LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification

Russ Tedrake, Ian R. Manchester, Mark Tobenkin and John W. Roberts

The International Journal of Robotics Research 2010; 29; 1038 originally published online Apr 22, 2010;
DOI: 10.1177/0278364910369189

Russ Tedrake
Ian R. Manchester
Mark Tobenkin
John W. Roberts

Computer Science and Artificial Intelligence Lab,
Massachusetts Institute of Technology,
Cambridge, MA 02139, USA
{russt, irm, mmt, jwr}@mit.edu

LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification*

Abstract

Advances in the direct computation of Lyapunov functions using convex optimization make it possible to efficiently evaluate regions of attraction for smooth non-linear systems. Here we present a feedback motion-planning algorithm which uses rigorously computed stability regions to build a sparse tree of LQR-stabilized trajectories. The region of attraction of this non-linear feedback policy "probabilistically covers" the entire controllable subset of state space, verifying that all initial conditions that are capable of reaching the goal will reach the goal. We numerically investigate the properties of this systematic non-linear feedback design algorithm on simple non-linear systems, prove the property of probabilistic coverage, and discuss extensions and implementation details of the basic algorithm.

KEY WORDS—randomized motion planning, Lyapunov verification, trajectory libraries

1. Introduction

This paper aims to build on advances in systems theory and randomized motion planning to design efficient and general algorithms for non-linear feedback control synthesis in constrained non-linear systems. Specifically, the controls verification community has developed a number of efficient algorithms for direct computation of Lyapunov functions for smooth non-linear systems using convex optimization (Johansen 2000; Parrilo 2000). These tools can be used to estimate conservative regions of attraction around a planned and stabilized trajectory even for very complicated dynamical systems. Combining this formal control verification with ideas from randomized motion planning opens up a number of interesting possibilities for algorithm development. In particular, we present the linear quadratic regulator (LQR)-trees algorithm, which employs locally optimal linear feedback control policies to stabilize planned trajectories computed by local trajectory optimizers and uses verification based on a sums-of-squares method to estimate the local regions of stability.

Specifically, the LQR-trees algorithm operates by growing a tree of stabilized and verified trajectories backwards from a goal state. At each step of the algorithm, a random sample is drawn from a bounded distribution over state space. If that point is already inside the region of attraction of one of the trajectories, then it is discarded. If it is outside, then a local trajectory optimizer attempts to find a new trajectory which connects this random sample to the tree (and therefore to the goal). This new trajectory is stabilized and verified, and then the process repeats. We demonstrate that this algorithm has an important notion of completeness: it provably covers the controllable subset of state space with a stabilizing controller. We call this property "probabilistic feedback coverage".

The aim of this work is to generate a class of algorithms capable of computing verified feedback policies for complicated[1] non-linear systems that are not amenable to feedback linearization and which have dimensionality beyond what might be accessible to grid-based algorithms such as dynamic programming. The use of local trajectory optimizers and local feedback stabilization scales well to higher dimensions, and reasoning about the feedback regions allows the algorithm to cover a bounded subset of state space which is controllable to the goal with a relatively sparse set of trajectories. In addition, the algorithms operate directly on the continuous state and action spaces and perform verification algebraically; thus they are not subject to the pitfalls of discretization which limit accu-

The International Journal of Robotics Research
Vol. 29, No. 8, July 2010, pp. 1038–1052
DOI: 10.1177/0278364910369189

Figures 1–4 appear in color online: http://ijr.sagepub.com

* A preliminary version of this paper was presented at the 2009 Robotics: Science and Systems Conference (Tedrake 2009).

1. Complicated systems might include those subject to underactuation constraints, input saturations, and/or other kinodynamic constraints, for example.

racy and scalability. By considering feedback during the planning process, the resulting plans are robust to disturbances and quite suitable for implementation on real robots.

The following sections introduce the details of the LQR-trees algorithm. After a review of related work in Section 2, Section 3 starts by describing the design, stabilization and local verification of a single feasible trajectory. Section 4 then introduces the method for growing the tree backwards from the goal, yielding a non-linear feedback policy over the state space. Section 5 demonstrates the algorithm on a few model robotic control problems. Section 6 contains the proof of probabilistic feedback coverage. In the final sections, we discuss a few of the more subtle implementation details and possibilities for future extensions.

2. Background

2.1. Feedback Motion Planning

The past decade has seen rapid progress in motion-planning algorithms, spurred by advances in sample-based, randomized algorithms such as probabilistic roadmaps (PRMs) (Kavraki et al. 1996) and rapidly exploring random trees (RRTs) (LaValle and Kuffner 2000). These algorithms have demonstrated the ability to produce feasible open-loop trajectories from an initial condition to a goal state in impressively high-dimensional and convincingly non-convex search problems (Kuffner et al. 2001; Frazzoli et al. 2002; Shkolnik and Tedrake 2009).

For implementation on real robots, open-loop trajectories generated by a motion-planning system are typically stabilized by a trajectory-stabilizing feedback system[2]. While this decoupled approach works for most problems, it is possible that a planned trajectory is not stabilizable or is very costly to stabilize compared to other, more desirable trajectories. *Feedback motion-planning algorithms*, which explicitly consider the feedback stabilization during the planning process, can avoid this pitfall and, as we shall see, can potentially use a local understanding of the capabilities of the feedback system to guide and optimize the search in a continuous state space.

Mason popularized the metaphor of a "funnel" (see Figure 1) for a feedback policy that collapses a large set of initial conditions into a smaller set of final conditions (Mason 1985). Peterfreund and Baram (1998) formalized this concept as "nested Lyapunov functions" and used it to analyze the time convergence of non-linear systems in which the equations of dynamics could be represented as Fourier series. Burridge et al. (1999) then painted a beautiful picture of feedback motion planning as a sequential composition of locally valid feedback policies, or funnels, which take a broad set of initial conditions to a goal region. At the time, the weakness of this approach lay in the difficulty in computing, or estimating by trial-and-error, the region of applicability (i.e. the mouth of the funnel, or preimage) for each local controller in a non-linear system. Consequently, in addition to the particular solution in Burridge et al. (1999), these ideas have mostly been limited to reasoning about vector fields on systems without dynamics (LaValle 2006).

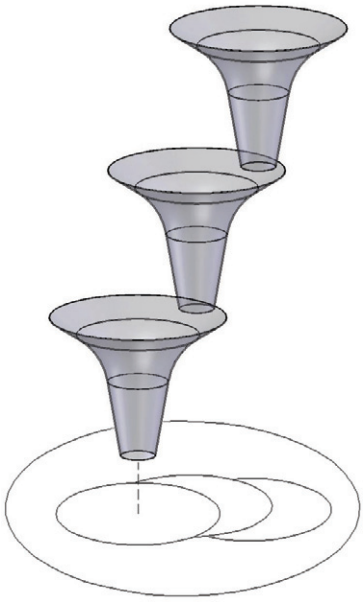


Fig. 1. Motion planning diagram with funnels in the spirit of Burridge et al. (1999).

2.2. Direct Computation of Lyapunov Functions

Burridge et al. (1999) also pointed out the strong connection between Lyapunov functions and motion-planning funnels. For a dynamical system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ with $\mathbf{f}(0) = 0$, a Lyapunov function is a differentiable positive definite output function, $V(\mathbf{x})$, for which $\frac{\partial V}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x})$ is negative definite. If these conditions are met over some ball B_r in state space containing the origin, then the origin is asymptotically stable. The ball B_r can then be interpreted as the preimage of the funnel. Lyapunov functions have played a central role in non-linear control theory, since they allow verification of a system's stability without actually computing solutions to the system (Khalil 2001). Indeed, if a dynamical system is asymptotically stable within a region of attraction, then there exists a Lyapunov function certifying that fact. However, computing such a Lyapunov function for non-linear systems is a challenging problem, and has been the focus of much of the history of non-linear control theory.

A number of computational approaches to computing Lyapunov functions for non-linear systems have emerged in the

2. Note that an increasingly plausible alternative is real-time, dynamic replanning.

past decade, often based on convex optimization; see, for example, Johansen (2000) and Parrilo (2000). One of these techniques, which forms the basis of the results reported here, is based on the realization that one can check the uniform positive definiteness of a polynomial expression using a *sums-of-squares* (SOS) optimization program (Parrilo 2000).

The main idea of SOS is quite straightforward. Suppose we want to check whether the following polynomial inequality is true:

$$x^4 + 2x^3 + 3x^2 - 2x + 2 \geq 0 \quad \forall x \in \mathbb{R}. \tag{1}$$

One way to verify the truth of this inequality is to note that

$$x^4 + 2x^3 + 3x^2 - 2x + 2 = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix},$$

which can be checked by matching the coefficients. Now, the matrix on the right is positive definite (its eigenvalues are approximately 3.88, 1.65 and 0.47), and this implies that for any value of x the polynomial in (1) is positive.

However, the above is not the only way to represent the polynomial as a quadratic form in the "monomials" 1, x and x^2. In general, we could have a matrix of the form

$$x^4 + 2x^3 + 3x^2 - 2x + 2 = \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix}$$

and, by comparing coefficients, we get $a = 2$, $2b = -2$, $d + 2c = 3$, $2e = 2$ and $f = 1$. Note that one can trade off the affine constraint between d and c and still represent the same polynomial. This same approach can be extended to multivariable polynomials; sums-of-squares programming refers to checking non-negativity of a polynomial by searching for a positive semidefinite matrix that satisfies a set of affine constraints which match coefficients. Furthermore, one can perform this optimization when the polynomial coefficients are free (decision) parameters. This results in a semidefinite program which can be solved efficiently by means of interior point methods (Nesterov and Nemirovskii 1994).

Freely available libraries, including SOSTOOLS (Prajna et al. 2004a) and the Systems Polynomial Optimization Toolbox (Megretski) make performing these computations in MATLAB feasible. As we shall see, the ability to check uniform positive (or negative) definiteness will offer a means of verifying candidate Lyapunov functions over a region of state space for smooth non-linear systems.

2.3. Feedback Synthesis by Sums-of-Squares Optimization

The compatibility of sums-of-squares methods with semidefinite programming, and the associated interior point method solvers, has had a notable impact on stability verification; see, e.g., Parrilo (2000), Rantzer (2001), Papachristodoulou and Prajna (2002), Prajna and Rantzer (2007), Tan and Packard (2008), Topcu et al. (2008), Papachristodoulou et al. (2009) and many others. However, an open and active area of research is focused on extending these approaches to controller synthesis. Given a system, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$, we wish to simultaneously generate a feedback $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ and a Lyapunov function $V(\mathbf{x})$ such that $\frac{\partial V}{\partial \mathbf{x}}\left[\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\boldsymbol{\pi}(\mathbf{x})\right]$ is negative definite. This has proven to be a difficult problem because the set of $V(\mathbf{x})$ and $\boldsymbol{\pi}(\mathbf{x})$ satisfying these conditions may not be convex or even connected (Rantzer 2001). To combat this, a number of solutions have been proposed based on density functions (Prajna et al. 2004b) and iterative algorithms for model-predictive control design (Franzè et al. 2009).

This paper takes a different approach to control synthesis. Rather than designing a non-linear feedback controller directly in the optimization, we rely on classical LQR synthesis (Anderson and Moore 1990) to design a series of locally valid controllers and then compose these controllers using feedback motion planning. This has the advantage of potentially working for hopelessly non-convex control problems, such as navigation through a complicated obstacle field, robotic manipulation, or legged locomotion over rough terrain, where randomized motion-planning algorithms have demonstrated success. We prove in Section 6 that, under some mild assumptions, for a class of smooth non-linear systems the feedback synthesis will eventually stabilize effectively every point that is controllable to the goal.

2.4. Other Related Work

Atkeson and Stephens (2008) used local trajectory optimizers and LQR stabilizers with randomized starting points to try to cover the space, with the hope of verifying global optimality (in the infinite-resolution case) by having consistent locally quadratic estimates of the value function on neighboring trajectories. The conditions for adding nodes in that work were based on the magnitude of the value function (not the region of guaranteed stability). In the work described here, we sacrifice direct attempts at obtaining optimal feedback policies in favor of computing "good enough" policies which probabilistically cover the controllable subset of state space with a region of attraction. As a result, we have stronger guarantees of getting to the goal and considerably sparser collections of sample paths.

Region-of-attraction verification for polynomial systems has recently been explored as a search for Lyapunov functions using sums-of-squares techniques (Tan and Packard 2008). The optimization required is bilinear in the decision variables, so optimization methods will find only a local solution. Simulation of the system dynamics can help find good initial guesses for the Lyapunov function (Topcu et al. 2008). Tomlin et al. (2005) explore verification and control of non-linear systems using reachable set computations and hybrid

automata. In general, this requires computing level sets of a Hamilton–Jacobi–Bellman–Isaacs partial differential equation, which must be approached numerically for all but the simplest systems (Mitchell and Templeton 2005). A recent approach to control design which has some philosophical connections to the method we propose is that of "patchy Lyapunov functions", in which a system is stabilized by covering the state space with an ordered family of locally valid control Lyapunov functions and stabilization is ensured via a switching feedback control (Goebel et al. 2009).

3. Linear Feedback Design and Verification

The LQR-trees algorithm is based on the ability to efficiently design trajectories of the robot through state space, to stabilize those trajectories using LQR feedback design, and to estimate the region of attraction of the feedback controller. In this section, we develop the procedure first for a stabilizable point and then for a single stabilizable trajectory.

3.1. Stabilizing a Goal State

Consider a smooth non-linear system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state of the system and $\mathbf{u} \in \mathbb{R}^m$ is the control input. We first examine the problem of stabilizing a goal state with an infinite-horizon LQR controller and approximating the closed-loop region of attraction. Consider a goal state $\mathbf{x}_G$, with $\mathbf{u}_G$ defined so that $\mathbf{f}(\mathbf{x}_G, \mathbf{u}_G) = 0$. Define

$$\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_G, \quad \bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}_G. \tag{3}$$

Now, linearize the system around $(\mathbf{x}_G, \mathbf{u}_G)$ to yield the dynamics

$$\dot{\bar{\mathbf{x}}}(t) \approx \mathbf{A}\bar{\mathbf{x}}(t) + \mathbf{B}\bar{\mathbf{u}}(t). \tag{4}$$

We assume that this linearization of $\mathbf{f}$ about the goal is controllable. Define the quadratic regulator cost-to-go function as

$$\begin{aligned} J(\bar{\mathbf{x}}') &= \int_0^\infty \left[\bar{\mathbf{x}}^{\mathrm{T}}(t)\mathbf{Q}\bar{\mathbf{x}}(t) + \bar{\mathbf{u}}^{\mathrm{T}}(t)\mathbf{R}\bar{\mathbf{u}}(t)\right] dt, \\ \mathbf{Q} &= \mathbf{Q}^{\mathrm{T}} \geq 0, \quad \mathbf{R} = \mathbf{R}^{\mathrm{T}} > 0, \quad \bar{\mathbf{x}}(0) = \bar{\mathbf{x}}'. \end{aligned} \tag{5}$$

The optimal cost-to-go function for the linear system is given by

$$J^*(\bar{\mathbf{x}}) = \bar{\mathbf{x}}^{\mathrm{T}}\mathbf{S}\bar{\mathbf{x}}, \tag{6}$$

where $\mathbf{S}$ is the positive definite solution to the equation

$$0 = \mathbf{Q} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S} + \mathbf{S}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{S} \tag{7}$$

(given by the MATLAB `lqr` function). The optimal feedback policy for the linear system is given by

$$\bar{\mathbf{u}}^* = -\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S}\bar{\mathbf{x}} = -\mathbf{K}\bar{\mathbf{x}}. \tag{8}$$

3.1.1. Time-invariant LQR Verification

We acquire an estimate of the region of attraction of the LQR controller by constructing a function, $V(\mathbf{x})$, which is a valid Lyapunov function for the non-linear system over some sub-level set

$$\mathcal{B}_G(\rho) = \{\mathbf{x} \mid 0 \leq V(\mathbf{x}) \leq \rho\}. \tag{9}$$

To demonstrate asymptotic stability, we require:

- $V(\mathbf{x})$ to be positive definite in $\mathcal{B}_G(\rho)$;
- $\dot{V}(\mathbf{x})$ to be negative definite in $\mathcal{B}_G(\rho)$.

Such a function demonstrates that the goal state is stabilized and, furthermore, that all initial conditions in $\mathcal{B}_G(\rho)$ will converge to $\mathbf{x}_G$ (Slotine and Li 1990).

Here we take $V(\mathbf{x}) = J^*(\bar{\mathbf{x}})$, the linear optimal cost-to-go function, which is a Lyapunov function for the linear system and hence serves well as a local Lyapunov function candidate for the non-linear system. By construction, this choice satisfies the positive definiteness constraint on $V(\mathbf{x})$. For the second condition, first observe that

$$\dot{V}(\mathbf{x}) = \dot{J}(\bar{\mathbf{x}}) = 2\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{S}\mathbf{f}(\mathbf{x}_G + \bar{\mathbf{x}}, \mathbf{u}_G - \mathbf{K}\bar{\mathbf{x}}). \tag{10}$$

SOS programs certify the *global* non-negativity (or non-positivity) of a polynomial; however, our condition on $\dot{J}^*$ is

$$\dot{J}^*(\bar{\mathbf{x}}) < 0 \quad \forall\, \bar{\mathbf{x}} \neq 0 \in \mathcal{B}_G(\rho), \quad \dot{J}^*(\mathbf{0}) = 0. \tag{11}$$

We take two steps to make our condition compatible with SOS; these are common in the SOS literature (Parrilo 2000). First, we transform the negativity constraint into a non-positivity constraint by requiring that

$$\dot{J}^*(\bar{\mathbf{x}}) \leq -\epsilon\|\bar{\mathbf{x}}\|_2^2 \quad \forall\, \bar{\mathbf{x}} \in \mathcal{B}_G(\rho) \tag{12}$$

for some small positive constant ϵ. Next, we remove the restriction of $\bar{\mathbf{x}} \in \mathcal{B}_G(\rho)$ by using a Lagrange multiplier. Take any non-negative polynomial $h(\bar{\mathbf{x}})$ and examine the inequality

$$\dot{J}^*(\bar{\mathbf{x}}) + h(\bar{\mathbf{x}})(\rho - J^*(\bar{\mathbf{x}})) \leq -\epsilon\|\bar{\mathbf{x}}\|_2^2. \tag{13}$$

Note that the sign of $(\rho - J^*(\bar{\mathbf{x}}))$ is positive for all $\bar{\mathbf{x}} \in \mathcal{B}_G(\rho)$ by definition, and is negative outside this ball. As a result, if (13) holds for any $h(\bar{\mathbf{x}}) \geq 0$, it immediately implies (12). This allows us to search for such an $h(\bar{\mathbf{x}})$:

$$\begin{aligned} \text{find} \quad & h(\bar{\mathbf{x}}) \\ \text{subject to} \quad & \dot{J}^*(\bar{\mathbf{x}}) + h(\bar{\mathbf{x}})\left(\rho - J^*(\bar{\mathbf{x}})\right) \leq -\epsilon\|\bar{\mathbf{x}}\|_2^2, \\ & h(\bar{\mathbf{x}}) \geq 0. \end{aligned} \tag{14}$$

The polynomial h must be of sufficiently large degree to counteract higher-order terms in $\dot{J}^*(\bar{\mathbf{x}})$. Throughout this paper we

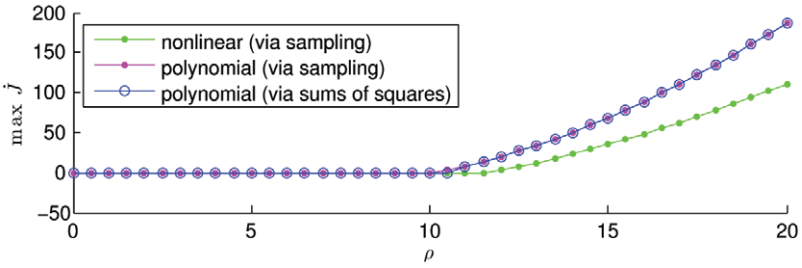


Fig. 2. Polynomial verification of linear time-invariant feedback on the damped simple pendulum ($m = 1$ kg, $l = 0.5$ m, $b = 0.1$ m^2 kg s^{-1}, $g = 9.8$ m s^{-2}, $\mathbf{Q} = \text{diag}([10, 1])$, $\mathbf{R} = 15$, $N_f = 3$, $N_m = 2$).

make frequent use of Lagrange multipliers such as $h(\cdot)$ to handle regional non-negativity conditions. In the case where $\mathbf{f}$ is polynomial in $\mathbf{x}$ and $\mathbf{u}$, demonstrating a feasible $h(\cdot)$ is sufficient for achieving an exact verification of our desired condition, although it is important to note that not all non-negative polynomials are SOS (Parrilo 2000).

For the case where $\mathbf{f}$ is not polynomial, we can approximate this condition by means of a Taylor expansion of $\mathbf{f}$ with order greater than one. Let us write the closed-loop dynamics as

$$\mathbf{f}^{(\text{cl})}(\mathbf{x}) = \mathbf{f}(\mathbf{x}, \mathbf{u}_G - \mathbf{K}(\mathbf{x} - \mathbf{x}_G)). \tag{15}$$

We denote the truncated Taylor expansion of $\mathbf{f}^{(\text{cl})}$ to order $N_f > 1$ by

$$\mathbf{f}^{(\text{cl})}(\mathbf{x}_G + \bar{\mathbf{x}}) \approx \hat{\mathbf{f}}^{(\text{cl})}(\bar{\mathbf{x}}). \tag{16}$$

For example, when $N_f = 2$, the kth component of the closed-loop dynamics is approximated by

$$\begin{aligned} \hat{\mathbf{f}}_k^{(\text{cl})}(\bar{\mathbf{x}}) &= \sum_i \bar{x}_i \left[\frac{\partial \mathbf{f}_k^{(\text{cl})}(\mathbf{x})}{\partial x_i} \right]_{\mathbf{x}=\mathbf{x}_G} \\ &+ \frac{1}{2} \sum_{i,j} \bar{x}_i \bar{x}_j \left[\frac{\partial^2 \mathbf{f}_k^{(\text{cl})}(\mathbf{x})}{\partial x_i \partial x_j} \right]_{\mathbf{x}=\mathbf{x}_G}, \end{aligned} \tag{17}$$

where we also have that $\mathbf{f}_k^{(\text{cl})}(\mathbf{x}_G) = 0$. When the polynomial approximation of the dynamics is used, we replace the $\dot{J}^*(\bar{\mathbf{x}})$ in the SOS program above by $\hat{\dot{J}}^*(\bar{\mathbf{x}})$, where

$$\hat{\dot{J}}(\bar{\mathbf{x}}) = 2\bar{\mathbf{x}}^{\mathrm{T}} \mathbf{S} \hat{\mathbf{f}}^{(\text{cl})}(\bar{\mathbf{x}}). \tag{18}$$

Figure 2 presents a numerical exploration of the estimated region of attraction for the simple pendulum. For increasing values of ρ, the maximum value of $\hat{\dot{J}}^*(\bar{\mathbf{x}})$ in $\mathcal{B}_G$ is plotted. Also plotted are the maximum value achieved by $\hat{\dot{J}}(\bar{\mathbf{x}})$ sampling the boundary of this ellipse and the maximum value of the true (non-polynomial) $\dot{J}^*(\bar{\mathbf{x}})$ sampled similarly. One cannot generally rely on the Taylor expansion to be conservative as in Figure 2. Since we are interested in the Taylor expansion over a bounded region (namely $\mathcal{B}_G(\rho)$), one can bound the error of the expansion if desired and ensure that the region of attraction estimate is conservative, as in Chesi (2009).

Finally, we formulate a convex optimization to find the largest region $\mathcal{B}_G(\rho)$ for which we can verify the second condition:

$$\begin{aligned} \text{maximize} \quad & \rho \\ \text{subject to} \quad & \hat{\dot{J}}^*(\bar{\mathbf{x}}) + h(\bar{\mathbf{x}}) \left(\rho - \hat{J}^*(\bar{\mathbf{x}}) \right) \leq -\epsilon \|\mathbf{x}\|_2^2, \\ & \rho > 0, \\ & h(\bar{\mathbf{x}}) \geq 0. \end{aligned} \tag{19}$$

This optimization can be performed via a line search on ρ. At each step of the line search the SOS program (14) is executed. If the program is feasible, then ρ can be increased; otherwise it must be decreased. Thus, the maximal ρ that can be verified is approached by a sequence of SOS programs.

3.2. Trajectory Optimization

In order to increase the set of states that can reach the goal beyond the time-invariant LQR design, we will design and stabilize a feasible trajectory for the system with an initial condition outside the verified region of attraction of the time-invariant controller. Trajectory design can be accomplished readily by trajectory optimization algorithms, including shooting methods, multiple shooting methods and direct collocation methods (Betts 2001), all of which are quite mature and can perform well even on very complicated non-linear systems. Given the non-linear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, we solve for a feasible trajectory $(\mathbf{x}_0(t), \mathbf{u}_0(t))$ of the system over the finite time interval $[t_0, t_f]$ which (locally) optimizes a cost function of the form

$$J = \int_{t_0}^{t_f} \left[1 + \mathbf{u}_0^{\mathrm{T}} \mathbf{R} \mathbf{u}_0 \right] dt, \tag{20}$$

often subject to a final-value constraint (for instance, $\mathbf{x}_0(t_f) = \mathbf{x}_G$). Throughout the paper, we perform trajectory optimization with a multiple shooting method with the initial trajectory

specified by a random duration "tape" of random control actions; this constrained non-linear optimization is implemented with sequential quadratic programming (SQP), using SNOPT (Gill et al. 2005).

3.3. Time-varying LQR

Given a nominal trajectory $(\mathbf{x}_0(t), \mathbf{u}_0(t))$, a solution of Equation (2) over a finite time interval $[t_0, t_f]$, we stabilize the trajectory using a time-varying LQR controller. Linearizing the system around the trajectory, we obtain

$$\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0(t), \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0(t), \tag{21}$$

$$\dot{\bar{\mathbf{x}}}(t) \approx \mathbf{A}(t)\bar{\mathbf{x}}(t) + \mathbf{B}(t)\bar{\mathbf{u}}(t). \tag{22}$$

We define a quadratic regulator (tracking) cost function:

$$\begin{aligned} J(\bar{\mathbf{x}}', t') &= \bar{\mathbf{x}}^{\mathrm{T}}(t_f)\mathbf{Q}_f\bar{\mathbf{x}}(t_f) \\ &\quad + \int_{t'}^{t_f} \left[\bar{\mathbf{x}}^{\mathrm{T}}(t)\mathbf{Q}\bar{\mathbf{x}}(t) + \bar{\mathbf{u}}^{\mathrm{T}}(t)\mathbf{R}\bar{\mathbf{u}}(t)\right] dt, \\ \mathbf{Q}_f &= \mathbf{Q}_f^{\mathrm{T}} > \mathbf{0}, \quad \mathbf{Q} = \mathbf{Q}^{\mathrm{T}} \geq \mathbf{0}, \\ \mathbf{R} &= \mathbf{R}^{\mathrm{T}} > \mathbf{0}, \quad \bar{\mathbf{x}}(t') = \bar{\mathbf{x}}'. \end{aligned} \tag{23}$$

In general, $\mathbf{Q}$ and $\mathbf{R}$ could easily be made functions of time as well. With time-varying dynamics, the resulting cost-to-go function is time-varying. It can be shown that the optimal cost-to-go function, J^*, is given by

$$J^*(\bar{\mathbf{x}}, t) = \bar{\mathbf{x}}^{\mathrm{T}}\mathbf{S}(t)\bar{\mathbf{x}}, \quad \mathbf{S}(t) = \mathbf{S}^{\mathrm{T}}(t) > \mathbf{0}, \tag{24}$$

where $\mathbf{S}(t)$ is the solution to

$$-\dot{\mathbf{S}} = \mathbf{Q} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{S} + \mathbf{S}\mathbf{A} + \mathbf{A}^{\mathrm{T}}\mathbf{S}, \quad \mathbf{S}(t_f) = \mathbf{Q}_f, \tag{25}$$

and the optimal feedback policy is given by

$$\bar{\mathbf{u}}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}(t)\mathbf{S}(t)\bar{\mathbf{x}}(t) = -\mathbf{K}(t)\bar{\mathbf{x}}(t). \tag{26}$$

3.3.1. Time-varying LQR Verification

In order to describe the stabilization of a finite-length nominal trajectory rather than asymptotic stability, we specify a bounded goal region $\mathcal{B}_f$ of state space and search for a time-varying region $\mathcal{B}(\rho(\cdot), t)$ for which the closed-loop system obeys

$$\mathbf{x}(t) \in \mathcal{B}(\rho(\cdot), t) \implies \mathbf{x}(t_f) \in \mathcal{B}_G \quad \forall\, t \in [t_0, t_f]. \tag{27}$$

We search for such a region in terms of a (now time-varying) positive definite, differentiable, radially unbounded function $V(\mathbf{x}, t)$. At a moment in time, the region is determined as a sub-level set

$$\mathcal{B}(\rho(\cdot), t) = \{\mathbf{x} \mid 0 \leq V(\mathbf{x}, t) \leq \rho(t)\}, \tag{28}$$

where $\rho(t)$ is chosen to ensure that (27) holds. We define the goal region similarly:

$$\mathcal{B}_f = \{\mathbf{x} \mid 0 \leq V(\mathbf{x}, t_f) \leq \rho_f\}. \tag{29}$$

To guarantee that (27) holds, we choose the function $\rho(t) : [t_0, t_f] \mapsto \mathbb{R}^+$ to have the following conservative properties:

- $\rho(t_f) \leq \rho_f$.
- $\rho(t)$ is discontinuous at finitely many points $\tau_1, \ldots, \tau_M$, and
$$\lim_{t \uparrow \tau_m} \rho(t) \leq \lim_{t \downarrow \tau_m} \rho(t). \tag{30}$$
- The right-hand derivative $\frac{d}{dt}_+\rho(t)$ exists, and
$$V(\mathbf{x}, t) = \rho(t) \implies \dot{V}(\mathbf{x}, t) \leq \frac{d}{dt}_+ \rho(t). \tag{31}$$

This ensures that the value $V(\mathbf{x}, t)$ decreases faster along trajectories than our level set $\rho(t)$. It is useful to note that given any two such functions, $\rho_1(t)$ and $\rho_2(t)$, their pointwise maximum $\rho(t) = \max\{\rho_1(t), \rho_2(t)\}$ also satisfies these conditions, which verifies that $\mathcal{B}(\rho_1(\cdot), t) \cup \mathcal{B}(\rho_2(\cdot), t)$ satisfies (27).

Again, here we choose to take $V(\mathbf{x}, t) = J^*(\mathbf{x}, t)$, which is positive definite since the LQR derivation ensures that $\mathbf{S}(t)$ is uniformly positive definite. In particular, this gives us

$$\mathcal{B}_f = \{\mathbf{x} \mid 0 \leq \bar{\mathbf{x}}^{\mathrm{T}}\mathbf{Q}_f\bar{\mathbf{x}} \leq \rho_f\}. \tag{32}$$

When $\mathbf{x}(t_f) = \mathbf{x}_G$, a stabilizable goal, one can choose $\mathbf{Q}_f = \mathbf{S}_{(\mathrm{lti})}$ and $\rho_f = \rho_{(\mathrm{lti})}$ to ensure that trajectories land in an infinite-horizon LQR controller's region of attraction.

Now we have

$$\begin{aligned} \dot{J}^*(\bar{\mathbf{x}}, t) &= \bar{\mathbf{x}}^{\mathrm{T}}\dot{\mathbf{S}}(t)\bar{\mathbf{x}} \\ &\quad + 2\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{S}(t)\mathbf{f}\left(\hat{\mathbf{x}}_0(t) + \bar{\mathbf{x}}, \hat{\mathbf{u}}_0(t) - \mathbf{K}(t)\bar{\mathbf{x}}\right). \end{aligned} \tag{33}$$

Here, even if $\mathbf{f}$ is polynomial in $\mathbf{x}$ and $\mathbf{u}$ and the input tape $\mathbf{u}_0(t)$ was polynomial, our analysis must make use of $\mathbf{x}_0(t)$, $\mathbf{S}(t)$ and $\mathbf{K}(t)$, which result from numerical integration (e.g., using `ode45` in Matlab). We approximate this temporal dependence with (elementwise) piecewise polynomials using splines of order N_t, where N_t is often chosen to be 3 (i.e. cubic splines), with the knot points at the time-steps output by the variable-step integration, which we denote by $t_0, t_1, \ldots, t_N$ with $t_N = t_f$. For example:

$$\forall t \in [t_k, t_k + 1],$$

$$S_{ij}(t) \approx \sum_{m=0}^{N_t} \alpha_{ijm}(t - t_k)^m = \hat{S}_{ij}(t), \tag{34}$$

$$\hat{J}^*(\bar{\mathbf{x}}, t) = \sum_i \sum_j \bar{x}_i \bar{x}_j \hat{S}_{ij}(t). \tag{35}$$

If $\mathbf{f}$ is non-polynomial, we approximate $\dot{J}^*(\bar{\mathbf{x}}, t)$ by first taking the Taylor expansion in $\bar{\mathbf{x}}$ to arrive at a polynomial at each knot t_k and then interpolating the coefficients of the resulting polynomials as piecewise polynomials of time.

3.3.2. Optimizing for $\rho(t)$

In general, we would desire the largest possible $\rho(t)$, perhaps measured by the volume of state space included in $\mathcal{B}(\rho(\cdot), t)$. However, searching over a suitably parameterized $\rho(t)$ over all time segments while verifying invariance appears intractable. Instead, we seek a piecewise-linear[3] $\rho(t)$ over the intervals $[t_k, t_{k+1})$:

$$\rho_k(t) = \beta_{1k} t + \beta_{0k}, \tag{36}$$

$$\rho(t) = \begin{cases} \rho_k(t) & \text{for } t \in [t_k, t_{k+1}), \\ \rho_f & \text{for } t = t_f, \end{cases} \tag{37}$$

which we construct backwards in time, starting with $k = N - 1$. For a given ρ_k, it is easy to test whether $\rho_k(t_{k+1}) = \beta_{1k} t_{k+1} + \beta_{0k} \leq \rho(t_{k+1})$. The more complicated step of verification requires that

$$\hat{J}^*(\bar{\mathbf{x}}, t) = \rho_k(t) \implies \dot{\hat{J}}^*(\bar{\mathbf{x}}, t) \leq \dot{\rho}_k(t) = \beta_{1k} \quad \forall t \in [t_k, t_{k+1}), \tag{38}$$

which can be tested by the following SOS feasibility program:

$$\begin{aligned} \text{find} \quad & h_1(\bar{\mathbf{x}}), h_2(\bar{\mathbf{x}}), h_3(\bar{\mathbf{x}}) \\ \text{subject to} \quad & \dot{\hat{J}}^*(\bar{\mathbf{x}}, t) - \dot{\rho}_k(t) + h_1(\bar{\mathbf{x}}, t)(\rho_k(t) - \hat{J}^*(\bar{\mathbf{x}}, t)) \\ & \quad + h_2(\bar{\mathbf{x}}, t)(t - t_k) + h_3(\bar{\mathbf{x}}, t)(t_{k+1} - t) \leq 0, \\ & h_2(\bar{\mathbf{x}}, t) \geq 0, \\ & h_3(\bar{\mathbf{x}}, t) \geq 0, \end{aligned} \tag{39}$$

where the three Lagrange multipliers h_1, h_2 and h_3 are chosen to be polynomials of sufficient order. Note that $h_1(\bar{\mathbf{x}})$ is used to restrict our attention to an equality constraint and thus does not need to be non-negative. This feasibility test enables a search for $\beta_{\cdot k}$. The feasible region is generally non-convex, but there are some guarantees as to the local minima.

Our assumptions about the system dynamics ensure that if the polynomial approximations of $\mathbf{x}_0(t)$ and $\dot{J}^*(\bar{\mathbf{x}}, t)$ are sufficiently close to true solutions of the dynamics and Riccati equations, then

$$\exists\, \epsilon_k > 0 \text{ such that } \forall\, t \in [t_k, t_{k+1}), \quad J^*(\bar{\mathbf{x}}, t) \leq \epsilon_k \implies \dot{\hat{J}}^*(\bar{\mathbf{x}}, t) \leq 0. \tag{40}$$

3. Higher-order polynomials are of course possible, but for small time-steps, piecewise-linear functions seem to perform well.

We formulate the optimization

$$\begin{aligned} \underset{\alpha_k}{\text{maximize}} \quad & \rho_k(t) = \alpha_k \\ \text{subject to} \quad & \alpha_k \leq \rho(t_{k+1}), \\ & \text{SOS program (39).} \end{aligned} \tag{41}$$

This program is amenable to line search, similar to that for the linear time-invariant region of attraction. A piecewise-constant $\rho(t)$ would be defined by these values, but it has the disadvantage of being monotonically increasing.

This value obtained from (39), α_k^*, can be used as an initial guess for optimizing $\rho_k(t)$ over a class of polynomials. A particularly successful optimization strategy has been the following:

$$\begin{aligned} \underset{\beta_{\cdot k}}{\text{maximize}} \quad & \rho_k(t_k) = \beta_{1k} t_k + \beta_{0k} \\ \text{subject to} \quad & \rho_k(t_{k+1}) \leq \rho(t_{k+1}), \\ & \text{SOS program (39).} \end{aligned} \tag{42}$$

This provides coefficients β^* such that

$$\rho_k(t) = \max\left\{\alpha_k^*, \beta_{1k} t + \beta_{0k}\right\}, \quad k = N - 1, \ldots, 1 \tag{43}$$

form a valid, piecewise-linear certificate, and avoids locally maximizing ρ_k at the expense of later steps.

3.4. Verification with Input Saturation

This section describes a modified condition for certificates covering systems with saturated inputs. We examine the single-input case where the control law $\mathbf{u}(t) = \mathbf{u}_0(t) - \mathbf{K}(t)\bar{\mathbf{x}}(t)$ is mapped through

$$g(\mathbf{u}(t)) = \begin{cases} \mathbf{u}_+ & \text{if } \mathbf{u}(t) \geq \mathbf{u}_+, \\ \mathbf{u}_- & \text{if } \mathbf{u}(t) \leq \mathbf{u}_-, \\ \mathbf{u}(t) & \text{otherwise.} \end{cases}$$

We begin by calculating the smallest level set ρ for which either constraint becomes active; for example,

$$\underset{\bar{\mathbf{x}}'}{\text{minimize}} \quad J^*(\bar{\mathbf{x}}') \tag{44}$$

$$\text{subject to} \quad \mathbf{u}_+ = \mathbf{u}_0 - \mathbf{K}\bar{\mathbf{x}}'. \tag{45}$$

Let $\rho_{\max}$ be the solution to this convex quadratic program. Similarly, $\rho_{\min}$ can be computed as the smallest level set for which $\mathbf{u}^*(t) = \mathbf{u}_-$. These values allow for a case-by-case analysis of the saturation.

We then compute a separate SOS condition for each active constraint, and choose the largest ρ which satisfies all of the constraints. Let

$$\dot{J}_-(\bar{\mathbf{x}}) = \frac{\partial}{\partial \bar{\mathbf{x}}} J(\bar{\mathbf{x}}) \mathbf{f}(\bar{\mathbf{x}}, \mathbf{u}_-) + \frac{\partial}{\partial t} J(\bar{\mathbf{x}}), \tag{46}$$

$$\dot{J}_K(\bar{\mathbf{x}}) = \frac{\partial}{\partial \bar{\mathbf{x}}} J(\bar{\mathbf{x}}) \mathbf{f}(\bar{\mathbf{x}}, \mathbf{u}_0 - \mathbf{K}\bar{\mathbf{x}}) + \frac{\partial}{\partial t} J(\bar{\mathbf{x}}), \tag{47}$$

$$\dot{J}_+(\bar{\mathbf{x}}) = \frac{\partial}{\partial \bar{\mathbf{x}}} J(\bar{\mathbf{x}}) \mathbf{f}(\bar{\mathbf{x}}, \mathbf{u}_+) + \frac{\partial}{\partial t} J(\bar{\mathbf{x}}). \tag{48}$$

Then the verification inequality (14) can be replaced by

$$\begin{aligned} & \dot{J}_-(\bar{\mathbf{x}}) + h_1(\bar{\mathbf{x}})(\rho - J(\bar{\mathbf{x}})) \\ + \; & h_2(\bar{\mathbf{x}})(\mathbf{u}_- - \mathbf{u}_0 + \mathbf{K}\bar{\mathbf{x}}) \leq -\epsilon \|\bar{\mathbf{x}}\|_2^2, \end{aligned} \tag{49}$$

$$\begin{aligned} & \dot{J}_K(\bar{\mathbf{x}}) + h_3(\bar{\mathbf{x}})(\rho - J(\bar{\mathbf{x}})) + h_4(\bar{\mathbf{x}})(\mathbf{u}_0 - \mathbf{K}\bar{\mathbf{x}} - \mathbf{u}_-) \\ + \; & h_5(\bar{\mathbf{x}})(\mathbf{u}_+ - \mathbf{u}_0 + \mathbf{K}\bar{\mathbf{x}}) \leq -\epsilon \|\bar{\mathbf{x}}\|_2^2, \end{aligned} \tag{50}$$

$$\begin{aligned} & \dot{J}_+(\bar{\mathbf{x}}) + h_6(\bar{\mathbf{x}})(\rho - J(\bar{\mathbf{x}})) \\ + \; & h_7(\bar{\mathbf{x}})(\mathbf{u}_0 - \mathbf{K}\bar{\mathbf{x}} - \mathbf{u}_+) \leq -\epsilon \|\bar{\mathbf{x}}\|_2^2, \end{aligned} \tag{51}$$

with each $h_i(\bar{\mathbf{x}})$ being a sum of squares. Again, these Lagrange multipliers serve to restrict the verification to the regions of interest. Note that (50) is always evaluated, as it corresponds to the unsaturated region. Inequalities (49) and (51) verify stability with $\mathbf{u}$ saturated at its minimum value and maximum value, respectively, and need only be evaluated if $\rho > \rho_{\min}$ and $\rho > \rho_{\max}$, respectively.

Analogous additions can be made to the linear time-varying verification inequality (39). It is straightforward to extend this method to higher input dimensions; however, in general there will be 3^m different combinations of saturated and unsaturated inputs, where m is the control dimension, which, for systems with many inputs, could quickly become computationally prohibitive.

4. The LQR-trees Algorithm

Given the ability to design, stabilize and verify a trajectory from an initial condition to the goal, the LQR-trees algorithm proceeds by growing a randomized tree of stabilizing controllers to reach the goal. Because both the feedback design and the verification work backwards in time, we grow the tree backwards, starting from the goal. The result is that the tree becomes a large web of stabilizing controllers which grab initial conditions and pull them towards the goal (with formal certificates of stability for the non-linear, continuous state and the action system).

The goal of the algorithm is to cover an entire region of interest—the set of points from which the goal state is reachable or a specified bounded subset of this region—with this stabilizing controller. In Section 6, we define this property as "probabilistic feedback coverage". To achieve this, we grow our tree in the fashion of an RRT, where new subgoals are chosen at random from a uniform distribution over the state space. Unlike the RRTs, we have additional information from the estimated regions of stability (funnel), and we can immediately discard sample points which are sampled inside the previously verified region. Define $\mathcal{C}_k$ to be the estimated region of stability for the entire tree after k iterations of the algorithm. The discarding of random samples from $\mathcal{C}_k$ occurs rapidly, due in part to our choice of a simple ellipsoidal estimate for the funnels, and leads to a dramatic improvement in the sparsity of the randomized tree.

One consequence of this sampling strategy is that the algorithm which attempts to connect the random sample back to the goal must be capable of establishing this connection over some finite distance through state space. For this reason, we use the trajectory optimization algorithms described briefly in Section 3.2 to grow a trajectory from the sampled point forward until it connects to the existing tree. These algorithms are local, and are subject to local minima, but we prove in Section 6 that they are sufficient for obtaining the desired coverage property of the algorithm. If the trajectory optimization fails to connect to the tree, then the sampled point is discarded to allow for the possibility that the tree is not reachable from that sampled point; if the tree is reachable, then this region will be sampled again in the future with a more extensive tree available to connect to. The trajectory optimization is set to timeout with a failure after some relatively small number of major iterations of the SQP algorithm, in order to facilitate fast sampling and discarding of new samples.

The most important component of the trajectory optimization that attempts to connect to the tree is the "final tree constraint", which specifies that the optimized trajectory must connect back to the tree but is free to connect to any part of the continuous manifold of points that form the tree. On the kth iteration of the algorithm, given a candidate trajectory terminating in $\mathbf{x}_0(t_f)$, we enforce the vector constraint that

$$\min_{\mathbf{x}' \in \mathcal{T}_{k-1}} \|\mathbf{x}_0(t_f) - \mathbf{x}'\|_2 = \mathbf{0}, \tag{52}$$

where $\mathcal{T}_k$ is the set of all points in the tree after k iterations of the algorithm. A well-implemented optimization algorithm can satisfy this constraint by designing a trajectory which connects to the tree and then continuing to "walk" the terminal condition along the constraint manifold of the tree to maximize the objective.

4.1. The Algorithm

The algorithm proceeds by producing a tree, T, with nodes containing the tuples $\{\mathbf{x}, \mathbf{u}, \mathbf{S}, \mathbf{K}, \rho_c, i\}$, where $J^*(\bar{\mathbf{x}}, t) =$

Algorithm 1 LQR-tree ($\mathbf{f}, \mathbf{x}_G, \mathbf{u}_G, \mathbf{Q}, \mathbf{R}$)

1: $[\mathbf{A}, \mathbf{B}] \Leftarrow$ linearization of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ around $(\mathbf{x}_G, \mathbf{u}_G)$
2: $[\mathbf{K}, \mathbf{S}] \Leftarrow \text{LQR}(\mathbf{A}, \mathbf{B}, \mathbf{Q}, \mathbf{R})$
3: $\rho_c \Leftarrow$ level set computed as described in §3.1.1
4: T.init($\{\mathbf{x}_g, \mathbf{u}_g, \mathbf{S}, \mathbf{K}, \rho_c, \text{NULL}\}$)
5: **for** $k = 1$ to K **do**
6: $\quad \mathbf{x}_{\text{rand}} \Leftarrow$ random sample
7: $\quad$ **if** $\mathbf{x}_{\text{rand}} \in \mathcal{C}_k$ **then**
8: $\quad\quad$ continue
9: $\quad$ **end if**
10: $\quad [t, \mathbf{x}_0(t), \mathbf{u}_0(t)]$ from trajectory optimization with a "final tree constraint"
11: $\quad$ **if** $\mathbf{x}_0(t_f) \notin \mathcal{T}_k$ **then**
12: $\quad\quad$ continue
13: $\quad$ **end if**
14: $\quad [\mathbf{K}(t), \mathbf{S}(t)]$ from time-varying LQR
15: $\quad \rho_c \Leftarrow$ level set computed as in §3.1.1
16: $\quad i \Leftarrow$ pointer to branch in T containing $\mathbf{x}_0(t_f)$
17: $\quad$ T.add-branch($\mathbf{x}_0(t), \mathbf{u}_0(t), \mathbf{S}(t), \mathbf{K}(t), \rho_c, i$)
18: **end for**

$\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{S}\bar{\mathbf{x}}$ is the local quadratic approximation of the value function, $\bar{\mathbf{u}}^* = -\mathbf{K}\bar{\mathbf{x}}$ is the feedback controller, $J^*(\bar{\mathbf{x}}, t) \leq \rho(t)$ is the funnel with $\rho(t)$ described by the vector of polynomial coefficients $\boldsymbol{\rho}_c$, and i is a pointer to the parent node.

4.2. Executing LQR-tree Feedback

The resulting LQR-tree policy is a function

$$\mathbf{u} = \pi_T(\mathbf{x}, t_T, b) \tag{53}$$

with internal controller state t_T, the time corresponding to the current execution, and a unique identifier b for the currently active branch of the tree. At the onset of control from an initial condition $\mathbf{x}_i$, these controller variables are initialized as

$$[t_T, b] = \underset{b \in T, t \in [t_0^b, t_f^b]}{\text{argmax}} \; c(\mathbf{x}_i, t, b), \tag{54}$$

$$c(\mathbf{x}, t, b) = \rho^b(t) - \left(\mathbf{x}_i - \mathbf{x}_0^b(t)\right)^{\mathrm{T}} \mathbf{S}^b(t) \left(\mathbf{x} - \mathbf{x}_0^b(t)\right), \tag{55}$$

where superscript b denotes the nominal times, trajectories and funnels on branch b of the tree (with b taken from an index set of branches). We refer to the quantity $c(t)$ as the *confidence*. A positive value of $c(t)$ corresponds to a verification of stability to the goal, and larger values of $c(t)$ mean that the current state is further inside the region of attraction. Although it is tempting to continually search over t_T as the policy is executed in order to maximize $c(t)$, this can lead to undesirable properties such as chattering. Therefore, in practice, we execute the policy with time evolving naturally ($\dot{t}_T = 1$), unless a disturbance occurs that takes the state outside the funnel, in which case we re-evaluate t_T and b. Similarly, the branch index b is held constant until the end of its tape, at which point b is set to the parent trajectory of the previous branch, closer to the goal.

5. Simulations

We illustrate the operation of the algorithm on a two-dimensional toy problem, for which we can carefully plot the entire state space and the covered set $\mathcal{C}_k$ at each iteration of the algorithm. Here we use the swing-up task for a torque-limited simple pendulum, $I\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = \tau$, with $m = 1, l = 0.5, b = 0.1, I = ml^2, g = 9.8$ and $|\tau| \leq 3$. Here $\mathbf{x} = [\theta, \dot{\theta}]^{\mathrm{T}}$ and $\mathbf{u} = \tau$. The parameters of the LQR-trees algorithm were $\mathbf{x}_G = [\pi, 0]^{\mathrm{T}}$, $\mathbf{u}_G = 0$, $\mathbf{Q} = \text{diag}([10, 1])$, $\mathbf{R} = 20$, $N_f = 3$ and $N_t = 3$.

Figure 3(a) shows the region of attraction (gray shaded region) after computing the linear time-invariant LQR solution around the unstable equilibrium. Figure 3(b) shows the entire nominal trajectory (in blue) to the first random sample point, together with the funnels that have been computed for this trajectory. Note that the state space of the pendulum lives on a cylinder, and the trajectory (and region of attraction) wraps around from the left to the right. Plots (c) and (d) show the region of attraction as it grows to fill the state space, with the black ellipses denoting the inlet and outlet of each funnel in the tree.

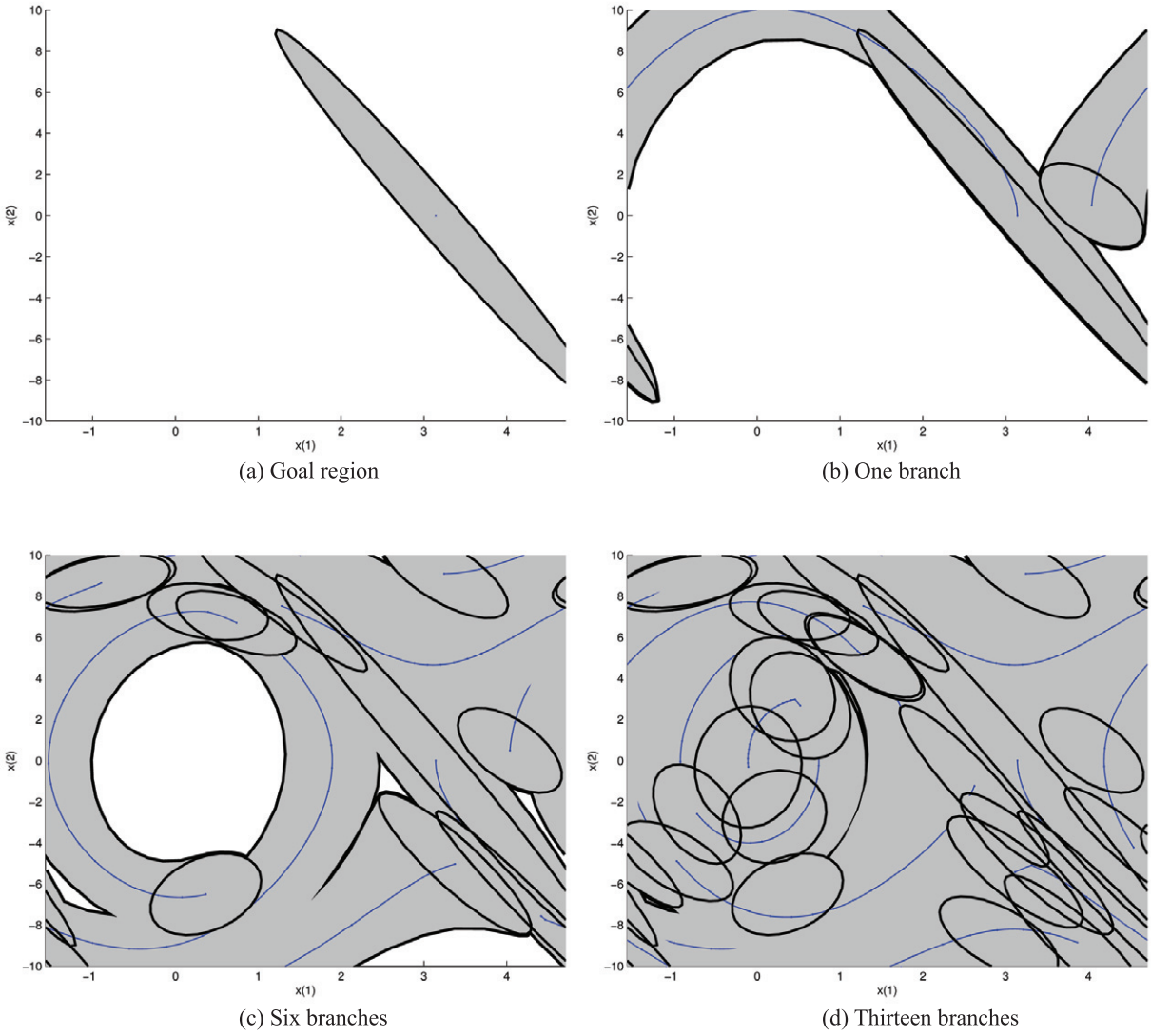
In this example, the entire space is probabilistically covered (1,000 random points chosen sequentially were all in the region of attraction) after the tree contained just 13 branches. On average, for the simple pendulum with these parameters, the algorithm terminates after 13.25 branches have been added. For contrast, Branicky and Curtiss (2002) shows a well-tuned single-directional RRT for the simple pendulum which has 5,600 nodes. However, the cost of adding each node is considerably greater here than in traditional RRT, dominated by the line search used to maximize the estimated region of stability.

6. LQR-trees Achieve Probabilistic Feedback Coverage

Consider a smooth system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{56}$$

with $\mathbf{x}(t) \in \mathcal{X}$ and $\mathbf{u}(t) \in \mathcal{U}$, where $\mathcal{X}$ and $\mathcal{U}$ are open subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. We assume that for all piecewise continuous functions $\mathbf{u} : [0, \infty) \to \mathcal{U}$, a unique solution of (56) exists, and that solutions have continuous dependence on controls and initial conditions. Note that there are many polynomial differential equations for which this is not true; however, for well-modelled physically motivated systems, these are natural assumptions.



(a) Goal region (b) One branch

(c) Six branches (d) Thirteen branches

Fig. 3. An LQR-tree for the simple pendulum. The x-axis shows $\theta \in [-\pi/2, 3\pi/2]$ (note that the state wraps around this axis), and the y-axis shows $\dot{\theta} \in [-10, 10]$. The gray shaded region represents the covered set $\mathcal{C}_k$ (i.e. the "funnels") after k branches have been added to the tree.

We wish to consider the performance of an iterative algorithm for the task of planning motions to a particular goal state $\mathbf{x}_G \in \mathcal{X}$.

We define the following sets:

- $\mathcal{R}(\mathbf{x}_G)$, the set of points from which a state $\mathbf{x}_G$ is reachable, possibly in infinite time; that is, $\mathbf{x}_0 \in \mathcal{R}(\mathbf{x}_G)$ if and only if there exists a piecewise-continuous control signal $\mathbf{u}^\star : [0, \infty) \to \mathcal{U}$ such that the system (56) with initial condition $\mathbf{x}_0$ and input $\mathbf{u}^\star$ asymptotically approaches $\mathbf{x}_G$.
- $\mathcal{C}_k$, the set of points covered (e.g. inside the region of attraction of some trajectory) by the feedback motion-planning algorithm after k iterations. This is the union of all funnels that have been added so far, hence $C_k \subseteq C_{k+1}$.
- $\mathcal{C}_\infty$, the limit set of the coverage sets $\mathcal{C}_k$; that is,

$$\mathcal{C}_\infty = \lim_{k \to \infty} \mathcal{C}_k. \tag{57}$$

  In other words, $\mathbf{x} \in \mathcal{C}_\infty$ if and only if there exists a k_0 such that $\mathbf{x} \in C_k$ for all $k > k_0$.

Note that the limit $\mathcal{C}_\infty$ is well-defined since $\mathcal{C}_k \subseteq \mathcal{C}_{k+1}$ for all k, which implies that $\limsup \mathcal{C}_k = \liminf \mathcal{C}_k$.

We are interested in proving the following property, which plays an analogous role in feedback motion planning to that

of probabilistic completeness in open-loop motion planning (LaValle and Kuffner 2000).

Definition 1. A feedback motion-planning algorithm achieves *probabilistic feedback coverage* for the goal state $\mathbf{x}_G$ if $\mathrm{cl}(\mathcal{C}_\infty) = \mathrm{cl}(\mathcal{R}(\mathbf{x}_G))$ with probability one[4].

The LQR-trees algorithm works by randomly sampling states and attempting to link them back to the tree. We make the following assumption.

Assumption 1. Each new state to be linked back to the tree is sampled from a distribution with non-zero probability density everywhere on $\mathcal{X}$.

For bounded $\mathcal{X}$ a uniform probability density is adequate, but for unbounded $\mathcal{X}$ more care will need to be taken to ensure that the density is integrable.

Assumption 2. We make the following assumptions.

(a) For the goal point $\mathbf{x}_G$, the linear time-invariant system is completely controllable.

(b) For any feasible trajectory $(\mathbf{x}^\star, \mathbf{u}^\star)$ of (56), the associated linear time-varying Riccati differential equation has a positive definite stabilizing solution from any positive definite final-time condition.

Assumption 2(b) is an assumption on local exponential stabilizability of the system dynamics by linear feedback. Taken together, these assumptions imply that $\mathcal{C}_k$ has non-empty interior for every k and hence that $\mathcal{R}(\mathbf{x}_G)$ has non-empty interior, since each $\mathcal{C}_k \subset \mathcal{R}(\mathbf{x}_G)$.

Let $\mathcal{T}_k$ be the set of states on the branches of the LQR-tree after k nodes have been added. We make the following assumption on the availability of an open-loop motion-planning algorithm.

Assumption 3. The open-loop motion planner used to reconnect to the tree has the following property: given any point $\mathbf{x}_0 \in \mathcal{R}(\mathbf{x}_G)$, the motion planner will, with non-zero probability, find a finite time t_f and a control signal $\mathbf{u}^\star(t)$ defined on $[0, t_f]$ such that the solution of $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}^\star)$ with $\mathbf{x}(0) = \mathbf{x}_0$ achieves $\mathbf{x}(t_f) \in \mathcal{T}_k$.

In Section 6.1 we discuss this assumption and how it can be satisfied in practice.

We are now ready to state the main result of this section.

4. Here $\mathrm{cl}(\cdot)$ denotes the closure of a set.

Theorem 1. *For any system and goal state satisfying Assumptions 1, 2 and 3, the LQR-trees algorithm achieves probabilistic feedback coverage.*

Proof. For a particular instance of the randomized tree, define

$$\mathcal{F} = \bigcap_{k=1}^{\infty} \{\mathcal{R}(\mathbf{x}_G)/\mathcal{C}_k\}, \tag{58}$$

i.e. the set of all states from which the goal is reachable that fail to be added to the tree. Note that $\mathcal{F}$ is the complement of $\mathcal{C}_\infty$ relative to $\mathcal{R}(\mathbf{x}_G)$. Now, $\mathrm{cl}(C_\infty)$ equals $\mathrm{cl}(\mathcal{R}(\mathbf{x}_G))$ if and only if $\mathcal{F}$ has empty interior. We prove that this must be the case by contradiction.

Suppose $\mathcal{F}$ has non-empty interior, denoted by $\mathrm{int}(\mathcal{F})$. In this case the Lebesgue measure of $\mathrm{int}(\mathcal{F})$ is positive and hence, by Assumption 1, there is a non-zero probability of sampling from $\mathrm{int}(\mathcal{F})$. Furthermore, by Assumption 3, for any such sample there is a non-zero probability that it can be connected back to the tree. Since all the samplings and trajectory searches are independent events, this implies that with probability one a sample $\mathbf{x}_i$ will eventually be found in $\mathrm{int}(\mathcal{F})$ which connects back to the tree with a feasible trajectory $\mathbf{x}^\star(t)$, i.e. one that has $\mathbf{x}^\star(t_i) = \mathbf{x}_i$ and $\mathbf{x}^\star(t_f) \in \mathcal{T}_k$ for some t_i and t_f.

Now, if such an $\mathbf{x}_i$ were found, then by Assumption 2 the solution of the Riccati equation associated with $\mathbf{x}^\star(t)$ would be positive definite for all $t \in [t_i, t_f]$. Since $\mathcal{X}$ and $\mathcal{U}$ are open sets and $f(\mathbf{x}, \mathbf{u})$ is smooth, this implies that around each point on the trajectory $x^\star(t)$ there is an ellipsoid of positive radius which will be added to $\mathcal{C}_{k+1}$. Since $\mathbf{x}_i \in \mathcal{F}$, some of these ellipsoids will have intersections of non-zero measure with $\mathrm{int}(\mathcal{F})$. However, this would contradict the definition of $\mathcal{F}$. Hence there is zero probability that $\mathcal{F}$ has non-empty interior. Therefore, with probability one, $\mathrm{cl}(\mathcal{C}_\infty)$ equals $\mathrm{cl}(\mathcal{R}(\mathbf{x}_G))$. □

Note that we have not assumed anywhere that $\mathcal{X}, \mathcal{U}$ or $\mathcal{R}(\mathbf{x}_G)$ is bounded. Unbounded sets will, however, require careful selection of sampling density, and in practice they may be problematic in regard to satisfying Assumption 3.

6.1. Discussion of the Motion-planning Assumption

We give a brief argument that Assumption 3 is reasonable in practice. First, owing to the smoothness of (56) and the complete controllability of the linearized system about $\mathbf{x}_G$, if from a sampled point the system can be made to asymptotically approach $\mathbf{x}_G$, then it can be made to reach $\mathbf{x}_G$ in finite time.

In the present implementation of the LQR-trees algorithm, a path back to the tree is sought by randomly sampling a piecewise-constant control signal $\mathbf{u}$, where the time horizon t_f is randomly sampled from a distribution which has non-zero density everywhere on $[0, T]$ for some large T and the sampling interval is a random variable with non-zero density on

$[0, t_f]$. Therefore, for sufficiently large T, arbitrarily fine approximations of any finite-length piecewise-continuous control signal can be sampled.

Using the sampled $\mathbf{u}$ as an initial guess, a search is performed over input signals with the cost function (20), subject to the constraints that $\mathbf{x}(t_2) \in \mathcal{T}_k$ and $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. The optimization is posed in direct-collocation or multiple-shooting form, and is solved numerically via an SQP method in which non-linear equality constraints are handled by l_1 regularized elastic variables (Gill et al. 2005).

Now, suppose that a sampled $\mathbf{x}_0$ lies in $\mathcal{R}(\mathbf{x}_G)$, i.e. there exists a piecewise-continuous function $\mathbf{u}_0 : [0, t_1] \rightarrow \mathcal{U}$ which drives the state from $\mathbf{x}_0$ to a point on the tree $\mathbf{x}(t_f) \in \mathcal{T}_k$. Since $\mathcal{U}$ is open and solutions of (56) are continuous with respect to controls, there exists an open neighborhood of piecewise-continuous functions $[0, t_2] \rightarrow \mathcal{U}$, $t_2 > t_1$, containing $\mathbf{u}_0$, which is the region of convergence to a local optimum for the optimization (20) considered over piecewise-continuous functions. Since our sampling can produce arbitrarily fine approximations of such functions, it can be argued that there is a non-zero probability of numerically finding a feasible solution of (20).

In practice, if the optimization fails to find a solution given the initial seed, the sampled point is discarded as temporarily uncontrollable; but after the tree has grown, a nearby point will eventually be sampled from which the search will be successful. Another choice for the open-loop motion planner would be a planner based on randomized sampling, such as the RRT (LaValle and Kuffner 2000), run for a finite (but random) number of steps.

6.2. Controllability-limited Example

The performance of the algorithm when searching near a boundary of controllability in state space is an important consideration, particularly if we intend to apply it to complicated, underactuated systems in which the location and form of such boundaries is unknown. Figure 4 demonstrates the performance of the algorithm when applied to the following second-order polynomial dynamical system:

$$\ddot{x} = x + u^2 - u - 3/4. \tag{59}$$

This system has a known controllable set defined by $\dot{x} < 1 - x$ (whose boundary is shown as a red line in Figure 4). Intuitively, the force applied to the system is constrained to be in the set $[-1, \infty)$. From this realization, energy considerations clearly show that the boundary of the controllable set is defined by the line $\dot{x} = 1 - x$. The algorithm will provably fill the sampled space (shown in light blue) which is capable of reaching the goal. As Figure 4 demonstrates, this is achieved by placing ever smaller funnels closer and closer to the boundary, until the boundary is approximated by the surfaces of the funnels.

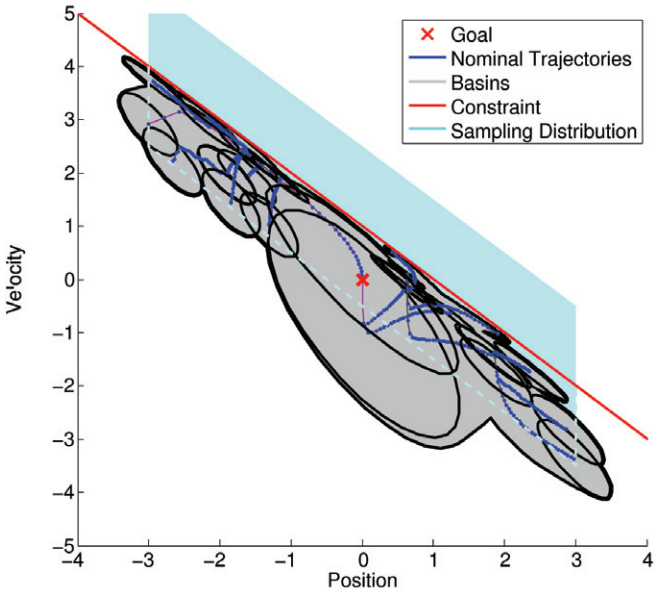


Fig. 4. Controllability-limited example.

7. Discussion

In this section, we briefly discuss the limitations of LQR-trees, possible variations of the main algorithm, and some implementation details.

7.1. Limitations of LQR-trees

While the LQR-trees algorithm has broad applicability in the realm of non-linear systems, it does have limitations with respect to the dimension of the systems that can be solved. Its scaling behavior can best be understood by considering the four components that make up the algorithm: (1) a local motion planner (e.g. multiple shooting); (2) a global motion planner which connects the local plans (RRTs); (3) local control design (time-varying LQR); and (4) local verification (SOS).

The local motion planner and the local control design (components 1 and 3) both scale well with dimension. The global motion planner (component 2) has poor asymptotic behavior but has been shown to work well on systems with 30 or more dimensions (Kuffner et al. 2001) and, for particular tasks, on systems with over 1,000 dimensions (Shkolnik and Tedrake 2009). The bottleneck in addressing systems of increasing dimensionality is component 4, verification using SOS. We have had success applying LQR-trees to systems of dimension up to five, and examples involving rigid body dynamics in six dimensions can be found in the literature; see, e.g., Gollu and Rodrigues (2007).

The reliance on linearized controller synthesis means that the LQR-trees algorithm is not directly applicable to non-linear systems which are controllable but have uncontrollable linearizations, such as non-holonomic systems.

7.2. Variations of the Algorithm

- **Discrete-time formulation**. Although the presentation here has focused on the continuous-time case, the approach has a natural discrete-time formulation. In many ways, the discrete-time formulation is simpler, mitigating many of the numerical subtleties to be described in Section 7.3, and may in fact be preferable for implementation on a sampled-data control system. The primary drawbacks are that: (1) it is more difficult to do a careful time discretization of a polynomial system than of a linear one; and (2) the tree will now consist of a collection of points rather than a smooth manifold, complicating any efforts to create a final-value constraint that can "walk along the tree".

- **Compatibility with optimal trajectories**. The LQR-trees algorithm provides a relatively efficient way to fill the controllable state space with funnels, but it does not make any claim as to the optimality of the resulting trajectories. If tracking particular trajectories, or optimal trajectories, is important for a given problem, then it is quite natural to seed the LQR-tree with one or more locally optimal trajectories and then use the random exploration to fill in any missing regions.

- **Early termination**. For higher-dimensional problems, covering the controllable state space may be unnecessary or impractical. Based on the RRTs, the LQR-trees can easily be steered towards a region of state space (e.g. by sampling from that region with slightly higher probability) containing important initial conditions. Termination could then occur when some important subspace is covered by the tree.

- **Bidirectional trees**. Although LQR-trees only grow backwards from the goal, a partial covering tree (from an early termination) could also serve as a powerful tool for real-time planning. Given a new initial condition, a forward RRT simply has to grow until it intersects with the *volume*, C_k, defined by the region of attraction of the backwards tree.

- **Other local stabilizers**. In this paper we have proposed using LQR as a local stabilizer for points and trajectories. The main reason for this choice is simplicity of computation, together with the fact that the "cost-to-go" automatically provides a reasonable candidate Lyapunov function. Other popular control design techniques, such as H^∞ or linear matrix inequality (LMI)-based methods, also supply candidate Lyapunov functions, and might be more appropriate for certain systems. In particular, some information about the non-linearities could be encoded in an uncertainty structure. These methods require more computational effort than does LQR; but if they are able to stabilize much larger regions, that may be a useful trade-off.

- **Other regional stability verifiers**. In this paper we have proposed SOS as a regional stability verifier. This allows a very large class of smooth non-linear systems to be handled within a single framework for which reliable computational tools are available. However, for systems of high dimension, SOS programs can result in very large semidefinite programs, which become a computational bottleneck. An alternative version which approximates the verification stage via sampling can be performed with much lower computational cost, at the expense of losing the rigorous stability guarantees (Reist and Tedrake 2010).

  Many interesting systems in robotics can be decomposed into "easy" subsystems (e.g. actuated links with linear or nearly linear dynamics) and "hard" subsystems (e.g. unactuated links with highly non-linear dynamics). Taking advantage of this structure could reap large benefits in the computation of stability regions. Taking this line even further, there appear to be deep connections between SOS methods and classical methods for non-linear stability analysis such as integral quadratic constraints (Megretski and Rantzer 1997), which we intend to investigate.

- **Multi-query algorithms**. Another very interesting question is the feasibility of reusing previous computational work when the goal state is changed. In the pendulum example, suppose that we have a new goal state $\mathbf{x}_G = [\pi + 0.1, 0]^{\mathrm{T}}$, which would of course require a non-zero torque to stabilize. To what extent could the tree generated for stabilizing $\mathbf{x}_G = [\pi, 0]^{\mathrm{T}}$ be used to stabilize this new fixed point? If one can find a trajectory to connect up the new goal state near the root of the tree, then the geometry of the tree could be preserved, but naively one might think that all of the stabilizing controllers and the verification would have to be re-calculated. Interestingly, there is also a middle road, whereby the existing feedback policy is kept for the original tree and the estimated funnels are not recomputed but simply scaled down to make sure that the funnels from the old tree transition completely into the funnel for the new tree. This can be accomplished very efficiently by just propagating a new $\rho_{\max}$ through the tree; however, it could come at the cost of losing coverage.

7.3. Implementation Details

The LQR-trees algorithm presented here, which relies on a continuous-time formulation and optimization over polynomials, requires the practitioner to address a number of potential numerical issues. We list a few of these here:

- **Balancing coordinates**. One drawback of working with polynomials is that their associated optimization problems are often poorly conditioned (high-order polynomials blow up quickly), and solutions to the Riccati differential equation can easily have a large range of eigenvalues. An essential step for getting good performance out of the semidefinite programming was finding a coordinate transformation

$$\mathbf{x}_b = \mathbf{T}\mathbf{x}, \tag{60}$$

  where $\mathbf{T}$ was selected to numerically condition the problem by making the matrices $\mathbf{T}'\mathbf{S}\mathbf{T}$ and $\mathbf{T}'\left(2\mathbf{S}\frac{\partial \mathbf{f}(\mathbf{x}_0+\bar{\mathbf{x}},\mathbf{u}_0-\mathbf{K}\bar{\mathbf{x}})}{\partial \mathbf{x}}+\dot{\mathbf{S}}\right)\mathbf{T}$ as close as possible to the identity matrix. Without this balancing step, the semidefinite solvers often return errors with infeasible programs.

- **Piecewise-polynomial approximations of numerical integration.** For the continuous-time formulation, it was essential to carefully manage the piecewise-polynomial interpolations of solutions from trajectory optimization and from the Riccati differential equation. Simple cubic spline representations often caused even the nominal trajectory to appear to be unstable. In the current implementation, we represent the output of the trajectory optimization using a cubic piecewise polynomial that, for every segment, matches the knot points and the derivatives of the knot points, which are known from the dynamics function. The Riccati differential equation is solved using `ode45` in MATLAB, interpolations of that solution are done using `deval`, and calculations of $\dot{\mathbf{S}}$ are done by explicitly re-evaluating the Riccati equation.

- **Time dependence on $\dot{\mathbf{J}}$**. Finally, one must be careful when performing verification on the time-varying Lyapunov candidate. Naively, if one used a cubic spline representation of the nominal trajectory and of $\mathbf{S}$, then $\dot{\mathbf{J}}$ quickly becomes of an unnaturally high order. Instead, we fit the best cubic spline directly to $\dot{J}$.

7.4. A Software Distribution

Owing to the potential complexity of getting a robust and high-performance implementation of the LQR-trees algorithm, the authors are making a MATLAB toolbox available at http://groups.csail.mit.edu/locomotion/software.html

8. Summary and Conclusions

Advances in direct computation of Lyapunov functions have enabled the development of a new class of feedback motion-planning algorithms for complicated dynamical systems. In this paper we have presented the LQR-trees algorithm, which uses Lyapunov computations to evaluate the regions of attraction of randomized trees stabilized with LQR feedback. We prove, and demonstrate through simulation examples, that this algorithm has the property of "probabilistic feedback coverage". Furthermore, initial experimental results suggest that this coverage occurs efficiently, requiring only a small number of stabilized trajectories to cover the controllable space.

Further investigation of this algorithm will likely result in a covering motion-planning strategy for underactuated systems with dimensionality greater than what is currently accessible by discretization algorithms such as dynamic programming, as well as early termination strategies that provide targeted coverage of state space in systems of much higher dimension. The resulting policies will have certificates guaranteeing their performance on the system model.

Acknowledgments

The authors wish to thank Alexandre Megretski and Pablo Parrilo for their invaluable assistance on the formulation and numerical implementation of the sums-of-squares methods. We also thank Philipp Reist and Steve LaValle for valuable discussions. This work was supported by the National Science Foundation (award numbers 0746194 and 0915148) and the Air Force Research Laboratory (contract FA8650-05-C-7262).

References

Anderson, B. D. O. and Moore, J. B. (1990). *Optimal control: linear quadratic methods*. Upper Saddle River, NJ, Prentice Hall.

Atkeson, C. G. and Stephens, B. (2008). Random sampling of states in dynamic programming. In Advances in Neural Information Processing Systems 20, Platt JC and Koller D and Singer Yand Roweis S (eds), Cambridge, MA: MIT Press. pp. 33–40.

Betts, J. T. (2001). *Practical Methods for Optimal Control Using Nonlinear Programming* (*Advances in Design and Control*). Philadelphia, PA, Society for Industrial and Applied Mathematics.

Branicky, M. and Curtiss, M. (2002). Nonlinear and hybrid control via RRTs. *Proceedings International Symposium on Mathematical Theory of Networks and Systems*.

Burridge, R. R., Rizzi, A. A. and Koditschek, D. E. (1999). Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, **18**(6): 534–555.

Chesi, G. (2009). Estimating the domain of attraction for non-polynomial systems via LMI optimizations. *Automatica*, **45**(6): 1536–1541.

Franzè, G., Casavola, A., Famularo, D. and Garone, E. (2009). An off-line MPC strategy for nonlinear systems based on SOS programming. *Nonlinear Model Predictive Control*, Magni, L., Raimondo, D. M. and Allgower, F. (eds) (*Lecture Notes in Control and Information Sciences*, Vol. 384). Berlin, Springer, pp. 491–499.

Frazzoli, E., Dahleh, M. A. and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, **25**(1): 116–129.

Gill, P. E., Murray, W. and Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, **47**(1): 99–131.

Goebel, R., Prieur, C. and Teel, A. R. (2009). Smooth patchy control Lyapunov functions. *Automatica*, **45**: 675–683.

Gollu, N. and Rodrigues, L. (2007). Control of large angle attitude maneuvers for rigid bodies using sum of squares. *American Control Conference, 2007 (ACC '07)*, July, pp. 3156–3161.

Johansen, T. A. (2000). Computation of Lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, **36**(11): 1617–1626.

Kavraki, L., Svestka, P., Latombe, J. and Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, **12**(4): 566–580.

Khalil, H. K. (2001). *Nonlinear Systems*, 3rd edition. Upper Saddle River, NJ, Prentice Hall.

Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M. and Inoue, H. (2001). Motion planning for humanoid robots under obstacle and dynamic balance constraints. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 692–698.

LaValle, S. and Kuffner, J. (2000). Rapidly-exploring random trees: Progress and prospects. *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge, Cambridge University Press.

Mason, M. (1985). The mechanics of manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 544–548.

Megretski, A. *SPOT: Systems Polynomial Optimization Tools*. http://web.mit.edu/ameg/www/.

Megretski, A. and Rantzer, A. (1997). System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, **42**(6): 819–830.

Mitchell, I. M. and Templeton, J. A. (2005). A toolbox of Hamilton-Jacobi solvers for analysis of nondeterministic continuous and hybrid systems. *Proceedings of Hybrid Systems Computation and Control*.

Nesterov, Y. and Nemirovskii, A. (1995). Interior-Point Polynomial Algorithms in Convex Programming, SIAM Review **36**(4): 682–683.

Papachristodoulou, A., Peet, M. M. and Lall, S. (2009). Analysis of polynomial systems with time delays via the sum of squares decomposition. *IEEE Transactions on Automatic Control*, **54**(5): 1058–1064.

Papachristodoulou, A. and Prajna, S. (2002). On the construction of Lyapunov functions using the sum of squares decomposition. *Proceedings of the 41st IEEE Conference on Decision and Control*.

Parrilo, P. A. (2000). *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD Thesis, California Institute of Technology.

Peterfreund, N. and Baram, Y. (1998). Convergence analysis of nonlinear dynamical systems by nested Lyapunov functions. *IEEE Transactions on Automatic Control*, **43**(8): 1179–1184.

Prajna, S., Papachristodoulou, A., Seiler, P. and Parrilo, P. A. (2004a). *SOSTOOLS Sum of Squares Optimization Toolbox for MATLAB: User's Guide*, version 2.00.

Prajna, S., Parrilo, P. and Rantzer, A. (2004b). Nonlinear control synthesis by convex optimization. *IEEE Transactions on Automatic Control*, **49**(2): 310–314.

Prajna, S. and Rantzer, A. (2007). Convex programs for temporal verification of nonlinear dynamical systems. *SIAM Journal on Control and Optimization*, **46**(3): 999–1021.

Rantzer, A. (2001). A dual to Lyapunov's stability theorem. *Systems and Control Letters*, **42**(3): 161–168.

Reist, P. and Tedrake, R. (2010). Simulation-based LQR-trees with input and state constraints. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, (in press).

Shkolnik, A. and Tedrake, R. (2009). Path planning in 1000+ dimensions using a task-space Voronoi bias. *Proceedings of the IEEE/RAS International Conference on Robotics and Automation (ICRA)*.

Slotine, J.-J. E. and Li, W. (1990). *Applied Nonlinear Control*. Upper Saddle River, NJ, Prentice Hall.

Tan, W. and Packard, A. (2008). Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming. *IEEE Transactions on Automatic Control*, **53**(2): 565–571.

Tedrake, R. (2009). LQR-Trees: Feedback motion planning on sparse randomized trees. *Proceedings of Robotics: Science and Systems (RSS)*, p. 8.

Tomlin, C. J., Mitchell, I. M., Bayen, A. M. and Oishi, M. K. M. (2005). Computational techniques for the verification and control of hybrid systems. *Multidisciplinary Methods for Analysis Optimization and Control of Complex Systems* (*Mathematics in Industry*). Berlin, Springer, pp. 151–175.

Topcu, U., Packard, A. and Seiler, P. (2008). Local stability analysis using simulations and sum-of-squares programming. *Automatica*, **44**(10): 2669–2675.