u}$. Because the integral is no longer defined, the HJB equation does not hold and a terminal cost $\mathcal{L}_T(\mathbf{x})$ must be incurred. In practice, this "clamping" constraint $V(\mathbf{x}) = \mathcal{L}_T(\mathbf{x})$ is enforced by scattering n_b points $\mathbf{x}_{j=1\ldots n_b} \in \partial X_2$ for which the error is defined to be $e_j = (V(\mathbf{x}_j) - \mathcal{L}_T(\mathbf{x}_j))$ rather than the usual HJB residual.
∂X_3	Regions where $\mathbf{f}(\mathbf{x},\mathbf{u})$ points into X for some values of $\mathbf{u}$ and out of X for other values. These regions are the most problematic for our purposes and can sometimes be avoided by a careful choice of X. The solution we used[5] was to modify the dynamics so that if $\mathbf{f}(\mathbf{x},\mathbf{u})$ points out of X, the component perpendicular to ∂X_3 is discarded and only the parallel component is retained. It should be noted that the resulting discontinuities in $\mathbf{f}$ do not invalidate the existence of HJB solutions.

D. Causes of Divergence

There are several difficulties associated with the approximate numerical solution of the HJB equation. First are multiple solutions admitted by the equation. These are either the result of the aforementioned discontinuities allowed in the solution (see [14] for examples) or the nonuniqueness which results when the minimization in (5) is realized as extremization in (6). This type of nonuniqueness is epitomized by the Riccati equation having two solutions, a positive–definite and a negative–definite one (see Section III-A).

Second are the positive feedback effects brought about by V and $\partial V/\partial \mathbf{x}$ having the same sign in (3). Because $\mathbf{u}^{\mathrm{G}}$ is a function of $\partial V/\partial \mathbf{x}$ in (6), greedy value iteration of type proposed here can lead to divergent phenomena. One type is the classical "rattling" effect, where repeated overshooting of the minimum leads to divergence. In our approach, this effect is mostly controlled by the Levenberg–Marquardt parameter μ. Another type of positive feedback phenomena emerges when using a nonlinear function approximator. Local biases in the approximation or "hidden extrapolation," especially at difficult-to-approximate discontinuous boundaries, can lead to false solutions (see, e.g., Fig. 4).

E. Promoting Convergence by Gradual Complexification

The both two methods proposed in the following involve the gradual modification of some parameter during the convergence loop [i.e., before every **repeat** $\rightarrow$ **until** iteration, steps 2)–10) in the algorithm], so that the problem is initially an "easier" variant which then progressively approaches the full problem. Although their success is not guaranteed and no general convergence proofs are provided, we show how these techniques enable us to address a wide range of problems with considerable success.

F. Modifying the Horizon Time Scale

Our first method involves the modification of the horizon time scale τ. An inspection of (3) leads to an interpretation of τ as the relative weighting coefficient of instantaneous and future costs. As mentioned previously, the introduction of the normalizing coefficient $1/\tau$ in (1) fixes the units of V to be those of $\mathcal{L}$. Specifically for $\tau = 0$, we have $V = \mathcal{L}$ and the problem reduces to simple function approximation. These insights motivate the following method: First, train the NN approximator with $\tau = 0$ until $V(\mathbf{x}, \mathbf{w})$ has converged to $\mathcal{L}$, and then, increase τ in small increments until the desired value is reached.

For linear–quadratic (LQR) problems, where more than one solution to the HJB equation is possible, starting with $\tau = 0$ is equivalent to starting the convergence process from within the positive–definite cone, thus avoiding the negative–definite solution (see Section III-A). Finally, it is important to note that this method is not appropriate for all types of problems. In many cases, especially when the goal is to bring the state into some target set, $\mathcal{L}$ is discontinuous in X while V is continuous. In these cases, the approximation is actually more difficult for small values of τ, and no advantage is gained by its gradual incrementation.

[5]Due to Rémi Munos.

G. Modifying the Stochastic Term

As described in Section I-C, the assumption of stochastic dynamics adds a second derivative term to the HJB equation and has a smoothing effect on the value function. Our second method for promoting convergence involves the gradual reduction of this term.[6] Intuitively, boundaries across which the value function is discontinuous for deterministic dynamics become fuzzy with stochasticity as noise may push the dynamics from one region to another. Besides the dispensation with the formal requirement for "viscosity solutions," smooth functions are easier to approximate and are far less susceptible to the destructive effects of local irregularities, which may form when a continuous function approximates a discontinuous one (as in Gibbs oscillations). In practice, all we need to do to enjoy the benefits of smoothness is to modify step 5) of the algorithm to be

$$e_i \leftarrow V(\mathbf{x}_i, \mathbf{w}) - \tau \frac{\partial V(\mathbf{x}_i, \mathbf{w})}{\partial \mathbf{x}} \cdot \dot{\mathbf{x}}_i - \mathcal{L}\left(\mathbf{x}_i, \mathbf{u}_i^{\mathrm{G}}\right) - \frac{\tau\gamma}{2} \mathrm{tr}\left(\frac{\partial^2 V(\mathbf{x}_i)}{\partial \mathbf{x}^2}\right)$$

which is equivalent to the assumption of white spherical process noise of variance $\mathbf{N} = \gamma \mathbf{I}$. The appropriate derivatives must also be computed when calculating $\partial \mathbf{e}_i/\partial \mathbf{w}$ in step 6)

$$\frac{\partial e_i}{\partial \mathbf{w}} \leftarrow \frac{\partial}{\partial \mathbf{w}} V(\mathbf{x}_i, \mathbf{w}) - \tau \frac{\partial}{\partial \mathbf{w}} \frac{\partial}{\partial \mathbf{x}} V(\mathbf{x}_i, \mathbf{w}) \cdot \dot{\mathbf{x}}_i - \frac{\tau\gamma}{2} \frac{\partial}{\partial \mathbf{w}} \mathrm{tr}\left(\frac{\partial^2 V(\mathbf{x}_i)}{\partial \mathbf{x}^2}\right).$$

It is important to note that while our calculations assume stochastic dynamics and enjoy the benefits of a smooth, differentiable value function, no actual noise is injected anywhere and the algorithm remains deterministic. Another benefit relative to regularization schemes of the Tikhonov, or similar types, is that rather than directly penalizing weights or nonsmoothness as an extra term in the squared error, this type of smoother acts inside the residual term and actually decreases the squared error by orders of magnitude. While these benefits are obvious, we pay a price: A sharp edge in the surface of the value function is often meaningful as a "switching curve," and smoothing it could result in a suboptimal policy, or even the system's failure to achieve its goal. Therefore, we end up with a logic similar to the previous method: Begin with large γ value for improved convergence during the initial approximation iterations, and decrease it gradually through the iterations until it reaches 0 or some other small value.

H. Scheduling and Combining the Methods

For both parameters, we found that parameter modification was best implemented with a sigmoidal schedule, to achieve a converge-track-converge behavior. In the first few iterations, the

[6]The idea of adding this term, hinted at in [29], was suggested to the authors by R. Munos.

values of τ or γ are kept fixed, so that the approximator can converge from its initial conditions to the "easy" problem. Then, the parameter changes gradually while the approximator tracks the resulting changes in the value function. Finally, the parameter remains fixed for the last iterations, allowing maximum convergence for the "hard" problem. The exact rate of change of the parameter during the tracking phase is immaterial, as long as it is slow enough for consecutive realizations of the value function to be good approximations of each other.

When using both methods concurrently, we found that an effective approach is to first let τ increase while γ is held large and constant and then decrease γ only after τ has plateaued. Intuitively, this is because the divergent phenomena described previously are all related to the predictive information contained in V as parameterized by τ. The smoothness benefits arising from γ are used to robustify the convergence while τ is increasing, and should, therefore, be decreased only once τ no longer changes.

III. Experiments

In this section, we demonstrate the proposed method on three optimal control problems. First, we describe the linear quadratic problem, derive the Riccati equation for the discounted horizon case, and discuss the relationship between the horizon time scale and stability. Our solution of a 4-D linear quadratic system is then compared to an accurate solution obtained by standard methods. Next, we present results for the 2-D car-on-the-hill problem. We show how discontinuity of the value function can lead to local divergence, and proceed to use the smoothing parameter γ to resolve this issue. Finally, the 4-D cart-pole problem is described and solved. All experiments were carried out using Pineda-type NNs with $\tanh$ sigmoids, as described in the Appendix.[7] Running time complexity is $O(nq)$ with n the number of sample points and q the number of weights. Experiments were performed on a 3-GHz Pentium 4 CPU.

In both of the nonlinear problems, we follow [12] and [30] and implement a bounded control $|u| < u_{\mathrm{MAX}}$ by using the convex cost function

$$\mathcal{L}_u(u) = c\int_0^u \tanh^{-1}\left(\frac{y}{u_{\max}}\right)dy$$

which, inserted in (6), results in the control law

$$u^{\mathrm{G}} = -u_{\max}\tanh\left(\frac{\tau}{c}\frac{\partial V(\mathbf{x})}{\partial \mathbf{x}}\cdot\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right)^{\mathrm{T}}. \tag{8}$$

We use the terminology of reinforcement learning (reward maximization) rather than control (cost minimization), when doing so aids with the comparison to previous work.

A. Linear Quadratic System

Let us derive the discounted Riccati equation. For linear dynamics $\mathbf{f}(\mathbf{x},\mathbf{u}) = \mathbf{Ax}+\mathbf{Bu}$ and quadratic costs $\mathcal{L}(\mathbf{x},\mathbf{u}) = \mathbf{x}^{\mathrm{T}}\mathbf{Qx}+\mathbf{u}^{\mathrm{T}}\mathbf{Ru}$ with $(\mathbf{A},\mathbf{B})$ stabilizable, $\mathbf{R} > 0$ and $\mathbf{Q} \geq 0$, the value function is itself quadratic $V(\mathbf{x}) = (1/\tau)\mathbf{x}^{\mathrm{T}}\mathbf{Sx}$. Premultiplication by $1/\tau$, which amounts to a rescaling of $\mathbf{S}$, is added for easier comparison to standard notations. Inserting $\mathbf{f}$ and $\mathcal{L}$ in (6), the greedy policy is $\mathbf{u}^{\mathrm{G}} = -\tau\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Sx}$. Substituting in (2), we have

$$\frac{1}{\tau}\mathbf{x}^{\mathrm{T}}\mathbf{Sx} = 2\mathbf{x}^{\mathrm{T}}\mathbf{S}(\mathbf{Ax}-\mathbf{BR}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Sx})+\mathbf{x}^{\mathrm{T}}\mathbf{Qx} +\mathbf{x}^{\mathrm{T}}\mathbf{SBR}^{-1}\mathbf{RR}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{Sx}.$$

Differentiating twice w.r.t. $\mathbf{x}$, dividing by two, and rearranging

$$\left(\mathbf{A}^{\mathrm{T}}-\frac{1}{2\tau}\mathbf{I}\right)\mathbf{S}+\mathbf{S}\left(\mathbf{A}-\frac{1}{2\tau}\mathbf{I}\right)-\mathbf{SBR}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S}+\mathbf{Q}=0$$

By comparison to the usual Riccati equation $\mathbf{A}^{\mathrm{T}}\mathbf{S}+\mathbf{SA}-\mathbf{SBR}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S}+\mathbf{Q}=0$, we see that a discounted linear quadratic system is equivalent to a nondiscounted system with a friction-like damping factor of $-(1/2\tau)\mathbf{I}$ added to the dynamics. The Riccati equation, essentially a quadratic equation, admits two solutions, the positive–definite "correct" solution, and a destabilizing negative–definite solution. This fact serves to motivate the method of increasing τ during the convergence. For a small enough τ, the dynamical system is extremely stable and any initial guess value function will provide a stable control. As τ is increased, the stabilizing controller found at the previous stages serves as "scaffolding" for the current stage. This is quite similar to other techniques [4], [24], [31], where the preexistence of a stabilizing controller is a condition for its further improvement. Another way of showing the same effect is to rescale $\mathbf{S} \leftarrow \tau\mathbf{S}$ and to see that $\tau = 0 \Rightarrow \mathbf{S} = \mathbf{Q}$, which is a positive–definite initial condition.

We experimented with the 4-D linear quadratic system given by

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ -2 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\end{pmatrix}\quad \mathbf{B} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\quad \mathbf{Q} = \mathbf{I}_4 \quad \mathbf{R} = 1.$$

For $u = 0$, this is a simple energy conserving model of two masses connected by springs. We generated 1000 quasi-uniform points in the ellipsoid of unit mechanical energy. Applying the *Naive* algorithm over these points to the nondiscounted equation ($\tau = \infty$) using a 93-weight NN, we got mixed results. In Fig. 1, we see one run converging to the "correct" positive–definite solution, another to the negative–definite solution, and yet another to no solution at all. Gradually increasing τ from 1 to ∞, as described, always resulted in correct convergence.

B. Car on the Hill

The car-on-the-hill problem, described in [32] and investigated in [11], is a 2-D optimal control problem. A "car" is postulated to roll[8] on a hill of shape

$$H(x) = \begin{cases} x^2+x, & \text{if } x<0,\\ x/\sqrt{1+5x^2}, & \text{if } x\geq 0.\end{cases}$$

[7]MATLAB code for implementing differential backpropagation is available at www.alice.nc.huji.ac.il/~tassa/

[8]We make the hairsplitting note that the dynamical equations, used in this paper to allow comparison to previous work and originally given in [32], ignore the centripetal force and are not "correct" in the Newton-mechanical sense.

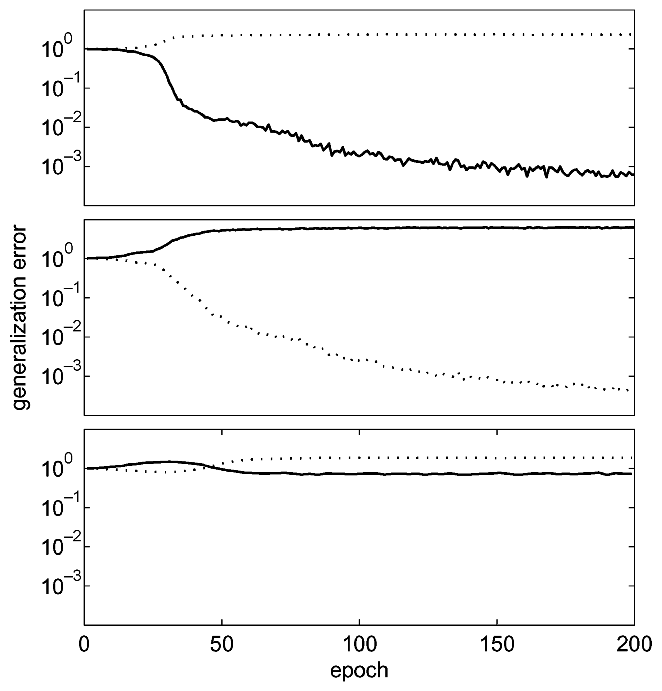


Fig. 1. Performance of three sample runs of the naive algorithm on the linear quadratic system with $\tau = \infty$. At every iteration, we measure the difference between approximated and analytical values over a set of 1000 random points generated anew. Solid (dotted) lines denote the difference between the approximated function and the positive–definite (negative–definite) analytical solution, computed with standard methods. The three figures show a converging, anticonverging, and misconverging run.

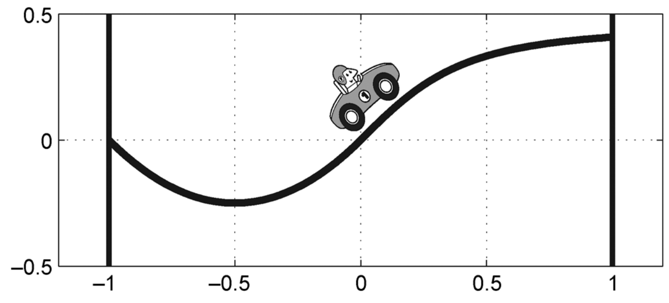


Fig. 2. Car-on-the-hill optimal control problem. The reward is identically zero inside the state space. Terminal rewards of -1 and $1 - \dot{x}/2$ are incurred upon reaching the left and right edges of the state space, respectively.

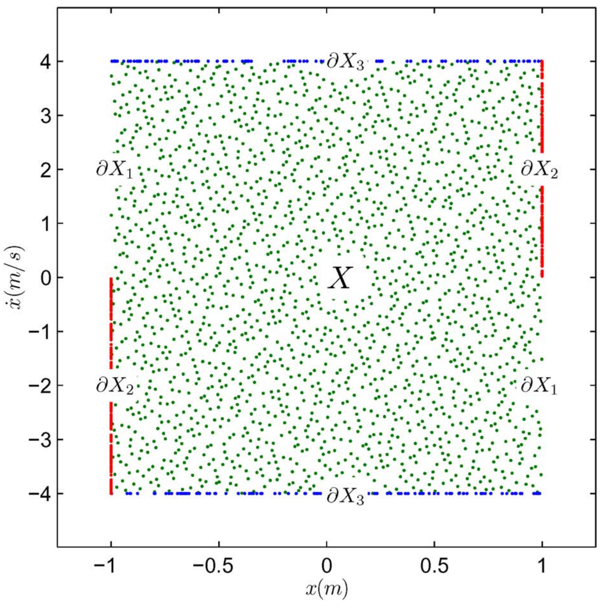


Fig. 3. Illustration of the points where the HJB residual was measured and minimized. Clamping constraints were applied to points on the "outbound" ∂X_2 by minimizing $(V(\mathbf{x}) - r(x))^2$. For points on ∂X_3, the dynamics were modified to never point outside the state space.

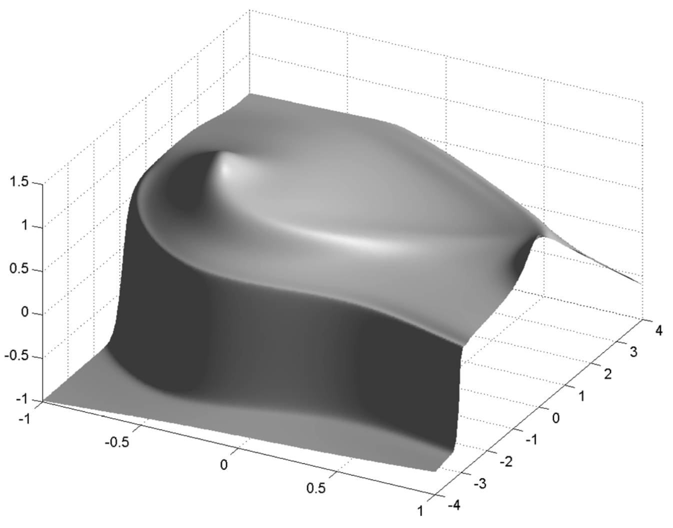


Fig. 4. Local divergence of the learning. The bump in the value function close to $(-0.5\ 0)$ begins as a local extrapolation near the discontinuous boundary and migrates to this position.

The stated goal of reaching the top of the hill with minimal velocity is achieved by setting terminal rewards of $r(x = -1) = -1$ and $r(x = 1) = 1 - \dot{x}/2$ on the left and right edges of the state space, respectively (Fig. 2), and $r(|x| < 1) = 0$ everywhere else. A discrete control space $U = \{-4, 4\}$ is well approximated by using the bounded control (8) with $u_{\max} = 4$ and $c = 10^{-4}$. The volume of state space where we solve this problem is $X = \{(x, \dot{x}) \in \mathbb{R}^2 : |x| < 1_m, |\dot{x}| < 4\text{m/s}\}$. We used three sets of points to find a solution, as shown in Fig. 3. First, we placed 2000 points evenly spread across X, using the pseudorandom Halton sequence. Next, we placed 400 points on the "outbound" $\partial X2$ ($x = \pm 1$ and, respectively, $\dot{x} \lesseqgtr 0$), where any trajectory would inevitably fall out of X. At this points, $V(\mathbf{x}) = r(x)$ was enforced. Finally, we placed 200 points on $\partial X3$, where $|x| < 1$ but $\dot{x} = \pm 4$. In these points, the dynamics were modified by limiting the maximal speed of the car to 4 m/s, a manipulation to the effect of "clipping" the outbound dynamics to be tangent to ∂X. The value function which solves this problem has a discontinuous ridge and a nondifferentiable ridge which make the problem difficult for essentially smooth function approximators like ours. Specifically, when running the *Naive* algorithm, a Gibbs-oscillation-type phenomenon at the discontinuous boundary was found to sometimes evolve into a false stationary point on the $\dot{x} = 0$ line (Fig. 4).

Since the reward function of this problem is discontinuous, setting $\tau = 0$ (and, consequently, $V = r$) would present the

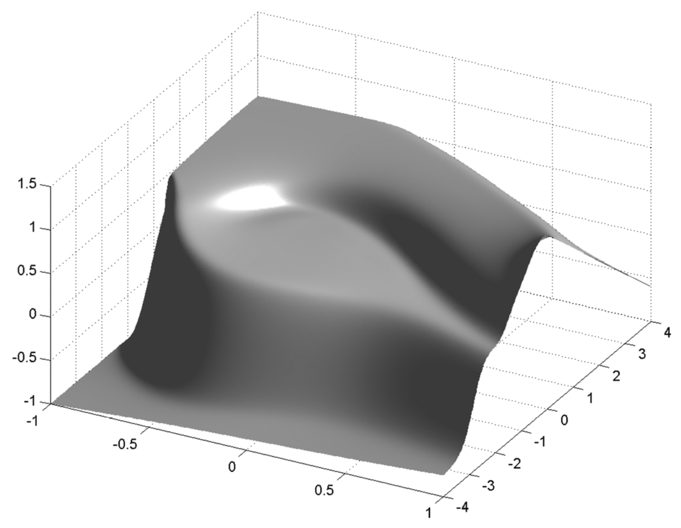


Fig. 5. Smooth value function obtained for $\gamma = 0.02$. We note that the HJB residual for this function was smaller by more than an order of magnitude than for the "sharp" solution.

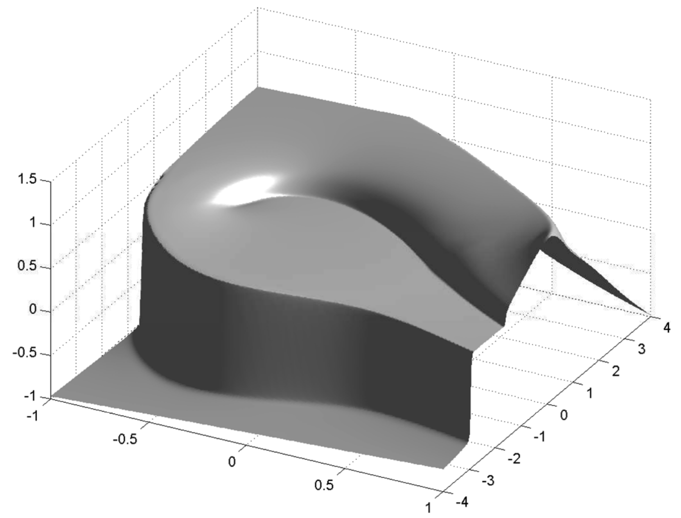


Fig. 6. Approximation of the optimal value function for the car-on-the-hill problem. Compare to [14, Figs. 3 and 4] (available at www.cmap.polytechnique.fr/~munos).

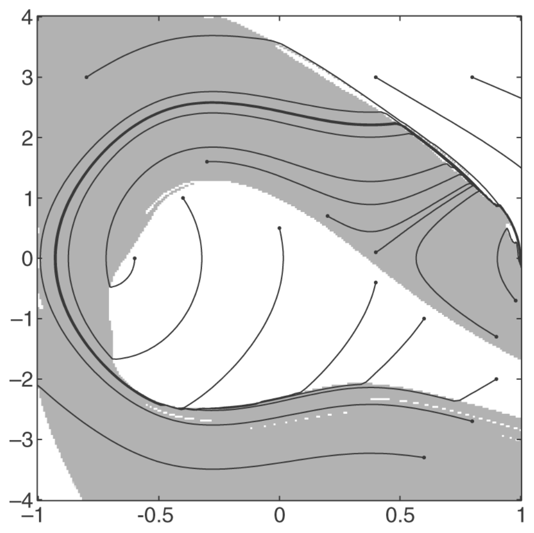


Fig. 7. Approximate optimal policy obtained for the car-on-the-hill problem. Gray and white areas indicate the maximum and minimum control values, respectively. Lines indicate some trajectories integrated using this policy. Compare to [11, Fig. 4].

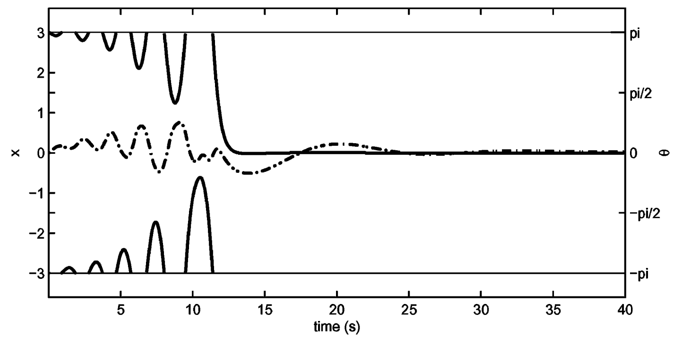


Fig. 8 Time course of the swing-up trajectory for the cart-pole dynamical system, starting from the motionless hanging-down position in the center of the track. The solid line denotes the angle of the pole and the dashed line denotes the position of the cart. Note that the horizon time scale τ is only 1.5 s, far shorter than the total swing-up time.

NN with an unpleasant challenge, and therefore, we are limited in this case to working with the smoothing term only. The smooth value function obtained by setting $\gamma = 0.02$ for all iterations (Fig. 5) has a small HJB residual and is quite immune to the local divergence described previously. By letting γ decrease from 10^{-1} to $2 \cdot 10^{-4}$, we end up with a "sharp" value function (Fig. 6), without exposing ourselves to the danger of local mis-extrapolation. A typical simulation usually takes about 100 iterations to converge in about 10-min time. The value and policy of Figs. 6 and 7, which were generated with an 800 weight NN, are comparable to the nearly optimal discretization-based solutions in [14], which are described by 66 000 parameters, and appear to be far better than the NN solution therein.

C. Cart Pole Swing-Up

Last, we demonstrate our algorithm on the 4-D system known as the cart-pole dynamical system in the RL community and the inverted-pendulum in the control community, as described in [5] and elsewhere. It consists of a mass (the cart) on a 1-D horizontal track to which another mass (the pole) is attached. The pole swings freely under the effect of gravity. The controller may apply force to the cart in order to achieve the goal of swinging the pole up and then stabilizing it over the cart in the center of the track. The volume of state space where we solve the problem is $X = \{(x, \dot{x}, \theta, \dot{\theta}) \in \mathbb{R}^4 : |x| < 3\,\text{m}, |\dot{x}| < 5\,\text{m/s}, |\theta| < \pi, |\dot{\theta}| < 10\,\text{rad/s}\}$. We used a reward function of $r(x,\theta) = \cos(\theta) - (1/9)x^2$, cart mass 1 kg, pole length 1 m, and pole mass 0.1 kg. The parameters of (8) were $u_{\max} = 3$ N and $c = 0.01$, and the maximal horizon time scale was $\tau = 1.5$ s. The learning took place over 10 000 points generated quasi-uniformly across the entire volume, and another 500 points for clamping constraints at exit regions from the state space. Here, we used both the increase of τ and the decrease of γ. As described earlier, we first increased τ with a high constant value of γ and then decreasing γ after τ had plateaued. Using a 1049-weight NN, a typical simulation took about 250

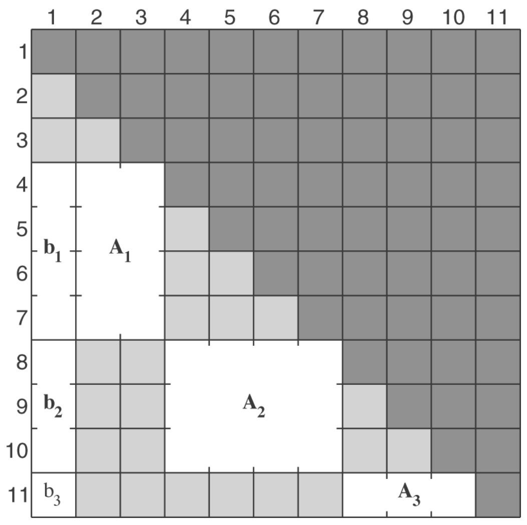


Fig. 9. Pineda weight matrix W for $q = 11$ neurons. Dark-gray squares indicate feedback (recurrent) connections, unused here. Light-gray subdiagonal squares indicate the maximally connected topology enabled by the formalism, of these, indicated by white, are the weights of a regular two hidden-layer network with two input neurons, a four-layer, a three-layer, and a scalar linear output, which would usually be written as $y(\mathbf{x}) = \mathbf{A}_3\sigma(\mathbf{A}_2\sigma(\mathbf{A}_1\mathbf{x}+\mathbf{b}_1)+\mathbf{b}_2)+b_3$. As described in the text, the first neuron is the bias neuron with constant output 1. In all of our experiments, the topology used was the maximal topology minus the weights $\mathbf{W}_{ij}$, for which $j + q + 1 < 2i$.

iterations to converge in about 4 h. In Fig. 8, we show the solution trajectory starting from the difficult position of the pole hanging straight down in the middle of the track with no linear or angular velocity. From this position, the controller must jiggle the cart for a while to accumulate energy in the pole, then swing the pole up and stabilize it around the origin. Quite remarkably, the swing-up process requires almost 30 s to complete which is considerably longer than the horizon time scale $\tau = 1.5$ s.

IV. Conclusion

In this paper, we have shown how using batch least squares minimization of the squared HJB residual is a simple and effective method for solving nonlinear optimal control problems. As a preliminary study, this paper can be extended in numerous ways. The two methods proposed here can probably be put to good use in many other numerical algorithms, and perhaps also in an analytical context. The use of fixed, uniformly distributed points over which the residual is minimized is an obvious place for improvement. A more disciplined approach would be to use some statistically sound method like Kalman filtering to sequentially estimate the update vector $\Delta\mathbf{w}$ at randomly generated points, and accept an update only when the estimate reaches some prescribed confidence threshold. Another approach would be to use advanced stochastic methods like those proposed in [33]. Recent advances in other types of function approximation schemes for the solution of differential equations, such as support vector regression [34] seem promising. Our hope is that by using these or other techniques, even more difficult nonlinear optimal control problems, such as those ostensibly solved by biological nervous systems, might soon become accessible.

Appendix I

The measurement and backpropagation w.r.t. differential quantities in NNs is a rather esoteric art. First proposed indirectly in [20], their description and formulation reappears, apparently independently, every several years [5], [21], [22]. We believe these approximators can find good use in many fields and, therefore, give their explicit formulation, up to the second order. The Pineda feedforward topology, first described in [25] and popularized by [35], is a generalization of classical layered topologies and allows for a maximally connected feedforward topology. Indexing the q neurons of the network in an order allowing feedforward computation, i.e., the ith neuron depending only on neurons $j < i$, we arrange the weights in a strictly lower triangular $q \times q$ matrix $\mathbf{W}$, the weight W_{ij} denoting the connection from neuron j to neuron i.

A. Regular Forward Propagation

For an input vector $\mathbf{x} \in \mathbb{R}^d$, we fix the outputs of the first $d + 1$ neurons to be the augmented input vector $\mathbf{y}_{1\ldots(d+1)} = (1, x_1, \ldots, x_d)^{\mathrm{T}}$, set $W_{ij} = 0$ for $i \le (d + 1)$, and then, progressively compute for $i = (d + 2)\ldots q$

$$a_i = \sum_{j<i} W_{ij} y_j, \qquad y_i = \sigma(a_i) \tag{9}$$

where a_i is the input to the ith neuron, and σ the nonlinearity. Here, we assume the scalar output of the network to be the value of the last neuron, with no squashing nonlinearity to avoid bounding the outputs: $F(\mathbf{x}, \mathbf{w}) = y_q = a_q$. By making some of the weights zero, or simply ignoring them in the calculations, any topology can be easily implemented, e.g., the W matrix of a layered network will have a chain of blocks below the diagonal (see Fig. 9).

B. Regular Backpropagation

When calculating $\partial F/\partial W_{ij}$, we first compute $\delta_i \equiv (\partial F/\partial a_i)$ in reverse order, i.e., $\delta_q = 1$, and then, for $i = (q - 1)\ldots 1$

$$\delta_i = \sigma'(a_i) \sum_{j>i} W_{ji} \delta_j \tag{10}$$

and then

$$\frac{\partial F}{\partial W_{ij}} = \frac{\partial F}{\partial a_i}\frac{\partial a_i}{\partial W_{ij}} = \delta_i y_j.$$

Note that in (9) we sum over the ith row of W, the weights into neuron i, and in (10), over the ith column, the weights out of neuron i.

C. First-Order Differential Propagation

We use the superscript $\mu = 1\ldots d$ to denote $\partial/\partial x_\mu$, differentiation w.r.t. the μth coordinate of the input vector. The initial input to the derivative network $\mathbf{y}^{\mu}_{1\ldots(d+1)} =$

$(0, 0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is a vector of all zeros with 1 in the $(\mu+1)$th place. Then, $(\partial/\partial x_\mu)F(\mathbf{x}, \mathbf{w})$ is given by

$$a_i^\mu = \sum_{j<i} W_{ij} y_j^\mu \qquad y_i^\mu = a_i^\mu \sigma'(a_i).$$

The backpropagation of the quantities $(\partial/\partial W_{ij})(\partial/\partial x_\mu)F(\mathbf{x}, \mathbf{w})$ is then given by first setting $\delta_q^\mu = 0$ and computing in reverse order

$$\delta_i^\mu = \sum_{j>i} W_{ji} \left(\delta_j^\mu \sigma'(a_i) + \delta_j \sigma''(a_i) a_i^\mu\right)$$

and then

$$\frac{\partial}{\partial W_{ij}} \frac{\partial F}{\partial x_\mu} = \frac{\partial}{\partial x_\mu} \left(\frac{\partial F}{\partial a_i} \frac{\partial a_i}{\partial W_{ij}}\right) = \delta_i^\mu y_j + \delta_i y_j^\mu.$$

D. Second-Order Differential Propagation

The initial input to the second-order derivative network $\mathbf{y}^{\mu\nu}_{1\ldots(d+1)} = (0, 0, \ldots 0)^{\mathrm{T}}$ is now a $d+1$ vector of all zeros. Then, $(\partial/\partial x_\mu)(\partial/\partial x_\nu)F(\mathbf{x}, \mathbf{w})$ is given by

$$a_i^{\mu\nu} = \sum_{j<i} W_{ij} y_j^{\mu\nu} \qquad y_i^{\mu\nu} = a_i^{\mu\nu} \sigma'(a_i) + a_i^\mu a_i^\nu \sigma''(a_i).$$

The backpropagation of the quantities $(\partial/\partial W_{ij})(\partial/\partial x_\mu)(\partial/(\partial x_\nu)F(\mathbf{x}, \mathbf{w})$ is given by setting $\delta_q^{\mu\nu} = 0$ and then

$$\delta_i^{\mu\nu} = \sum_{j>i} W_{ji} \left(\delta_j^{\mu\nu} \sigma'(a_i) + \delta_j^\mu \sigma''(a_i) a_i^\nu + \delta_j^\nu \sigma''(a_i) a_i^\mu + \delta_j \sigma''(a_i) a_i^{\mu\nu} + \delta_j \sigma'''(a_i) a_i^\mu a_i^\nu\right)$$

and then

$$\frac{\partial}{\partial W_{ij}} \frac{\partial}{\partial x_\mu} \frac{\partial}{\partial x_\nu} F = \delta_i^{\mu\nu} y_j + \delta_i^\mu y_j^\nu + \delta_i^\nu y_j^\mu + \delta_i y_j^{\mu\nu}.$$

Acknowledgment

The authors would like to thank M. Margaliot and R. Munos for their helpful comments.

References

[1] M. Crandall and P. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Trans. Amer. Math. Soc.*, vol. 277, 1983.

[2] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 369–376.

[3] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, pp. 183–192, 1989.

[4] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[5] R. Coulom, "Reinforcement learning using neural networks, with applications to motor control," Ph.D. dissertation, Institut National Polytechnique de Grenoble, Grenoble, France, 2002.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 1995.

[7] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: Elsevier, 1970.

[8] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[9] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, pp. 1593–1599, 1997.

[10] M. G. Lagoudakis, R. E. Parr, and M. L. Littman, "Least-squares methods in reinforcement learning for control," in *Proc. 2nd Hellenic Conf. Artif. Intell.*, 2002, vol. 2308, pp. 249–260.

[11] R. Munos and A. W. Moore, "Variable resolution discretization for high-accuracy solutions of optimal control problems," in *Proc. Int. Joint Conf. Artif. Intell.*, 1999, pp. 1348–1355.

[12] K. Doya, "Temporal difference learning in continuous time and space," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. Cambridge, MA: MIT Press, 1996, vol. 8.

[13] G. Tesauro, "Practical issues in temporal difference learning," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kaufmann, 1992, vol. 4, pp. 259–266.

[14] R. Munos, L. Baird, and A. Moore, "Gradient descent approaches to neural net-based solutions of the Hamilton-Jacobi-Bellman equation," in *Proc. Int. Joint Conf. Neural Netw.*, 1999, pp. 1316–1323.

[15] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proc. 1993 Connectionist Models Summer School*, M. Mozer, P. Smolensky, D. Touretzky, J. Elman, and A. Weigend, Eds., 1993, pp. 255–263.

[16] R. Beard and T. McLain, "Successive galerkin approximation algorithms for nonlinear optimal and robust control," *Proc. Int. J. Control: Special Issue Breakthroughs Control Nonlinear Syst.*, vol. 71, no. 5, pp. 717–743, 1998.

[17] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[18] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[19] C. J. Goh, "On the nonlinear optimal regulator problem," *Automatica*, vol. 29, no. 3, pp. 751–756, 1993.

[20] P. Simard, B. Victorri, Y. LeCun, and J. Denker, "Tangent prop—A formalism for specifying selected invariances in an adaptive network," in *Neural Information Processing Systems*, J. M. R. Lippman and S. J. Hanson, Eds. San Mateo, CA: Morgan Kaufmann, 1992, vol. 4.

[21] J. W. Lee and J. H. Oh, "Hybrid learning of mapping and its Jacobian in multilayer neural networks," *Neural Comput.*, vol. 9, pp. 937–958, 1997.

[22] R. Masuoka, "Neural networks learning differential data," *IEICE Trans. Inf. Syst.*, vol. E83-D, no. 6, pp. 1291–1300, 2000.

[23] W. Fleming and H. Soner, *Controlled Markov Processes and Viscosity Solutions*. New York: Springer-Verlag, 1993.

[24] G. Saridis and C. S. Lee, "An approximation theory of optimal control for trainable manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 3, pp. 152–159, Mar. 1979.

[25] F. Pineda, "Generalization of back-propagation to recurrent neural networks," *Phys. Rev. Lett.*, vol. 19, no. 59, pp. 2229–2232, 1987.

[26] M. S. Bazarraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York: Wiley, 1993.

[27] D. H. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights," in *Proc. 1st IEEE Int. Joint Conf. Neural Netw.*, 1990, vol. 3, pp. 21–26.

[28] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, pp. 84–90, 1960.

[29] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, 2000.

[30] S. E. Lyshevski and A. U. Meyer, "Control system analysis and design upon the Lyapunov method," in *Proc. Amer. Control Conf.*, Jun. 1995, pp. 3219–3223.

[31] D. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. Autom. Control*, vol. 13, no. 1, pp. 114–115, Feb. 1968.

[32] A. Moore and C. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces," *Mach. Learn.*, vol. 21, pp. 1–36, 1995.

[33] N. N. Schraudolph, "Local gain adaptation in stochastic gradient descent ISDIA, Lugano, Switzerland, Tech. Rep. IDSIA-09-99, 1999, p. 8.

[34] M. Lazaro, I. Santamaria, F. Perez-Cruz, and A. Artes-Rodriguez, "Support vector regression for the simultaneous learning of a multivariate function and its derivatives," *Neurocomput.*, vol. 69, pp. 42–61, 2005.

[35] B. A. Pearlmutter, "Fast exact multiplication by the Hessian," *Neural Comput.*, vol. 6, no. 1, pp. 147–160, 1994.

Chapter 4

Receding-Horizon Differential Dynamic Programming

Receding Horizon Differential Dynamic Programming

Yuval Tassa [*] **Tom Erez & Bill Smart** [†]

Abstract

We introduce a method for the solution of high-dimensional, continuous, non-linear optimal-control problems. By extending Differential Dynamic Programming, a second-order trajectory optimization algorithm, with a receding horizon scheme reminiscent of Model Predictive Control, we learn locally quadratic models of the time-independent Value Function along likely trajectories. A global policy is generalized in the control phase from several such trajectories using a nearest-neighbor rule. We demonstrate the effectiveness of our approach on a class of high-dimensional problems using a simulated multi-link swimming robot. These experiments show that our approach effectively circumvents dimensionality issues, and is capable of dealing with problems of (at least) 24 state and 9 action dimensions. A real-time MATLAB interaction package is made available at alice.nc.huji.ac.il/∼tassa.

1 Introduction

We are interested in learning controllers for high-dimensional, highly non-linear dynamical systems, continuous in state, action, and time. Local, trajectory-based methods, using techniques such as Differential Dynamic Programming (DDP), are an active field of research in the Reinforcement Learning and Control communities. Local methods do not model the value function or policy over the entire state space by focusing computational effort along likely trajectories. Featuring algorithmic complexity polynomial in the dimension, local methods are not directly affected by dimensionality issues as space-filling methods.

In this paper, we introduce Receding Horizon DDP (RH-DDP), a set of modifications to the classic DDP algorithm, which allows us to construct stable and robust controllers based on local-control trajectories in highly non-linear, high-dimensional domains. Our new algorithm is reminiscent of Model Predictive Control, and enables us to form a time-independent value function approximation along a trajectory. We aggregate several such trajectories into a library of locally-optimal linear controllers which we then select from, using a nearest-neighbor rule.

Although we present several algorithmic contributions, a main aspect of this paper is a conceptual one. Unlike much of recent related work (below), we are not interested in learning to follow a pre-supplied reference trajectory. We define a reward function which represents a global measure of performance relative to a high level objective, such as swimming towards a target. Rather than a reward based on distance from a given desired configuration, a notion which has its roots in the control community's definition of the problem, this global reward dispenses with a "path planning" component and requires the controller to solve the entire problem.

We demonstrate the utility of our approach by learning controllers for a high-dimensional simulation of a planar, multi-link swimming robot. The *swimmer* is a model of an actuated chain of links in a viscous medium, with two location and velocity coordinate pairs, and an angle and angular

[*]Y. Tassa is with the Hebrew University, Jerusalem, Israel.

[†]T. Erez and W.D. Smart are with the Washington University in St. Louis, MO, USA.

velocity for each link. The controller must determine the applied torque, one action dimension for each articulated joint. We reward controllers that cause the swimmer to swim to a target, brake on approach and come to a stop over it.

We synthesize controllers for several swimmers, with state dimensions ranging from 10 to 24 dimensions. The controllers are shown to exhibit complex locomotive behaviour in response to real-time simulated interaction with a user-controlled target.

1.1 Related work

Optimal control of continuous non-linear dynamical systems is a central research goal of the RL community. Even when important ingredients such as stochasticity and on-line learning are removed, the exponential dependence of computational complexity on the dimensionality of the domain remains a major computational obstacle. Methods designed to alleviate the curse of dimensionality include adaptive discretizations of the state space [1], and various domain-specific manipulations [2] which reduce the effective dimensionality.

Local trajectory-based methods such as DDP were introduced to the NIPS community in [3], where a local-global hybrid method is employed. Although DDP is used there, it is considered an aid to the global approximator, and the local controllers are constant rather than locally-linear. In this decade DDP was reintroduced by several authors. In [4] the idea of using the second order local DDP models to make locally-linear controllers is introduced. In [5] DDP was applied to the challenging high-dimensional domain of autonomous helicopter control, using a reference trajectory. In [6] a minimax variant of DDP is used to learn a controller for bipedal walking, again by designing a reference trajectory and rewarding the walker for tracking it. In [7], trajectory-based methods including DDP are examined as possible models for biological nervous systems. Local methods have also been used for purely policy-based algorithms [8, 9, 10], without explicit representation of the value function.

The best known work regarding the swimming domain is that by Ijspeert and colleagues (e.g. [11]) using Central Pattern Generators. While the inherently stable domain of swimming allows for such open-loop control schemes, articulated complex behaviours such as turning and tracking necessitate full feedback control which CPGs do not provide.

2 Methods

2.1 Definition of the problem

We consider the discrete-time dynamics $x^{k+1} = F(x^k, u^k)$ with states $x \in \mathbb{R}^n$ and actions $u \in \mathbb{R}^m$. In this context we assume $F(x^k, u^k) = x^k + \int_0^{\Delta t} f(x(t), u^k)dt$ for a continuous f and a small Δt, approximating the continuous problem and identifying with it in the $\Delta t \to 0$ limit. Given some scalar reward function $r(x, u)$ and a fixed initial state x^1 (superscripts indicating the time index), we wish to find the policy which maximizes the total reward[1] acquired over a finite temporal horizon:

$$\pi^*(x^k, k) = \underset{\pi(\cdot,\cdot)}{\text{argmax}}[\sum_{i=k}^{N} r(x^i, \pi(x^i, i))].$$

The quantity maximized on the RHS is the *value function*, which solves Bellman's equation:

$$V(x, k) = \max_u [r(x, u) + V(F(x, u), k+1)]. \tag{1}$$

Each of the functions in the sequence $\{V(x, k)\}_{k=1}^N$ describes the optimal reward-to-go of the *optimization subproblem* from k to N. This is a manifestation of the *dynamic programming principle*. If $N = \infty$, essentially eliminating the distinction between different time-steps, the sequence collapses to a global, time-independent value function $V(x)$.

[1]We (arbitrarily) choose to use phrasing in terms of reward-maximization, rather than cost-minimization.

2.2 DDP

Differential Dynamic Programming [12, 13] is an iterative improvement scheme which finds a locally-optimal trajectory emanating from a fixed starting point x^1. At every iteration, an approximation to the time-dependent value function is constructed along the current trajectory $\{x^k\}_{k=1}^N$, which is formed by iterative application of F using the current control sequence $\{u^k\}_{k=1}^N$. Every iteration is comprised of two sweeps of the trajectory: a *backward* and a *forward* sweep.

In the *backward sweep*, we proceed backwards in time to generate local models of V in the following manner. Given quadratic models of $V(x^{k+1}, k+1)$, $F(x^k, u^k)$ and $r(x^k, u^k)$, we can approximate the *unmaximised* value function, or Q-function,

$$Q(x^k, u^k) = r(x^k, u^k) + V^{k+1}(F(x^k, u^k)) \tag{2}$$

as a quadratic model around the present state-action pair (x^k, u^k):

$$Q(x^k + \delta x, u^k + \delta u) \approx Q_0 + Q_x \delta x + Q_u \delta u + \frac{1}{2}[\delta x^T \; \delta u^T] \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} \tag{3}$$

Where the coefficients $Q_{\star\star}$ are computed by equating coefficients of similar powers in the second-order expansion of (2)

$$\begin{array}{llll} Q_x & = r_x + V_x^{k+1} F_x^k & Q_{xx} & = r_{xx} + F_x^k V_{xx}^{k+1} F_x^k + V_x^{k+1} F_{xx}^k \\ Q_u & = r_u + V_x^{k+1} F_u^k & Q_{uu} & = r_{uu} + F_u^k V_{xx}^{k+1} F_u^k + V_x^{k+1} F_{uu}^k \\ & & Q_{xu} & = r_{xu} + F_x^k V_{xx}^{k+1} F_u^k + V_x^{k+1} F_{xu}^k. \end{array} \tag{4}$$

Once the local model of Q is obtained, the maximizing δu is solved for

$$\delta u^* = \underset{\delta u}{\operatorname{argmax}}[Q(x^k + \delta x, u^k + \delta u)] = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta x) \tag{5}$$

and plugged back into (3) to obtain a quadratic approximation of V^k:

$$V_0^k = V_0^{k+1} - Q_u (Q_{uu})^{-1} Q_u \tag{6a}$$

$$V_x^k = Q_x^{k+1} - Q_u (Q_{uu})^{-1} Q_{ux} \tag{6b}$$

$$V_{xx}^k = Q_{xx}^{k+1} - Q_{xu} (Q_{uu})^{-1} Q_{ux}. \tag{6c}$$

This quadratic model can now serve to propagate the approximation to V^{k-1}. Thus, equations (4), (5) and (6) iterate in the backward sweep, computing a local model of the Value function along with a modification to the policy in the form of an open-loop term $-Q_{uu}^{-1} Q_u$ and a feedback term $-Q_{uu}^{-1} Q_{ux} \delta x$, essentially solving a local linear-quadratic problem in each step. In some senses, DDP can be viewed as dual to the Extended Kalman Filter (though employing a higher order expansion of F).

In the *forward sweep* of the DDP iteration, both the open-loop and feedback terms are combined to create a new control sequence $(\hat{u}^k)_{k=1}^N$ which results in a new nominal trajectory $(\hat{x}^k)_{k=1}^N$.

$$\hat{x}^1 = x^1 \tag{7a}$$

$$\hat{u}^k = u^k - Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} (\hat{x}^k - x^k) \tag{7b}$$

$$\hat{x}^{k+1} = F(\hat{x}^k, \hat{u}^k) \tag{7c}$$

We note that in practice the inversion in (5) must be conditioned. We use a Levenberg Marquardt-like scheme similar to the ones proposed in [14]. Similarly, the u-update in (7b) is performed with an adaptive line search scheme similar to the ones described in [15].

2.2.1 Complexity and convergence

The leading complexity term of one iteration of DDP itself, assuming the model of F as required for (4) is given, is $O(Nm^{\gamma_1})$ for computing (6) N times, with $2 < \gamma_1 < 3$, the complexity-exponent of inverting Q_{uu}. In practice, the greater part of the computational effort is devoted to the measurement of the dynamical quantities in (4) or in the propagation of collocation vectors as described below.

DDP is a second order algorithm with convergence properties similar to, or better than Newton's method performed on the full vectorial u^k with an exact $Nm \times Nm$ Hessian [16]. In practice, convergence can be expected after 10-100 iterations, with the stopping criterion easily determined as the size of the policy update plummets near the minimum.

2.2.2 Collocation Vectors

We use a new method of obtaining the quadratic model of Q (Eq. (2)), inspired by [17][2]. Instead of using (4), we fit this quadratic model to samples of the value function at a cloud of collocation vectors $\{x_i^k, u_i^k\}_{i=1..p}$, spanning the neighborhood of every state-action pair along the trajectory. We can directly measure $r(x_i^k, u_i^k)$ and $F(x_i^k, u_i^k)$ for each point in the cloud, and by using the approximated value function at the next time step, we can estimate the value of (2) at every point:

$$q(x_i^k, u_i^k) = r(x_i^k, u_i^k) + V^{k+1}(F(x_i^k, u_i^k))$$

Then, we can insert the values of $q(x_i^k, u_i^k)$ and (x_i^k, u_i^k) on the LHS and RHS of (3) respectively, and solve this set of p linear equations for the $Q_{\star\star}$ terms. If $p > (3(n+m) + (m+n)^2)/2$, and the cloud is in general configuration, the equations are non-singular and can be easily solved by a generic linear algebra package.

There are several advantages to using such a scheme. The full nonlinear model of F is used to construct Q, rather than only a second-order approximation. F_{xx}, which is an $n \times n \times n$ tensor need not be stored. The addition of more vectors can allow the modeling of noise, as suggested in [17]. In addition, this method allows us to more easily apply general coordinate transformations in order to represent V in some internal space, perhaps of lower dimension.

The main drawback of this scheme is the additional complexity of an $O(Np^{\gamma_2})$ term for solving the p-equation linear system. Because we can choose $\{x_i^k, u_i^k\}$ in way which makes the linear system sparse, we can enjoy the $\gamma_2 < \gamma_1$ of sparse methods and, at least for the experiments performed here, increase the running time only by a small factor.

In the same manner that DDP is dually reminiscent of the Extended Kalman Filter, this method bears a resemblance to the test vectors propagated in the Unscented Kalman Filter [18], although we use a quadratic, rather than linear number of collocation vectors.

2.3 Receding Horizon DDP

When seeking to synthesize a global controller from many local controllers, it is essential that the different local components operate synergistically. In our context this means that local models of the value function must all model the same function, which is not the case for the standard DDP solution. The local quadratic models which DDP computes around the trajectory are approximations to $V(x, k)$, the time-dependent value function. The standard method in RL for creating a global value function is to use an exponentially discounted horizon. Here we propose a fixed-length non-discounted Receding Horizon scheme in the spirit of Model Predictive Control [19].

Having computed a DDP solution to some problem starting from many different starting points x_1, we can discard all the models computed for points $x^{k>1}$ and save only the ones around the x_1's. Although in this way we could accumulate a time-independent approximation to $V(x, N)$ only, starting each run of N-step DDP from scratch would be prohibitively expensive. We therefore propose the following: After obtaining the solution starting from x^1, we save the local model at $k = 1$ and proceed to solve a new N-step problem starting at x^2, this time initialized with the policy obtained on the previous run, shifted by one time-step, and appended with the last control $u_{new} = [u^2, u^3 ... u^N u^N]$. Because this control sequence is very close to the optimal solution, the second-order convergence of DDP is in full effect and the algorithm converges in 1 or 2 sweeps. Again saving the model at the first time step, we iterate. We stress the that without the fast and exact convergence properties of DDP near the maximum, this algorithm would be far less effective.

2.4 Nearest Neighbor control with Trajectory Library

A run of DDP computes a locally quadratic model of V and a locally linear model of u, expressed by the gain term $-Q_{uu}^{-1}Q_{ux}$. This term generalizes the open-loop policy to a tube around the trajectory, inside of which a basin-of-attraction is formed. Having lost the dependency on the time k with the receding-horizon scheme, we need some space-based method of determining which local gain model we select at a given state. The simplest choice, which we use here, is to select the nearest Euclidian neighbor.

[2]Our method is a specific instantiation of a more general algorithm described therein.

Outside of the basin-of-attraction of a single trajectory, we can expect the policy to perform very poorly and lead to numerical divergence if no constraint on the size of u is enforced. A possible solution to this problem is to fill some volume of the state space with a library of local-control trajectories [20], and consider all of them when selecting the nearest linear gain model.

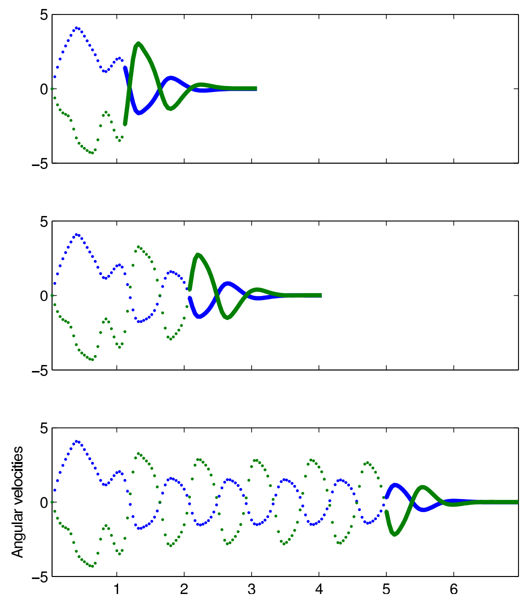
3 Experiments

3.1 The *swimmer* dynamical system

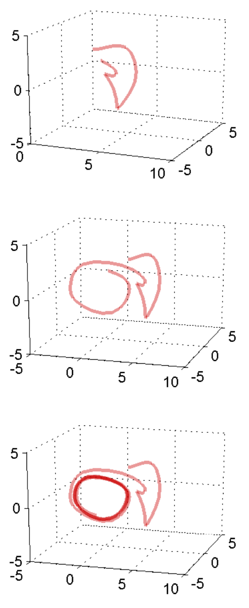
We describe a variation of the *d-link swimmer* dynamical system [21]. A stick or *link* of length l, lying in a plane at an angle θ to some direction, parallel to $\hat{\mathbf{t}} = \left(\begin{smallmatrix}\cos(\theta)\\ \sin(\theta)\end{smallmatrix}\right)$ and perpendicular to $\hat{\mathbf{n}} = \left(\begin{smallmatrix}-\sin(\theta)\\ \cos(\theta)\end{smallmatrix}\right)$, moving with velocity $\dot{\mathbf{x}}$ in a viscous fluid, is postulated to admit a normal frictional force $-k_n l \hat{\mathbf{n}}(\dot{\mathbf{x}} \cdot \hat{\mathbf{n}})$ and a tangential frictional force $-k_t l \hat{\mathbf{t}}(\dot{\mathbf{x}} \cdot \hat{\mathbf{t}})$, with $k_n > k_t > 0$. The swimmer is modeled as a chain of d such links of lengths l_i and masses m_i, its configuration described by the generalized coordinates $\mathbf{q} = \left(\begin{smallmatrix}\mathbf{x}_{cm}\\ \boldsymbol{\theta}\end{smallmatrix}\right)$, of two center-of-mass coordinates and d angles. Letting $\bar{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{x}_{cm}$ be the positions of the link centers WRT the center of mass , the Lagrangian is

$$L = \tfrac{1}{2}\dot{\mathbf{x}}_{cm}^2 \sum_i m_i + \tfrac{1}{2}\sum_i m_i \dot{\bar{\mathbf{x}}}_i^2 + \tfrac{1}{2}\sum_i I_i \dot{\theta}_i^2$$

with $I_i = \frac{1}{12} m_i l_i^2$ the moments-of-inertia. The relationship between the relative position vectors and angles of the links is given by the $d-1$ equations $\bar{\mathbf{x}}_{i+1} - \bar{\mathbf{x}}_i = \frac{1}{2} l_{i+1}\hat{\mathbf{t}}_{i+1} + \frac{1}{2} l_i \hat{\mathbf{t}}_i$, which express the joining of successive links, and by the equation $\sum_i m_i \bar{\mathbf{x}}_i = 0$ which comes from the



(a) Time course of two angular velocities.



(b) State projection.

Figure 1: RH-DDP trajectories. (a) three snapshots of the receding horizon trajectory (dotted) with the current finite-horizon optimal trajectory (solid) appended, for two state dimensions. (b) Projections of the same receding-horizon trajectories onto the largest three eigenvectors of the full state covariance matrix. As described in Section 3.3, the linear regime of the reward, here applied to a 3-swimmer, compels the RH trajectories to a steady swimming gait – a limit cycle.

definition of the $\bar{\mathbf{x}}_i$'s relative to the center-of-mass. The function

$$F = -\tfrac{1}{2}k_n \sum_i \left[l_i(\dot{\mathbf{x}}_i \cdot \hat{\mathbf{n}}_i)^2 + \tfrac{1}{12}l_i^3\dot{\theta}_i^2\right] - \tfrac{1}{2}k_t \sum_i l_i(\dot{\mathbf{x}}_i \cdot \hat{\mathbf{t}}_i)^2$$

known as the *dissipation function*, is that function whose derivatives WRT the $\dot{q}_i$'s provide the postulated frictional forces. With these in place, we can obtain $\ddot{\mathbf{q}}$ from the $2{+}d$ Euler-Lagrange equations:

$$\tfrac{d}{dt}\left(\tfrac{\partial}{\partial q_i}L\right) = \tfrac{\partial}{\partial \dot{q}_i}F + \mathbf{u}$$

with $\mathbf{u}$ being the external forces and torques applied to the system. By applying $d-1$ torques τ_j in action-reaction pairs at the joints $u_i = \tau_i - \tau_{i-1}$, the isolated nature of the dynamical system is preserved. Performing the differentiations, solving for $\ddot{\mathbf{q}}$, and letting $\mathbf{x} = \binom{\mathbf{q}}{\dot{\mathbf{q}}}$ be the $4{+}2d$-dimensional state variable, finally gives the dynamics $\dot{\mathbf{x}} = \binom{\dot{\mathbf{q}}}{\ddot{\mathbf{q}}} = f(\mathbf{x}, \mathbf{u})$.

3.2 Internal coordinates

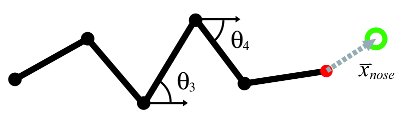
The two coordinates specifying the position of the center-of-mass and the d angles are defined relative to an external coordinate system, which the controller should not have access to. We make a coordinate transformation into *internal* coordinates, where only the $d-1$ relative angles $\{\hat{\theta}_j = \theta_{j+1} - \theta_j\}_{j=1}^{d-1}$ are given, and the location of the target is given relative to coordinate system fixed on one of the links. This makes the learning isotropic and independent of a specific location on the plane. The collocation method allows us to perform this transformation directly on the vector cloud without having to explicitly differentiate it, as we would have had to using classical DDP. Note also that this transformation reduces the dimension of the state (one angle less), suggesting the possibility of further dimensionality reduction.

3.3 The reward function

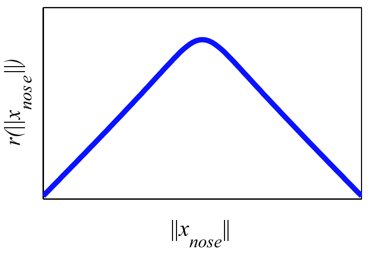
The reward function we used was

$$r(x, u) = -c_x \frac{||x_{nose}||^2}{\sqrt{||x_{nose}||^2 + 1}} - c_u||u||^2 \tag{8}$$

Where $x_{nose} = [x_1 x_2]^{\mathrm{T}}$ is the 2-vector from some designated point on the swimmer's body to the target (the origin in internal space), and c_x and c_u are positive constants. This reward is maximized when the nose is brought to rest on the target under a quadratic action-cost penalty. It should not be confused with the *desired state* reward of classical optimal control since values are specified only for 2 out of the $2d + 4$ coordinates. The functional form of the target-reward term is designed to be linear in $||x_{nose}||$ when far from the target and quadratic when close to it (Figure 2(b)). Because



(a) Swimmer



(b) Reward

Figure 2: (a) A *5-swimmer* with the "nose" point at its tip and a ring-shaped target. (b) The functional form of the planar reward component $r(x_{nose}) = -||x_{nose}||^2/\sqrt{||x_{nose}||^2 + 1}$. This form translates into a steady swimming gait at large distances with a smooth braking and stopping at the goal.

of the differentiation in Eq. (5), the solution is independent of V_0, the constant part of the value. Therefore, in the linear regime of the reward function, the solution is independent of the distance from the target, and all the trajectories are quickly compelled to converge to a one-dimensional manifold in state-space which describes steady-state swimming (Figure 1(b)). Upon nearing the target, the swimmer must initiate a braking maneuver, and bring the nose to a standstill over the target. For targets that are near the swimmer, the behaviour must also include various turns and jerks, quite different from steady-state swimming, which maneuver the nose into contact with the target. Our experience during interaction with the controller, as detailed below, leads us to believe that the behavioral variety that would be exhibited by a hypothetical exact optimal controller for this system to be extremely large.

4 Results

In order to asses the controllers we constructed a real-time interaction package[3]. By dragging the target with a cursor, a user can interact with controlled swimmers of 3 to 10 links with a state dimension varying from 10 to 24, respectively. Even with controllers composed of a single trajectory, the swimmers perform quite well, turning, tracking and braking on approach to the target.

All of the controllers in the package control swimmers with unit link lengths and unit masses. The normal-to-tangential drag coefficient ratio was $k_n/k_t = 25$. The function F computes a single 4th-order Runge-Kutta integration step of the continuous dynamics $F(x^k, u^k) = x^k + \int_t^{t+\Delta t} f(x^k, u^k)dt$ with $\Delta t = 0.05_s$. The receding horizon window was of 40 time-steps, or 2 seconds.

When the state doesn't gravitate to one of the basins of attraction around the trajectories, numerical divergence can occur. This effect can be initiated by the user by quickly moving the target to a "surprising" location. Because nonlinear viscosity effects are not modeled and the local controllers are also linear, exponentially diverging torques and angular velocities can be produced. When adding as few as 20 additional trajectories, divergence is almost completely avoided.

Another claim which may be made is that there is no guarantee that the solutions obtained, even on the trajectories, are in fact optimal. Because DDP is a local optimization method, it is bound to stop in a local minimum. An extension of this claim is that even if the solutions are optimal, this has to do with the swimmer domain itself, which might be inherently convex in some sense and therefore an "easy" problem.

While both divergence and local minima are serious issues, they can both be addressed by appealing to our panoramic motivation in the biology. Real organisms cannot apply unbounded torque. By hard-limiting the torque to large but finite values, non-divergence can be guaranteed[4]. Similarly, local minima exist even in the motor behaviour of the most complex organisms, famously evidenced by Fosbury's reinvention of the high jump.

Regarding the easiness or difficulty of the swimmer problem – we made the documented code available and hope that it might serve as a useful benchmark for other algorithms.

5 Conclusions

The significance of this work lies at its outlining of a new kind of tradeoff in nonlinear motor control design. If biological realism is an accepted design goal, and physical and biological constraints taken into account, then the expectations we have from our controllers can be more relaxed than those of the control engineer. The unavoidable eventual failure of any specific biological organism makes the design of truly robust controllers a futile endeavor, in effect putting more weight on the mode, rather than the tail of the behavioral distribution. In return for this forfeiture of global guarantees, we gain very high performance in a small but very dense sub-manifold of the state-space.

[3]Available at http://alice.nc.huji.ac.il/$\sim$tassa/

[4]We actually constrain angular velocities since limiting torque would require a stiffer integrator, but theoretical non-divergence is fully guaranteed by the viscous dissipation which enforces a Lyapunov function on the entire system, once torques are limited.

Since we make use of biologically grounded arguments, we briefly outline the possible implications of this work to biological nervous systems. It is commonly acknowledged, due both to theoretical arguments and empirical findings, that some form of dimensionality reduction must be at work in neural control mechanisms. A common object in models which attempt to describe this reduction is the *motor primitive*, a hypothesized *atomic* motor program which is combined with other such programs in a small "alphabet", to produce complex behaviors in a given context. Our controllers imply a different reduction: a set of complex prototypical motor programs, each of which is near-optimal only in a small volume of the state-space, yet in that space describes the entire complexity of the solution. Giving the simplest building blocks of the model such a high degree of task specificity or context, would imply a very large number of these *motor prototypes* in a real nervous system, an order of magnitude analogous, in our linguistic metaphor, to that of words and concepts.

References

[1] Remi Munos and Andrew W. Moore. Variable Resolution Discretization for High-Accuracy Solutions of Optimal Control Problems. In *International Joint Conference on Artificial Intelligence*, pages 1348–1355, 1999.

[2] M. Stilman, C. G. Atkeson, J. J. Kuffner, and G. Zeglin. Dynamic programming in reduced dimensional spaces: Dynamic planning for robust biped locomotion. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 2399–2404, 2005.

[3] Christopher G. Atkeson. Using local trajectory optimizers to speed up global optimization in dynamic programming. In *NIPS*, pages 663–670, 1993.

[4] C. G. Atkeson and J. Morimoto. Non-parametric representation of a policies and value functions: A trajectory based approach. In *Advances in Neural Information Processing Systems 15*, 2003.

[5] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems 19*, 2007.

[6] J. Morimoto and C. G. Atkeson. Minimax differential dynamic programming: An application to robust bipedwalking. In *Advances in Neural Information Processing Systems 14*, 2002.

[7] Emanuel Todorov and Wei-Wei Li. Optimal control methods suitable for biomechanical systems. In *25th Annual Int. Conf. IEE Engineering in Medicine and Biology Society*, 2003.

[8] R. Munos. Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791, 2006.

[9] J. Peters and S. Schaal. Reinforcement learning for parameterized motor primitives. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2006)*, 2006.

[10] Tom Erez and William D. Smart. Bipedal walking on rough terrain using manifold control. In *IEEE/RSJ International Conference on Robots and Systems (IROS)*, 2007.

[11] A. Crespi and A. Ijspeert. AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th International Conference on Climbing and Walking Robots (CLAWAR 2006)*, pages 19–27, 2006.

[12] D. Q. Mayne. A second order gradient method for determining optimal trajectories for non-linear discrete-time systems. *International Journal of Control*, 3:85–95, 1966.

[13] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming*. Elsevier, 1970.

[14] L.-Z. Liao and C. A. Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692–706, 1991.

[15] S. Yakowitz. Algorithms and computational techniques in differential dynamic programming. *Control and Dynamic Systems: Advances in Theory and Applications*, 31:75–91, 1989.

[16] L.-Z. Liao and C. A. Shoemaker. Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems. Technical Report 92-097, Cornell Theory Center, 1992.

[17] E. Todorov. Iterative local dynamic programming. Manuscript under review, available at `www.cogsci.ucsd.edu/~todorov/papers/ildp.pdf`, 2007.

[18] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proceedings of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, 1997.

[19] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice. *Automatica*, 25:335–348, 1989.

[20] M. Stolle and C. G. Atkeson. Policies based on trajectory libraries. In *Proceedings of the International Conference on Robotics and Automation (ICRA 2006)*, 2006.

[21] R. Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

Chapter 5

Smoothing Contact with Stochasticity

Stochastic Complementarity for Local Control of Discontinuous Dynamics

Yuval Tassa
Interdisciplinary Center for Neural Computation
Hebrew University
tassa@alice.nc.huji.ac.il

Emo Todorov
Computer Science and Engineering
University of Washington
todorov@cs.washington.edu

***Abstract*— We present a method for smoothing discontinuous dynamics involving contact and friction, thereby facilitating the use of local optimization techniques for control. The method replaces the standard Linear Complementarity Problem with a Stochastic Linear Complementarity Problem. The resulting dynamics are continuously differentiable, and the resulting controllers are robust to disturbances. We demonstrate our method on a simulated 6-dimensional manipulation task, which involves a finger learning to spin an anchored object by repeated flicking.**

I. Introduction

Classic control methods focus on forcing a dynamical system to a reference trajectory. This approach is powerful but limited. For complex behaviors in underactuated domains, planning the desired trajectory cannot be easily separated from the control strategy. Optimal Control offers a comprehensive framework that solves both the planning and control problems by finding a policy which minimizes future costs. However, global methods for finding optimal policies scale exponentially with the state dimension, making them prohibitively expensive.

Local Optimal Control methods, or *trajectory optimizers*, play a key role in the search for nonlinear feedback controllers that are on par with biological motor systems. These methods find a solution in a small part of the state-space, but because their complexity scales polynomially, they may constitute the only feasible approach to tackling high-dimensional problems.

Locomotion and *hand-manipulation*, behaviors which encompass some of the most interesting control problems, are crucially dependent on contact and friction. These phenomena pose a problem for efficient variants of local methods, which require differentiability to some order. Hard contacts and joint limits are examples of dynamic discontinuities where velocities change instantaneously upon impact. Though these two phenomena can conceivably be modeled with stiff nonlinear spring-dampers, friction is inherently discontinuous and does not readily admit a smooth approximation.

If our goal is to produce an accurate simulation, it might be sensible to use a deterministic model of the dynamics, since macro-physical systems are often well-modeled as such. If however we wish to control the system, *noise* might qualitatively alter the optimal behavior. We argue that when the dynamics are discontinuous, an optimal controller must account for stochasticity in order to generate an acceptable policy. Because stochastic dynamics are inherently smooth (in the mean), differentiability issues automatically disappear.

A popular and established method for modeling hard unilateral constraints and friction, involves defining *complementarity conditions* on forces and velocities. Simulators based on this principle are called *time-stepping integrators* and solve a Linear Complementarity Problem at each time step. We propose to instead solve a Stochastic Linear Complementarity Problem, using the approach proposed in [1]. This method transforms the complementarity problem into a smooth nonlinear optimization which is readily solved.

The solution effectively describes new deterministic dynamics, that implicitly take into account noise around the contact. These modified dynamics can qualitatively be described as featuring a fuzzy "force-field" that extends out from surfaces, allowing both contact and friction to act at a distance. The size and shape of this layer are naturally determined by the noise distribution, without any free parameters.

We test our method on a simplified manipulation task. A two link planar finger must learn to spin an ellipse that is anchored to the wall by flicking at it. We use an off-line Model Predictive Control strategy to patch together a global, time-independent policy, which robustly performs the task for both the smoothed system and the original discontinuous one. We constructed an interactive simulation which allows the user to actively perturb the controlled system. The controller proved to be robust, withstanding all these disturbances.

II. A Qualitative Argument

Consider the simple and familiar task of holding an object. The force which prevents the object from falling is friction, related by Coulomb's law to the normal force exerted by the fingers. If we now slowly loosen our grip, there is no discernable change in the positions or velocities of the system until suddenly, when the the weight of the object penetrates the friction cone, sticking changes into slipping and the object drops from our hand.

In the Optimal Control context, the control signal realizes a tradeoff between a state-cost and a control-cost. Because the state-cost cannot change until the object begins to slip, a controller that is optimal with respect to deterministic dynamics will attempt to hold the object with the *minimum possible force*, i.e. on the very edge of the friction cone.

This delicate grip would be disastrously fragile – the smallest perturbation would cause the object to fall. If, however, there is uncertainty in our model, we can only maintain our grip *in probability*. By grasping with more force than is strictly necessary, we push probability-mass from the slip-regime into the stick-regime.

The moral is that optimal control of discontinuous dynamics cannot give acceptable results with a deterministic model. Our intuition of what constitutes a reasonable solution implicitly contains the notion of robustness, which requires the explicit modeling of noise near the discontinuities.

III. Background

A. Dynamics with unilateral contacts

The modeling and simulation of multibody systems with contacts and friction is a broad and active field of research [2]. One reasonable approach, not investigated here, is to model discontinuous phenomena with a continuous-time *hybrid* dynamical system which undergoes switching [3]. These methods require accurate resolution of collision and separation times, so fixed time-step integration with a fixed computational cost is impossible. Moreover, collision events can in principle occur infinitely often in a finite time, as when a rigid elastic ball bounces to rest. The appropriate control strategy would be to chop the trajectory into several first-exit problems, where contact surfaces serve as an exit manifold for one segment, while the post-collision state serves as an initial condition for the next one. It is not clear how a trajectory-optimizer for such a system could deal with unforeseen changes to the switch sequence, as when a foot makes grazing contact. Furthermore, such a scheme would mandate that every contact/stick-slip *configuration* be considered a hybrid system component, so their number would grow combinatorially.

Time-stepping avoids all of these problems. We give a brief exposition here and refer the interested reader to [4]. The equations of motion of a controlled mechanical system are

$$\mathbf{M}\dot{\mathbf{v}} = \mathbf{r} + \mathbf{u}$$
$$\dot{\mathbf{q}} = \mathbf{v}$$

where $\mathbf{q}$ and $\mathbf{v}$ are respectively the generalized coordinates and velocities, $\mathbf{M} = \mathbf{M}(\mathbf{q})$ is the mass matrix, $\mathbf{r} = \mathbf{r}(\mathbf{q}, \mathbf{v})$ the vector of total external forces (gravity, drag, centripetal, coriolis etc.) and $\mathbf{u}$ is the applied control signal (e.g. motor torques).

It is non-trivial to model discontinuous phenomena like rigid contact and Coulomb friction in continuous time. Doing so requires instantaneous momentum transfers via unbounded forces and the formalism of Measure Differential Inclusions to fully characterize solutions [5]. Time-stepping integrators rewrite the equations of motion in a discrete-time momentum-impulse formulation. Because the integral of force (the impulse) is always finite, unbounded quantities are avoided, and both *bounded* and *impulsive* forces can be properly addressed in one step.

For a timestep h, an Euler integration step for the momenta[1] $\mathbf{M}\mathbf{v}' = \mathbf{M}\mathbf{v}(t+h)$ and coordinates $\mathbf{q}' = \mathbf{q}(t+h)$ is:

$$\mathbf{M}\mathbf{v}' = h(\mathbf{r} + \mathbf{u}) + \mathbf{M}\mathbf{v}$$
$$\mathbf{q}' = \mathbf{q} + h\mathbf{v}'.$$

We want a unilateral constraint vector function $\mathbf{d}(\mathbf{q})$ to remain non-negative, for example a *signed distance* between objects, which is positive for separation, zero for contact and negative for penetration. We therefore demand that impulses λ be applied so that

$$\mathbf{d}(\mathbf{q}') \approx \mathbf{d}(\mathbf{q}) + h\mathbf{J}\mathbf{v}' \geq 0,$$

where $\mathbf{J} = \nabla\mathbf{d}(\mathbf{q})$. This leads to the following complementarity problem for $\mathbf{v}'$ and λ:

$$\mathbf{M}\mathbf{v}' = h(\mathbf{r} + \mathbf{u}) + \mathbf{M}\mathbf{v} + \mathbf{J}^\mathsf{T}\lambda \tag{1a}$$
$$\lambda \geq 0, \tag{1b}$$
$$\mathbf{d}(\mathbf{q}) + h\mathbf{J}\mathbf{v}' \geq 0, \tag{1c}$$
$$\lambda^\mathsf{T}(\mathbf{d}(\mathbf{q}) + h\mathbf{J}\mathbf{v}') = 0. \tag{1d}$$

Conditions (1b) and (1c) are read element-wise, and respectively constrain the contact impulse to be non-adhesive, and the distance to be non-penetrative. Condition (1d) asserts that $\mathbf{d}(\mathbf{q}') > 0$ (broken contact) and $\lambda > 0$ (collision impact), are mutually exclusive.

Since the mass matrix is always invertible, we can solve (1a) for $\mathbf{v}'$, and plug into (1c). Defining

$$\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^\mathsf{T} \tag{2a}$$
$$\mathbf{b} = \mathbf{d}(\mathbf{q})/h + \mathbf{J}\mathbf{v} + h\mathbf{J}\mathbf{M}^{-1}(\mathbf{r} + \mathbf{u}), \tag{2b}$$

we can now write (1) in standard LCP form:

$$\text{Find } \lambda \quad s.t. \quad 0 \leq \lambda \ \perp \ \mathbf{A}\lambda + \mathbf{b} \geq 0. \tag{3}$$

In order to solve for frictional impulses λ_f, the Coulomb friction law $\|\lambda_f\| \leq \mu\lambda$ must be incorporated. To maintain linearity, a polyhedral approximation to the friction cone is often used, though some new methods can handle a smooth one [7]. In the model of [8], the matrix $\mathbf{A}$ is no longer positive-definite, though a solution is guaranteed to exist. In the relaxed model of [9], $\mathbf{A}$ remains positive-definite. With either model the problem retains the LCP structure (3).

B. Smoothing Methods for LCPs

The Linear Complementarity Problem (3) arises in many contexts and has stimulated considerable research [10][11]. One method of solving LCPs involves the use of a so-called NCP function $\phi(\cdot, \cdot)$ whose root satisfies the complementarity condition

$$\phi(a, b) = 0 \quad \Longleftrightarrow \quad a \geq 0,\ b \geq 0,\ ab = 0.$$

Due to the element-wise nature of complementarity, is suffices to consider NCP functions with scalar arguments. Two popular examples are

$$\phi(a, b) = \min(a, b) \tag{4}$$

[1]In the derivation we assume $\mathbf{M}$ and $\mathbf{r}$ to be constant throughout the time-step, but for our simulations used a more accurate approximation [6].

and

$$\phi(a,b) = a + b - \sqrt{a^2 + b^2}.$$

These functions reformulate the complementarity problem as a system of (possibly nonsmooth) nonlinear equations, whose residual can then be minimized. A particular method, closely related to the one used here, was presented by Chen and Mangasarian in [12] and proceeds as follows. Rewrite (4) as $\min(a,b) = a - \max(a-b,0)$ and replace $\max(\cdot,0)$ with a smooth approximation

$$s(x,\epsilon) \;\to\; \max(x,0) \quad \text{as} \quad \epsilon \downarrow 0.$$

One of several such functions proposed there is

$$s(x,\epsilon) = \epsilon \log(1 + e^{x/\epsilon}). \tag{5}$$

The authors then present an iterative algorithm whereby for a positive ϵ, the squared residual

$$r(a,b,\epsilon) = \big(a - s(a-b,\epsilon)\big)^2 \tag{6}$$

is minimized with a standard nonlinear minimization technique, ϵ is subsequently decreased, and the procedure repeated until a satisfactory solution is obtained.

C. Local Optimization

Local Optimal Control methods, with roots in the venerable Maximum Principle [13], solve the control problem along a single *trajectory*, giving either an open-loop or a closed-loop policy that is valid in some limited volume around it. They are efficient enough to be used for real-time control of fast dynamics [14], and scale well enough to handle high-dimensional nonlinear mechanisms [15]. These methods can get stuck in local minima, though if neural controllers set the golden standard, it might be noted that biological suboptimal minima exist, somewhat anecdotally evidenced by the high-jump before Dick Fosbury. Their main detraction is that they solve the problem in only a small volume of space, namely around the trajectory. To remedy this, new methods [16] use Sum of Squares verification tools to measure the size of local basins-of-attraction and constructively patch together a global feedback controller. Finally, using the control-estimation duality, the powerful framework of estimation on graphical models is being brought to bear [17] on trajectory optimizers.

Though policy gradient methods like *shooting* can also be considered, we concentrate on Local Dynamic Programming, i.e. the construction of a local approximation to the Value function, and restrict ourselves to the finite-horizon case.

The control $\mathbf{u} \in \mathbb{R}^m$ affects the propagation of the state $\mathbf{x} \in \mathbb{R}^n$ through the general Markovian dynamics

$$\mathbf{x}' = \mathbf{f}(\mathbf{x},\mathbf{u}). \tag{7}$$

The cost-to-go starting from state $\mathbf{x}$ at time i with a control sequence $\mathbf{u}_{i:N-1} \equiv \{\mathbf{u}_i, \mathbf{u}_{i+1} \dots, \mathbf{u}_{N-1}\}$, is the sum of running costs[2] $\ell(\mathbf{x},\mathbf{u})$ and final cost $\ell_f(\mathbf{x})$:

$$J_i(\mathbf{x}_i, \mathbf{u}_{i:N-1}) = \sum_{k=i}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \ell_f(\mathbf{x}_N),$$

[2]The possible dependence of ℓ on i is suppressed for compactness.

We define the optimal Value function at time i as the the cost-to-go given the minimizing control sequence

$$V_i^*(\mathbf{x}) \equiv \min_{\mathbf{u}_{i:N-1}} J_i(\mathbf{x}, \mathbf{u}_{i:N-1}).$$

Setting $V_N^*(\mathbf{x}) \equiv \ell_f(\mathbf{x}_N)$, the Dynamic Programming principle reduces the minimization over the entire sequence of controls to a sequence of minimizations over a single control, proceeding backwards in time:

$$V_i^*(\mathbf{x}) = \min_{\mathbf{u}} [\ell(\mathbf{x},\mathbf{u}) + V_{i+1}^*(\mathbf{f}(\mathbf{x},\mathbf{u}))] \tag{8}$$

1) First-Order Dynamic Programming: To derive a discrete-time equivalent of the Maximum Principle, we observe the following: Given a first-order approximation of the Value at $i{+}1$, if $\mathbf{f}$ is affine in $\mathbf{u}$ (which holds for mechanical systems) and ℓ is convex and smooth in $\mathbf{u}$ (so that $\nabla_{\mathbf{u}}\ell$ is invertible), then the minimizing $\mathbf{u}$ is given by:

$$\mathbf{u}_i^* = -\nabla_{\mathbf{u}}\ell^{-1}\left(\nabla_{\mathbf{u}}\mathbf{f}^{\mathsf{T}} \nabla_{\mathbf{x}} V_{i+1}\right) \tag{9}$$

with dependencies on $\mathbf{x}$ and $\mathbf{u}$ suppressed for readability. Once $\mathbf{u}_i^*$ is known, the approximation at time i is given by

$$\nabla_{\mathbf{x}} V_i(\mathbf{x}) = \nabla_{\mathbf{x}}\big(\ell(\mathbf{x},\mathbf{u}_i^*) + V_{i+1}(\mathbf{f}(\mathbf{x},\mathbf{u}_i^*))\big). \tag{10}$$

The first-order local dynamic programming algorithm proceeds by alternatingly propagating the dynamics forward with (7), and propagating $\mathbf{u}_i$ and $\nabla_{\mathbf{x}} V_i(\mathbf{x})$ backward with (9) and (10).

2) Second-Order Dynamic Programming: By propagating a quadratic model of $V_i(\mathbf{x})$, second-order methods can compute locally-linear policies. These provide both quadratic convergence rate and a more accurate, closed-loop controller. We define the *unminimized* Value function

$$Q_i(\delta\mathbf{x},\delta\mathbf{u}) = \ell(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}) + V_{i+1}(\mathbf{f}(\mathbf{x}+\delta\mathbf{x}, \mathbf{u}+\delta\mathbf{u}))$$

and expand to second order

$$\approx \frac{1}{2}\begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} Q_0 & Q_x{}^{\mathsf{T}} & Q_u{}^{\mathsf{T}} \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}. \tag{11}$$

Solving for the minimizing $\delta\mathbf{u}$ we have

$$\delta\mathbf{u}^* = \underset{\delta\mathbf{u}}{\operatorname{argmin}}[Q_i(\delta\mathbf{x},\delta\mathbf{u})] = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta\mathbf{x}), \tag{12}$$

giving us both open-loop and linear-feedback control terms. This control law can then be plugged back into (11) to obtain a quadratic approximation of V_k. As in the first-order case, these methods proceed by alternating a forward pass which propagates the dynamics using the current policy, and a backward pass which re-estimates the Value and produces a new policy.

D. Optimal Control of Discontinuous Systems

To date, there has not been a profusion of research on Optimal Control of discontinuous dynamics. Stewart and Anitescu's recent contribution [18] includes a literature review. In that paper, the authors describe a method whereby a smooth approximation replaces the discontinuities. They show that in some limit, the solution of the smooth system converges to the solution of the nonsmooth one, and proceed to apply their method to the "Michael Schumacher" racing car problem. We would argue that though a solution is obtained, it might be a solution to the wrong problem. Qualitatively, it reaches the very limits of tire traction, and passes within a whisker of the walls. The smallest disturbance would send the car crashing. Their example is appropriate since this extreme driving well-describes the car racing profession, but the optimum is clearly non-robust. A driver who thinks that the road is slippery or the steering wheel is inaccurate, would most likely be considered a "better driver" by other standards.

The method presented in the next section describes a specific way of accounting for stochasticity when controlling dynamics with complementarity conditions, but in general, noise has a profound effect on optimal solutions. For deterministic continuous-time systems, even infinitely differentiable ones, the Value function, which satisfies the Hamilton Jacobi Bellman PDE, is often discontinuous. A solution always exists in the *Viscosity Solution* sense [19], but the fact that a special formalism is required is conspicuous. If, however, the dynamics are a stochastic diffusion with positive-definite noise covariance, the HJB equation gains a second-order Itô term, and the solution is always unique and smooth, regardless of discontinuities in the underlying dynamics.

IV. The Proposed Method

Instead of solving for contact impulses with an LCP, we propose instead to solve a Stochastic LCP. The incorporation of stochasticity makes the contact impulses a differentiable function of the state, facilitating the use of local control methods. Because a controller must always overcome noise (process, observation, modeling), the noise-induced smooth dynamics constitute a better model for control purposes, even if they are a worse approximation of the real physical mechanism.

A. New and old formulas for SLCPs

The study of Stochastic Linear Complementarity Problems is a fairly recent endeavor, see [20] for a survey. The general problem could be written

$$0 \leq \mathbf{x} \;\perp\; \mathbf{A}(\omega)\mathbf{x} + \mathbf{b}(\omega) \geq 0. \tag{13}$$

Where ω is random variable. In this form the problem is obviously not well-posed, since it is not clear in what sense $\mathbf{x}$ satisfies the constraints. The Expected Residual Minimization approach of [1], proposes that we minimize

$$r_{\text{ERM}}(\mathbf{x}) = \mathbf{E}\left[\|\Phi(\mathbf{x},\omega)\|^2\right] \tag{14}$$

where

$$\Phi(\mathbf{x},\omega)_i = \phi\Big(\mathbf{x}_i,\ \big(\mathbf{A}(\omega)\mathbf{x} + \mathbf{b}(\omega)\big)_i\Big)$$

is a vector of NCP residuals.

As detailed in section IV-B, in our case $\mathbf{A}$ can be considered constant to first order. This variant is discussed in section 3 of [1], where after a simple proof that $r_{\text{ERM}}(\mathbf{x})$ is continuously differentiable, it is explicitly computed for a uniformly distributed $\mathbf{b}$. Because we are ultimately trying to model a diffusion, the Normal distribution would be more appropriate. Considering scalar arguments a and $b_{\mathcal{N}}$, with

$$b_{\mathcal{N}} \sim p(b_{\mathcal{N}}) = \mathcal{N}(b_{\mathcal{N}}|b,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\tfrac{1}{2}\left(\tfrac{b_{\mathcal{N}}-b}{\sigma}\right)^2\right)$$

and cumulative distribution

$$P(b_{\mathcal{N}}) = \int_{-\infty}^{b_{\mathcal{N}}} p(t)dt = \frac{1}{2}\left(1 + \text{erf}\left(\tfrac{b_{\mathcal{N}}-b}{\sigma\sqrt{2}}\right)\right),$$

the expectation produces

$$\mathbf{E}\left[\min(a,b_{\mathcal{N}})^2\right] = \\ a^2 - \sigma^2(a+b)p(a) + \left(\sigma^2 + b^2 - a^2\right)P(a). \tag{15}$$

A possible variant of equation (14), would be to exchange squaring and expectation, giving what might be termed Residual Expectation Minimization:

$$r_{\text{REM}}(\mathbf{x}) = \|\,\mathbf{E}\left[\Phi(\mathbf{x},\omega)\right]\|^2 \tag{16}$$

Due to Jensen's inequality, $r_{\text{REM}}(\mathbf{x}) \leq r_{\text{ERM}}(\mathbf{x})$, and is thus a weaker bound, though their minima coincide as $\sigma \downarrow 0$. A benefit of this residual is that the expectation gives simpler formulae. In particular, if $b_{\mathcal{L}}$ has a Logistic distribution

$$b_{\mathcal{L}} \sim \mathcal{L}(b_{\mathcal{L}}|b,\sigma) = \frac{1}{\sigma}\Big(\exp\left(\tfrac{b_{\mathcal{L}}-b}{2\sigma}\right) + \exp\left(\tfrac{b-b_{\mathcal{L}}}{2\sigma}\right)\Big)^{-2}$$

then the expectation produces

$$\mathbf{E}[\min(a,b_{\mathcal{L}})] = a - \sigma\log(1 + e^{\frac{a-b}{\sigma}}). \tag{17}$$

It is clear that this residual is identical to (6) with the smoothing function (5), where σ takes the place of ϵ. This immediately provides us with a new interpretation to the method of [12], and gives us access to the literature which investigates it (e.g. [21]).

With these smooth approximations[3] to ϕ, the minimizing $\mathbf{x}$ is now a differentiable function of $\mathbf{A}$ and $\mathbf{b}$.

[3] Although we only used formulae (15) and (17) in our experiments, for completeness we also provide the expectation of the $\min(\cdot,\cdot)$ function with a Normally distributed argument:

$$\mathbf{E}[\min(a,b_{\mathcal{N}})] = a - \sigma^2 p(a) - (a-b)P(a),$$

and of $\min(\cdot,\cdot)^2$ with a Logistically distributed argument:

$$\mathbf{E}[\min(a,b_{\mathcal{L}})^2] = a^2 - 2a\sigma\log(1 + e^{\frac{a-b}{\sigma}}) + 2\sigma^2\,\text{Li}_2(-e^{\frac{a-b}{\sigma}}),$$

where $\text{Li}_2(x)$ is the Dilogarithm function $\text{Li}_2(x) = -\int_x^0 \frac{\log(1-t)}{t}dt$.

B. Determining the noise covariance

How should the noise covariance σ be determined? Assume that a gaussian state distribution is estimated from observations by some filter, and we are given

$$\Sigma_{\mathbf{q}} = \mathbf{E}[\mathbf{q}\mathbf{q}^\mathsf{T}] \quad \text{and} \quad \Sigma_{\mathbf{v}} = \mathbf{E}[\mathbf{v}\mathbf{v}^\mathsf{T}].$$

Examining (2a) we see that $\mathbf{A}$ is a function of the mass matrix $\mathbf{M}$ and the constraint Jacobian $\mathbf{J}$. Both of these depend on $\mathbf{q}$ but often smoothly and slowly. In contrast, since h must be small due to the Euler integration, its presence in the denominator of the first term of the RHS of (2b), suggests that the appropriate first-order approximation is

$$\Sigma_{\mathbf{b}} = \mathbf{E}[\mathbf{b}\mathbf{b}^\mathsf{T}] \approx \mathbf{J}\Sigma_{\mathbf{q}}\mathbf{J}^\mathsf{T} h^{-2}$$

To account for the second term as well, one would use

$$\Sigma_{\mathbf{b}} \approx \mathbf{J}(\Sigma_{\mathbf{q}} h^{-2} + \Sigma_{\mathbf{v}})\mathbf{J}^\mathsf{T}.$$

The element-wise nature of the complementarity conditions allows us to ignore the off-diagonal terms and use

$$\mathbf{b}(\omega)_i \sim \mathcal{N}(\omega|\mathbf{b}_i, \sqrt{(\Sigma_{\mathbf{b}})_{ii}}) \quad \text{or} \quad \mathcal{L}(\omega|\mathbf{b}_i, \sqrt{(\Sigma_{\mathbf{b}})_{ii}}), \tag{18}$$

to define the vector residual as

$$\Phi(\mathbf{x},\omega)_i = \phi\Big(\mathbf{x}_i,\ \mathbf{A}_i\mathbf{x} + \mathbf{b}(\omega)_i\Big).$$

C. Making global controllers from trajectories

The output of the second-order trajectory optimizer of section III-C.2 is a time-dependent sequence of linear feedback policies $\mathbf{u}()_{1:N-1}$

$$\mathbf{u}(\mathbf{x})_i = \mathbf{u}_i - Q_{i\,uu}^{-1} Q_{i\,ux}(\mathbf{x} - \mathbf{x}_i).$$

Ideally, we would like to use an online Model Predictive Control strategy, whereby we iteratively use $\mathbf{u}()_1$ for one time-step and re-solve the problem. Our simulations were not fast enough for that (see below), so we resort to an off-line MPC strategy, as in [15]. Once a solution trajectory is obtained for some $\mathbf{x}_1$, we save $\mathbf{u}()_1$, and proceed to solve for a new trajectory starting at $\mathbf{x}_{1+d}$ for a small offset d. We can use the previous control sequence shifted by $-d$ and appended with d copies of the last control

$$\mathbf{u}^{\text{NEW}}()_{1:N-1} = [\mathbf{u}()_{1+d:N-1}\ \ \mathbf{u}()_{N-1}\ \ \mathbf{u}()_{N-1} \cdots], \tag{19}$$

as our initial guess for the policy of the shifted trajectory. Because we are usually not far from the optimum, we enjoy the quadratic convergence properties of second-order methods. The trajectory thus propagates forward, leaving behind it a trail of time-independent controllers $\mathbf{u}()_k$, all with horizon $T = hN$. We now use this collection of local linear controllers to construct a global controller that can be used online, by following a simple nearest neighbor rule:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x})_j \quad \text{with} \quad j = \operatorname*{argmin}_k \|\mathbf{x} - \mathbf{x}_k\|^2. \tag{20}$$

This is similar to the Trajectory Library concept [22]. If, as in the case below, the solution lies on a limit cycle, the time-independent trajectory will converge to it. Now we can perturb the initial state $\mathbf{x}_1$ every several time steps, and explore the state space around the limit cycle.

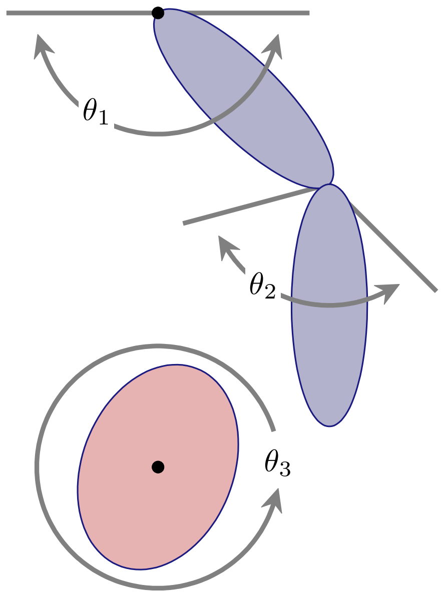


Fig. 1. The "flicking finger" dynamical system. The controller actuates θ_1 and θ_2 in order to spin the free ellipse around θ_3.

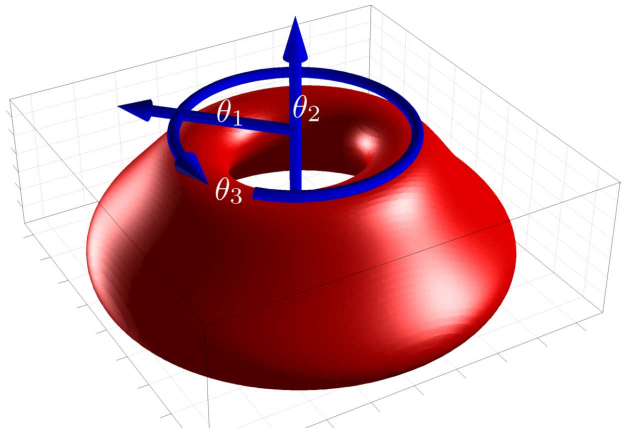


Fig. 2. The contact surface in the configuration space $[\theta_1\ \ \theta_2\ \ \theta_3]^\mathsf{T}$. The torus-like shape is the zero-valued isosurface of the distance function $\mathbf{d}(\mathbf{q})$. It corresponds to the set of points where the finger-tip makes contact with the free ellipse. Points inside and outside the surface correspond to penetration and broken contact respectively. A cylindrical coordinate system was chosen because θ_3 is a periodic variable (with period π rather than 2π due to symmetry).

V. Experiments

A. Setup

We performed our experiments with the 6-dimensional system of Figure 1. This planar system is composed of two articulated ellipses, with angles θ_1 and θ_2, and a free-spinning ellipse, whose angle is given by θ_3. The full state of the system is thus

$$\mathbf{x} = [\theta_1\ \theta_2\ \theta_3\ \dot{\theta}_1\ \dot{\theta}_2\ \dot{\theta}_3]^\mathsf{T}$$

The controller can apply two torques $\mathbf{u} = [u_1\ \ u_2]^\mathsf{T}$ to θ_1 and θ_2. We make it the controller's goal to spin the free ellipse in the positive direction, by defining a negative state-cost proportional to θ_3 with a quadratic control-cost:

$$\ell(\mathbf{x}, \mathbf{u}) = -c_x\theta_3 + c_u\|\mathbf{u}\|^2$$

Note that it would be very difficult to solve this task with control techniques that force the system to a pre-planned trajectory, since it is not clear how such a trajectory would be found.

In Figure 2, we depict the $\mathbf{d}(\mathbf{q}) = 0$ isosurface in the configuration space $[\theta_1\ \theta_2\ \theta_3]^\mathsf{T}$.

B. Parameters and methods

The following values can all be assumed to have the appropriate units of a self consistent unit system (e.g. MKS). The major and minor radii of the finger ellipses are .8 and .25 respectively. Those of the free ellipse are .7 and .5. The masses and moments of inertia correspond to a mass density of 1. The vertical distance between the two anchors (black dots in Figure 1) is 3. Angle limits are $-\pi \le \theta_1 \le 0$ and $-2\pi/3 \le \theta_2 \le 0$. The drag coefficients of the finger joints and of the free ellipse axis are .2, .2 and .7, respectively. Gravity in the vertical direction is -4. The control-cost coefficients were $c_u = 0.05$ and $c_x = 1$. The time step $h = 0.05$ and the number of time steps per trajectory $N = 75$, for a time horizon of $T = 3.5$.

Both angle constraints and the contact constraint were satisfied with impulses, as described in section III-A. We used a friction coefficient of $\mu = 0.5$, with the friction model of [9]. Since we did not estimate the dynamics, we used a constant $\sqrt{(\Sigma_\mathbf{b})_{ii}} = 0.1$, which was chosen as a trade-off between good convergence and small softening of the contact (described below). For the SLCP residual, both (15) and (17) gave qualitatively similar results, and we used (17) in the results below, because it is computationally cheaper.

We avoided the matrix inversions of (2) by directly solving the (stochastic) mixed complementarity problem of (1). This is easy to do with NCP functions, by setting $\phi(a, b) = b$ for those indexes where equality is desired.

We propagated the receding-horizon trajectory 400 times, with an offset $d = 4$, giving us a library of 400 locally linear feedback controllers. Every 15 time-steps, the velocities of initial state were perturbed by normal noise of variance 4. The reason only the velocities were perturbed is that random perturbations of the angles might lead to illegal (penetrative) configurations. When selecting the nearest neighbor of equation (20), we used a Euclidian norm scaled by the covariance of all the $\mathbf{x}_i$. In Figure 3, we plot the locations of the origins $\mathbf{x}_i$, projected onto the first 3 dimensions of the state-space.

Our simulator[4] was written in MATLAB for versatility and code readability, but is therefore not very efficient. One forward-backward pass of the local method took $\sim 5_s$ on a quad-core Core i7 machine. Starting from the zero policy, convergence was attained in $\sim$60 passes. In receding-horizon mode, starting from the shifted previous policy (19), $\sim$8 passes sufficed, due to quadratic convergence.

[4]Available online at http://alice.nc.huji.ac.il/~tassa/

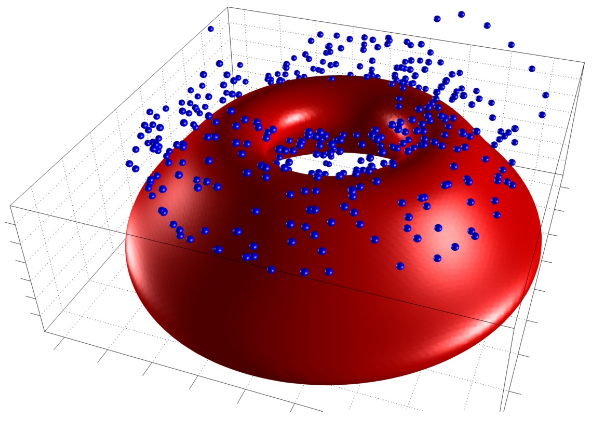


Fig. 3. The trajectory library. The dots correspond to the origins of 400 linear feedback controllers which we select with the nearest-neighbor rule (20). The global policy is effectively a Voronoi tessellation of these policies.

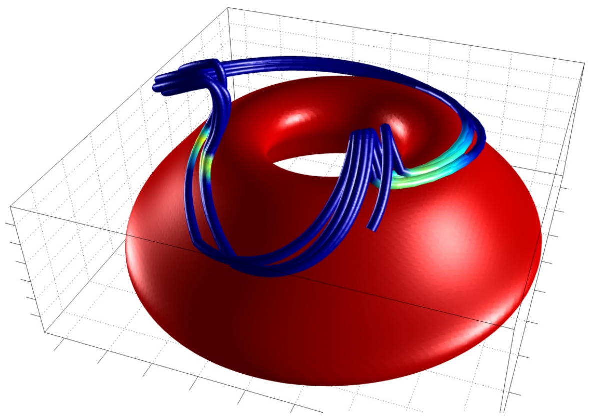


Fig. 4. The limit-cycle of a controlled trajectory. The color of the trajectory is proportional to the contact impulses, showing the points when flicking occurs.

C. Results

Our use of the SLCP when solving for velocities and impulses has the qualitative effect of creating a “force field” which extends from surfaces and lets contact and friction impulses act at a distance. If we could simulate the true stochastic system, some probability mass would experience contact before the mean would. The “fuzzy” force layer in our new deterministic system is an approximation to the mean force that would act on the mean of the distributed state.

The controller that was synthesized is very robust. We tested it on both the smoothed system and the original non-smooth one. For both systems, the controller dependably spun the ellipse, and recovered from disturbances. As an indicator of robustness, the controller worked well for time-steps different than the one which it was trained. We built a real-time simulator with a graphical front-end, which allows the user to perturb the system as it is being controlled, by interactively pulling on the ellipses with the mouse cursor. We were unable to perturb the system out of the controller’s basin-of-attraction.

In order to assess the contribution of the smoothing to robustness, we solved again, with a smaller noise covariance

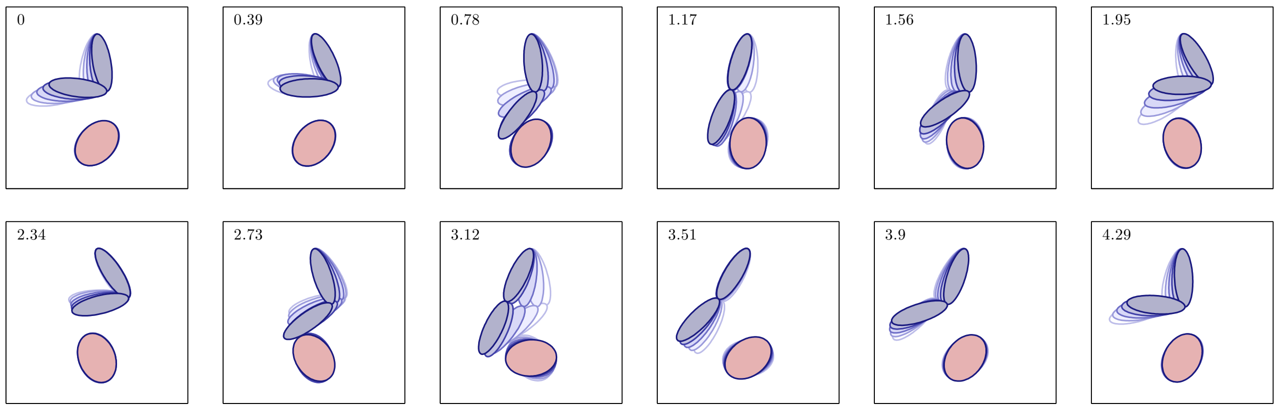


Fig. 5. Animation frames from the limit cycle. The third frame (time=0.78) and eighth frame (time=2.73) correspond to the contacts on the left and right side of Figure 4, respectively.

$\sqrt{(\Sigma_\mathbf{b})_{ii}} = 0.05$. The resulting controller was less robust in all of the senses described above. In particular, this controller sometimes entered into a non-productive limit-cycle where no contact was achieved. For smaller noise covariances the trajectory optimizer did not converge, so a direct comparison was not possible.

In Figure 4, we show the limit-cycle that a controlled trajectory makes in configuration space, with $h = 0.03$. The dynamics used were those of the original, nonsmooth system. The color of the trajectory is proportional to the contact impulses, showing the points when flicking occurs. The contact on the left of the figure corresponds to a weak tap that repositions the ellipse at a favorable angle, while the second one on the right side, exploits this position to deliver a stronger flick, that spins the ellipse. In Figure 5 we show animation frames from this sequence. The first and second contacts correspond to frame 3 (t=0.78) and frame 8 (t=2.73).

VI. Discussion

Our purpose here was to investigate the applicability of local methods to nonlinear control problems involving contact and friction. Our proposed method involves modifying the contact solver of the modeled dynamics, in a way which takes into account the controller's uncertainty regarding the state of the system. Though we present promising preliminary results that include a working robust controller, open issues remain.

Folding the noise into the dynamics, as we do here, is an approximation of what we would really like to do, namely simulate the full stochastic system, and measure costs WRT distributions rather than points. This option however has its own problems. The actual distributions which are propagated by the true dynamics become multimodal upon contact and require either a nontrivial parameterization, or a large number of "particles" for a non-parametric representation. In the case where distributions are represented as a mixture of samples, whether particles or σ-points, the dynamics would still be non-differentiable.

Local minima are still a significant problem for local methods like the ones used here. If we had not included gravity in the simulation, the initial trajectory with $\mathbf{u}_i = 0$ would have never achieved contact with the free ellipse, and the controller wouldn't have "known" about the possibility of contact. One option is to inject noise into the system during the initial stages of learning, for exploration purposes. Another option is to use the action-at-a-distance effect of the SLCP solution. By first solving for a system with very large postulated noise covariance, and then gradually reducing it, a scaffolding effect might be achieved.

The SLCP solution effectively replaces hard contact with a nonlinear spring-damper. However, unlike some arbitrary spring, it does not require setting the unknown parameters (spring and damper coefficients, form of the nonlinearity), and is derived in a principled way from the noise covariance. Additionally, unlike conventional springs which deform in proportion to the applied force, the forces which we compute scale with the effective inertia, so that a heavy and a light object experience the same smoothing.

A non-differentiable distance function $\mathbf{d}(\mathbf{q})$ would result in non-differentiable dynamics. For the ellipses used here, the signed distance is indeed differentiable, but this is not true for other shapes.

The complementarity conditions in (1) apply for quantities of different units: λ is an impulse while $\mathbf{d}(\mathbf{q}')$ is a distance. Because the smoothed NCP function $\phi(\cdot,\cdot)$ effectively mixes these quantities, a different choice of units would ostensibly lead to different results, which is clearly undesirable. This is a fundamental issue that requires further investigation. As famously observed by Stewart [4], "in many ways it is easier to write down a numerical method for rigid-body dynamics than it is to say exactly what the method is trying to compute".

Acknowledgment

This work was supported by the US National Science Foundation.

References

[1] X. Chen and M. Fukushima, "Expected residual minimization method for stochastic linear complementarity problems," *Math. Oper. Res.*, vol. 30, no. 4, pp. 1022–1038, 2005.

[2] F. Pfeiffer and C. Glocker, *Multibody dynamics with unilateral contacts*. Wiley-VCH, 1996.

[3] E. Westervelt, *Feedback control of dynamic bipedal robot locomotion*. Boca Raton: CRC Press, 2007.

[4] D. E. Stewart, "Rigid-Body dynamics with friction and impact," *SIAM Review*, vol. 42, no. 1, pp. 3–39, 2000.

[5] J. J. Moreau, "Unilateral contact and dry friction in finite freedom dynamics," *Nonsmooth mechanics and Applications*, p. 1–82, 1988.

[6] F. A. Potra, M. Anitescu, B. Gavrea, and J. Trinkle, "A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction," *International Journal for Numerical Methods in Engineering*, vol. 66, no. 7, pp. 1079–1124, 2006.

[7] E. Todorov, "Implicit nonlinear complementarity: a new approach to contact dynamics," in *International Conference on Robotics and Automation*, 2010.

[8] M. Anitescu and F. A. Potra, "Formulating dynamic Multi-Rigid-Body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, Nov. 1997.

[9] M. Anitescu, "Optimization-based simulation of nonsmooth rigid multibody dynamics," *Mathematical Programming*, vol. 105, no. 1, pp. 113–143, 2006.

[10] R. W. Cottle, J. Pang, and R. E. Stone, *The Linear Complementarity Problem*. SIAM, Oct. 2009.

[11] S. C. Billups and K. G. Murty, "Complementarity problems," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 303–318, Dec. 2000.

[12] C. Chen and O. L. Mangasarian, "A class of smoothing functions for nonlinear and mixed complementarity problems," *Computational Optimization and Applications*, vol. 5, no. 2, pp. 97–138, Mar. 1996.

[13] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, *The mathematical theory of optimal processes*. Interscience New York, 1962.

[14] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, 2007, p. 1.

[15] Y. Tassa, T. Erez, and W. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, p. 1465–1472.

[16] R. Tedrake, "LQR-Trees: feedback motion planning on sparse randomized trees," in *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[17] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[18] D. E. Stewart and M. Anitescu, "Optimal control of systems with discontinuous differential equations," *Numerische Mathematik*, 2009.

[19] W. H. Fleming and H. M. Soner, *Controlled Markov processes and viscosity solutions*. Springer Verlag, 2006.

[20] M. Fukushima and G. lin, "Stochastic equilibrium problems and stochastic mathematical programs with equilibrium constraints: A survey," *Pacific Journal of Optimization*, vol. to appear, 2010.

[21] X. Chen and P. Tseng, "Non-Interior continuation methods for solving semidefinite complementarity problems," *Mathematical Programming*, vol. 95, no. 3, pp. 431–474, Mar. 2003.

[22] M. Stolle and C. G. Atkeson, "Policies based on trajectory libraries," in *Proceedings of the International Conference on Robotics and Automation (ICRA 2006)*, 2006.

Chapter 6

Solving for Limit-Cycles

6.1 Introduction

As we've shown in the previous chapters, the only algorithms that can presently be expected to work off-the-shelf for high-dimensional nonlinear systems are local methods. They generate either an open-loop trajectory, as in Pontryagin's maximum principle and pseudo-spectral methods (Stengel, 1994; Ross and Fahroo, 2004), or a trajectory and a local feedback control law, as in Differential Dynamic Programming (Jacobson and Mayne, 1970) and iterative linear-quadratic-Gaussian control (Todorov and Li, 2005). The local nature of these methods is of course a limitation, however many interesting behaviors involve stereotypical movements, and the ability to discover those movements and generate them in a stable manner is very useful.

This chapter aims to address one of the major shortcomings of local methods – which is that they are limited to finite-horizon problem formulations. Instead we would like to have methods with similar efficiency but capable of solving infinite-horizon problems, in particular problems that give rise to complex periodic movements such as walking, running, swimming, flying (with wings), turning a screwdriver, etc. This requires optimization over cycles. Such optimization is difficult to cast as an optimal control problem because the underlying system becomes non-Markov. Here we overcome this difficulty by replacing the control problem with a dual Bayesian inference problem, and performing inference over a graphical model with loops. We use linear-Gaussian machinery (as in DDP and iLQG), thus when the dynamics are nonlinear we have to solve a sequential Bayesian inference problem: the solution at each iteration is used to re-linearize the system and define the inference problem for the next iteration. When the algorithm converges, the mean of the posterior gives the locally-optimal trajectory while the covariance of the posterior gives the local feedback control law. Since computing the correct covariance is important here, we perform inference using the variational approach of Mitter and Newton (2004), leading to an algorithm based on sparse matrix factorization rather than loopy belief propagation (where only the mean is guaranteed to be correct, see Weiss and Freeman (2001)).

The estimation-control duality which is at the heart of our new method arises within the recently-developed framework of linearly-solvable optimal control (Kappen, 2005; Todorov, 2009). Several control algorithms exploiting this duality (sometimes implicitly) have been developed (Attias, 2003; Toussaint, 2009; Kappen et al., 2009), however they are limited to finite-horizon formulations – which DDP and iLQG can already handle, with comparable efficiency as far as we can tell. The present paper exploits the estimation-control duality in more general graph structures for the first time.

6.2 Related work

The method presented below is related to three lines of research.

The first is classic local trajectory-optimization methods, such as the Maximum Principle of Pontryagin et al. (1962) and Differential Dynamic Programming of Jacobson and Mayne (1970). It is possible to use these methods to solve for limit cycles by "attaching" the first and last states. This can be done either approximately, by imposing a final-cost over distance from the initial state, or exactly, by employing multipliers which enforce the state constraint, as in the method of Lantoine and Russell (2008). We tried both of these approaches, and the inevitable result was a noticeable asymmetry around the attachment point, either in the state trajectory (when using final-cost), or in the controls (when using multipliers). The main insight is that these algorithms assume Markovity, which does not hold for a loop. One could also use a finite-horizon method with a very long horizon, that loops around the limit-cycle several times. By truncating the transients at both ends, we can get a decent approximation to the infinite-horizon solution. Clearly this is an inefficient use of computational resources, but can serve as a useful validation procedure for our algorithm.

The second related body of work involves directly optimizing the total cost of a limit-cycle, while enforcing periodicity. Wampler and Popovic (2009) and Ackermann and den Bogert (2010) are two recent examples, respectively from the computer graphics and biomechanics communities. The log-likelihood that we end up maximizing below is indeed analogous to such a cost, however our method generates a feedback controller around the limit-cycle, rather than simply open-loop controls.

Finally, the last several years have seen research into the subclass of stochastic nonlinear Optimal Control problems which are dual to Bayesian estimation. Specifically, Toussaint (2009) explores message-passing algorithms (Expectation Propagation) for the solution of Optimal Control problems. Murphy et al. (1999) and others have shown that when a graph has a loopy structure, message passing converges to the right mean but the wrong covariance. The procedure we describe below does not suffer from this drawback.

6.3 Optimal control via Bayesian inference

The basic intuition behind the duality we exploit here is that the negative log-likelihood in estimation corresponds to a state-dependent cost in control, and the difference (KL divergence) between the prior and the posterior corresponds to a control-dependent cost. The class of stochastic optimal control problems which have Bayesian inference duals in the above sense have received a lot of attention recently, because these problems have a number of other interesting properties, including the fact that the (Hamilton-Jacobi) Bellman equation becomes linear after exponentiation (Kappen, 2005; Todorov, 2008).

6.3.1 Linearly-Solvable Framework

A linearly-solvable MDP (or LMDP) is defined by a state cost $q(x) \geq 0$ and a transition probability density $p(x'|x)$ corresponding to the notion of passive dynamics. The controller is free to specify any transition probability density $\pi(x'|x)$ with the restriction that $\pi(x'|x) = 0$ whenever $p(x'|x) = 0$. In infinite-horizon average-cost problems p, π are further required to be ergodic. The cost rate function is

$$\ell(x, \pi(\cdot|x)) = q(x) + D_{\text{KL}}[\pi(\cdot|x) || p(\cdot|x)]$$

The KL divergence term is a control cost which penalizes deviations from the passive dynamics. Defining the *desirability* function $z(x) \triangleq \exp(-v(x))$ where $v(x)$ is the optimal cost-to-go, the optimal control is

$$\pi(x'|x) \propto p(x'|x) z(x')$$

The exponentiated Bellman equation becomes linear in z. In particular, for finite horizon problems this equation is

$$z_t(x) = \exp(-q(x)) \sum\nolimits_{x'} p(x'|x)\, z_{t+1}(x') \tag{1}$$

with $z_N(x)$ initialized from the final cost. One can also define the function $r(x)$ as the solution to the transposed equation:

$$r_{t+1}(x') = \sum\nolimits_{x} \exp(-q(x))\, p(x'|x)\, r_t(x) \tag{2}$$

with $r_0(x)$ being a delta function over the fixed initial state. Then the marginal density under the optimally-controlled stochastic dynamics can be shown to be

$$\mu_t(x) \propto r_t(x)\, z_t(x) \tag{3}$$

The duality to Bayesian inference is now clear: z is the backward filtering density, r is the forward filtering density, p is the dynamics prior, $q(x)$ is the negative log-likelihood (of some unspecified measurements), and μ is the marginal of the Bayesian posterior. We can also write down the density p^* over trajectories generated by the optimally-controlled stochastic dynamics, and observe that it matches the Bayesian posterior over trajectories in the estimation problem:

$$p^*(x_1, x_2, \cdots x_N|x_0) \propto \prod\nolimits_{t=1}^{N} \exp(-q(x_t))\, p(x_t|x_{t-1}). \tag{4}$$

These LMDPs can be used to model the continuous systems we are primarily interested in as follows. It has been shown that for controlled Ito diffusions in the form

$$dx = a(x)\, dt + B(x)(u\, dt + \sigma d\omega) \tag{5}$$

and cost functions in the form

$$\ell(x,u) = q(x) + \frac{1}{2\sigma^2}\|u\|^2 \tag{6}$$

the stochastic optimal control problem is a limit of continuous-state discrete-time LMDPs. The LMDP passive dynamics are obtained via explicit Euler discretization with time step h:

$$p(x'|x) = \mathcal{N}\left(x + ha(x),\ h\,\sigma^2 B(x)\, B(x)^{\mathsf{T}}\right) \tag{7}$$

where $\mathcal{N}$ denotes a Gaussian. The LMDP state cost is simply $hq\,(x)$. Note that the h-step transition probability of the controlled dynamics (with $u \neq 0$) is a Gaussian with the same covariance as (7) but the mean is shifted by $h\,B\,(x)\,u$. Using the formula for KL divergence between Gaussians, the general KL divergence control cost reduces to a more traditional control cost quadratic in u.

6.3.2 Periodic optimal control as Bayesian inference

Our goal now is to write down the trajectory probability p^* for infinite-horizon average-cost problems, and then interpret it as a Bayesian posterior. This cannot be done exactly, because here (4) involves infinitely-long trajectories which we cannot even represent unless they are periodic. Therefore we will restrict the density to the subset of periodic trajectories with period N. This of course is an approximation, but the hope is that most of the probability mass lies in the vicinity of such trajectories. Then $p^*\,(x_1, x_2, \cdots x_N)$ is the same as (4), except we have now defined $x_0 = x_N$.

While the trajectory probability for the control problem is no longer exact, (4) is still a perfectly valid Bayesian posterior for a graphical model with a loop. More precisely, $\exp\,(-q\,(x_t))$ are single-node potentials which encode evidence, while $p\,(x_t|x_{t-1})$ are pair-wise potentials which encode the prior. One caveat here is that, since the state space is continuous, the density may not be integrable. In practice however we approximate $p\,(x_t|x_{t-1})$ with a Gaussian, so integrability comes down to making sure that the joint covariance matrix is positive definite – which can be enforced in multiple ways (see below).

Once the Bayesian posterior over limit-cycle trajectories is computed, we need to recover the underlying control law for the stochastic control problem. The obvious approach is to set

$$\pi\,(x_t|x_{t-1}) = \frac{p^*\,(x_t, x_{t-1})}{p^*\,(x_{t-1})}$$

where $p^*\,(x_t, x_{t-1})$ and $p^*\,(x_{t-1})$ are the corresponding marginals of the trajectory probability, and then recover the physical control signal $u\,(x_t)$ by taking the mean. However this yields N different conditional distributions, and we need to somehow collapse them into a single conditional $\pi\,(x'|x)$ because the control problem we are solving is time-invariant. We have explored the two obvious ways to do the combination: average the π's weighted by the marginals $p^*(x_t)$, or use the π corresponding to the nearest neighbor. Empirically we found that averaging blurs the density too much, while the nearest neighbor approach works well. An even better approach is to combine all the $p^*(x_t, x_{t-1})$ into a mixture density, and then compute the conditional $\pi(x'|x)$ of the entire mixture. This can be done efficiently when the mixture components are Gaussians.

6.4 Algorithm

Probabilistic graphical models (Jordan, 1998) are an efficient way of describing conditional independence structures. A cycle-free directed graph (a tree) represents a joint probability as a product of conditionals

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\text{parents}(x_k))$$

This equation represents the factorization properties of p. *Message Passing* algorithms, which involve sequentially propagating local distributions along directed graphs, provably converge to the true posterior.

An alternative to directed graphical models are Markov Fields, whose graph is undirected, and may contain cycles. The joint distribution of a Markov Field is given by

$$p(\mathbf{x}) \propto \prod_{c \in C} \psi_c(x_c),$$

where C is the set of maximal cliques in the graph, and the ψ are called *potential functions*. Message Passing algorithms are not guaranteed to converge on this type of model. In particular, for models where the nodes are distributed as gaussians (as we will assume below), Weiss and Freeman (2001) had shown that posteriors and marginals converge to correct means, but not to the correct variances.

6.4.1 Potential Functions

Let $\{x_i\}_{i=1}^{N}$ be a set of state variables $x_i \in \mathbb{R}^n$, with the conditional dependency structure of a cycle. Let ij index over the pairs of sequential states

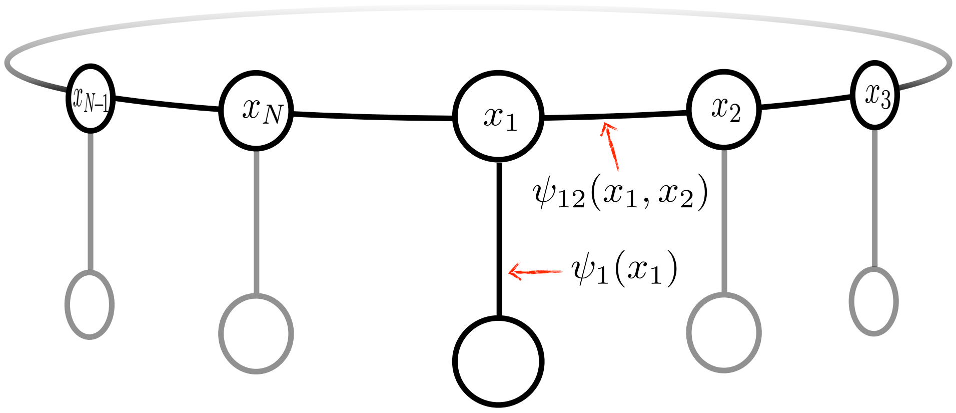


Figure 6.1: Illustration of probabilistic graphical model. State costs are encoded in the leaf potentials $\psi_i(x_i)$. Dynamics and control costs are encoded in the edge potentials $\psi_{ij}(x_i, x_j)$. See section 6.4.1.

$ij \in \{(1,2), (2,3), \ldots, (N,1)\}$. The discrepancy between the controlled dynamics and the discrete-time passive dynamics for each pair is

$$a_{ij} = x_i + ha(x_i) - x_j.$$

The gaussian noise leads to pairwise potentials, corresponding to $p(x_j|x_i)$,

$$\psi_{ij}(x_i, x_j) = p(x_j|x_i) = \exp(-\tfrac{1}{2} a_{ij}^{\mathsf{T}} \Sigma_i^{-1} a_{ij}),$$

where $\Sigma_i = \; h\,\sigma^2 B\,(x_i)\,B\,(x_i)^{\mathsf{T}}$, as in (7). The leaf potentials $\psi_i(x_i)$ are composed of two parts, the state-cost $q(x_i)$, and an optional prior on x_i. This prior, not used below, could be useful when we wish to clamp certain states to specified values. For example, in the finite-horizon case where the graph is a chain, we could place a gaussian prior on a known initial state

$$\psi_1(x_1) = \exp(-q(x_1))\mathcal{N}(x_1|m_1, \Sigma_1).$$

The joint distribution of the entire model is

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_N) \sim \prod_{i=1}^{N} \psi_i(x_i) \prod_{ij=1}^{N} \psi_{ij}(x_i, x_j)$$

Where $\mathbf{x} = \text{stack}\{x_i\} = [x_1^\mathsf{T} x_2^\mathsf{T} \cdots x_N^\mathsf{T}]^\mathsf{T}$ is the stacked vector of all states. The negative log-likelihood is

$$l(\mathbf{x}) = \sum_i q_i(x_i) + \sum_{ij} \tfrac{1}{2} a_{ij}^\mathsf{T} \Sigma_i^{-1} a_{ij}. \tag{8}$$

The first term is the total state-cost and the second term is the total control-cost.

6.4.2 Gaussian approximation

Modeling $p(\cdot)$ as a gaussian: $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{x}|\bar{\mathbf{x}}, \mathbf{S})$, is equivalent to fitting a quadratic model to the negative log likelihood.

$$l(\mathbf{x}) \approx \tfrac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^\mathsf{T} \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}}) = l_0 + \mathbf{x}^\mathsf{T}\mathbf{g} + \tfrac{1}{2}\mathbf{x}^\mathsf{T} H \mathbf{x}. \tag{9}$$

The normalization term $\frac{1}{2}\log(\det(2\pi\mathbf{S}))$ is folded into the constant l_0. The mean $\bar{\mathbf{x}}$ (which maximizes the likelihood) and the covariance $\mathbf{S}$ are given by

$$\bar{\mathbf{x}} = -H^{-1}\mathbf{g} \tag{10a}$$

$$\mathbf{S} = H^{-1} \tag{10b}$$

6.4.3 Iterative inference

Given a current approximation to the mean $\bar{\mathbf{x}}$ of the Bayesian posterior over trajectories, we expand $l(\bar{\mathbf{x}} + \delta\mathbf{x})$ to second-order in $\delta\mathbf{x}$ by computing H and g, and then find a new mean as

$$\bar{\mathbf{x}}' = \bar{\mathbf{x}} + \underset{\delta\mathbf{x}}{\operatorname{argmin}}\, l(\bar{\mathbf{x}} + \delta\mathbf{x}) = \bar{\mathbf{x}} - H(\bar{\mathbf{x}})^{-1}\mathbf{g}(\bar{\mathbf{x}}), \tag{11}$$

until convergence. The actual inversion of the precision matrix or Hessian H, as required by (10b), can be performed only once, at the end. During the iterations of (11), we can use iterative methods (e.g. preconditioned conjugate gradients) to solve $H\mathbf{y} = \mathbf{g}$, which are very cheap for sparse systems. Again, this process can be interpreted either as repeated estimation of a joint gaussian model, or sequential quadratic minimization of the total cost.

6.4.4 Dynamics model

We now turn to the computation of H and $\mathbf{g}$. Placing the cost Hessians on a the block-diagonal of the matrix $Q = \text{diag}\{\frac{\partial^2}{\partial x^2} q_i(\bar{x}_i)\} \in \mathbb{R}^{nN \times nN}$ and stacking the local cost gradients $q_{\mathbf{x}} = \text{stack}\{\frac{\partial}{\partial x} q_i(\bar{x}_i)\} \in \mathbb{R}^{nN}$, the last term of (8) can be approximated

$$\sum_i q_i(\bar{x}_i + \delta x_i) \approx \sum_i q_i(\bar{x}_i) + \delta\mathbf{x}^\mathsf{T} q_{\mathbf{x}} + \tfrac{1}{2}\delta\mathbf{x}^\mathsf{T} Q \delta\mathbf{x}$$

In order to quadratize the last term of (8), we must approximate the non-linear dynamics (or rather the dynamic discrepancies a_{ij}). We can do this using either a linear or a quadratic model. The former is faster to compute at each iteration, while the latter is more accurate and thus could yield convergence in fewer iterations. Which approach is better probably depends on the problem; in the examples given below we found that the quadratic model works better.

Linear dynamics approximation:

We expand the dynamic discrepancies to first order around our current approximation,

$$a_{ij}(\bar{x}_i + \delta x_i, \bar{x}_j + \delta x_j) = \bar{a}_{ij} + a_x(\bar{x}_i)\delta x_i - \delta x_j.$$

We construct the sparse matrix $A \in \mathbb{R}^{nN \times nN}$ as a stack of N block-rows of dimension $n \times nN$. For each pair ij, we place a negative identity matrix $-I_n$ on the j-th column-block and the dynamics Jacobians $a_x(x_i)$ on the i-th column-block. Additionally letting $\mathbf{a} = \text{stack}\{a_i\} \in \mathbb{R}^{nN}$, we have in matrix form

$$\mathbf{a}(\bar{\mathbf{x}} + \delta\mathbf{x}) = \bar{\mathbf{a}} + A\delta\mathbf{x}.$$

Defining $M = \text{diag}\{\Sigma_i^{-1}\} \in \mathbb{R}^{nN \times nN}$, the last term of (8) becomes $\frac{1}{2}(\bar{\mathbf{a}} + A\delta\mathbf{x})^\mathsf{T} M(\bar{\mathbf{a}} + A\delta\mathbf{x})$, and the second-order expansion around $\bar{\mathbf{x}}$ is seen to be

$$l(\bar{\mathbf{x}} + \delta\mathbf{x}) = l(\bar{\mathbf{x}}) + \delta\mathbf{x}^\mathsf{T}(q_{\mathbf{x}} + A^\mathsf{T} M\bar{\mathbf{a}}) + \tfrac{1}{2}\delta\mathbf{x}^\mathsf{T}(Q + A^\mathsf{T} MA)\delta\mathbf{x}.$$

Comparison with (9) shows that

$$\mathbf{g} = q_{\mathbf{x}} + A^\mathsf{T} M\bar{\mathbf{a}}$$

$$H = Q + A^{\mathsf{T}} M A$$

Quadratic dynamics approximation:

We can achieve a more accurate approximation by considering a quadratic model of the passive dynamics

$$a_{ij}(\bar{x}_i + \delta x_i, \bar{x}_j + \delta x_j) = \bar{a}_{ij} + a_x(\bar{x}_i)\delta x_i + \tfrac{1}{2}\delta x_i^{\mathsf{T}} a_{xx}(\bar{x}_i)\delta x_i - \delta x_j,$$

where the left and right multiplications with the 3-tensor a_{xx} are understood as contractions on the appropriate dimensions. Though the gradient $\mathbf{g}$ is unaffected by the second order term, the Hessian picks up the product of second-order and zeroth-order terms. Let the set of $n \times n$ matrices $U_{ij} = a_{ij}^{\mathsf{T}} \Sigma_i^{-1} a_{xx}(\bar{x}_i)$, contracting with the leading dimension of the tensor a_{xx}. Now define the block-diagonal matrix $U = \text{diag}\{U_{ij}\} \in \mathbb{R}^{nN \times nN}$. The new approximation is now

$$\mathbf{g} = q_{\mathbf{x}} + A^{\mathsf{T}} M \bar{\mathbf{a}}$$
$$H = Q + A^{\mathsf{T}} M A + U$$

In the experiments described below, the addition of the U term was found to significantly improve convergence.

6.4.5 Computing the policy

In this section, we describe how to obtain a local feedback control policy from the posterior marginals, that is the means $\bar{x}_i$ and the covariance $\mathbf{S}$. Let $S_i = \text{cov}(x_i)$ be the i-th diagonal $n \times n$ block of $\mathbf{S}$ and S_{ij} be the cross-covariance of x_i and x_j, the $n \times n$ block of $\mathbf{S}$ at the i-th n-column and j-th n-row, so that the mean of the conditional is

$$E[x_j|x_i] = \bar{x}_j + S_{ij} S_i^{-1}(x_i - \bar{x}_i).$$

The feedback policy is then that control which produces the expected controlled dynamics:

$$u(x_i) = B^{-1}(\bar{x}_j + S_{ij} S_i^{-1}(x_i - \bar{x}_i) - a(\bar{x}_i)) \tag{14}$$

6.4.6 Algorithm summary

Given an initial approximation of $\bar{\mathbf{x}}$:

(a) Repeat until convergence:

> Compute $\mathbf{g}(\bar{\mathbf{x}})$ and $H(\bar{\mathbf{x}})$ with (13).
>
> Recompute $\bar{\mathbf{x}}$ with (11).

(b) Compute $\mathbf{S}$ with (10b), and the feedback control law with (14).

6.5 Experiments

We demonstrate our algorithm on two simulated problems. A toy problem with 2 state dimensions and and a simulated walking robot with 23 state dimensions.

6.5.1 2D problem

The continuous diffusion consists of a non-linear spring damper system, subject to process noise in the velocity variable:

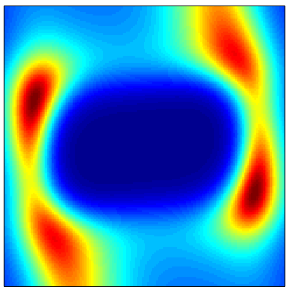
$$\begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -(x_1^3 + x_2^3)/6 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 1 \end{bmatrix} (udt + \sigma d\omega)$$

and cost function is

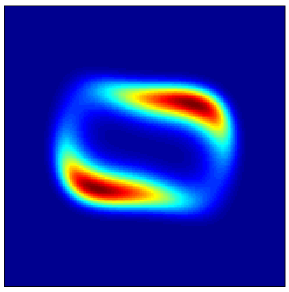
$$\ell(x_2, u) = c_x \left(1 - e^{-(x_2-2)^2} - e^{-(x_2+2)^2}\right) + \frac{u^2}{2\sigma^2}$$

The state cost coefficient is $c_x = 4$ and the noise variance is $\sigma^2 = 1/4$. In Fig. 6.2(g), we show the cost function, overlayed by a vector plot of the drift, and one integrated trajectory of the passive dynamics.

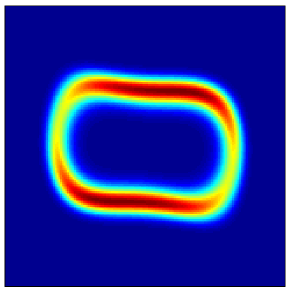
We first solved this problem by discretizing the state-space and solving the resulting MDP. We used a 201×201 grid, leading to a 40401×40401 state transition matrix with 1.2×10^6 nonzeros. Discrete LMDPs can be solved by finding the leading eigenvector of a related matrix (Todorov, 2007). The results are shown in Figures 6.2(a)-6.2(d). Solving the MDP (using MATLAB's "eigs" function) took $86s$ on a standard PC.



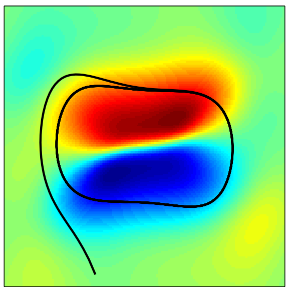
(a) $z(x)$, MDP.



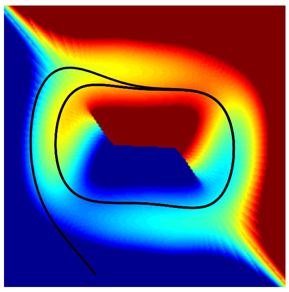
(b) $r(x)$, MDP.



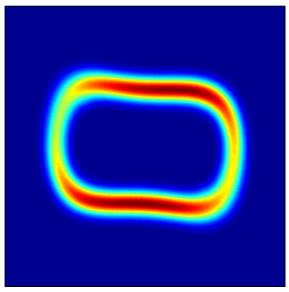
(c) $\mu(x) = z(x)r(x)$, MDP.



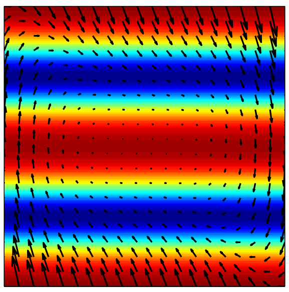
(d) $u(x)$, MDP.



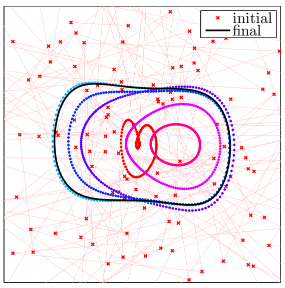
(e) $u(x)$, approximation.



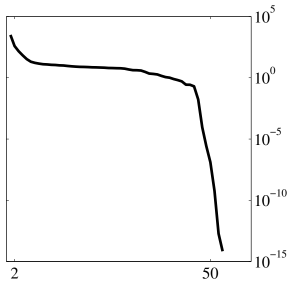
(f) $\mu(x)$, approximation.



(g) Cost and passive dynamics.



(h) Snapshots.



(i) Cost reduction V. iterations.

Figure 6.2: **(a)-(h)** show the area $[-4, 4]^2 \in (x_1 \times x_1)$. **(a)-(d)**, MDP solutions for a discretized state-space. **(e)**, **(f)**, **(h)**, **(i)**, solution obtained by the proposed algorithm. **(a)** The exponentiated negative value function $z(x)$. **(b)** The forward filtering density $r(x)$. **(c)** The optimally controlled steady-state distribution, formed by elementwise product of $z(x)$ and $r(x)$. **(d)** The policy generated by the MDP solution, superimposed with one instantiation of the controlled dynamics. **(e)** The policy generated by the approximate solution, superimposed with one instantiation of the controlled dynamics. The color map is clipped to the values in **(d)**, so saturated areas indicate misextrapolation. **(f)** The approximate distribution generated by our algorithm. For each pixel we measure the marginal of the state whose mean is nearest in the Euclidean sense. Note the similarity to **(c)**. **(g)** The cost function, superimposed with a vector field of the passive dynamics. **(h)** Snapshots of the means for a particular convergence sequence, showing 8 configurations out of a total of 40. The red $\times$'s are the random initialization, followed by a jump to the center to decrease dynamical inconsistency, followed by a gradual convergence to the limit-cycle solution. **(i)** Convergence of the cost. Averaged over 15 runs with random initialization.

We then solved the problem with our proposed algorithm. With 150 variables on the ring, the matrix H was 300×300, with 1800 nonzeros. Full convergence from a random initialization took an average $0.3s$. Of course this is not a fair comparison, since the MDP solver finds a global rather than local solution, yet the difference is striking. Once convergence of equation (11) has been achieved, we compute the posterior with (10). In order to plot the resulting distribution, for every pixel in Fig. 6.2(f), we plot the value of the marginal of the closest (euclidean) gaussian. The similarity of figures 6.2(c) and 6.2(f) is remarkable. Of course, our proposed method is still local, and comparing Figures 6.2(d) and 6.2(e), we see that the generated policy is valid only close to the limit-cycle. In 6.2(h), we see snapshots of the convergence of the means. Starting from a random initialization, the means jump to the center in order to decrease dynamical inconsistency, followed by a gradual convergence to the limit-cycle solution. In 6.2(i), we show the cost averaged over 15 runs, relative to the minimum cost achieved over all the runs. We see that all runs converged to the global minimum, with a quadratic convergence rate towards the end.

6.5.2 Simulated walking robot

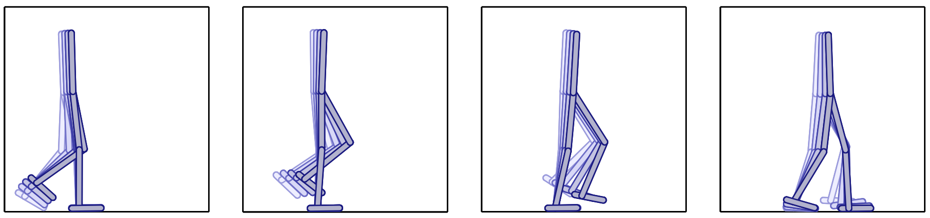


Figure 6.3: Frames from the limit-cycle solution of an optimal walking gait. See section 6.5.2.

Our planar walking model is made of two legs and a trunk, each leg having three segments (thigh, shin and foot). The following parameter values can all be assumed to have the appropriate units of a self-consistent system (e.g. MKS). The length of the foot segments is 1, and all other segments are of length 2. The segment masses are 0.1 for the foot, 0.4 for the shin, 1 for the thigh, and 4 for the trunk. A control signal of dimension 6 acts on the joints (hips, knees, ankles), but not directly. In order to model the excitation-activation dynamics associated with muscular activity, we augment the state space with 6 first-order filters of the control signal, with a time constant of 25/1000. The seven segment angles, together with the planar position of center-of-mass, make for a system with 9 degrees-of-freedom, or 18 state dimensions. In order to allow the gait to take a limit-cycle form, we remove the horizontal position dimension of the center-of-mass, for a total of 23 state dimensions.

The equations of motion are simulated using our own general purpose simulator[1]. We imposed joint-angle constraints that ensure a biomechanically-realistic posture. Ground reaction forces are computed using the method described by Tassa and Todorov (2010). We used 80 time-steps of length 1/80, for a step period of 1. The matrix H was 1840×1840, with 105840 non-zeros. Each iteration took 0.2 seconds, with an average of 300 iterations until convergence (depending on initial conditions).

In order to produce upright walking, we use a cost function with three terms: First, a quadratic penalty for deviation of the center-of-mass's horizontal velocity v_x from a desired value of 2. Second, a linear reward for the vertical hight of the trunk's upper tip h_T, to promote upright posture. Third, a cost term that is quadratic in the muscle activation dimensions c. The total weighted state-cost was

$$q(x) = (v_x - 2)^2 - 0.1h_T + 0.01\|c\|^2.$$

Convergence for this problem was robust, with different initializations converging to the same solution. The resulting gait is demonstrated in Fig. 6.3.

6.5.3 Feedback control law

One of the main advantages of the algorithm presented here is that the generated control law (14) includes feedback terms, forming an effective basin-of-attraction around the optimal limit-cycle. In Fig. 6.4, we illustrate convergence to the limit cycle from perturbed states, of the simulated walking robot. The means $\bar{x}_i$ are projected onto $\{d_1, d_2\} \in \mathcal{R}^{23}$, the two leading eigenvectors of the covariance $\text{cov}(\bar{x}_i)$ (i.e. the two leading PCA directions). One state is then randomly perturbed, and used as an initial state for a controlled trajectory. The distribution of the perturbations was chosen so that most trajectories (solid) are within the basin-of-attraction and converge to the limit-cycle (successful walking), while several (dashed) diverge (falling

[1]Available at `alice.nc.huji.ac.il/~tassa/`

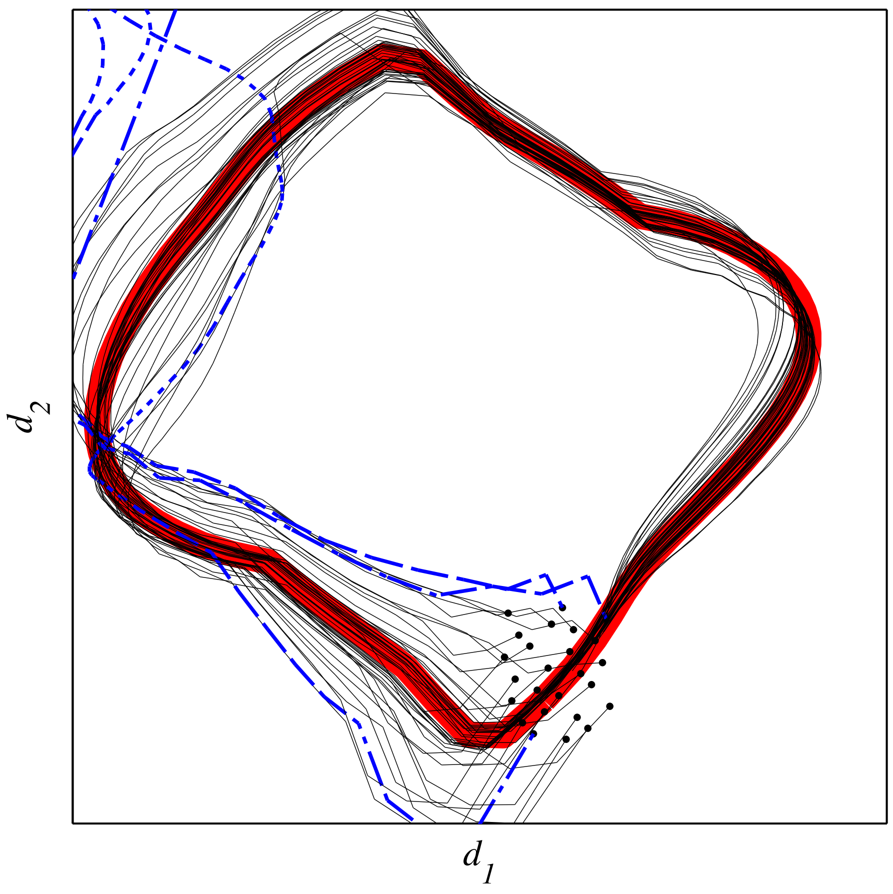


Figure 6.4: Robustness to perturbations of the simulated walking robot, under the feedback control law. The axes d_1 and d_2 are the largest eigenvectors of the covariance of the $\bar{x}_i$. The optimal limit cycle is in thick red, superimposed with simulated trajectories. Converging trajectories are thin solid lines, diverging ones are thicker dashed lines. See section 6.5.3.

down).

6.6 Conclusions

We presented a method of solving Optimal Control problems with a periodic solution. Using the control-estimation duality which holds for problems of the form (5),(6), we recast the problem as Bayesian inference, and maximized the likelihood of a gaussian approximation. The computational complexity of the methods scales either linearly or quadratically with the state dimension, depending on the order of the dynamics approximation.

Several interesting questions remain open.

Here we focused on limit cycles, but the presented algorithm is also valid for simple chains, where message passing algorithms of the type described by Toussaint (2009) are applicable. A performance comparison would be interesting.

The algorithm produces a local feedback control law around the trajectory, but the volume of the basin-of-attraction is limited by the validity of the gaussian approximation. One way of increasing it is by using higher-order approximations, like a mixture-of-gaussians. Another possibility is to combine this offline method with online methods like Model Predictive Control, by using the infinite-horizon cost-to-go as a final-cost for a finite-horizon trajectory optimizer.

Chapter 7

Discussion and Future Work

This thesis describes several contributions, mostly to algorithmic aspects of optimal control. They can be discussed on two levels – engineering developments and biological implications.

In engineering terms, the headline is that we are at the threshold of synthesizing controllers which are as powerful as neural systems. The type of algorithm which seems most promising is some combination of local and global methods. In particular if the algorithm is based on Value approximation, a rough approximation computed by a global method can be used as the final cost of a trajectory optimizer. Local methods are fast enough to be applied online, a procedure known as "Model Predictive Control". The implied combination of local-online/global-offline therefore seems promising. The largest computational burden of all of these methods is the physics simulator. Simulators which exploit the power of modern multi-core CPUs and GPUs will be useful, as will fast methods of approximating the dynamics, perhaps by fitting an approximator the simulation. The explicit modeling of stochasticity appears indispensable for global methods. Noise makes the Value function smooth, greatly reducing the required representational power of approximators.

Biological implications are far more conjectural. The basic machineries of optimization, namely gradient descent and Newton's method, require accurate linear algebra to work efficiently. There is no good evidence for neural circuits that implement these computations. However, the central feature shared by all efficient methods is the explicit use of derivatives, in particular derivatives of the dynamics model and of the cost. Perhaps the most likely neuroanatomical candidate for derivative computation is the cerebellum. This is because it exists even in basic nervous systems, and has a simple repetitive structure.

The estimation-control duality of linearly-solvable optimal control described in Section 6.3.1 provides a rich pasture for interpretation. The recasting of optimal control in Bayesian terms holds the promise of similar algorithms and representations in sensory and motor areas. The exponentiated Value function z (Eq. 1) is a good candidate for representation by neurons due to its non-negativity. The forward filtering density r (Eq. 2) can be used to implicitly represent the optimally-controlled steady-state distribution μ (Eq. 3), which is the natural quantity for determining the relative importance of points in state-space. These observations are very preliminary, and it is clear that we have only just scratched the surface of this important duality.

Bibliography

M. Ackermann and A. J Van den Bogert. Optimality principles for model-based prediction of human gait. *Journal of biomechanics*, 2010.

H. Attias. Planning by probabilistic inference. In *Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics*, 2003.

N. Bernstein. *The co-ordination and regulation of movements*, volume 1. Pergamon Press Oxford:, 1967.

Dimitri Bertsekas. *Dynamic programming and optimal control.* Athena Scientific, Belmont Mass., 2nd ed. edition, 2000. ISBN 9781886529083.

M. G Crandall and P. L Lions. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, 277(1):142, 1983.

A. G. Feldman. Functional tuning of the nervous system with control of movement or maintenance of a steady posture. II. controllable parameters of the muscle. *Biophysics*, 11(3):565578, 1966.

T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7): 1688, 1985.

A. P Georgopoulos, J. F Kalaska, R. Caminiti, and J. T Massey. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, 2(11): 1527, 1982.

D. H Hubel and T. N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962. ISSN 0022-3751.

D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming.* Elsevier, 1970.

M. I Jordan. *Learning in graphical models.* Kluwer Academic Publishers, 1998.

B. Kappen, V. Gomez, and M. Opper. Optimal control as a graphical model inference problem. *arXiv*, 901, 2009.

H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005: P11011, 2005.

A. D. Kuo. An optimal control model for analyzing human postural balance. *IEEE Transactions on Biomedical Engineering*, 42(1):87101, 1995. ISSN 0018-9294.

M. G Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:11071149, 2003.

G. Lantoine and R. P Russell. A hybrid differential dynamic programming algorithm for robust low-thrust optimization. In *AAS/AIAA Astrodynamics Specialist Conference and Exhibit*, 2008.

L. Z Liao and C. A Shoemaker. Convergence in unconstrained discrete-time differential dynamic programming. *IEEE Transactions on Automatic Control*, 36(6):692, 1991.

L. Z. Liao and C. A Shoemaker. Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems. *Cornell University, Ithaca, NY*, 1992.

G. E. Loeb, W. S. Levine, and J. He. Understanding sensorimotor feedback through optimal control. In *Cold Spring Harbor symposia on quantitative biology*, volume 55, page 791, 1990.

D. Marr. *Vision: A computational investigation into the human representation and processing of visual information.* Henry Holt and Co., Inc. New York, NY, USA, 1982. ISBN 0716715678.

D. Q. Mayne. A second-order gradient method of optimizing non-linear discrete time systems. *Int J Control*, 3:8595, 1966.

P. A. Merton. Speculations on the servo-control of movement. In *The Spinal Cord, Ciba Foundation Symposium*, page 247260, 1953.

D. E Meyer, R. A Abrams, S. Kornblum, C. E Wright, and J. E.K Smith. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review*, 95(3):340370, 1988.

S. K Mitter and N. J Newton. A variational approach to nonlinear estimation. *SIAM Journal on Control and Optimization*, 42(5):18131833, 2004.

R. Munos and A. Moore. Variable resolution discretization in optimal control. *Machine learning*, 49(2):291323, 2002.

K. Murphy, Y. Weiss, and M. I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, page 467475, 1999.

B. A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607609, 1996. ISSN 0028-0836.

L. S Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The mathematical theory of optimal processes.* Interscience New York, 1962.

I. Ross and Fariba Fahroo. Legendre pseudospectral approximations of optimal control problems. In *New Trends in Nonlinear Dynamics and Control and their Applications*, pages 327–342. 2004.

R. A Schmidt. A schema theory of discrete motor skill learning. *Psychological review*, 82(4):225260, 1975. ISSN 0033-295X.

W. Schultz, P. Dayan, and P. R Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593, 1997.

Charles Sherrington. *The integrative action of the nervous system.* CUP Archive, 1923.

Robert F. Stengel. *Optimal Control and Estimation.* Dover Publications, September 1994. ISBN 0486682005.

Richard Sutton. *Reinforcement learning : an introduction.* MIT Press, Cambridge Mass., 1998. ISBN 9780262193986.

Y. Tassa and E. Todorov. Stochastic complementarity for local control of discontinuous dynamics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2010.

Yuval Tassa and Tom Erez. Least squares solutions of the HJB equation with neural network Value-Function approximators. *IEEE Transactions on Neural Networks*, 18(4):1031–1041, 2007. ISSN 1045-9227. doi: 10.1109/TNN.2007.899249.

Yuval Tassa, Tom Erez, and William Smart. Receding horizon differential dynamic programming. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, page 1465. MIT Press, Cambridge, MA, 2008.

E. Todorov. Linearly-solvable markov decision problems. *Advances in neural information processing systems*, 19:1369, 2007.

E. Todorov. General duality between optimal control and estimation. In *proceedings of the 47th ieee conf. on decision and control*, 2008.

E. Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478, 2009.

E. Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306, Portland, OR, USA, 2005. doi: 10.1109/ACC.2005.1469949.

Emanuel Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, 2004. ISSN 1097-6256. doi: 10.1038/nn1309.

M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

J. N Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1):5994, 1996. ISSN 0885-6125.

J. N Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674690, 2002. ISSN 0018-9286.

N. J Wade and S. Finger. The eye as an optical instrument: from camera obscura to helmholtz's perspective. *Perception*, 30(10):11571178, 2001. ISSN 0301-0066.

K. Wampler and Z. Popovic. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, 28(3):18, 2009.

Y. Weiss and W. T Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural computation*, 13(10): 21732200, 2001.

Summary in Hebrew

תיאוריה ויישומים של בקרת תנועה ביולוגית

חיבור לשם קבלת תואר דוקטור לפילוסופיה

מאת

יובל טסה

הוגש לסנט האוניברסיטה העברית בירושלים בשנת תשע׳א

עבודה זו נעשתה בהנחייתם של
פרופ׳ נפתלי תשבי ופרופ׳ עמנואל טודורוב

תקציר

בקרה משמעה – פעולה על מערכת דינמית באופן שמממש מטרה. מוחות הם הבקרים הביולוגים שבפעולתם על השרירים ודרכם הגוף והסביבה, ממשים מטרות מורכבות. זאת למרות אי-לינאריות, רעש ועיכובים במערכת, והפרעות חיצוניות.

חיבור זה עוסק בעיצובם של בקרים מלאכותיים שיוכלו להתחרות בזריזות, בעמידות וביכולת הלמידה של מערכות עצבים. תקוותנו היא שהבנה יסודית ומופשטת של בעיית הבקרה תוביל לתובנות אודות בקרים ביולוגיים. דוגמא והשראה לכך הן ההתפתחויות בעשורים האחרונים בהסקה בייסאנית ותורת האיפורמציה, והתובנות העמוקות שהתקבלו מהן אודות עיבוד חושי במוח. על אף הבדלים בין מנגנוני החישוב הביולוגיים והספרתיים, אותם עקרונות כלליים נמצאו תקפים.

המסגרת התיאורטית שבה נעשה שימוש היא תורת הבקרה המיטבית. תורה זו מתארת את בחירתן של פעולות באופן שיקטין מחיר עתידי, בהנתן מחיר כלשהו. מעבר להיותה כללית ובת-יישום לבעיות בהן נתמקד, תכונות מסויימות של התנהגות ביולוגית מוסברות היטב על ידי ההנחה שבבסיסה העצבי מתקיים תהליך אופטימיזציה.

במהלך ארבעת הניסויים שמתוארים להלן נפתרו מספר בעיות בקרה, ושתי תובנות עיקריות התגבשו אודות עקרונות העיצוב והאילוצים של בקר אופטימלי. הראשונה היא היעילות הגדולה של שיטות מקומיות שעושות שימוש במשתני מצב רציפים, והשניה היא החשיבות במידולו של רעש.

בניסוי הראשון השתמשנו ברשת ניורונים רב-שכבתית קלאסית, כדי לקרב את פונקצית הערך על ידי הקטנת השארית הריבועית של משוואת המילטון-יעקבי. שיטה זו איננה מקומית ומאבדת מיעילותה במרחבי מצבים בעלי מימד גדול מארבע. כאן גילינו לראשונה את היתרון הגדול שטמון במידול מפורש של הרעש הדינמי, ואת השיפור שהוא מקנה בהתכנסות.

בניסוי השני אנו עושים שימוש באלגוריתם המקומי "תכנון דינמי דיפרנציאלי" כדי לבנות ספרייה של בקרים מקומיים שממשים שחיה. גישה זו היא רק ריבועית במימד המצב, ומצליחה להתמודד עם עד עשרים וארבעה מימדים.

בניסוי השלישי אנו עוסקים בבעיית המגע. היעילות של השיטות המקומיות שפיתחנו נשענת על גזירותה של הדינמיקה, אך מגע וחיכוך גורמים לאי-גזירות ואי-רציפות שלה, בקירוב ראשון. אנו משתמשים בתובנה מהניסוי הראשון כדי להטמיע את אי הוודאות שלנו אודות המצב לתוך המודל של הדינמיקה. באופן אפקטיבי אנו מקנים לגופים בחלל גבולות עמומים, וכך מחליקים את הדינמיקה שתהיה גזירה. אנו משתמשים במודל זה כדי לפתור בעיה שבה על אצבע לסובב גוף על ציר תוך שימוש במגע. אנו עושים שימוש ברעיון ספריית הבקרים שפותח בניסוי השני.

בניסוי הרביעי אנו מתארים אלגוריתם מקומי חדש לבעיות בקרה עם פתרון מחזורי. שלא כמו במקרים הקודמים, הפתרון המחזורי איננו מרקובי, וכן לא ניתן להשתמש בתכנון דינמי. אנו עושים שימוש בדואליות בין אמידה בייסיאנית ובקרה מיטבית כדי להמיר את בעיית הבקרה לבעיה של היסק על מודל גרפי טבעתי, ואז פותרים את בעיית ההיסק. בעזרת שיטה אנו בונים בקר שממש הליכה ברובוט בעל עשרים ושלושה מימדי מצב. מודל החיכוך בין כף הרגל לריצפה הוא זה שפותח בניסוי השלישי.