# Extended LQR: Locally-Optimal Feedback Control for Systems with Non-Linear Dynamics and Non-Quadratic Cost

Jur van den Berg

**Abstract** We present Extended LQR, a novel approach for locally-optimal control for robots with non-linear dynamics and non-quadratic cost functions. Our formulation is conceptually different from existing approaches, and is based on the novel concept of *LQR-smoothing*, which is an LQR-analogue of Kalman smoothing. Our approach iteratively performs both a backward Extended LQR pass, which computes approximate *cost-to-go* functions, and a forward Extended LQR pass, which computes approximate *cost-to-come* functions. The states at which the *sum* of these functions is minimal provide an approximately optimal sequence of states for the control problem, and we use these points to linearize the dynamics and quadratize the cost functions in the subsequent iteration. Our results indicate that Extended LQR converges quickly and reliably to a locally-optimal solution of the non-linear, non-quadratic optimal control problem. In addition, we show that our approach is easily extended to include *temporal optimization*, in which the duration of a trajectory is optimized as part of the control problem. We demonstrate the potential of our approach on two illustrative non-linear control problems involving simulated and physical differential-drive robots and simulated quadrotor helicopters.

## 1 Introduction

Optimal control is an important problem in robotics. The problem is generally defined in terms of a description of the robot's dynamics and a control objective in the form of a cost function that is to be minimized, and the goal is to compute an optimal feedback control policy that tells the robot what control to apply given the state it is in. The linear-quadratic regulator (LQR) [4] provides a closed-form optimal solution in case the dynamics of the robot are linear and the cost function is quadratic. The linear-quadratic control problem is known to be closely related to the linear-Gaussian state estimation problem [21], for which the Kalman filter provides the

Jur van den Berg
School of Computing, University of Utah. e-mail: berg@cs.utah.edu

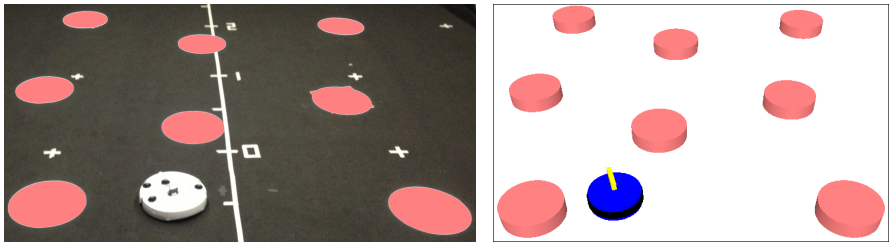


**Fig. 1** A physical and simulated iRobot Create navigating an environment with obstacles using Extended LQR. See **http://arl.cs.utah.edu/research/extendedlqr/** for videos of our experiments.

optimal closed-form solution. Since the *Extended* Kalman filter provides a natural extension to state estimation of non-linear systems [1], a natural question is whether LQR can be extended in a similar fashion to systems with general non-linear dynamics and non-quadratic cost functions. The main challenge here is that the LQR controller is derived using a recursion backward in time starting at the final time, and that the future states and control inputs of the robot are unknown at the time of designing the controller. So, choosing suitable points to linearize the dynamics and quadratize the cost functions about is non-trivial.

To address this, we propose *Extended LQR*, a novel approach to the non-linear, non-quadratic control problem that is based on the novel concept of *LQR-smoothing*. The LQR-smoother consists, analogous to the Kalman smoother [14], of a standard backward LQR pass that computes *cost-to-go* functions, and a *forward* LQR pass that computes *cost-to-come* functions. The sum of these functions give *total-cost* functions, and the states at which the total-cost functions are minimal provide an *optimal sequence of states* for the linear-quadratic control problem. To extend this to systems with non-linear dynamics and non-quadratic cost, Extended LQR iteratively performs a backward and a forward pass (analogous to the extended Kalman smoother [2]) to progressively obtain a better idea of the robot's future trajectory. The states at which the approximate total-cost functions are minimal are used to linearize the dynamics and quadratize the cost functions in each iteration, while the control policies computed in each pass provide control inputs to linearize and quadratize about. We will show that this procedure converges quickly and reliably to a locally-optimal solution to the non-linear, non-quadratic control problem.

There is a large body of literature on the non-linear, non-quadratic control problem, and many approaches based on linear-quadratic approximations have previously been proposed (as we discuss in detail in Section 2). Even though we will show that Extended LQR improves upon existing methods such as Iterative LQR (iLQR) [12], the main purpose of this paper is to introduce a conceptually novel approach to non-linear, non-quadratic control. Our formulation remains strictly within the LQR framework and does not use the duality between control and estimation [21] (in contrast to e.g. Approximate Inference Control (AICO) [22]), resulting in a conceptually intuitive approach that is easy to implement. Also, we will show that Extended LQR can naturally be applied to *temporal optimization* problems, in which the duration of the trajectory is to be optimized as part of the optimal con-

trol problem [15]. We have made source code of our approach publicly available at **http://arl.cs.utah.edu/research/extendedlqr/**.

We experimented with our approach on two illustrative non-linear control problems with and without temporal optimization, and compared performance to iLQR. Experiments involve both a physical and simulated differential-drive robot in a 2-D environment with obstacles (see Fig. 1), and a simulated quadrotor helicopter with a 12-D state space in 3-D environments with obstacles. Our results indicate that Extended LQR converges more quickly and reliably than iLQR even without providing it with an initial trajectory or implementing special convergence measures.

The remainder of this paper is organized as follows. We discuss related work in Section 2. In Section 3, we formally define the problem we address in this paper. In Section 4 we review the LQR controller and introduce the novel concept of linear-quadratic *smoothing*. We use this in Section 5 to develop our Extended LQR approach, and show how it can be applied to temporal optimization problems in Section 6. We present experimental results in Section 7 and conclude in Section 8.

## 2 Related Work

Our approach is conceptually different from existing approaches to approximate optimal control such as Iterative LQR (iLQR) [12] and Differential Dynamic Programming (DDP) [9]. These approaches linearize the dynamics and quadratize the cost functions about a given (dynamically feasible) nominal trajectory and use LQR to compute a control policy. This control policy is then executed to compute a new nominal trajectory, and the procedure is repeated until convergence. These approaches require special measures such as line search [26] to ensure convergence, as the control policies may drive a new trajectory too "far away" from where the LQ-approximation is valid. Our approach, in contrast, relinearizes/requadratizes in both the backward pass and the forward pass, about a sequence of states and control inputs that is dynamically feasible only upon convergence. We will show that as a result, Extended LQR converges reliably without special convergence measures.

Sequential Quadratic Programming (SQP) methods [13, 3] also iteratively relinearize the dynamics and requadratize the cost functions, with the distinction that it formulates the problem in each iteration as a convex optimization problem that allows for the inclusion of convex constraints on the state and the control input. SQP approaches, however, typically do not compute a feedback control policy, but instead give an optimal open-loop sequence of control inputs for the control problem. The approaches of [17, 27] are closely related, and focus on *trajectory optimization* among obstacles for the specific class of robots with holonomic dynamics. Extended LQR can be used for trajectory optimization as well. In this case, like the mentioned approaches, the initial trajectory need not be dynamically feasible.

Approximate Inference Control (AICO) [22] uses the duality between control and estimation [21], and formulates the optimal control problem in terms of Kullback-Leibler divergence minimization [16]. Even though the derivation of Extended LQR differs considerably from that of AICO, ultimately the qualitative differences are subtle. A key technical difference is that AICO focuses on computing

an optimal sequence of states and does not compute control policies during iterations. This limits AICO to cost functions that are explicitly quadratic in the control input, and it requires "local" iterations for each stage along the trajectory in addition to global forward and backward passes. Our approach computes control policies in each pass, which are used to select control inputs to linearize/quadratize about in the subsequent pass. Extended LQR is therefore applicable to general non-quadratic cost functions and does not use local iterations.

Extended LQR assumes deterministic dynamics, implicitly relying on the fact that the optimal LQR solution is independent from the process noise variance. Other approaches more directly account for stochastic dynamics: AICO lets the process noise variance interact with the cost matrices in the Ricatti equations, resulting in a form of risk-sensitive control [25, 16], and iLQR and DDP have been extended to explicitly take into account state and control input-dependent process noise [20, 19].

The problem of temporal optimization has previously been addressed in [15], which extends AICO with an EM-approach that alternatingly optimizes the trajectory and its duration. We present a more direct approach that includes the time-step in the state, and penalizes for the duration of the trajectory in the cost function. Extended LQR can then be applied as is to the augmented control problem.

One of the concepts underpinning our approach is *Forward LQR*. While forward Ricatti recursion has been explored in optimal control [6, 24], these works do not use it to compute *cost-to-come* functions as part of an *LQR-smoothing* framework. Our work also has similarities in spirit to [7], which employs both a forward and a backward Dijkstra's algorithm in discrete gridworlds.

## 3 Problem Definition

Let $\mathbb{X} \subset \mathbb{R}^n$ and $\mathbb{U} \subset \mathbb{R}^m$ be the state space and the control input space, respectively, of the robot, and let its deterministic discrete-time dynamics be given by:

$$\mathbf{x}_{t+1} = \mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t), \tag{1}$$

with $\mathbf{g}_t \in \mathbb{X} \times \mathbb{U} \to \mathbb{X}$, where $\mathbf{x}_t \in \mathbb{X}$ and $\mathbf{u}_t \in \mathbb{U}$ denote the state and the control input of the robot at stage $t$. Let the control objective be defined as minimizing a cost function:

$$c_\ell(\mathbf{x}_\ell) + \textstyle\sum_{t=0}^{\ell-1} c_t(\mathbf{x}_t, \mathbf{u}_t), \tag{2}$$

for given horizon $\ell$ and local cost functions $c_\ell \in \mathbb{X} \to \mathbb{R}$ and $c_t \in \mathbb{X} \times \mathbb{U} \to \mathbb{R}$. Now, the optimal control problem is defined as finding a *control policy* $\pi_t \in \mathbb{X} \to \mathbb{U}$ for all $0 \leq t < \ell$, such that selecting controls $\mathbf{u}_t = \pi_t(\mathbf{x}_t)$ minimizes Eq. (2).

A general solution approach computes the *cost-to-go* functions $s_t \in \mathbb{X} \to \mathbb{R}$ and the optimal control policies $\pi_t$ using a backward recursion procedure:

$$s_\ell(\mathbf{x}_\ell) = c_\ell(\mathbf{x}_\ell), \qquad s_t(\mathbf{x}_t) = \min_{\mathbf{u}_t}(c_t(\mathbf{x}_t, \mathbf{u}_t) + s_{t+1}(\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t))), \tag{3}$$

$$\pi_t(\mathbf{x}_t) = \operatorname{argmin}_{\mathbf{u}_t}(c_t(\mathbf{x}_t, \mathbf{u}_t) + s_{t+1}(\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t))). \tag{4}$$

The cost-to-go function $s_t(\mathbf{x}_t)$ gives the total future cost that will be accrued between stage $t$ and stage $\ell$ by a minimal-cost sequence of states and control inputs that starts in $\mathbf{x}_t$ at stage $t$. In general, there is no explicit parametric expression for the cost-to-go functions $s_t$, except in the case where the dynamics are linear and the local cost functions are quadratic (as we will review in Section 4.1). The objective of this paper is to extend this approach to create (locally) optimal solutions to the general control problem with non-linear dynamics and non-quadratic cost.

Even though we address the discrete-time control problem, we assume the *continuous-time* dynamics are given:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)). \tag{5}$$

A continuous-time formulation is typically the most natural way to describe the dynamics of (non-linear) robotic systems, and it allows us to evaluate the discrete-time dynamics $\mathbf{g}_t$ (Eq. (1)), as well as its *inverse*, for any given time-step using e.g. Runge-Kutta integration. The *inverse* discrete-time dynamics are denoted by:

$$\mathbf{x}_t = \bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \tag{6}$$

where $\bar{\mathbf{g}}_t$ is defined such that $\mathbf{g}_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \mathbf{u}_t) = \mathbf{x}_{t+1}$ and, equivalently, $\bar{\mathbf{g}}_t(\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t), \mathbf{u}_t) = \mathbf{x}_t$. It is obtained by integrating Eq. (5) backward in time. We further assume that the local cost functions have positive-(semi)definite Hessians:

$$\frac{\partial^2 c_\ell}{\partial \mathbf{x}_\ell \partial \mathbf{x}_\ell} > 0, \qquad \frac{\partial^2 c_t}{\partial \mathbf{u}_t \partial \mathbf{u}_t} > 0, \qquad \frac{\partial^2 c_t}{\partial \left[\begin{smallmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{smallmatrix}\right] \partial \left[\begin{smallmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{smallmatrix}\right]} \geq 0. \tag{7}$$

## 4 Linear-Quadratic Control and Smoothing

We begin this section by reviewing LQR, which computes *cost-to-go* functions and provides a closed-form optimal solution to the linear-quadratic control problem. We then show how *cost-to-come* functions can be computed by a procedure similar to LQR, but one that runs *forward* in time, and introduce the concept of *linear-quadratic smoothing*, in which the cost-to-go and the cost-to-come functions are combined to create an LQR analogue of the Kalman smoother [14] that provides the optimal sequence of states for the linear-quadratic control problem. These concepts are at the foundation of our Extended LQR approach for non-linear, non-quadratic control, which we discuss in Section 5.

### *4.1 LQR Control*

The linear-quadratic control problem is a special case of the general problem defined above for which the *LQR controller* provides a closed-form optimal solution [4]. In this case, the dynamics are linear and given by:

$$\mathbf{x}_{t+1} = \mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t) = A_t \mathbf{x}_t + B_t \mathbf{u}_t + \mathbf{c}_t, \tag{8}$$

with $A_t \in \mathbb{R}^{n\times n}$, $B_t \in \mathbb{R}^{n\times m}$, and $\mathbf{c}_t \in \mathbb{R}^n$ given for all $t$. The local cost functions are quadratic, and given by:

$$c_\ell(\mathbf{x}_\ell) = \frac{1}{2}\mathbf{x}_\ell^T Q_\ell \mathbf{x}_\ell + \mathbf{x}_\ell^T \mathbf{q}_\ell, \quad c_t(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^T \begin{bmatrix}Q_t & P_t^T \\ P_t & R_t\end{bmatrix}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^T \begin{bmatrix}\mathbf{q}_t \\ \mathbf{r}_t\end{bmatrix}, \quad (9)$$

where $Q_t \in \mathbb{R}^{n\times n}$, $R_t \in \mathbb{R}^{m\times m}$, $P_t \in \mathbb{R}^{m\times n}$, $\mathbf{q}_t \in \mathbb{R}^n$, and $\mathbf{r}_t \in \mathbb{R}^m$ are given for all $t$. Matrices $Q_\ell > 0$ and $R_t > 0$ are positive-definite, and $\left[\begin{smallmatrix}Q_t & P_t^T \\ P_t & R_t\end{smallmatrix}\right] \geq 0$ is positive-semidefinite, in accordance with Eq. (7).

For this control problem, the cost-to-go functions $s_t$ have the following explicit quadratic formulation:

$$s_t(\mathbf{x}_t) = \tfrac{1}{2}\mathbf{x}_t^T S_t \mathbf{x}_t + \mathbf{x}_t^T \mathbf{s}_t + k, \quad (10)$$

where $k$ is a constant, and $S_t \in \mathbb{R}^{n\times n} > 0$ and $\mathbf{s}_t \in \mathbb{R}^n$ are computed using backward recursion. For final stage $\ell$, we have $S_\ell = Q_\ell$ and $\mathbf{s}_\ell = \mathbf{q}_\ell$, and for stage $\ell > t \geq 0$:

$$S_t = D_t - C_t^T E_t^{-1} C_t, \qquad \mathbf{s}_t = \mathbf{d}_t - C_t^T E_t^{-1}\mathbf{e}_t, \quad (11)$$

where:

$$C_t = P_t + B_t^T S_{t+1} A_t, \quad D_t = Q_t + A_t^T S_{t+1} A_t, \quad E_t = R_t + B_t^T S_{t+1} B_t,$$
$$\mathbf{d}_t = \mathbf{q}_t + A_t^T \mathbf{s}_{t+1} + A_t^T S_{t+1}\mathbf{c}_t, \quad \mathbf{e}_t = \mathbf{r}_t + B_t^T \mathbf{s}_{t+1} + B_t^T S_{t+1}\mathbf{c}_t.$$

The optimal feedback control policies $\pi_t$ have an explicit linear formulation:

$$\pi_t(\mathbf{x}_t) = L_t \mathbf{x}_t + \mathbf{l}_t, \qquad L_t = -E_t^{-1} C_t, \qquad \mathbf{l}_t = -E_t^{-1}\mathbf{e}_t, \quad (12)$$

## *4.2 Cost-to-Come Functions and Forward LQR*

As mentioned above, the cost-to-go functions $s_t(\mathbf{x}_t)$ give the total future cost that will be accrued between stage $t$ and stage $\ell$ (*including* the cost incurred at stage $t$) by a minimal-cost sequence of states and control inputs that starts in $\mathbf{x}_t$ at stage $t$. We use the similar concept of *cost-to-come* functions [11], denoted $\bar{s}_t(\mathbf{x}_t)$, which give the total *past* cost that was accrued between stage 0 and stage $t$ (*excluding* the cost incurred at stage $t$) by a minimal-cost sequence of states and controls that arrives in $\mathbf{x}_t$ at stage $t$. Given the *inverse* dynamics (see Eq. (6)), the cost-to-come functions are generally defined by the following *forward* recursion procedure:

$$\bar{s}_0(\mathbf{x}_0) = 0, \quad \bar{s}_{t+1}(\mathbf{x}_{t+1}) = \min_{\mathbf{u}_t}(c_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \mathbf{u}_t) + \bar{s}_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t))), \quad (13)$$
$$\bar{\pi}_t(\mathbf{x}_{t+1}) = \operatorname{argmin}_{\mathbf{u}_t}(c_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t), \mathbf{u}_t) + \bar{s}_t(\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t))), \quad (14)$$

Here, $\bar{\pi}_t$ is an *inverse* control policy that given a state $\mathbf{x}_{t+1}$ at stage $t+1$ computes the control input $\mathbf{u}_t = \bar{\pi}_t(\mathbf{x}_{t+1})$ that was applied at stage $t$ in order to arrive at $\mathbf{x}_{t+1}$ with minimal cost-to-come.

If the dynamics are linear and given by Eq. (8), and the local cost functions are quadratic and given by Eq. (9), the cost-to-come functions $\bar{s}_t$ have an explicit

quadratic formulation, similar to the cost-to-go functions in LQR:

$$\bar{s}_t(\mathbf{x}_t) = \tfrac{1}{2}\mathbf{x}_t^T \bar{S}_t \mathbf{x}_t + \mathbf{x}_t^T \bar{\mathbf{s}}_t + \bar{k}, \tag{15}$$

where $\bar{S}_t \in \mathbb{R}^{n \times n} \geq 0$ and $\bar{\mathbf{s}}_t \in \mathbb{R}^n$. The recursive update equations for $\bar{S}_t$ and $\bar{\mathbf{s}}_t$ run forward in time, and can be derived in a similar fashion as those of standard LQR, given that the linear dynamics are expressed in their inverse form:

$$\mathbf{x}_t = \bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t) = \bar{A}_t \mathbf{x}_{t+1} + \bar{B}_t \mathbf{u}_t + \bar{\mathbf{c}}_t, \tag{16}$$

with $\bar{A}_t = A_t^{-1}$, $\bar{B}_t = -A_t^{-1}B_t$, and $\bar{\mathbf{c}}_t = -A_t^{-1}\mathbf{c}_t$ (as follows from solving Eq. (8) for $\mathbf{x}_t$). Then, we initially have $\bar{S}_0 = 0$ and $\bar{\mathbf{s}}_0 = \mathbf{0}$, and for stage $0 \leq t < \ell$:

$$\bar{S}_{t+1} = \bar{D}_t - \bar{C}_t^T \bar{E}_t^{-1} \bar{C}_t, \qquad \bar{\mathbf{s}}_{t+1} = \bar{\mathbf{d}}_t - \bar{C}_t^T \bar{E}_t^{-1} \bar{\mathbf{e}}_t, \tag{17}$$

where

$$\bar{C}_t = \bar{B}_t^T(\bar{S}_t + Q_t)\bar{A}_t + P_t\bar{A}_t,\ \bar{D}_t = \bar{A}_t^T(\bar{S}_t + Q_t)\bar{A}_t,\ \bar{E}_t = \bar{B}_t^T(\bar{S}_t + Q_t)\bar{B}_t + R_t + P_t\bar{B}_t + \bar{B}_t^T P_t^T,$$
$$\bar{\mathbf{d}}_t = \bar{A}_t^T(\bar{\mathbf{s}}_t + \mathbf{q}_t) + \bar{A}_t^T(\bar{S}_t + Q_t)\bar{\mathbf{c}}_t, \quad \bar{\mathbf{e}}_t = \mathbf{r}_t + P_t\bar{\mathbf{c}}_t + \bar{B}_t^T(\bar{\mathbf{s}}_t + \mathbf{q}_t) + \bar{B}_t^T(\bar{S}_t + Q_t)\bar{\mathbf{c}}_t.$$

The inverse control policies $\bar{\pi}_t$ have, similar to the control policies in LQR, an explicit linear formulation:

$$\bar{\pi}_t(\mathbf{x}_{t+1}) = \bar{L}_t \mathbf{x}_{t+1} + \bar{\mathbf{l}}_t, \qquad \bar{L}_t = -\bar{E}_t^{-1}\bar{C}_t, \qquad \bar{\mathbf{l}}_t = -\bar{E}_t^{-1}\bar{\mathbf{e}}_t. \tag{18}$$

In the remainder of this paper, we refer to the above procedure as *Forward LQR*.

### *4.3 Linear-Quadratic Smoothing*

Performing both the standard LQR procedure (following Section 4.1) and the forward LQR procedure (following Section 4.2) for a given linear-quadratic control problem gives both the cost-to-go functions $s_t$ and cost-to-come functions $\bar{s}_t$. The sum of the cost-to-go $s_t(\mathbf{x}_t)$ and the cost-to-come $\bar{s}_t(\mathbf{x}_t)$ gives the total (past and future) cost $\hat{s}_t(\mathbf{x}_t)$ accrued between stage 0 and stage $\ell$ by a minimal-cost sequence of states and controls that visits $\mathbf{x}_t$ at stage $t$:

$$\hat{s}_t(\mathbf{x}_t) = s_t(\mathbf{x}_t) + \bar{s}_t(\mathbf{x}_t) = \tfrac{1}{2}\mathbf{x}_t^T(S_t + \bar{S}_t)\mathbf{x}_t + \mathbf{x}_t^T(\mathbf{s}_t + \bar{\mathbf{s}}_t) + \hat{k}. \tag{19}$$

Let $\hat{\mathbf{x}}_t$ denote the state at stage $t$ for which the total-cost function $\hat{s}_t$ is minimal:

$$\hat{\mathbf{x}}_t = \operatorname{argmin}_{\mathbf{x}_t} \hat{s}_t(\mathbf{x}_t) = -(S_t + \bar{S}_t)^{-1}(\mathbf{s}_t + \bar{\mathbf{s}}_t). \tag{20}$$

(Note that this inverse exists since $S_t > 0$.) Then, the sequence of states $\{\hat{\mathbf{x}}_0, \ldots, \hat{\mathbf{x}}_\ell\}$ is the minimum-cost sequence of states for the given linear-quadratic control problem. The associated controls are given by the (inverse) control policies:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{g}_t(\hat{\mathbf{x}}_t, \pi_t(\hat{\mathbf{x}}_t)), \qquad \hat{\mathbf{x}}_t = \bar{\mathbf{g}}_t(\hat{\mathbf{x}}_{t+1}, \bar{\pi}_t(\hat{\mathbf{x}}_{t+1})). \tag{21}$$

Note also that $\pi_t(\hat{\mathbf{x}}_t) = \bar{\pi}_t(\hat{\mathbf{x}}_{t+1})$.

The above can be seen as an LQR-analogue of the *Kalman smoother* [14]. The Kalman smoother performs both a forward and a backward Kalman filter to compute the posterior distributions of the state given all (past and future) observations. The mean of these distributions gives the maximum-likelihood (and maximum-a-posteriori) sequence of states. In fact, it can be shown that the above procedure is the exact dual of the Kalman smoother. We use the insights of Eqs. (20) and (21) below to develop Extended LQR.

## 5 Extended LQR

In this section we extend LQR and Forward LQR to the case of general non-linear dynamics and non-quadratic local cost functions, in a similar way as how the extended Kalman filter extends the Kalman filter to non-linear systems. In our (forward) Extended LQR approach, we use the same equations as in standard LQR and forward LQR to update the cost-to-go and cost-to-come functions, respectively. The challenge is how to linearize the dynamics and quadratize the local cost functions in each cycle of the recursion, since there are a-priori no obvious candidates for the state and control input to linearize/quadratize about.

The idea of our approach to tackle this problem is to iteratively perform backward and forward Extended LQR passes to compute increasingly better approximations of the total-cost functions $\hat{s}_t$, analogous to the (iterated) extended Kalman smoother [2]. In each pass, we use the states at which the current approximations of the total-cost functions are minimal to linearize and quadratize, and use the control policies from the preceding pass to select control inputs to linearize/quadratize about.

The iteration continues until convergence, i.e. when the minimum-total-cost states no longer change. These states then provide a locally-optimal sequence of states for the non-linear, non-quadratic control problem, and the control policies computed by the last backward pass provide an approximately optimal feedback control policy. The iteration starts with a backward pass, which we discuss first. We then discuss the forward pass, and discuss convergence properties.

### *5.1 Backward Extended LQR*

We assume during the backward pass that the cost-to-come functions, as defined by $\bar{S}_t$ and $\bar{\mathbf{s}}_t$, as well as the inverse control policies $\bar{\pi}_t$ are available for all $t$ from the preceding forward pass. Also, an initial quadratization point $\hat{\mathbf{x}}_\ell$ for stage $\ell$ is available from the forward pass. If this is the first backward pass (and no forward pass preceded it), one can assume $\bar{S}_t = 0$, $\bar{\mathbf{s}}_t = \mathbf{0}$, $\bar{\pi}_t(\mathbf{x}_{t+1}) = \mathbf{0}$, and $\hat{\mathbf{x}}_\ell = \mathbf{0}$, or alternatively set these values in accordance with any available prior information (such as a given initial trajectory).

The backward pass closely follows the standard LQR approach of Section 4.1. We initialize the backward pass by setting $S_\ell = Q_\ell$ and $\mathbf{s}_\ell = \mathbf{q}_\ell$ as in standard LQR, where matrix $Q_\ell$ and vector $\mathbf{q}_\ell$ are obtained by quadratizing the final cost function $c_\ell(\mathbf{x}_\ell)$ about the given point $\hat{\mathbf{x}}_\ell$, which puts it in the form of Eq. (9), with:

$$Q_\ell = \frac{\partial^2 c_\ell}{\partial \mathbf{x}_\ell \partial \mathbf{x}_\ell}(\hat{\mathbf{x}}_\ell), \qquad \mathbf{q}_\ell = \frac{\partial c_\ell}{\partial \mathbf{x}_\ell}(\hat{\mathbf{x}}_\ell) - Q_\ell \hat{\mathbf{x}}_\ell. \tag{22}$$

We then proceed by performing recursive updates of the cost-to-go function for $\ell > t \geq 0$. In each step of the recursion, we compute $S_t$ and $\mathbf{s}_t$ given $S_{t+1}$ and $\mathbf{s}_{t+1}$, starting with $t = \ell - 1$. For this, we use Eq. (11) as in standard LQR, where matrices $A_t$, $B_t$, and $\mathbf{c}_t$ are obtained by linearizing the dynamics $\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t)$, and matrices $P_t$, $Q_t$, $R_t$, $\mathbf{q}_t$, and $\mathbf{r}_t$ are obtained by quadratizing the local cost function $c_t(\mathbf{x}_t, \mathbf{u}_t)$.

The challenge is to choose a state $\hat{\mathbf{x}}_t$ and control input $\hat{\mathbf{u}}_t$ about which to linearize the dynamics and quadratize the local cost function. In principle, we would like to set $\hat{\mathbf{x}}_t$ to the minimum-total-cost state at stage $t$, as defined by Eq. (20). However, we do not yet have an estimate of the total-cost function at stage $t$, since $S_t$ and $\mathbf{s}_t$ are not yet computed (at least not in the current backward pass). We do have a current-best estimate of the state with minimal total-cost at stage $t+1$, though:

$$\hat{\mathbf{x}}_{t+1} = -(S_{t+1} + \bar{S}_{t+1})^{-1}(\mathbf{s}_{t+1} + \bar{\mathbf{s}}_{t+1}). \tag{23}$$

We can then set the state $\hat{\mathbf{x}}_t$ and the control input $\hat{\mathbf{u}}_t$ about which to linearize and quadratize in accordance with Eq. (21), using the inverse dynamics and the inverse control policy $\bar{\pi}_t$ available from the preceding forward pass:

$$\hat{\mathbf{u}}_t = \bar{\pi}_t(\hat{\mathbf{x}}_{t+1}), \qquad \hat{\mathbf{x}}_t = \bar{\mathbf{g}}_t(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t). \tag{24}$$

Linearizing the (forward) dynamics $\mathbf{g}_t(\mathbf{x}_t, \mathbf{u}_t)$ about $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ then puts it in the form of Eq. (8), with:

$$A_t = \frac{\partial \mathbf{g}_t}{\partial \mathbf{x}_t}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t), \qquad B_t = \frac{\partial \mathbf{g}_t}{\partial \mathbf{u}_t}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t), \qquad \mathbf{c}_t = \hat{\mathbf{x}}_{t+1} - A_t \hat{\mathbf{x}}_t - B_t \hat{\mathbf{u}}_t, \tag{25}$$

and quadratizing the local cost function $c_t(\mathbf{x}_t, \mathbf{u}_t)$ about $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ puts it in the form of Eq. (9), with:

$$\begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} = \frac{\partial^2 c_t}{\partial \left[\begin{smallmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{smallmatrix}\right] \partial \left[\begin{smallmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{smallmatrix}\right]}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t), \qquad \begin{bmatrix} \mathbf{q}_t \\ \mathbf{r}_t \end{bmatrix} = \frac{\partial c_t}{\partial \left[\begin{smallmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{smallmatrix}\right]}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) - \begin{bmatrix} Q_t & P_t^T \\ P_t & R_t \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_t \\ \hat{\mathbf{u}}_t \end{bmatrix}. \tag{26}$$

Given these matrices and vectors we can compute $S_t$ and $\mathbf{s}_t$ using Eq. (11), and the control policy $\pi_t$ using Eq. (12). This recursion is repeated until $t = 0$.

## *5.2 Forward Extended LQR*

We assume during the forward pass that the cost-to-go functions, as defined by $S_t$ and $\mathbf{s}_t$, as well as the control policies $\pi_t$ are available for all $t$ from the preceding backward pass. The forward pass closely follows the Forward LQR approach of Section 4.2. We initialize the forward pass by setting $\bar{S}_0 = 0$ and $\bar{\mathbf{s}}_0 = \mathbf{0}$, and then proceed by performing recursive updates of the cost-to-come functions for $0 \leq t < \ell$.

In each step of the recursion, we compute $\bar{S}_{t+1}$ and $\bar{\mathbf{s}}_{t+1}$ given $\bar{S}_t$ and $\bar{\mathbf{s}}_t$, starting with $t = 0$. For this, we use Eq. (17), where matrices and vector $\bar{A}_t$, $\bar{B}_t$, and $\bar{\mathbf{c}}_t$ are

obtained by linearizing the inverse dynamics $\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t)$, and matrices and vectors $P_t$, $Q_t$, $R_t$, $\mathbf{q}_t$, and $\mathbf{r}_t$ are obtained by quadratizing the local cost function $c_t(\mathbf{x}_t, \mathbf{u}_t)$.

We select linearization and quadratization points in a similar manner as in the backward pass. We do not yet have an estimate of the minimum-total-cost state at stage $t+1$ as defined by Eq. (20) to linearize the inverse dynamics about, since $\bar{S}_{t+1}$ and $\bar{\mathbf{s}}_{t+1}$ are not yet computed, but we do have a current-best estimate of the state with minimal total-cost at stage $t$:

$$\hat{\mathbf{x}}_t = -(S_t + \bar{S}_t)^{-1}(\mathbf{s}_t + \bar{\mathbf{s}}_t). \tag{27}$$

Using the forward dynamics and the control policy $\pi_t$ available from the backward pass, we then set $\hat{\mathbf{x}}_{t+1}$ and $\hat{\mathbf{u}}_t$ in accordance with Eq. (21):

$$\hat{\mathbf{u}}_t = \pi_t(\hat{\mathbf{x}}_t), \qquad \hat{\mathbf{x}}_{t+1} = \mathbf{g}_t(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t). \tag{28}$$

Linearizing the inverse dynamics $\bar{\mathbf{g}}_t(\mathbf{x}_{t+1}, \mathbf{u}_t)$ about $\hat{\mathbf{x}}_{t+1}$ and $\hat{\mathbf{u}}_t$ and quadratizing the local cost function about $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{u}}_t$ then gives:

$$\bar{A}_t = \frac{\partial \bar{\mathbf{g}}_t}{\partial \mathbf{x}_{t+1}}(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t), \quad \bar{B}_t = \frac{\partial \bar{\mathbf{g}}_t}{\partial \mathbf{u}_t}(\hat{\mathbf{x}}_{t+1}, \hat{\mathbf{u}}_t), \quad \bar{\mathbf{c}}_t = \hat{\mathbf{x}}_t - \bar{A}_t \hat{\mathbf{x}}_{t+1} - \bar{B}_t \hat{\mathbf{u}}_t, \tag{29}$$

and $P_t$, $Q_t$, $R_t$, $\mathbf{q}_t$, and $\mathbf{r}_t$ as in Eq. (26). We can then compute $\bar{S}_{t+1}$ and $\bar{\mathbf{s}}_{t+1}$ using Eq. (17), and the inverse control policy $\bar{\pi}_t$ using Eq. (18). This recursion is repeated until $t = \ell - 1$. Finally, we set an initial quadratization point for stage $\ell$ for the subsequent backward pass:

$$\hat{\mathbf{x}}_\ell = -(S_\ell + \bar{S}_\ell)^{-1}(\mathbf{s}_\ell + \bar{\mathbf{s}}_\ell). \tag{30}$$

### *5.3 Convergence Properties*

Upon convergence, the properties of Eq. (21) hold, and the minimum-total-cost states $\hat{\mathbf{x}}_t$ form an exact *locally-optimal* sequence of states for the non-linear, non-quadratic control problem. The (linear) control policies $\pi_t$ from the last backward pass then provide a first-order Taylor approximation about the minimum-total-cost states $\hat{\mathbf{x}}_t$ of the true locally-optimal control policy, and the computed (quadratic) cost-to-go functions provide a second-order Taylor approximation of the true locally-optimal cost-to-go functions. Using the analogy with the iterated Kalman smoother [2], our approach can be shown to perform Gauss-Newton updates towards a local optimum and thus should exhibit a rate of convergence approaching second-order [5]. The running-time per iteration is $O(\ell n^3)$.

Before convergence is achieved, the sequence of minimum-total-cost states is not necessarily consistent with the non-linear dynamics. Also, as the minimal-total-cost states are in a way an "average" between the minimal-cost-to-go states and the minimal-cost-to-come states, of which only one is updated in each pass, the minimal-total-cost states smoothly evolve. This is why our approach can converge reliably without implementing convergence measures such as line search.

## 6 Temporal Optimization

In many cases it is desirable to optimize the duration of the trajectory as part of the optimal control problem [15]. Such an objective naturally fits within our framework, by keeping the number of steps $\ell$ constant, but making the *time-step* $\tau$ part of the state, and penalizing linearly for the duration of the trajectory in the cost function.

Since the time-step must be strictly positive, we make its logarithm $\lambda = \log \tau$ part of the state. The augmented state $\tilde{\mathbf{x}}$ and the augmented (continuous-time) dynamics $\dot{\tilde{\mathbf{x}}}(t) = \tilde{\mathbf{f}}(t, \tilde{\mathbf{x}}(t), \mathbf{u}(t))$ are then defined as:

$$\tilde{\mathbf{x}} = [\mathbf{x}^T, \lambda]^T, \qquad \dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), \qquad \dot{\lambda}(t) = 0, \tag{31}$$

where we use a time-step of $\tau = \exp \lambda_t$ when evaluating the discrete-time dynamics $\mathbf{g}_t(\tilde{\mathbf{x}}_t, \mathbf{u}_t)$ (or its inverse). In addition, we create augmented cost functions $\tilde{c}_t$ that add a term $\exp \lambda_t$ to the local cost functions $c_t$ of each stage $0 < t \leq \ell$ (excluding 0), such that in total the duration of the trajectory is linearly penalized:

$$\tilde{c}_\ell(\tilde{\mathbf{x}}_\ell) = \exp \lambda_\ell + c_\ell(\mathbf{x}_\ell), \quad \tilde{c}_t(\tilde{\mathbf{x}}_t, \mathbf{u}_t) = \begin{cases} c_t(\mathbf{x}_t, \mathbf{u}_t) & \text{if } t = 0, \\ \exp \lambda_t + c_t(\mathbf{x}_t, \mathbf{u}_t) & \text{if } 0 < t < \ell. \end{cases} \tag{32}$$

Since the second derivative of $\exp \lambda_t$ with respect to $\lambda_t$ is positive, the augmented cost functions obey the requirements of Eq. (7).

With these definitions of augmented state, dynamics, and cost functions, we can use Extended LQR as is to solve control problems with temporal optimization. Upon convergence, the value of $\lambda_t$ in the minimum-total-cost states $\hat{\tilde{\mathbf{x}}}_t$ is the same for all $t$, as follows from Eq. (21) and the fact that $\dot{\lambda}(t) = 0$ in the augmented dynamics.

## 7 Experiments

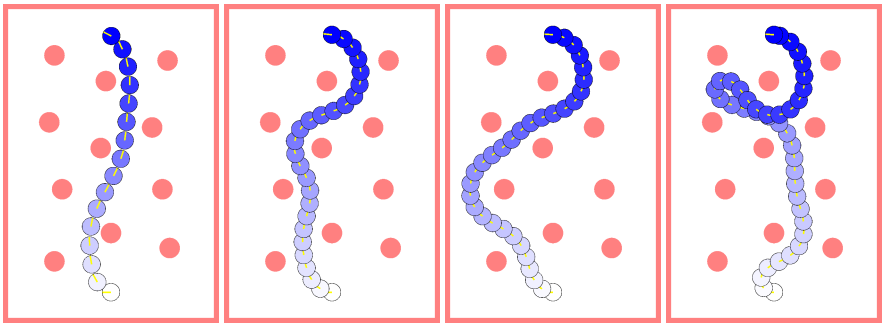
We experimented with Extended LQR on two systems; an iRobot Create differential-drive robot, which we use mainly for illustrative purposes to gain insight into the working of our approach and temporal optimization, and a simulated quadrotor helicopter, on which we perform extensive quantitative analysis and performance comparison with Iterative LQR (iLQR).

### *7.1 iRobot Create Differential-Drive Robot*

Our first experiment involves a simulated and physical iRobot Create differential-drive robot. Its state $\mathbf{x} = [p_x, p_y, \theta]^T$ is described by its two-dimensional position $(p_x, p_y)$ (m) and orientation $\theta$ (rad), and its control input $\mathbf{u} = [v_\ell, v_r]^T$ consists of the speeds (m/s) of the left and right wheel, respectively. The dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ of the robot are non-linear and given by:

$$\dot{p}_x = \tfrac{1}{2}(v_\ell + v_r)\cos\theta, \qquad \dot{p}_y = \tfrac{1}{2}(v_\ell + v_r)\sin\theta, \qquad \dot{\theta} = (v_r - v_\ell)/w,$$

where $w = 0.258$m is the distance between the wheels of the iRobot Create.



(a) $\tau = \frac{1}{10}$s (b) $\tau = \frac{1}{6}$s, $\tau \rightarrow 0.168$s (c) $\tau = \frac{1}{5}$s (d) $\tau = \frac{1}{4}$s

**Fig. 2** Trajectories resulting from the differential-drive robot experiments. The robot is shown every second. See **http://arl.cs.utah.edu/research/extendedlqr/** for videos of our experiments.

We use the following cost functions in our experiments:

$$\begin{aligned} c_\ell(\mathbf{x}) &= \tfrac{1}{2}(\mathbf{x}-\mathbf{x}_\ell^\star)^T Q(\mathbf{x}-\mathbf{x}_\ell^\star), \\ c_0(\mathbf{x},\mathbf{u}) &= \tfrac{1}{2}(\mathbf{x}-\mathbf{x}_0^\star)^T Q(\mathbf{x}-\mathbf{x}_0^\star) + \tfrac{1}{2}(\mathbf{u}-\mathbf{u}^\star)^T R(\mathbf{u}-\mathbf{u}^\star), \\ c_t(\mathbf{x},\mathbf{u}) &= \tfrac{1}{2}(\mathbf{u}-\mathbf{u}^\star)^T R(\mathbf{u}-\mathbf{u}^\star) + q\textstyle\sum_i \exp(-d_i(\mathbf{x})), \end{aligned}$$

for $0 < t < \ell$, where $\mathbf{x}_\ell^\star$ is the target state, $\mathbf{x}_0^\star$ the initial state, and $\mathbf{u}^\star$ the nominal control input, which we set to $(0.25, 0.25)$m/s for the iRobot Create (which has maximum wheel speeds of $v_\ell, v_r \in [-0.5, 0.5]$m/s). Matrices $Q$ and $R$ and scalar $q$ are positive weight factors. The function $d_i(\mathbf{x})$ gives the (signed) distance between the robot configured at $\mathbf{x}$ and the $i$'th obstacle in the environment. The term $q\sum_i \exp(-d_i(\mathbf{x}))$ makes this local cost function non-quadratic in the state $\mathbf{x}$. Since its Hessian is not always positive-semidefinite, counter to the requirement of Eq. (7), it is *regularized* when quadratizing the cost function. That is, its eigendecomposition is computed and its negative eigenvalues are set to zero [8].

We experimented in the environment of Fig. 2, which measures 4m by 6m. The obstacles each have a radius of 0.2m, and the iRobot Create has a physical radius of 0.17m. The initial state was set to $\mathbf{x}_0^\star = (0, -2.5, \pi)$ and the target state to $\mathbf{x}_\ell^\star = (0, 2.5, \pi)$. We ran our approach for various fixed time-steps $\tau$ and temporal optimization, with a fixed number of steps $\ell = 150$. Extended LQR was not seeded with an initial trajectory, and we let the algorithm run until the relative improvement dropped below $10^{-4}$. Fig. 2 shows the resulting trajectories. Table 1(a) gives quantitative results, where the second column gives the number of iterations until convergence, the third column the computation time required (for a C++ implementation on an Intel i5 1.60GHz with 4GB RAM), and the fourth column gives the average speed of the robot along the trajectory.

To appreciate these results, it is important to note that the nominal control input $\mathbf{u}^\star$ was set to 0.25m/s. Hence, the robot is penalized for driving either faster or slower. For a time-step of $\tau = \frac{1}{10}$s, the robot only has $\ell\tau = 15$s to reach the goal; it therefore chooses a short trajectory that comes close to the obstacles with a speed

Differential-Drive Robot ($n = 3$, $\ell = 150$)

| | Extended LQR | | |
|---|---|---|---|
| time-step (s) | #iters | time (s) | speed (m/s) |
| $\tau = 1/10$ | 8 | 0.014 | 0.359 |
| $\tau = 1/6$ | 7 | 0.012 | 0.251 |
| $\tau = 1/5$ | 8 | 0.014 | 0.238 |
| $\tau = 1/4$ | 14 | 0.027 | 0.242 |
| $\tau \to 0.168$ | 7 | 0.016 | 0.250 |

(a)

Quadrotor Helicopter ($n = 12$, $\ell = 150$)

| | Extended LQR | | Iterative LQR | |
|---|---|---|---|---|
| time-step (s) | #iters | time (s) | #iters | time (s) |
| $\tau = 1/40$ | 9.39 | 0.33 | 27.7 | 0.48 |
| $\tau = 1/30$ | 12.9 | 0.46 | 37.9 | 0.66 |
| $\tau = 1/20$ | 17.61 | 0.63 | 50.08 | 0.86 |
| $\tau = 1/10$ | 24.22 | 0.87 | 92.66 | 1.59 |
| $\tau \to 0.070$ | 20.85 | 0.81 | N/A | |

(b)

**Table 1** (a): Results of the simulations of Section 7.1 with a differential-drive robot. (b): Results of the simulations of Section 7.2 with a quadrotor helicopter, averaged over 100 queries.

far above the nominal. For $\tau = \frac{1}{6}$s, the robot has 25s to reach the goal, and we see a speed close to the nominal and a trajectory that takes a safer margin with respect to the obstacles. For $\tau = \frac{1}{4}$s, on the other hand, the robot has as much as 37.5s to reach the goal, while it still wants to keep driving at a speed of 0.25m/s. The result is that the robot takes a long detour, while maintaining a speed slightly below the nominal.

The above results clearly show why temporal optimization can be useful. The results of temporal optimization are shown the bottom row of Table 1(a). In this case, the resulting trajectory is visually indistinguishable from the trajectory resulting from $\tau = 1/6$s (Fig. 2(b)) and the resulting time-step (0.168s) is only slightly longer to allow for an average speed of exactly the nominal. The results suggest that including temporal optimization does not significantly affect the convergence rate of Extended LQR. The computation time slightly increases, which reflects the higher dimension of the state after adding the time-step.

We also successfully executed the control policy resulting from these experiments on a physical iRobot Create (see Fig. 1) in a laboratory with motion capture for state estimation. This shows that the control policy can account for disturbances to which a real robot is inherently subject, despite the fact that Extended LQR does not specifically take into account motion uncertainty.

## *7.2 Quadrotor Helicopter in 3-D Environment*

Our second experiment considers a simulated quadrotor helicopter, modeled after the Ascending Technologies' ResearchPilot. Its state $\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T, \mathbf{r}^T, \mathbf{w}^T]^T$ is 12-dimensional, and consists of its position $\mathbf{p}$ (m), velocity $\mathbf{v}$ (m/s), orientation $\mathbf{r}$ (rotation about axis $\mathbf{r}$ by angle $\|\mathbf{r}\|$ (rad)), and angular velocity $\mathbf{w}$ (rad/s). Its control input $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ (N) consists of the forces exerted by each of the four rotors. The dynamics are non-linear, and given by:

$$
\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{v}, \\
\dot{\mathbf{v}} &= -g\mathbf{e}_3 + ((u_1 + u_2 + u_3 + u_4)\exp([\mathbf{r}])\mathbf{e}_3 - k_v\mathbf{v})/m, \\
\dot{\mathbf{r}} &= \mathbf{w} + \tfrac{1}{2}[\mathbf{r}]\mathbf{w} + (1 - \tfrac{1}{2}\|\mathbf{r}\|/\tan(\tfrac{1}{2}\|\mathbf{r}\|))[\mathbf{r}]^2\mathbf{w}/\|\mathbf{r}\|^2, \\
\dot{\mathbf{w}} &= J^{-1}(\rho(u_2 - u_4)\mathbf{e}_1 + \rho(u_3 - u_1)\mathbf{e}_2 + k_m(u_1 - u_2 + u_3 - u_4)\mathbf{e}_3 - [\mathbf{w}]J\mathbf{w}),
\end{aligned}
$$

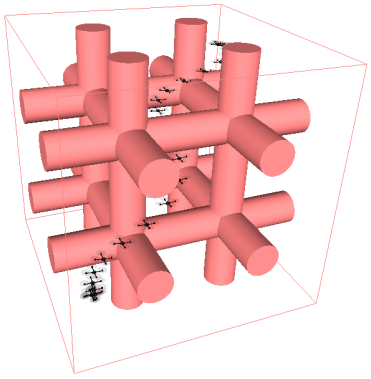
where $\mathbf{e}_i$ are the standard basis vectors, $g = 9.8\text{m/s}^2$ is the gravity, $k_v = 0.15$ is a constant relating the velocity to an opposite force (caused by rotor drag and induced inflow), $m = 0.5\text{kg}$ is the mass, $J = 0.05I$ (kg m$^2$) is the moment of inertia matrix, $\rho = 0.17\text{m}$ is the distance between the center of mass and the center of the rotors, and $k_m = 0.025$ is a constant relating the force of a rotor to its torque. The notation $[\mathbf{a}]$ refers to the skew-symmetric cross-product matrix of $\mathbf{a}$. We used the same local cost functions as in Sec. 7.1, where the nominal control input $\mathbf{u}^\star$ was set to $\frac{1}{4}mg$ N for each of the rotors, which is the force required to let the quadrotor hover.

We experimented in the 3-D environment of Fig. 3(a) measuring 6m by 6m by 6m for varying initial and target states and time-steps, and compared the performance of Extended LQR to iLQR. Neither algorithm was initialized with a given trajectory, but iLQR was implemented with line search. In all simulations we used a fixed number of steps $\ell = 150$, and modeled the geometry of the quadrotor as a sphere with a radius of 0.3m. The initial state $\mathbf{x}_0^\star$ was set to $(\mathbf{p}, \mathbf{0}, \mathbf{0}, \mathbf{0})$, where initial position $\mathbf{p}$ was randomly sampled from the edges of the environment. The target state $\mathbf{x}_\ell^\star = -\mathbf{x}_0^\star$ was set to the antipodal point in the environment. Due to the multiple homotopy classes in the environment, Extended LQR and iLQR often converge to different local optima. Therefore, we averaged the computation time and the number of iterations for both methods over the same 100 random queries for each value of the time-step, which we let range from $\tau = \frac{1}{40}\text{s}$ to $\tau = \frac{1}{10}\text{s}$. Fig. 3(b) graphs the quantitative results, and Table 1(b) gives actual numbers for part of the experiments.

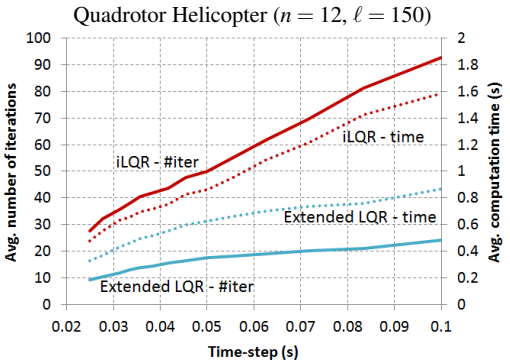
These results suggest that Extended LQR requires on average about a factor 3 less iterations than iLQR. However, per iteration, Extended LQR requires about twice the computation time of iLQR (35ms vs. 17ms). This is not surprising, as Extended LQR relinearizes and requadratizes in both the backward and the forward pass, whereas iLQR only does so in its backward pass. Combining these effects, the performance gain of Extended LQR over iLQR is about a factor 1.5. The graph also shows that the number of iterations required increases as the time-step (and hence the duration of the trajectory) increases. This is true for both Extended LQR and iLQR, although our results suggest that this effect is stronger for iLQR. The cause of this is not entirely clear; an intuitive explanation may be that the "well" of the cost-potential is less "deep" for longer trajectories, as there is less hurry to arrive at the goal. For the quadrotor simulations Extended LQR required about 18 times more computation time per iteration than for the differential-drive robot simulations, reflecting the trebling of the dimension $n$ of the state.

We also ran Extended LQR with temporal optimization on the same 100 queries (see the bottom row of Table 1(b); an example trajectory is shown in Fig. 3(a)). In this case the resulting time-step was on average 0.070s, and it took on average 20.85 iterations until convergence. This is consistent with the number of iterations required for a fixed time-step of 0.07s (see Fig. 3(b)), confirming that temporal optimization does not negatively affect the performance of Extended LQR.

Overall, we observed that the local optimum Extended LQR converges to is relatively sensitive to the parameter settings. This is because we did not seed our approach with an initial trajectory, and in the first few iterations the minimum-cost-states "bounce around" relatively unpredictably. In most cases Extended LQR con-



(a) $\tau \to 0.080$s




(b)

**Fig. 3** (a): A trajectory resulting from the quadrotor helicopter experiments of Section 7.2 with temporal optimization. The robot is shown every half second. For videos of our experiments, see **http://arl.cs.utah.edu/research/extendedlqr/**. (b): Chart indicating the relative performance of Extended LQR (blue lines) and iLQR (red lines) for varying values of the time-step in terms of computation time (dotted lines) and number of iterations (solid lines) averaged over 100 queries.

verged quickly, where the relative improvement typically declined by about a constant factor with each iteration. A second-order convergence rate was not observed in our experiments, for neither iLQR nor Extended LQR. This is likely the result of the way the obstacle-cost term is quadratized, in which negative second-order information is essentially "thrown away" in order to ensure positive-semidefiniteness. In all experiments, Extended LQR converged reliably without line search.

## 8 Conclusion

We presented Extended LQR, a novel approach to the non-linear, non-quadratic optimal control problem. Experiments showed that our approach converges quickly to a locally-optimal solution, outperforming iLQR, and does not require additional convergence measures for reliability. In addition, this paper introduced the novel concept of *LQR-smoothing*, which is at the foundation of Extended LQR. We also showed that Extended LQR can naturally be applied to temporal optimization problems. We made source code of Extended LQR publicly available for download at **http://arl.cs.utah.edu/research/extendedlqr/**.

We have presented our approach for deterministic dynamics, implicitly relying on the independence of the optimal LQR solution to the process noise variance. An interesting question for future work is whether our approach can be extended for the risk-sensitive control problem or the stochastic optimal control problem with state and control input-dependent process noise. In [25] and [20] it is shown that LQR can naturally be extended for such settings. Our approach would require a formulation of the *inverse* stochastic dynamics, which seems to be the main challenge.

Potential application domains of Extended LQR include *optimal kinodynamic motion planning*, where Extended LQR could serve as a local planner in RRT* [10]

or as part of LQR-trees [18], and *belief space planning*, where Extended LQR could improve upon the iLQR-based approach of [23].

## References


1. Y. Bar-Shalom, R. Li, T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*, Wiley-Interscience, 2004.
2. B. Bell. The iterated Kalman smoother as a Gauss-Newton method. *SIAM Journal on Optimization* 4(3):626–636, 1994.
3. J. Betts. *Practical methods for optimal control and estimation using nonlinear programming*, vol. 19, SIAM, 2009.
4. D. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2001.
5. A. Björck. *Numerical methods for least squares problems*. SIAM Philadelphia, 1996.
6. M.-S. Chen and C.-Y. Kao. Control of linear time-varying systems using forward Riccati equation. *Journal of Dynamic Systems, Measurement, and Control* 119(3):536540, 1997.
7. Y. Fujita, Y. Nakamura, Z. Shiller. Dual Dijkstra search for paths with different topologies. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2003.
8. N. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications* 103:103–118, 1988.
9. D. Jacobsen, D. Mayne. *Differential Dynamic Programming*. Elsevier New York, 1970.
10. S. Karaman, E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research* 30(7):846–894, 2011.
11. S. Lavalle. *Planning Algorithms*. Cambridge University Press, 2006.
12. W. Li, E. Todorov. Iterative linear-quadratic regulator design for nonlinear biological movement systems. *Proc. Int. Conf. on Informatics in Control, Automation and Robotics*, 2004.
13. J. Nocedal, S. Wright. *Numerical Optimization*. Springer Science+ Business Media, 2006.
14. H. Rauch, F. Tung, C. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal* 3(8):1445-1450, 1965.
15. K. Rawlik, M. Toussaint, S. Vijayakumar. An approximate inference approach to temporal optimization in optimal control. *Advances in Neural Information Processing Systems*, 2010.
16. K. Rawlik, M. Toussaint, S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proc. Robotics: Science and Systems*, 2012.
17. J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. *Robotics: Science and Systems*, 2013.
18. R. Tedrake, I. Manchester, M. Tobenkin, J. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *Int. Journal of Robotics Research* 29(8):1038–1052, 2010.
19. E. Theodorou, Y. Tassa, E. Todorov. Stochastic differential dynamic programming. *Proc. American Control Conference*, 2010.
20. E. Todorov, W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *Proc. American Control Conference*, 2005.
21. E. Todorov. General duality between optimal control and estimation. *Proc. IEEE Conf. on Decision and Control*, 2008.
22. M. Toussaint. Robot trajectory optimization using approximate inference. *Proc. Int. Conf. on Machine Learning*, 2009.
23. J. van den Berg, S. Patil, R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *Int. Journal of Robotics Research* 31(11):1263–1278, 2012.
24. A. Weiss, I. Kolmanovsky, D. Bernstein. Forward-integration Riccati-based output-feedback control of linear time-varying systems. *American Control Conference*, 2012.
25. P. Whittle. Risk-sensitive linear/quadratic/Gaussian control. *Advances in Applied Probability* 13(4):764–777, 1981.
26. S. Yakowitz. Algorithms and computational techniques in differential dynamic programming. *Control and Dynamic Systems* 31:75-91, 1989.
27. M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. Bagnell, S. Srinivasa. CHOMP: Covariant Hamiltonian optimization for motion planning. *Int. Journal of Robotics Research*, 2013.