2014 American Control Conference (ACC)
June 4-6, 2014. Portland, Oregon, USA

Iterated LQR Smoothing for Locally-Optimal Feedback Control of Systems with Non-Linear Dynamics and Non-Quadratic Cost

Jur van den Berg

***Abstract*— This paper introduces the novel concept of *LQR smoothing*, which analogous to Kalman smoothing consists of both a backward pass and a forward pass. In the backward pass the *cost-to-go* function is computed using the standard LQR Riccati equation that runs backward in time, and in the forward pass the *cost-to-come* function is computed using a Riccati equation that runs forward in time. The sum of the cost-to-go and the cost-to-come function gives the *total-cost* function, and we will show that the states for which the total-cost function is minimal constitute the minimum-cost trajectory for the linear-quadratic optimal control problem. This insight is used to construct a fast-converging iterative procedure to compute a locally-optimal feedback control policy for systems with non-linear dynamics and non-quadratic cost, where in each iteration the current minimal-total-cost states provide natural points about which the dynamics can be linearized and the cost quadratized. We demonstrate the potential of our approach on two illustrative non-linear control problems involving physical differential-drive robots and simulated quadrotor helicopters in environments with obstacles, and show that our approach converges in only about a third of the number of iterations required by existing approaches such as Iterative LQR.**

I. Introduction

Optimal control is a fundamental problem in many application domains, and has a long and rich history as a topic of research (see, e.g., [5], [11]). The problem is generally defined in terms of a description of the system's dynamics and a control objective in the form of a cost index that is to be minimized, and the goal is to compute an optimal feedback control policy that tells the system what control to apply given the state it is in. Only for specific instances of the problem, however, closed-form optimal solutions can be obtained. In particular, when the dynamics of the system are linear and the cost index is quadratic, the linear-quadratic regulator (LQR) provides a closed-form optimal solution.

The LQR controller is closely related to the Kalman filter for linear-Gaussian state estimation; in fact, they are each other's dual [20]. Since the Kalman filter is naturally applied to non-linear systems by continually linearizing the dynamics about the current-best estimate of the state (this is called the Extended Kalman filter [1]), a natural question is whether LQR can be extended in a similar fashion to general non-linear dynamics and non-quadratic cost functions [24]. The main challenge here is that the Ricatti equations of LQR, in constrast to those of the Kalman filter, run *backward* in time, and that at the time of designing the controller it is unknown what the future states of the system will be. This makes choosing suitable points to linearize the dynamics and quadratize the cost functions about non-trivial.

Jur van den Berg is with the School of Computing at the University of Utah. E-mail: berg@cs.utah.edu.

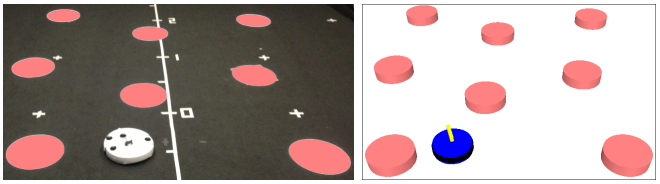

Fig. 1. A physical and simulated iRobot Create navigating an environment with obstacles using iterated LQR smoothing. For videos and source code of our experiments, see **http://arl.cs.utah.edu/research/lqrsmoothing/**.

To address this challenge, we introduce the novel concept of *LQR smoothing*. The LQR smoother is the dual of the Kalman smoother [14], and consists of both a backward pass and a forward pass. In the backward pass the *cost-to-go* function is computed using the standard LQR Riccati equations that run backward in time, and in the *forward* pass the *cost-to-come* function is computed using Riccati equations that run forward in time. The sum of these functions give the *total-cost* function, and we will show that the states for which the total-cost function is minimal constitute the *minimum-cost trajectory* for the linear-quadratic control problem.

This insight is used to construct a locally-optimal controller for systems with non-linear dynamics and non-quadratic cost, by iteratively performing both a backward "extended" LQR pass and a forward "extended" LQR pass (analogous to the iterated extended Kalman smoother [2]) to progressively obtain a better idea of the system's future trajectory. The states for which the approximate total-cost function is minimal are used to linearize the dynamics and quadratize the cost function 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 been proposed previously (as we discuss in more detail in Section II). We consider the main contribution of this paper to be the introduction of a novel concept (LQR smoothing) to address the problem, but we will also show that our approach improves quantitatively upon existing methods such as Iterative LQR (iLQR) [12]. Our formulation remains strictly within the LQR framework and does not explicitly use the duality between control and estimation [20] (in contrast to e.g. Approximate Inference Control (AICO) [21]), resulting in

978-1-4799-3274-0/$31.00 ©2014 AACC

a conceptually intuitive approach that is easy to implement.

For mathematical clarity, we introduce our approach in the context of the continuous-time formulation of the optimal control problem, but our results apply equally to the discrete-time variant (we refer the reader to [23] for details on the discrete-time derivation). We perform experiments using a discrete-time implementation of our approach on two illustrative non-linear control problems, and compared performance to iLQR. Our 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. The results suggest that our approach based on LQR smoothing converges more quickly and reliably than iLQR even without providing it with an initial trajectory or implementing special convergence measures. On average, our approach converges in only about a third of the number of iterations required by iLQR. We have made source code of our approach publicly available at **http://arl.cs.utah.edu/research/lqrsmoothing/**.

The remainder of this paper is organized as follows. We discuss related work in Section II and formally define the problem in Section III. In Section IV we introduce the concept of LQR smoothing, which is used in Section V to develop our iterated smoothing approach. We discuss experimental results in Section VI and conclude in Section VII.

II. Related Work

There is a large amount of previous work on non-linear, non-quadratic control, and many approaches are based on extending linear-quadratic synthesis together with Lyapunov analysis to ensure stability (see e.g. [4], [17]). A particularly popular and relatively recent approach is Iterative LQR (iLQR) [12], [19], which is a simplified version of Differential Dynamic Programming (DDP) [9], [18]. This approach linearizes the dynamics and quadratizes the cost function about a given (dynamically feasible) nominal trajectory and then uses 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. The approach requires special measures such as line search [26] to ensure convergence, as the control policies may drive a new trajectory too far from where the LQ-approximation is valid. Our approach is related but conceptually different, and relinearizes/requadratizes in both the backward pass and the forward pass, about a trajectory that is dynamically feasible only upon convergence. We will show that as a result, iterated LQR smoothing converges reliably without special convergence measures and requires only about a third of the number of iterations compared to iLQR.

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 [16], [27] are closely related, and focus on *trajectory optimization* among obstacles for the specific class of robots with holonomic dynamics. Our approach 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) [21] uses the duality between control and estimation [20], and formulates the optimal control problem in terms of Kullback-Leibler divergence minimization [15]. Even though the derivation of our approach 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. Our approach is hence applicable to general non-quadratic cost functions without requiring local iterations.

One of the concepts underpinning our approach is *Forward LQR*, which is based on Riccati equations that run forward in time. Forward Riccati equations have previously been explored in optimal control [7], [24]. However, these works do not use it to compute a *cost-to-come* function or as part of an *LQR-smoothing* framework.

III. Problem Definition

We consider the optimal control problem with general non-linear dynamics as given by a function $\mathbf{f}$:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{1}$$

where $\mathbf{x}(t)$ is the state of the system at time t, and $\mathbf{u}(t)$ is the control input of the system at time t. The cost index to be minimized is:

$$c_0(\mathbf{x}(0)) + c_\ell(\mathbf{x}(\ell)) + \int_0^\ell c(\mathbf{x}(t), \mathbf{u}(t), t)\, \mathrm{d}t, \tag{2}$$

where ℓ is the final time, and c_0, c_ℓ, and c are given non-quadratic initial, final, and local cost functions, respectively, for which $\frac{\partial^2 c_0}{\partial \mathbf{x} \partial \mathbf{x}} \geq 0$, $\frac{\partial^2 c_\ell}{\partial \mathbf{x} \partial \mathbf{x}} \geq 0$, $\frac{\partial^2 c}{\partial [\begin{smallmatrix}\mathbf{x}\\ \mathbf{u}\end{smallmatrix}] \partial [\begin{smallmatrix}\mathbf{x}\\ \mathbf{u}\end{smallmatrix}]} \geq 0$, and $\frac{\partial^2 c}{\partial \mathbf{u} \partial \mathbf{u}} > 0$. We specify the cost index with an initial cost c_0 such that the control problem has a well defined minimum-cost trajectory in absence of a given fixed initial state $\mathbf{x}(0)$.

The goal is to compute an optimal feedback control policy $\boldsymbol{\pi}$, such that applying controls

$$\mathbf{u}(t) = \boldsymbol{\pi}(\mathbf{x}(t), t) \tag{3}$$

minimizes the cost index of Eq. (2).

IV. LQR Smoothing

In this section we introduce the concept of *LQR Smoothing*. The LQR smoother provides a minimum-cost state trajectory for the optimal control problem in case the dynamics are linear and the local cost function is quadratic. It

consists of both a backward pass and a forward pass. In the backward pass the cost-to-go function is computed using a standard LQR Riccati equation that runs backward in time (as reviewed in Section IV-A), and in the forward pass the *cost-to-come* function is computed using a Riccati equation that runs forward in time (as introduced in Section IV-B). The sum of the cost-to-go function and the cost-to-come function forms the *total-cost* function, whose minima define the minimum-cost trajectory for the linear-quadratic control problem (as proven in Section IV-C). The LQR smoother provides the foundation for our iterative approach to the non-linear, non-quadratic control problem, which we discuss in Section V.

In this section, we assume that the dynamics of Eq. (1) are linear and given by:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}, t) = A(t)\mathbf{x} + B(t)\mathbf{u} + \mathbf{c}(t), \tag{4}$$

and that the initial, final, and local cost functions that constitute the cost index of Eq. (2) are quadratic and given by:

$$c_0(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T Q_0 \mathbf{x} + \mathbf{x}^T \mathbf{q}_0 + q_0, \tag{5}$$

$$c_\ell(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T Q_\ell \mathbf{x} + \mathbf{x}^T \mathbf{q}_\ell + q_\ell, \tag{6}$$

$$\begin{aligned} c(\mathbf{x}, \mathbf{u}, t) = {} & \tfrac{1}{2}\mathbf{x}^T Q(t)\mathbf{x} + \tfrac{1}{2}\mathbf{u}^T R(t)\mathbf{u} + \\ & \mathbf{u}^T P(t)\mathbf{x} + \mathbf{x}^T \mathbf{q}(t) + \mathbf{u}^T \mathbf{r}(t) + q(t), \end{aligned} \tag{7}$$

where Q_0, Q_ℓ, and $\left[\begin{smallmatrix} Q(t) & P(t)^T \\ P(t) & R(t) \end{smallmatrix}\right]$ are positive-semidefinite, and $R(t)$ is positive-definite.

A. LQR and Cost-To-Go

For the linear-quadratic control problem as defined by Eqs. (4)-(7), the LQR controller provides the optimal feedback control policy in closed form [11]. It computes a *cost-to-go* function v with the explicit quadratic form;

$$v(\mathbf{x}, t) = \mathbf{x}^T S(t)\mathbf{x} + \mathbf{x}^T \mathbf{s}(t) + s(t), \tag{8}$$

which gives the minimal cost that is accrued between time t and time ℓ by a trajectory that starts in $\mathbf{x}$ at time t. The positive-semidefinite matrix $S(t)$, vector $\mathbf{s}(t)$, and scalar $s(t)$ are given by Riccati differential equations that run backward in time:

$$\begin{aligned} -\dot{S} = {} & Q - P^T R^{-1} P + S(A - BR^{-1}P) + \\ & (A - BR^{-1}P)^T S - SBR^{-1}B^T S, \end{aligned} \tag{9}$$

$$\begin{aligned} -\dot{\mathbf{s}} = {} & \mathbf{q} - P^T R^{-1}\mathbf{r} + S(\mathbf{c} - BR^{-1}\mathbf{r}) + \\ & (A - BR^{-1}P)^T \mathbf{s} - SBR^{-1}B^T \mathbf{s}, \end{aligned} \tag{10}$$

$$\begin{aligned} -\dot{s} = {} & q - \tfrac{1}{2}\mathbf{r}^T R^{-1}\mathbf{r} + \\ & \mathbf{s}^T(\mathbf{c} - BR^{-1}\mathbf{r}) - \tfrac{1}{2}\mathbf{s}^T BR^{-1}B^T \mathbf{s}, \end{aligned} \tag{11}$$

where the dependencies on time t have been suppressed for readability, and the boundary conditions are given by:

$$S(\ell) = Q_\ell, \qquad \mathbf{s}(\ell) = \mathbf{q}_\ell, \qquad s(\ell) = q_\ell. \tag{12}$$

The optimal feedback control policy $\boldsymbol{\pi}$ has an explicit linear form, and is given by:

$$\boldsymbol{\pi}(\mathbf{x}, t) = -R^{-1}(B^T S + P)\mathbf{x} - R^{-1}(B^T \mathbf{s} + \mathbf{r}), \tag{13}$$

where again dependencies on time t have been suppressed.

B. Forward LQR and Cost-To-Come

Analogous to the cost-to-go function v, one can also define a *cost-to-come* function $\bar{v}(\mathbf{x}, t)$ that gives the minimal cost that is accrued between time 0 and time t by a trajectory that *arrives* in $\mathbf{x}$ at time t. For the linear-quadratic control problem, the cost-to-come function has, as the cost-to-go function, an explicit quadratic form:

$$\bar{v}(\mathbf{x}, t) = \mathbf{x}^T \bar{S}(t)\mathbf{x} + \mathbf{x}^T \bar{\mathbf{s}}(t) + \bar{s}(t), \tag{14}$$

where $\bar{S}(t)$, $\bar{\mathbf{s}}(t)$, and $\bar{s}(t)$ are given by differential equations that run *forward* in time:

$$\begin{aligned} \dot{\bar{S}} = {} & Q - P^T R^{-1} P - \bar{S}(A - BR^{-1}P) - \\ & (A - BR^{-1}P)^T \bar{S} - \bar{S}BR^{-1}B^T \bar{S}, \end{aligned} \tag{15}$$

$$\begin{aligned} \dot{\bar{\mathbf{s}}} = {} & \mathbf{q} - P^T R^{-1}\mathbf{r} - \bar{S}(\mathbf{c} - BR^{-1}\mathbf{r}) - \\ & (A - BR^{-1}P)^T \bar{\mathbf{s}} - \bar{S}BR^{-1}B^T \bar{\mathbf{s}}, \end{aligned} \tag{16}$$

$$\begin{aligned} \dot{\bar{s}} = {} & q - \tfrac{1}{2}\mathbf{r}^T R^{-1}\mathbf{r} - \\ & \bar{\mathbf{s}}^T(\mathbf{c} - BR^{-1}\mathbf{r}) - \tfrac{1}{2}\bar{\mathbf{s}}^T BR^{-1}B^T \bar{\mathbf{s}}, \end{aligned} \tag{17}$$

with boundary conditions:

$$\bar{S}(0) = Q_0, \qquad \bar{\mathbf{s}}(0) = \mathbf{q}_0, \qquad \bar{s}(0) = q_0. \tag{18}$$

In addition, one can define an optimal *inverse* control policy $\bar{\boldsymbol{\pi}}$ that if applied backwards in time from a state $\mathbf{x}$ at time t will trace back the minimum-cost trajectory to arrive in $\mathbf{x}$ at time t. The optimal inverse control policy $\bar{\boldsymbol{\pi}}$ has, as the optimal control policy $\boldsymbol{\pi}$, an explicit linear form:

$$\bar{\boldsymbol{\pi}}(\mathbf{x}, t) = R^{-1}(B^T \bar{S} - P)\mathbf{x} + R^{-1}(B^T \bar{\mathbf{s}} - \mathbf{r}). \tag{19}$$

Note that Eqs. (15), (16), (17), and (19) are of a similar form as Eqs. (9), (10), (11), and (13), respectively, with the difference that the sign of A, B, and $\mathbf{c}$ has flipped.

C. LQR Smoothing and Total-Cost

The *sum* of the cost-to-go function $v(\mathbf{x}, t)$ and the cost-to-come function $\bar{v}(\mathbf{x}, t)$ defines a *total-cost* function $\hat{v}(\mathbf{x}, t)$ that gives the minimal cost that is accrued between time 0 and time ℓ by a trajectory that *visits* $\mathbf{x}$ at time t:

$$\begin{aligned} \hat{v}(\mathbf{x}, t) &= v(\mathbf{x}, t) + \bar{v}(\mathbf{x}, t) \\ &= \tfrac{1}{2}\mathbf{x}^T(S(t) + \bar{S}(t))\mathbf{x} + \mathbf{x}^T(\mathbf{s}(t) + \bar{\mathbf{s}}(t)) + s(t) + \bar{s}(t). \end{aligned} \tag{20}$$

Let $\hat{\mathbf{x}}(t)$ denote the state for which the total-cost function $\hat{v}$ at time t is minimal:

$$\begin{aligned} \hat{\mathbf{x}}(t) &= \operatorname{argmin}_{\mathbf{x}} \ \hat{v}(\mathbf{x}, t) \\ &= -(S(t) + \bar{S}(t))^{-1}(\mathbf{s}(t) + \bar{\mathbf{s}}(t)), \end{aligned} \tag{21}$$

then the states $\hat{\mathbf{x}}(t)$ for $t \in [0, \ell]$ define the *minimum-cost trajectory* for the linear-quadratic optimal control problem of Eqs. (4)-(7):

Lemma 1 *The minimum-cost trajectory for the linear-quadratic optimal control problem of Eqs.* (4)-(7) *is given by the function* $\hat{\mathbf{x}}(t)$ *as defined in Eq.* (21).

To prove this, we show that $\hat{\mathbf{x}}(t)$ is consistent with the dynamics of Eq. (4) under both the optimal control policy and the optimal inverse control policy, i.e.:

$$\begin{aligned}\dot{\hat{\mathbf{x}}}(t) &= A(t)\hat{\mathbf{x}}(t) + B(t)\boldsymbol{\pi}(\hat{\mathbf{x}}(t), t) + \mathbf{c}(t) \\ &= A(t)\hat{\mathbf{x}}(t) + B(t)\bar{\boldsymbol{\pi}}(\hat{\mathbf{x}}(t), t) + \mathbf{c}(t). \end{aligned} \quad (22)$$

Indeed, for the minimum-total-cost states $\hat{\mathbf{x}}(t)$, the optimal control policy and the optimal inverse control policy give the same controls. The proofs are given in the appendix.

Our approach 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) state trajectory. In fact, it can be shown that the LQR smoother is the exact dual of the Kalman smoother [20]. We use the insights of Lemma 1 below to develop an iterative procedure to compute a locally-optimal feedback control policy for the non-linear, non-quadratic optimal control problem.

V. Iterated Extended LQR Smoothing

The challenge of extending LQR to non-linear, non-quadratic systems similar to how an Extended Kalman filter extends the Kalman filter to systems with non-linear dynamics and measurement models lies in the fact that it is not trivial to select states and control inputs about which to linearize the dynamics and quadratize the local cost functions. Our idea to address this challenge is to iteratively perform backward and forward "extended" LQR passes to compute increasingly better approximations of the total-cost functions, analogous to the iterated extended Kalman smoother [2]. In each pass, we linearize the dynamics and quadratize the cost function about the states for which the current approximation of the total-cost function is minimal, and about the control inputs as given by the control policy from the preceding pass.

The iteration continues until convergence, i.e. when the minimum-total-cost trajectory no longer changes. This trajectory then provides a locally-optimal state trajectory for the non-linear, non-quadratic control problem, and the control policy computed by the last backward pass provides an approximately optimal feedback control policy. The iteration may start with either a backward pass or a forward pass. Here, we discuss the backward pass first, then discuss the forward pass, and finally discuss convergence properties.

A. Backward Extended LQR Pass

We assume during the backward pass that the cost-to-come function, as defined by $\bar{S}(t)$ and $\bar{\mathbf{s}}(t)$ (in the remainder, we ignore the scalar terms of all cost functions, as they do not influence the control policy), and the inverse control policy $\bar{\boldsymbol{\pi}}$ are available from the preceding forward pass. Also, an initial quadratization point $\hat{\mathbf{x}}_\ell$ for the final cost function is available from the forward pass. If this is the first backward pass (and no forward pass preceded it), one can assume $\bar{S}(t) = 0$, $\bar{\mathbf{s}}(t) = \mathbf{0}$, $\bar{\boldsymbol{\pi}}(\mathbf{x}, t) = \mathbf{0}$, 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 objective of the backward LQR pass is to compute the cost-to-go function as defined by $S(t)$ and $\mathbf{s}(t)$, and the control policy $\boldsymbol{\pi}$. For this, we use exactly the same equations as Eqs. (9), (10), (12), and (13), but where the dynamics $\mathbf{f}$ and cost functions c_ℓ and c are put in the linear and quadratic forms of Eqs. (4), (6), and (7) through linearization and quadratization, respectively.

More specifically, we initialize the backward pass by setting $S(\ell)$ and $\mathbf{s}(\ell)$ as in Eq. (12), where Q_ℓ and $\mathbf{q}_\ell$ are obtained by quadratizing the final cost function c_ℓ about the given point $\hat{\mathbf{x}}_\ell$. This gives:

$$Q_\ell = \frac{\partial^2 c_\ell}{\partial \mathbf{x} \partial \mathbf{x}}(\hat{\mathbf{x}}_\ell), \qquad \mathbf{q}_\ell = \frac{\partial c_\ell}{\partial \mathbf{x}}(\hat{\mathbf{x}}_\ell) - Q_\ell \hat{\mathbf{x}}_\ell. \quad (23)$$

We then proceed by integrating Eqs. (9) and (10) backward in time to compute $S(t)$ and $\mathbf{s}(t)$, where $A(t)$, $B(t)$, and $\mathbf{c}(t)$ are obtained by linearizing the dynamics $\mathbf{f}$;

$$A(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t), \;\; B(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t), \quad (24)$$

$$\mathbf{c}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t) - A(t)\hat{\mathbf{x}}(t) - B(t)\hat{\mathbf{u}}(t), \quad (25)$$

and where $Q(t)$, $P(t)$, $R(t)$, $\mathbf{q}(t)$, and $\mathbf{r}(t)$ are obtained by quadratizing the local cost function c:

$$\begin{bmatrix} Q(t) & P(t)^T \\ P(t) & R(t) \end{bmatrix} = \frac{\partial^2 c}{\partial [\begin{smallmatrix}\mathbf{x}\\ \mathbf{u}\end{smallmatrix}] \partial [\begin{smallmatrix}\mathbf{x}\\ \mathbf{u}\end{smallmatrix}]}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t), \quad (26)$$

$$\begin{bmatrix} \mathbf{q}(t) \\ \mathbf{r}(t) \end{bmatrix} = \frac{\partial c}{\partial [\begin{smallmatrix}\mathbf{x}\\ \mathbf{u}\end{smallmatrix}]}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t) - \begin{bmatrix} Q(t) & P(t)^T \\ P(t) & R(t) \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}(t) \\ \hat{\mathbf{u}}(t) \end{bmatrix}. \quad (27)$$

Throughout the integration, the state $\hat{\mathbf{x}}(t)$ and control input $\hat{\mathbf{u}}(t)$ about which we linearize and quadratize are set in accordance with Eq. (21) and using the inverse control policy $\bar{\boldsymbol{\pi}}$ available from the preceding forward pass:

$$\hat{\mathbf{x}}(t) = -(S(t) + \bar{S}(t))^{-1}(\mathbf{s}(t) + \bar{\mathbf{s}}(t)), \quad (28)$$

$$\hat{\mathbf{u}}(t) = \bar{\boldsymbol{\pi}}(\hat{\mathbf{x}}(t), t). \quad (29)$$

The control policy $\boldsymbol{\pi}$ is then given by Eq. (13).

Finally, we set an initial quadratization point $\hat{\mathbf{x}}_0$ for the subsequent forward pass:

$$\hat{\mathbf{x}}_0 = -(S(0) + \bar{S}(0))^{-1}(\mathbf{s}(0) + \bar{\mathbf{s}}(0)). \quad (30)$$

B. Forward Extended LQR Pass

The forward pass proceeds similarly to the backward pass. We assume during the forward pass that the cost-to-go function, as defined by $S(t)$ and $\mathbf{s}(t)$, and the control policy $\boldsymbol{\pi}$ are available from the preceding backward pass. Also, an initial quadratization point $\hat{\mathbf{x}}_0$ for the initial cost function is available from the backward pass.

The objective of the forward LQR pass is to compute the cost-to-come function as defined by $\bar{S}(t)$ and $\bar{\mathbf{s}}(t)$, and the inverse control policy $\bar{\boldsymbol{\pi}}$. For this, we use exactly the same equations as Eqs. (15), (16), (18), and (19), but where the dynamics $\mathbf{f}$ and cost functions c_0 and c are put in the linear and quadratic forms of Eqs. (4), (6), and (7) through linearization and quadratization, respectively.

More specifically, we initialize the forward pass by setting $\bar{S}(0)$ and $\bar{\mathbf{s}}(0)$ as in Eq. (18), where Q_0 and $\mathbf{q}_0$ are obtained by quadratizing the initial cost function c_0 about the given point $\hat{\mathbf{x}}_0$. This gives:

$$Q_0 = \frac{\partial^2 c_0}{\partial \mathbf{x} \partial \mathbf{x}}(\hat{\mathbf{x}}_0), \qquad \mathbf{q}_0 = \frac{\partial c_0}{\partial \mathbf{x}}(\hat{\mathbf{x}}_0) - Q_0 \hat{\mathbf{x}}_0. \tag{31}$$

We then proceed by integrating Eqs. (15) and (16) forward in time to compute $\bar{S}(t)$ and $\bar{\mathbf{s}}(t)$, where $A(t)$, $B(t)$, and $\mathbf{c}(t)$ are obtained as in Eqs. (24) and (25) by linearizing the dynamics $\mathbf{f}$, and where $Q(t)$, $P(t)$, $R(t)$, $\mathbf{q}(t)$, and $\mathbf{r}(t)$ are obtained as in Eqs. (26) and (27) by quadratizing the local cost function c. Throughout the integration, the state $\hat{\mathbf{x}}(t)$ about which to linearize and quadratize is set as in Eq. (28), and we use the control policy $\boldsymbol{\pi}$ available from the preceding backward pass to select the control input $\hat{\mathbf{u}}(t)$ to linearize and quadratize about:

$$\hat{\mathbf{u}}(t) = \boldsymbol{\pi}(\hat{\mathbf{x}}(t), t). \tag{32}$$

The inverse control policy $\bar{\boldsymbol{\pi}}$ is then given by Eq. (19).

Finally, we set an initial quadratization point $\hat{\mathbf{x}}_\ell$ for the subsequent backward pass:

$$\hat{\mathbf{x}}_\ell = -(S(\ell) + \bar{S}(\ell))^{-1}(\mathbf{s}(\ell) + \bar{\mathbf{s}}(\ell)). \tag{33}$$

C. Convergence Properties

Upon convergence, the properties of Eq. (22) hold, and the minimum-total-cost states $\hat{\mathbf{x}}(t)$ form an exact *locally-optimal* state trajectory for the non-linear, non-quadratic control problem. The (linear) control policies $\boldsymbol{\pi}(t)$ 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 [6].

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

VI. Experiments

We implemented a discrete-time variant of our approach (see [23] for details on the discrete-time variant of Iterated LQR Smoothing), and experimented on two systems; a physical iRobot Create differential-drive robot, which we use mainly for illustrative purposes, and a simulated quadrotor helicopter, on which we perform extensive quantitative analysis and performance comparison with Iterative LQR (iLQR).

A. 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 θ (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:

$$\dot{p}_x = \frac{v_\ell + v_r}{2}\cos\theta, \quad \dot{p}_y = \frac{v_\ell + v_r}{2}\sin\theta, \quad \dot{\theta} = \frac{v_r - v_\ell}{w},$$

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

We use the following cost functions in our experiments:

$$\begin{aligned} c_\ell(\mathbf{x}) &= \tfrac{1}{2}(\mathbf{x} - \mathbf{x}^\star_\ell)^T Q (\mathbf{x} - \mathbf{x}^\star_\ell), \\ c_0(\mathbf{x}) &= \tfrac{1}{2}(\mathbf{x} - \mathbf{x}^\star_0)^T Q (\mathbf{x} - \mathbf{x}^\star_0), \\ c_t(\mathbf{x}, \mathbf{u}, t) &= \tfrac{1}{2}(\mathbf{u} - \mathbf{u}^\star)^T R (\mathbf{u} - \mathbf{u}^\star) + q \textstyle\sum_i \exp(-d_i(\mathbf{x})), \end{aligned}$$

where $\mathbf{x}^\star_\ell$ is the target state, $\mathbf{x}^\star_0$ 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 ± 0.5m/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. (2), 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 $\mathbf{x}^\star_0 = (0, -2.5, \pi)$ and the target state to $\mathbf{x}^\star_\ell = (0, 2.5, \pi)$. We ran our approach for a final time of $\ell = 25$s and a time-step of $\frac{1}{6}$s. Our approach was not seeded with an initial trajectory, and we let the algorithm run until the relative improvement in cost dropped below 10^{-4}. Fig. 2 shows the resulting trajectory. It took 7 iterations until convergence, and a total of 0.012s computation time (for a C++ implementation on an Intel i5 1.60GHz with 4GB RAM).

We also successfully executed the control policy resulting from this experiment 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 our approach does not specifically take into account motion uncertainty.

B. 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

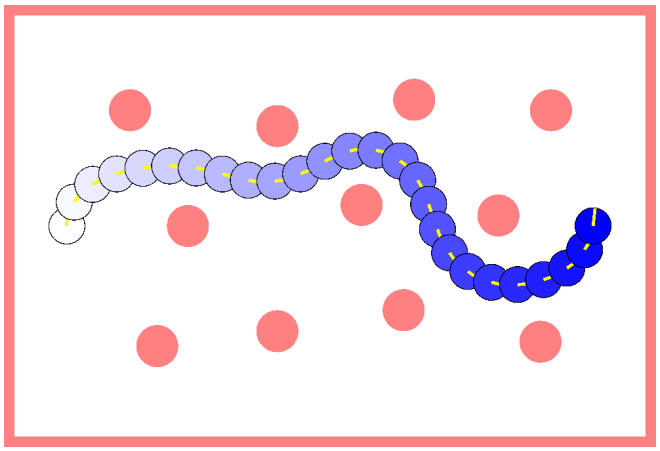

Fig. 2. Trajectory resulting from the differential-drive robot experiment. The robot is shown every second. For videos of our experiments, see **http://arl.cs.utah.edu/research/lqrsmoothing/**.

$\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ (N) consists of the forces exerted by each of the four rotors. The dynamics are non-linear, and given by:

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

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. VI-A, 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 measuring 6m by 6m by 6m for varying initial and target states, and compared the performance of our approach to iLQR. Neither algorithm was initialized with a given trajectory, but iLQR was implemented with line search. In all simulations we used a fixed final time of $\ell = 5\text{s}$ and a time step of $\frac{1}{30}\text{s}$, 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, our approach 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. Table I gives the results.

These results suggest that our approach requires on average about a third of the iterations that iLQR requires. However, per iteration, our approach requires about twice the computation time of iLQR (35ms vs. 17ms). This is not surprising, as our approach 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 our approach over iLQR is about a factor 1.5, meaning that our approach requires about two thirds of the computation time of iLQR. For the quadrotor simulations our approach required about 18 times more computation time per iteration than for the differential-drive robot simulations (when keeping other factors equal), reflecting the quadrupling of the dimension of the state.

TABLE I

RESULTS OF QUADROTOR HELICOPTER SIMULATIONS, AVERAGED OVER 100 QUERIES.

method	#iters	time (s)
Our approach	12.9	0.46
iLQR	37.9	0.66

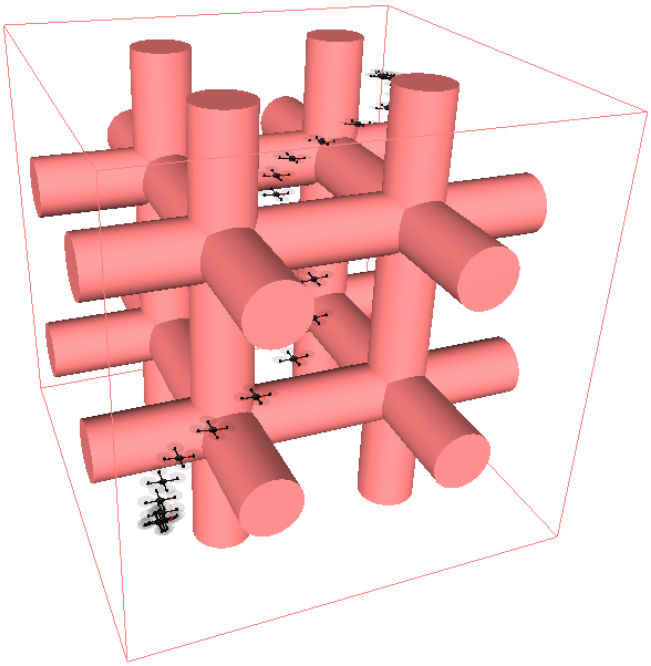

Fig. 3. A trajectory resulting from the quadrotor helicopter experiments of Section VI-B. The robot is shown every half second. For videos of our experiments, see **http://arl.cs.utah.edu/research/lqrsmoothing/**.

Overall, we observed that the local optimum our approach 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 our approach converged 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 our approach. 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, our approach

converged reliably without line search.

VII. Conclusion

We presented the novel concept of LQR smoothing, which lies at the basis of a new 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. We made source code of our approach publicly available for download at **http://arl.cs.utah.edu/research/lqrsmoothing/**.

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 stochastic optimal control problem with state and control input-dependent process noise. In [19] 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 our approach include *optimal kinodynamic motion planning*, where our approach could serve as a local planner in RRT* [10], [25], and *belief space planning*, where our approach could improve upon the iLQR-based approach of [22].

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. Bernstein. Nonquadratic cost and nonlinear feedback control. *Int. Journal of Robust and Nonlinear Control* 3(3):211-229, 1993.

[5] D. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2001.

[6] A. Björck. *Numerical methods for least squares problems*. SIAM Philadelphia, 1996.

[7] 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.

[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] F. Lewis, V. Syrmos. *Optimal Control*. John Wiley & Sons, New York, 1995.

[12] W. Li, E. Todorov. Iterative linear-quadratic regulator design for non-linear 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. On stochastic optimal control and reinforcement learning by approximate inference. *Proc. Robotics: Science and Systems*, 2012.

[16] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, P. Abbeel. Finding locally optimal, collisi-on-free trajectories with sequential convex optimization. *Robotics: Science and Systems*, 2013.

[17] 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.

[18] E. Theodorou, Y. Tassa, E. Todorov. Stochastic differential dynamic programming. *Proc. American Control Conference*, 2010.

[19] E. Todorov, W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *Proc. American Control Conference*, 2005.

[20] E. Todorov. General duality between optimal control and estimation. *Proc. IEEE Conf. on Decision and Control*, 2008.

[21] M. Toussaint. Robot trajectory optimization using approximate inference. *Proc. Int. Conf. on Machine Learning*, 2009.

[22] 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.

[23] J. van den Berg. Extended LQR: locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost. *Proc. Int. Symp. on Robotics Research*, 2013.

[24] A. Weiss, I. Kolmanovsky, D. Bernstein. Forward-integration Riccati-based output-feedback control of linear time-varying systems. *American Control Conference*, 2012.

[25] D. Webb, J. van den Berg. Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2013.

[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.

Appendix

A. Proof that $\boldsymbol{\pi}(\hat{\mathbf{x}}(t),t) = \bar{\boldsymbol{\pi}}(\hat{\mathbf{x}}(t),t)$.

The dependencies on time are suppressed for readability.

$$
\begin{aligned}
&\boldsymbol{\pi}(\hat{\mathbf{x}},t)\\
&= -R^{-1}B^T S\hat{\mathbf{x}} - R^{-1}B^T\mathbf{s} - R^{-1}(P\hat{\mathbf{x}}+\mathbf{r})\\
&= -R^{-1}B^T(S+\bar{S})\hat{\mathbf{x}} + R^{-1}B^T\bar{S}\hat{\mathbf{x}} -\\
&\quad R^{-1}B^T(\mathbf{s}+\bar{\mathbf{s}}) + R^{-1}B^T\bar{\mathbf{s}} - R^{-1}(P\hat{\mathbf{x}}+\mathbf{r})\\
&= R^{-1}B^T(S+\bar{S})(S+\bar{S})^{-1}(\mathbf{s}+\bar{\mathbf{s}}) + R^{-1}B^T\bar{S}\hat{\mathbf{x}} -\\
&\quad R^{-1}B^T(\mathbf{s}+\bar{\mathbf{s}}) + R^{-1}B^T\bar{\mathbf{s}} - R^{-1}(P\hat{\mathbf{x}}+\mathbf{r})\\
&= R^{-1}B^T\bar{S}\hat{\mathbf{x}} + R^{-1}B^T\bar{\mathbf{s}} - R^{-1}(P\hat{\mathbf{x}}+\mathbf{r})\\
&= \bar{\boldsymbol{\pi}}(\hat{\mathbf{x}},t). \qquad (34)
\end{aligned}
$$

B. Proof that $\dot{\hat{\mathbf{x}}}(t) = A(t)\hat{\mathbf{x}}(t) + B(t)\boldsymbol{\pi}(\hat{\mathbf{x}}(t),t) + \mathbf{c}(t)$.

The dependencies on time are suppressed for readability.

$$
\begin{aligned}
\dot{\hat{\mathbf{x}}} &= -\{(S+\bar{S})^{-1}\}'(\mathbf{s}+\bar{\mathbf{s}}) - (S+\bar{S})^{-1}\{\mathbf{s}+\bar{\mathbf{s}}\}'\\
&= (S+\bar{S})^{-1}\{\dot{S}+\dot{\bar{S}}\}(S+\bar{S})^{-1}(\mathbf{s}+\bar{\mathbf{s}}) - (S+\bar{S})^{-1}\{\dot{\mathbf{s}}+\dot{\bar{\mathbf{s}}}\}\\
&= (S+\bar{S})^{-1}\{-(S+\bar{S})(A-BR^{-1}P) - (A-PR^{-1}B)^T\cdot\\
&\quad (S+\bar{S}) + SBR^{-1}B^TS - \bar{S}BR^{-1}B^T\bar{S}\}(S+\bar{S})^{-1}(\mathbf{s}+\bar{\mathbf{s}})\\
&\quad - (S+\bar{S})^{-1}\{-(S+\bar{S})(\mathbf{c}-BR^{-1}\mathbf{r}) - (A-PR^{-1}B)^T\cdot\\
&\quad (\mathbf{s}+\bar{\mathbf{s}}) + SBR^{-1}B^T\mathbf{s} - \bar{S}BR^{-1}B^T\bar{\mathbf{s}}\}\\
&= \{-(A-BR^{-1}P) + (S+\bar{S})^{-1}(SBR^{-1}B^TS -\\
&\quad \bar{S}BR^{-1}B^T\bar{S})\}(S+\bar{S})^{-1}(\mathbf{s}+\bar{\mathbf{s}}) +\\
&\quad \mathbf{c} - BR^{-1}\mathbf{r} - (S+\bar{S})^{-1}(SBR^{-1}B^T\mathbf{s} - \bar{S}BR^{-1}B^T\bar{\mathbf{s}})\\
&= \{-(A-BR^{-1}P) + (S+\bar{S})^{-1}((S+\bar{S})BR^{-1}B^TS -\\
&\quad \bar{S}BR^{-1}B^T(S+\bar{S}))\}(S+\bar{S})^{-1}(\mathbf{s}+\bar{\mathbf{s}}) + \mathbf{c} - BR^{-1}\mathbf{r} -\\
&\quad (S+\bar{S})^{-1}((S+\bar{S})BR^{-1}B^T\mathbf{s} - \bar{S}BR^{-1}B^T(\mathbf{s}+\bar{\mathbf{s}}))\\
&= (A-BR^{-1}P-BR^{-1}B^TS)\hat{\mathbf{x}} + \mathbf{c} - BR^{-1}\mathbf{r} - BR^{-1}B^T\mathbf{s}\\
&= A\hat{\mathbf{x}} + B(-R^{-1}(B^TS+P)\hat{\mathbf{x}} - R^{-1}(B^T\mathbf{s}+\mathbf{r})) + \mathbf{c}\\
&= A\hat{\mathbf{x}} + B\boldsymbol{\pi}(\hat{\mathbf{x}},t) + \mathbf{c}. \qquad (35)
\end{aligned}
$$