High-Order Local Dynamic Programming

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

Emanuel Todorov
Applied Mathematics and Computer Science & Engineering
University of Washington, Seattle, USA
todorov@cs.washington.edu

***Abstract*—We describe a new local dynamic programming algorithm for solving stochastic continuous Optimal Control problems. We use cubature integration to both propagate the state distribution and perform the Bellman backup. The algorithm can approximate the local policy and cost-to-go with arbitrary function bases. We compare the classic quadratic cost-to-go/linear-feedback controller to a cubic cost-to-go/quadratic policy controller on a 10-dimensional simulated swimming robot, and find that the higher order approximation yields a more general policy with a larger basin of attraction.**

I. INTRODUCTION

Optimal Control describes the choice of actions which minimizes future costs and promises a simple, principled approach to controller synthesis: quantify task performance, and a numerical algorithm will automatically generate the optimal policy. In practice, the exponential scaling of computational complexity with the state-dimension makes this impossible for all but the simplest systems.

In the continuous nonlinear case, *local methods* are the only class of algorithms which successfully solve general, high-dimensional Optimal Control problems. They are based on the observation that optimal solutions form extremal trajectories, i.e. are solutions to a calculus of variations problem. Problems which are sufficiently smooth and deterministic, e.g. space flight, have been successfully solved using such methods, which characterize the solution on a single optimal trajectory.

When the dynamics are stochastic, an open-loop controller does not suffice, and feedback terms must be incorporated. Second-order local dynamic programming algorithms like DDP [1] and iterative-LQG [2] compute a quadratic approximation of the cost-to-go around a trajectory and correspondingly, a local linear-feedback controller. While controllers constructed with these algorithms are indeed more robust, they do not take the noise explicitly into account. Instead, they rely on the *certainty-equivalence* principle, which asserts that for linear-quadratic problems noise has no effect on the policy. This neglect of noise can hold only under Linear-Quadratic-Gaussian assumptions, which in some cases cease to be reasonable (see Discussion).

As a result of certainty-equivalence, locally-quadratic methods can measure the dynamics and cost and represent the Value only in an *infinitesimal neighborhood*, effectively a truncated Taylor expansion. In order to compute a higher-than-quadratic local approximation of the Value function, as we do here, we must explicitly measure and represent these quantities in a finite volume around the trajectory. This means that the propagation of the dynamics must involve a process akin to "Unscented" or "particle" filters, rather than the standard Extended Kalman Filter. Similarly, in the backward pass which approximates the Value, costs must be measured within this finite volume, rather than at a single point. We address this by using *cubature* integration formulas (see section IV).

We present a local dynamic-programming algorithm which uses cubature-based integration of a single trajectory and can accept general function bases for the approximation of both the cost-to-go and the policy. The direct representation of the distribution as a set of weighted cubature vectors allows us to increase the region of validity of our approximation, and to construct controllers with larger basins of attraction.

We test the algorithm on a simulated swimming robot with 10 state and 2 control dimensions. We compare a cubic-Value, quadratic-policy controller and the standard quadratic-Value, linear-policy controller, and find that it can generalize more effectively from a swimming behaviour to a twisting-in-place behaviour, in a target reaching task.

The work here extends [2], where we presented the idea of using high-order approximators, without the specific machinery of cubature integration as an accurate method of propagating the state-distribution and the Value back-up.

II. OVERVIEW OF LOCAL METHODS

We restrict ourselves to the discrete-time finite-horizon problem, though extensions to continuous time and first-exit formulations are possible. We begin with deterministic dynamics and turn to the stochastic case in Sec. III. The control $\mathbf{u} \in \mathbb{R}^m$ affects the propagation of the state $\mathbf{x} \in \mathbb{R}^n$ via the dynamics

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{1}$$

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 $\ell(\mathbf{x}, \mathbf{u})$[1] and final cost $\ell_f(\mathbf{x})$:

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

where the $\mathbf{x}_k$ for $k > i$ are propagated with (1). Defining the optimal Value function at time i as the cost-to-go given the minimizing control sequence

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

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

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

$$V_i^*(\mathbf{x}) = \min_{\mathbf{u}}[\ell(\mathbf{x},\mathbf{u}) + V_{i+1}^*(\mathbf{f}(\mathbf{x},\mathbf{u}))] \tag{2}$$

A. First-Order Dynamic Programming

To derive a discrete-time equivalent of the Maximum Principle, we observe that 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:

$$\mathbf{u}_i = -\nabla_{\mathbf{u}}\ell^{-1}\left(\nabla_{\mathbf{u}}\mathbf{f}^{\mathsf{T}}\nabla_{\mathbf{x}}V_{i+1}\right) \tag{3}$$

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

$$\nabla_{\mathbf{x}}V_i(\mathbf{x}) = \nabla_{\mathbf{x}}\big(\ell(\mathbf{x},\mathbf{u}_i) + V_{i+1}(\mathbf{f}(\mathbf{x},\mathbf{u}_i))\big). \tag{4}$$

The first-order local dynamic programming algorithm proceeds by alternatingly propagating the dynamics forward with (1), and propagating $\mathbf{u}_i$ and $\nabla_{\mathbf{x}}V_i(\mathbf{x})$ backward with (3) and (4). In other derivations the quantity $\nabla_{\mathbf{x}}V_i(\mathbf{x})$ is called the *co-state* vector.

B. 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[2] and a more accurate, closed-loop controller, albeit at a higher computational cost. 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

$$\approx \frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}^{\mathsf{T}}\begin{bmatrix}2Q_0 & Q_x{}^{\mathsf{T}} & Q_u{}^{\mathsf{T}}\\ Q_x & Q_{xx} & Q_{xu}\\ Q_u & Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}. \tag{5}$$

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

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

which gives us both open-loop and linear-feedback control terms. This control law can then be plugged back into (5) 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 reestimates the Value and produces a new policy.

[2]i.e. convergence like Newton's method rather than like gradient descent.

III. STOCHASTIC DYNAMICS

Small noise in discrete time is usually described as a short-time integral of the diffusion

$$d\mathbf{x} = \mathbf{f}_c(\mathbf{x},\mathbf{u})dt + F(\mathbf{x},\mathbf{u})d\omega$$

where $\mathbf{f}_c$ is the underlying continuous dynamics. For a short integration interval h the discrete dynamics are a conditional distribution that is guaranteed to be near-gaussian:

$$\begin{aligned}p(\mathbf{x}'|\mathbf{x},\mathbf{u}) &\propto \mathcal{N}(\mathbf{x}'\,;\mathbf{f}(\mathbf{x},\mathbf{u}),\Sigma(\mathbf{x},\mathbf{u}))\\ \mathbf{f}(\mathbf{x},\mathbf{u}) &= \mathbf{x} + h\mathbf{f}_c(\mathbf{x},\mathbf{u})\\ \Sigma(\mathbf{x},\mathbf{u}) &= hF(\mathbf{x},\mathbf{u})F(\mathbf{x},\mathbf{u})^{\mathsf{T}}\end{aligned}$$

Second-order methods provide a stabilizing linear state-feedback component, but they are mostly[3] invariant to the noise term ω. This is because quadratics are indifferent to convolution with a gaussian, apart from the constant term. If the noise is indeed small this might be considered a feature, but if the noise is large and has considerable effect on the dynamics, we would want our method to explicitly take it into account.

The propagation of a state distribution in the general case is given by

$$p(\mathbf{x}') = \iint p(\mathbf{x}'|\mathbf{x},\mathbf{u})p(\mathbf{u}|\mathbf{x})p(\mathbf{x})d\mathbf{u}d\mathbf{x} \tag{6}$$

The policy is now modeled as a stochastic function of $\mathbf{x}$, which is then perturbed by a white noise term which can be additive, multiplicative or generally state-dependent: $\mathbf{u} = \boldsymbol{u}(\mathbf{x}) + \xi(\mathbf{x})$. We then redefine the cost-to-go as an expectation

$$\begin{aligned}&J_i(\mathbf{x},\mathbf{u}_{i..N-1})\\ &\quad= \sum_{k=i}^{N-1}\iint \ell(\mathbf{x}_k,\mathbf{u}_k)p(\mathbf{u}_k|\mathbf{x}_k)p(\mathbf{x}_k)d\mathbf{x}_k d\mathbf{u}_k\\ &\quad+ \int \ell_f(\mathbf{x}_N)p(\mathbf{x}_N)d\mathbf{x}_N\end{aligned} \tag{7}$$

The Bellman equation for the Value is then

$$V_i^*(\mathbf{x}) = \min_{\boldsymbol{u}(\cdot)}\underset{\substack{\mathbf{u}\sim\\ p(\mathbf{u}|\mathbf{x})}}{\mathbf{E}}\Bigg[\ell(\mathbf{x},\mathbf{u}) + \underset{\substack{\mathbf{x}'\sim\\ p(\mathbf{x}'|\mathbf{x},\mathbf{u})}}{\mathbf{E}} V_{i+1}^*(\mathbf{x}')\Bigg] \tag{8}$$

which involves two nested expectations: an expectation of the immediate cost-to-go over the *current* state distribution, which depends on the expected cost-to-go of the *next* state.

IV. CUBATURE FORMULAE

Cubature is a term describing the approximation of an integral with a sum. Given a particular density function $g(\mathbf{x})$ defined over some volume, the expectation of an arbitrary smooth function $f(\mathbf{x})$ WRT $g(\mathbf{x})$, is approximated as a weighted sum of measurements:

$$\int f(\mathbf{x})g(\mathbf{x})d\mathbf{x} \approx \sum_j^K \alpha_j f(\mathbf{a}_j).$$

[3]Control-dependent covariance $\Sigma(\mathbf{u})$ can be taken into account, as in [3].

The set of vector and weight pairs $\{\mathbf{a}_j, \alpha_j\}_{j=1\dots K}$ is known as a *cubature formula*. There are extensive tables of such formulas in the literature for various domains and weight functions, e.g. [4], [5]. Different formulas differ in the quality of the approximation. This quality is usually measured as the minimum degree of a polynomial $f(\mathbf{x})$ for which the formula is exact. In general, the number of measurements K required to achieve accuracy of degree d in a space of dimension n grows roughly as

$$K(d,n) \approx \frac{2\,n^{d/2}}{(d/2)!}.$$

In our case, because we are dealing with short-time-integrals of diffusions, it makes sense to choose the gaussian $g(\mathbf{x}) \equiv \mathcal{N}(\mathbf{x};\mu,\Sigma)$ as the local approximation of the state distribution. The idea of using cubature for the forward propagation of diffusions is not new, and has recently been used to construct the Cubature Kalman Filter [6]. Given a formula $\{\tilde{\mathbf{a}}_j, \alpha_j\}$ for the unit gaussian centered at the origin, it is easy to transform it into the coordinate system of a general gaussian

$$\int f(\mathbf{x})\mathcal{N}(\mathbf{x}\,;\mu,\Sigma)d\mathbf{x} \approx \sum_j^K \alpha_j f(\mathbf{a}_j) \tag{9}$$

by setting

$$\mathbf{a}_j = V \cdot D^{\frac{1}{2}} \cdot \tilde{\mathbf{a}}_j + \mu \tag{10}$$

where $VDV^{\mathsf{T}} = \Sigma$ is the eigendecomposition of Σ.

Formulas $\{\tilde{\mathbf{a}}_j, \alpha_j\}$ for the unit gaussian can either be found in the literature or constructed ad-hoc by guessing the $\tilde{\mathbf{a}}_j$ and then solving for the α_j given some known values of expectations of some functions (e.g. moments). We discuss our specific choice of cubature formulae in section VI.

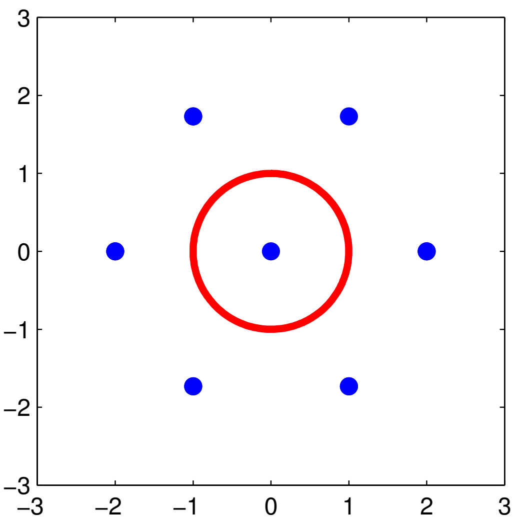

Fig. 1. Cubature integration for a two-dimensional unit gaussian. The gaussian is represented by the standard deviation circle at radius 1. The cubature vectors $\tilde{\mathbf{a}}_j$ are the seven dots, one at the origin and six in a hexagon at radius 2. The weights in this case are $\alpha_1 = \frac{1}{2}$ for the central vector and $\alpha_{j>1} = \frac{1}{12}$ for the others. This is the 2-D instantiation of the first formula described in [5], which is the also one used in the experiments here.

V. THE ALGORITHM

In a nutshell, the algorithm below proceeds as follows. Given some parametrization of the policy, we propagate the discrete dynamics forward in time using two nested cubature formulas, corresponding to the two nested integrals of equation (6). At every time-step we re-gaussianize the state-distribution using cubature integration to measure the new mean and covariance. In the backward pass we fit a general function approximator to the Value, and then use the cubature vectors in $\mathbf{u}$-space, corresponding to the $d\mathbf{u}$ integral in (6), to efficiently compute the gradient WRT the parameters of the policy. While in principle this gradient can be used to find the minimum as required by equation (8), we find that it is sufficient to perform a single Newton step.

A. *General linear approximators*

We let our Value approximation at time-step i take the general form

$$V_i(\mathbf{x},\theta_i) = T(\mathbf{x})^{\mathsf{T}}\theta_i \tag{11}$$

where $T(\cdot) \in \mathbb{R}^{n_\theta}$ is a vector of n_θ basis functions and θ is a weight vector. Similarly we let the policy at time-step i take the form

$$\boldsymbol{u}_i(\mathbf{x},\phi_i) = \sigma(\mathbf{P}(\mathbf{x})^{\mathsf{T}}\phi_i) \tag{12}$$

where $\mathbf{P}(\cdot) \in \mathbb{R}^{n_\phi \times m}$ is an $n_\phi \times m$ matrix of basis functions and ϕ is a weight vector. The optional squashing nonlinearity $\sigma(\cdot)$ prevents divergence when extrapolating and may increase the region where the policy is sensible.

B. *Forward Pass*

We assume that at a given time the state is normally distributed $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}\,;\mu,\Sigma)$. This a reasonable assumption for dynamics which are continuous in $\mathbf{x}$, but becomes unreasonable when phenomena such as rigid contact and friction are modeled. We also assume that the distribution of the control signal is gaussian $p(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mathbf{u}\,;\boldsymbol{u}(\mathbf{x}),\Sigma_u(\mathbf{x}))$. This can always be safely assumed since this distribution is not integrated in time. A cubature set $\{\mathbf{a}_j,\alpha_j\}$ with $\mathbf{a}_j \in \mathbb{R}^n$ is generated for the gaussian $\mathcal{N}(\mathbf{x}\,;\mu,\Sigma)$, and then *for each* $\mathbf{a}_j$ a cubature set $\{\mathbf{b}_k,\beta_k\}$ with $\mathbf{b}_k \in \mathbb{R}^m$ is generated for the gaussian $\mathcal{N}(\mathbf{u}\,;\boldsymbol{u}(\mathbf{a}_j),\Sigma_u(\mathbf{a}_j))$. The number of cubature points in these formula are n_j and n_k respectively, we discuss appropriate choices of formula in Section V-C.

We can now propagate an approximation of equation (6) with these *nested* cubature formulas, by computing

$$\mathbf{c}_{jk} = \mathbf{f}(\mathbf{a}_j, \boldsymbol{u}(\mathbf{a}_j,\phi_i) + \mathbf{b}_k(\mathbf{a}_j)), \tag{13}$$

and then finding the new estimate for $p(\mathbf{x}') = \mathcal{N}(\mathbf{x}'\,;\mu',\Sigma')$ by plugging the definition of the mean and covariance into equation (9):

$$\begin{aligned} \mu' &= \sum_{jk} \alpha_j\beta_k\mathbf{c}_{jk} \\ \Sigma' &= \sum_{jk} \alpha_j\beta_k(\mathbf{c}_{jk}-\mu')(\mathbf{c}_{jk}-\mu')^{\mathsf{T}} \end{aligned} \tag{14}$$

The cubature-based approximation of the total cost (7) for the entire trajectory is now

$$J_1(\mathbf{x}_1, \phi_{i..N-1}) = \sum_{i=1}^{N} \left[\sum_{jk} \alpha_j \beta_k \ell(\mathbf{a}_j, \boldsymbol{u}(\mathbf{a}_j, \phi) + \mathbf{b}_k(\mathbf{a}_j)) \right] + \sum_j \alpha_j \ell_f(\mathbf{a}_j) \tag{15}$$

where the dependence of $\mathbf{a}, \alpha, \mathbf{b}, \beta$ and ϕ on the time-step i has been dropped for readability.

C. Backward Pass

Using the cubature sets, the argument of the minimization in (8) at time i can now be approximated as

$$\begin{aligned} Q_i(\mathbf{x}, \phi_i) =& \sum_j \alpha_j \Big[\ell(\mathbf{a}_j, \boldsymbol{u}_i(\mathbf{a}_j, \phi_i)) + \sum_k \beta_k V_{i+1}(\mathbf{c}_{jk}, \theta_{i+1}) \Big] \\ =& \sum_j \alpha_j \Big[\ell(\mathbf{a}_j, \sigma(\mathbf{P}(\mathbf{a}_j)^\mathsf{T} \phi_i)) \\ &+ \sum_k \beta_k T(\mathbf{c}_{jk}(\phi_i))^\mathsf{T} \theta_{i+1} \Big], \end{aligned} \tag{16}$$

and the Bellman backup (8) is

$$V_i(\mathbf{x}) = \min_{\phi_i} \Big[Q_i(\mathbf{x}, \phi_i) \Big]$$

To show that the minimization can indeed be carried out in an efficient manner, we show that it is easy to compute $\partial Q_i / \partial \phi_i$, and thence to minimize with a general-purpose optimizer.

To differentiate the first term in the RHS of (16) we observe that $\partial \boldsymbol{u} / \partial \phi$ is known from our choice of σ and $\mathbf{P}$ in equation (12) and that knowing the control cost we can compute $\partial \ell / \partial \mathbf{u}$. It makes sense to choose $\sigma(\cdot) = \partial \ell^{-1} / \partial \mathbf{u}$ as in equation (3), i.e. to match the control cost with the squashing function. A quadratic cost corresponds to the standard linear gain, while the integral of the inverse of some sigmoid would be matched to a control attenuation by that sigmoid.

To differentiate the second term we again begin by observing that $\partial T / \partial \mathbf{c}_{jk}$ is known by design from (11). Differentiating equation (13) to obtain $\partial \mathbf{c}_{jk} / \partial \phi_i$ is similarly cheap since $\partial \mathbf{f} / \partial \mathbf{u}$ is known from knowing the dynamics and $\partial \boldsymbol{u}_i / \partial \phi_i$ is again known by design. If the dynamics are given as a "black-box" simulator rather than analytic function, we can use the inner cubature vectors $\mathbf{b}_k(\mathbf{a}_j)$ to expand the dynamics to fist-order in $\mathbf{u}$

$$\mathbf{c}_{jk} = \mathbf{f}(\mathbf{a}_j, \boldsymbol{u}(\mathbf{a}_j, \phi)) + \tfrac{\partial \mathbf{f}}{\partial \mathbf{u}} \mathbf{b}_k(\mathbf{a}_j)$$

and solve for $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$ in the least-squares sense. This is quite accurate since the Euler-discretized dynamics $\mathbf{f}$ inherit the control-affine nature of $\mathbf{f}_c$, and the dependence of the "control transition matrix" $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$, on $\mathbf{x}$ is usually quite weak. For the least-squares computation to be overdetermined we must have more equations than unknowns or

$$n_j n_k \geq n_\phi n. \tag{17}$$

Once we have $\partial Q_i / \partial \phi_i$, we can find the minimizing ϕ_i with standard algorithms such as conjugate-gradient or pseudo-Newton methods. While these methods are efficient, there is no reason to run them until full convergence since each backward pass is itself an interim step in a minimization process. Because of the difficulty of finding a stopping criterion that will give a uniform partial-convergence rate across the trajectory, we found that it is both simpler and more effective to perform a single Newton minimization step at each time-step. We compute the Hessian $\partial^2 Q_i / \partial \phi_i^2$ by finite-differencing the gradient (although analytical differentiation is not impossible) and then solve

$$\phi_i^* = \phi_i - (\partial^2 Q_i / \partial \phi_i^2)^{-1} \partial Q_i / \partial \phi_i.$$

Once we've found ϕ_i^*, we can plug it back to approximate the cost-to-go at the $\mathbf{a}_j$, and then solve

$$T(\mathbf{a}_j)^\mathsf{T} \theta_i = \ell(\mathbf{a}_j, \sigma(\mathbf{P}(\mathbf{a}_j)^\mathsf{T} \phi_i^*)) + \sum_k \beta_k T(\mathbf{c}_{jk}(\phi_i^*))^\mathsf{T} \theta_{i+1} \tag{18}$$

for θ_i in the least-squares sense. For these equations to be overdetermined we must have that

$$n_j \geq n_\theta. \tag{19}$$

In the first step of the backup, we fit θ_N with

$$T(\mathbf{a}_j)^\mathsf{T} \theta_N = \ell_f(\mathbf{a}_j, \sigma(\mathbf{P}(\mathbf{a}_j)^\mathsf{T} \phi_{N-1}^*)) \tag{20}$$

D. Regularization

As with second-order methods, we must provide some regularization of the backward pass. This is done by introducing a Leveberg-Marquardt parameter for the Hessian inversion

$$\phi_i^* = \phi_i - (\partial^2 Q_i / \partial \phi_i^2 + \lambda \mathbf{I})^{-1} \partial Q_i / \partial \phi_i \tag{21}$$

λ is increased whenever $\partial^2 Q_i / \partial \phi_i^2 + \lambda \mathbf{I}$ is no longer positive-definite for some i (and the back-pass restarted), and decreased otherwise.

E. Summary of the Algorithm

Repeat until convergence:

1. Forward pass: Begin with an initial point μ_1 and covariance Σ_1. For $i = 1 \ldots N-1$, generate the nested cubature vectors $\{\mathbf{a}_j, \alpha_j\}_i$ and $\{\mathbf{b}_k, \beta_k\}_i$, propagate them through the dynamics using the current policy with (13), and re-gaussianize the distribution using (14). At the end of the pass compute the total cost with (15), and terminate if the decrease from the previous iteration is small enough.

2. Backward pass: Begin by fitting the final Value approximation with (20). For $i = N-1 \ldots 1$, compute

$$\begin{aligned} \frac{\partial Q_i}{\partial \phi_i} =& \sum_j \alpha_j \Bigg[\frac{\partial \ell}{\partial \mathbf{u}_i} \frac{\partial \boldsymbol{u}_i}{\partial \phi_i} \\ &+ \sum_k \beta_k \left(\frac{\partial T}{\partial \mathbf{c}_{jk}} \frac{\partial \mathbf{c}_{jk}}{\partial \boldsymbol{u}_i} \frac{\partial \boldsymbol{u}_i}{\partial \phi_i} \right)^\mathsf{T} \theta_{i+1} \Bigg], \end{aligned}$$

and then the Hessian $\partial^2 Q_i/\partial\phi_i^2$, either analytically or by finite-differencing. If a Cholesky factorization fails, increase λ and restart the backward-pass. Otherwise solve (21) and then find the new θ_i from (18). At the end of a successful backward-pass decrease λ.

F. Complexity

We assume that the most expensive part of the algorithm is the computation of the dynamics. For second order methods, a single step in the backward pass is dominated by (5) and therefore requires $O((n+m)^2)$ evaluations of $\mathbf{f}$. In the algorithm above, the cost of a backward step is $O(n_j n_k n_\phi)$, where n_j, n_k and n_ϕ are the number of $\mathbf{a}_j$ and $\mathbf{b}_k$ vectors, and ϕ policy bases, respectively. The last coefficient comes from the finite-differencing of the Hessian.

Though we are free to make different choices of cubature sets and policy bases, let us make the most conservative choices for the same linear-quadratic accuracy. A locally linear policy $\boldsymbol{u}(\mathbf{x}) = \mathbf{P}\mathbf{x}$ with $\mathbf{P} \in \mathbb{R}^{m\times n}$ implies $n_\phi \propto nm$. A quadratic Value approximation and equality at (19) implies

$$n_\theta = (n^2+n)/2 \quad \Longrightarrow \quad n_j \propto n^2$$

and equality at (17) implies

$$n^2 n_k = nm\cdot n \quad \Longrightarrow \quad n_k \propto m,$$

for a total complexity of $O(n_j n_k n_\phi) = O(n^3 m^2)$. This higher computational complexity is indeed the central drawback of our approach, see Discussion.

Note that a fast analytical computation of the Hessian would bring complexity down to a reasonable $O(n^2 m)$, but in our experience analytical differentiation is usually nearly as expensive as finite-differencing, though of course more accurate (to machine precision).

VI. Experiment

We compare our new algorithm to second-order methods with a control problem involving a simulated swimming robot. The *k-swimmer* dynamical system, introduced in [7], describes a chain of k rigid links in a planar viscous medium. By applying $k-1$ torques at the joints, the controller can propel the swimmer in the plane. For our experiment we used a 3-link swimmer, described by 3 joints and two cartesian center-of-mass coordinates, and the respective time-derivatives, for a total of 10 state dimensions.

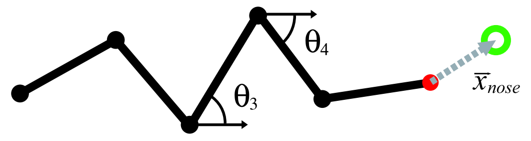

Fig. 2. A 5-link swimmer. The state-cost is a function of the 2D $\bar{x}_{\text{nose}}$ vector from the "nose" point to a ring-shaped target.

The cost function used is

$$\ell(\mathbf{x},\mathbf{u}) = c_x \frac{\|\bar{x}_{\text{nose}}\|^2}{\sqrt{\|\bar{x}_{\text{nose}}\|^2+1}} + c_u\|u\|^2$$

where $\bar{x}_{\text{nose}}$ is the 2-dimensional vector from a certain "nose" point on the mechanism to a movable target (dashed gray in Figure 2). The form of the state-dependent part of this function is illustrated in Figure 3. This type of cost function is good in exemplifying the power of optimal-control methods. Even though the state-cost involves only the 2 Cartesian coordinates out of the 10 dimensions of $\mathbf{x}$, during the backward pass it "leaks" into the cost-to-go in the angular dimensions, to reflect the possibility of actuating the angles in a way that minimizes the distance to the target – i.e. swimming.

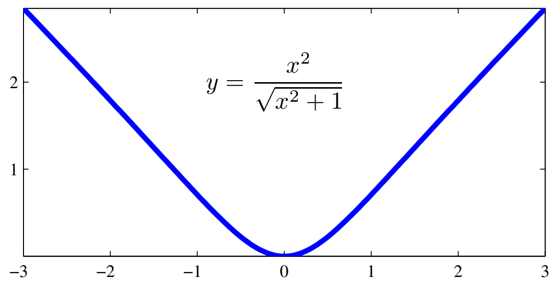

Fig. 3. The functional form of the state-cost component.

The linear regime of the cost function translates into steady-state swimming behavior, while the smooth minimum translates into a braking and stopping motion. Around this minimum, of both the cost and cost-to-go, it is possible to reach the target by twisting-in-place, i.e. without accelerating the center-of-mass. In the following experiment we compare the ability of two controllers of different approximation-order to generalize from the accelerate-swim-brake behavior to the twist-in-place behavior.

In order to transform the time-dependent policy into a time-independent one, we use the receding horizon technique described in [8]: solving for a length-N trajectory emanating from a point $\mathbf{x}(1)$, we save only the local policy at the first time-step, and re-solve for the next state $\mathbf{x}(2)$ initializing with the current policy. Iterating, we end up with a set of local policies, all with an N-step horizon. This can be thought of as an offline version of Model Predictive Control, see Figure 4. The local controllers are then selected with a nearest-neighbor rule to form a global controller which is a Voronoi tessellation of local controllers around the saved states.

We used $N = 60$ time-steps of length $h = 0.03_s$ with additive control noise $\Sigma_{\mathbf{u}} = .05\mathbf{I}_2$. The cost coefficients were $c_x = 1$ and $c_u = 0.02$. We use the first cubature formula presented in [5], though experiments with different formulas, including ad-hoc ones, also provided good accuracy, as long as inequalities (17) and (19) were maintained.

In the experiment, we test how well a controller which had been trained on a trajectory to a fixed target, describing the accelerate-swim-brake sequence, can generalize to perform the "twisting" behaviour. The receding-horizon learning sequence

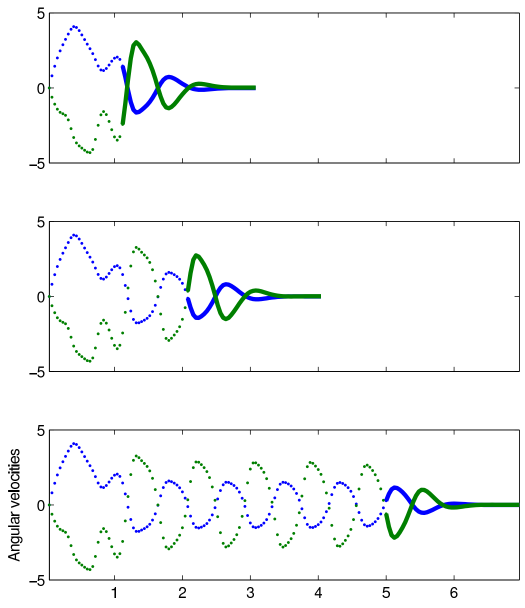

Fig. 4. Receding-horizon trajectories. Three snapshots of the receding horizon trajectory (dotted) appended with the finite-horizon trajectory (solid), for the two internal angles of a *3-swimmer*.

involved a single trajectory, with an initial state away from the stationary target, swimming to it and stopping. This trajectory was solved-for with the classic linear-quadratic scheme, and with a quadratic-cubic controller, i.e. the θ_i and ϕ_i are monomials up to 3rd and 2nd-order, respectively. We then used the global controllers to solve the same task, but this time while perturbing the target on some Lissajous curve.

As shown in the video at http://goo.gl/B1MW3 and figure 5, the quadratic-policy cubic-Value controller is able track the small perturbations, while the standard linear-policy quadratic-Value controller swims around the target in an inefficient manner, due to its insufficient generalization ability.

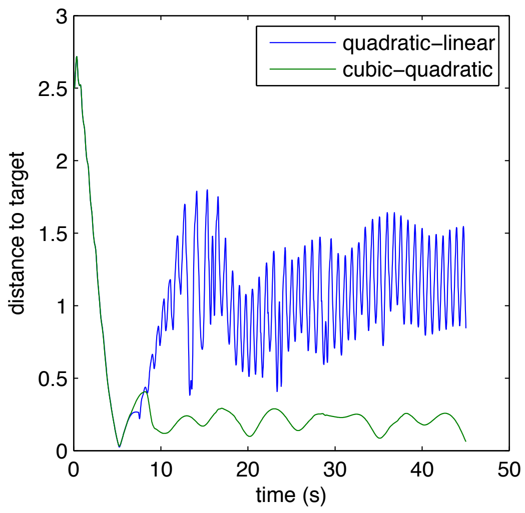

Fig. 5. Tracking behaviour of the two controllers. The distance to the target $||\bar{x}_{\text{nose}}||$ is plotted as a function of time, during the behaviour recorded in the video sequence http://goo.gl/B1MW3.

VII. DISCUSSION

While the results are promising, it seems that the algorithm presented here is not a drop-in replacement for lower-order algorithms due to the added computational burden, as described in Section V-F. In that sense it is possible that the linear-quadratic combination represents a computational "sweet-spot". It might be sensible to first optimize a trajectory with a cheap first or second-order method, and then run this algorithm to construct a higher order approximation with a larger region of validity. Since the state-control trajectory is already near-optimal, the algorithm will converge quickly. This would be useful for deterministic as well as stochastic systems.

An attractive feature is the prospect of using non-gaussian distributions. The very general nature of our algorithm allows us to easily replace the single-gaussian with a mixture-of-gaussians representation. This might be useful in the case where the trajectory is expected to be multi-modal. The most prominent case where gaussianity is an unreasonable assumption, is the one where the dynamics are discontinuous, as with rigid contact and friction. When the state distribution reaches a contact or slippage in state-space, it splits into two spatially distinct modes of pre- and post-discontinuity, and any attempt to model them as a single gaussian is bound to fail. A simple way to generalize the algorithm herein, is to replace the re-gaussianization in (14) with the Expectation Maximization algorithm for a mixture of 2 gaussians, effectively allowing the trajectory to split (and merge) when required. This possibility will be examined in future work.

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation.

REFERENCES

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

[2] E. Todorov and Y. Tassa, "Iterative local dynamic programming," in *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Nashville, TN, USA, 2009, pp. 90–95.

[3] E. Todorov and W. 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.*, Portland, OR, USA, 2005, pp. 300–306.

[4] A. Stroud, *Approximate calculation of multiple integrals*. Englewood Cliffs N.J.: Prentice-Hall, 1971.

[5] J. Lu and D. L. Darmofal, "Higher-Dimensional integration with gaussian weight for applications in probabilistic design," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, p. 613, 2004.

[6] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *Automatic Control, IEEE Transactions on*, vol. 54, no. 6, p. 12541269, 2009.

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

[8] 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.