LQR-Trees: Feedback Motion Planning on Sparse Randomized Trees

Russ Tedrake
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: russt@mit.edu

***Abstract*— Recent advances in the direct computation of Lyapunov functions using convex optimization make it possible to efficiently evaluate regions of stability for smooth nonlinear systems. Here we present a feedback motion planning algorithm which uses these results to efficiently combine locally valid linear quadratic regulator (LQR) controllers into a nonlinear feedback policy which probabilistically covers the reachable area of a (bounded) state space with a region of stability, certifying that all initial conditions that are capable of reaching the goal will stabilize to the goal. We investigate the properties of this systematic nonlinear feedback control design algorithm on simple underactuated systems and discuss the potential for control of more complicated control problems like bipedal walking.**

I. Introduction

Consider the problem of stabilizing a periodic (limit cycle) trajectory for a bipedal walking robot. Although many well-developed tools exist for local stabilization[25, 15], dynamic constraints due to actuator saturation and/or underactuation limit the validity of these solutions to a small neighborhood around the nominal trajectory. Dynamic programming approaches based on discretizing the state and action spaces require potentially very fine resolution to deal with the discontinuous dynamics of impact, and require many simplifications for application to even the simplest walking models[5].

This paper aims to build on recent advances from control theory and from randomized motion planning to design efficient and general algorithms for nonlinear feedback control synthesis in nonlinear underactuated systems like bipedal walking. Specifically, the controls community has recently developed a number of efficient algorithms for direct computation of Lyapunov functions for smooth nonlinear systems, using convex optimization [9, 17]. These tools can plug into motion planning algorithms to automatically compute planning "funnels" for even very complicated dynamical systems, and open a number of interesting possibilities for algorithm development. In particular, we present the LQR-Tree algorithm, which uses locally optimal linear feedback control policies to stabilize planned trajectories computed by local trajectory optimizers, and computational Lyapunov verification based on a sum-of-squares method to create the funnels.

The aim of this work is to generate a class of algorithms capable of computing verified feedback policies for underactuated systems with dimensionality beyond what might be accessible to grid-based algorithms like dynamic programming. The use of local trajectory optimizers and local feedback stabilization scales well to higher-dimensions, and reasoning about the feedback "funnels" allows the algorithm to cover a bounded, reachable subset of state space with a relatively sparse set of trajectories. In addition, the algorithms operate directly on the continuous state and action spaces, and thus are not subject to the pitfalls of discretization. By considering feedback during the planning process, the resulting plans are certifiably robust to disturbances and quite suitable for implementation on real robots. Although scaling is the driving motivation of this approach, this paper focuses on the coverage properties of the LQR-Tree algorithm by carefully studying a simple 2D example (the torque-limited simple pendulum), which reveals the essential properties of the algorithm on a problem where the control synthesis procedure can be easily visualized.

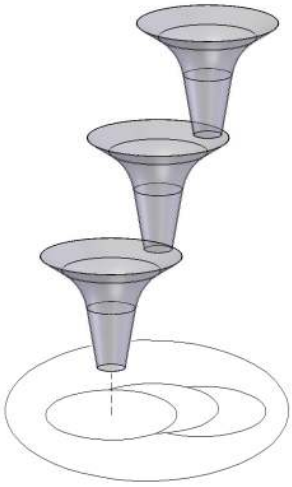


Fig. 1: Cartoon of motion planning with funnels in the spirit of [4].

II. Background

A. Feedback motion planning

For implementation on real robots, open-loop trajectories generated by a motion planning system are commonly stabilized by a feedback control system.[1] While this decoupled approach works for most problems, it is possible that a

[1]Note that an increasingly plausible alternative is real-time, dynamic re-planning.

planned trajectory is not stabilizable, or very costly to stabilize compared to other, more desirable trajectories. Algorithms which explicitly consider the feedback stabilization during the planning process can avoid this pitfall, and as we will 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 for a feedback policy which collapses a large set of initial conditions into a smaller set of final conditions[16]. Burridge, Rizzi, and Koditschek 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[4] (see Figure 1). At the time, the weakness of this approach was the difficulty in computing, or estimating by trial-and-error, the region of applicability - the mouth of the funnel, or preimage - for each local controller in a nonlinear system. Consequently, besides the particular solution in [4], these ideas have mostly been limited to reasoning about vector-fields on systems without dynamics[12].

B. Direct computation of Lyapunov functions

Burridge et al. also pointed out the strong connection between Lyapunov functions and these motion planning funnels[4]. A Lyapunov function is a differentiable positive-definite output function, $V(\mathbf{x})$, for which $\dot{V}(\mathbf{x}) < 0$ as the closed-loop dynamics of the system evolve. If these conditions are met over some ball in state space, B_r, 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 an incredibly important role in nonlinear control theory, but can be difficult to discover analytically for complicated systems.

The last few years has seen the emergence of a number of computational approaches to discovering Lyapunov functions for nonlinear systems, often based on convex optimization(e.g., [9, 17]). 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 (even with constant coefficients as free parameters) using a *sums of squares* (SOS) optimization program[17]. Sums of squares programs can be recast into semidefinite programs and solved using convex optimization solvers (such as interior point methods); the freely available SOSTOOLS library makes it quite accessible to perform these computations in MATLAB[18]. As we will see, the ability to check uniform positive (or negative) definiteness will offer the ability to verify candidate Lyapunov functions over a region of state space for smooth (nonlinear) polynomial systems.

These tools make it possible to automate the search for Lyapunov functions. Many researchers have used this capability to find stability proofs that didn't previously exist for nonlinear systems[17]. In this paper, we begin to explore the implications for planning of being able to efficiently compute planning funnels.

C. Other related work

The ideas presented here are very much inspired by the randomized motion planning literature, especially rapidly-exploring randomized trees (RRTs)[11] and probabilistic roadmaps (PRMs)[10]. This work was also inspired by [14] and [19] who point out a number of computational advantages to using sample-paths as a fundamental representation for learning policies which cover the relevant portions of state space.

In other related work, [1] 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 reachable state space with the basin of attraction. As a result, we have stronger guarantees of getting to the goal and considerably sparser collections of sample paths.

III. The LQR-Tree Algorithm

Like many other randomized planning algorithms, the proposed algorithm creates a tree of feasible trajectories by sampling randomly over some bounded region of state space, and growing the existing tree towards this random sample point. Here, when each new trajectory "branch" is added to the tree, we do some additional work by creating a trajectory stabilizing controller and by immediately estimating the basin of attraction of this controller using semi-definite programming. Because both the feedback design and the stability analysis work backwards in time, we perform these computations on only a backwards tree, starting from the goal. The result is that the backwards tree becomes a large web of local controllers which grab initial conditions and pull them towards the goal (with formal certificates of stability for the nonlinear, continuous state and action system). We terminate the algorithm when we determine (probabilistically) that all initial conditions which are capable of reaching the goal are contained in the basin of attraction of the tree.

Although many trajectory stabilizing feedback controller designs are possible (and potentially compatible with this approach), we have selected to use a time-varying linear quadratic regulator (LQR) design. LQR, iterative LQR (iLQR)[21, 23], and the closely related differential dynamic programming (DDP)[8] are common tools for roboticists, and have demonstrated success in a number of applications. LQR control synthesis has the additional benefit that it returns the quadratic cost-to-go function for the linear system, which is also a valid Lyapunov function for the nonlinear system over some region in the vicinity of the trajectory. We design a conservative approximation of this region using sums-of-squares optimization. Finally, we use the computed basin of attraction to influence the way that our tree grows, with the

goal of filling the reachable state space with the basin of attraction of a sparse set of trajectories.

The details of each of these steps are described in the remainder of this section.

A. Essential components

1) Time-varying LQR feedback stabilization: Let us first consider the subproblem of designing a time-varying LQR feedback based on a time-varying linearization along a nominal trajectory. Consider a controllable, smoothly differentiable, nonlinear system:

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

with a stabilizable goal state, $\mathbf{x}_G$. Define a nominal trajectory (a solution of equation 1) which reaches the goal in a finite time: $\mathbf{x}_0(t), \mathbf{u}_0(t)$, with $\forall t \geq t_G, \mathbf{x}_0(t) = \mathbf{x}_G$ and $\mathbf{u}_0(t) = \mathbf{u}_G$. Define

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

Now linearize the system around the trajectory, so that we have

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

Define a quadratic regulator (tracking) cost function as

$$J(\mathbf{x}', t') = \int_{t'}^{\infty} \left[\bar{\mathbf{x}}^T(t)\mathbf{Q}\bar{\mathbf{x}}(t) + \bar{\mathbf{u}}^T(t)\mathbf{R}\bar{\mathbf{u}}(t)\right] dt,$$
$$\mathbf{Q} = \mathbf{Q}^T \geq \mathbf{0}, \mathbf{R} = \mathbf{R}^T > \mathbf{0}, \mathbf{x}(t) = \mathbf{x}'.$$

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

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

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

$$-\dot{\mathbf{S}} = \mathbf{Q} - \mathbf{S}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S} + \mathbf{S}\mathbf{A} + \mathbf{A}^T\mathbf{S}, \tag{2}$$

and the boundary condition $\mathbf{S}(t_G)$ is the positive-definite solution to the equation:

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

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

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

2) LTI verification: We first estimate the basin of attraction of the linear time-invariant (LTI) feedback controller, $\mathbf{K}(t_G)$, executed for $t \geq t_G$. We verify that this controller stabilizes the fixed point given by $(\mathbf{x}_G, \mathbf{u}_G)$ by demonstrating that a function, $V(\mathbf{x})$, is a valid Lyapunov function for the nonlinear system over a bounded region of state-space, $\mathcal{B}$, defined by

$$\mathcal{B}(\rho) : \{\mathbf{x} | 0 \leq V(\mathbf{x}) \leq \rho\}$$

where ρ is a positive scalar. The origin is asymptotically stable if

- $V(\mathbf{x})$ is positive definite in $\mathcal{B}(\rho)$,

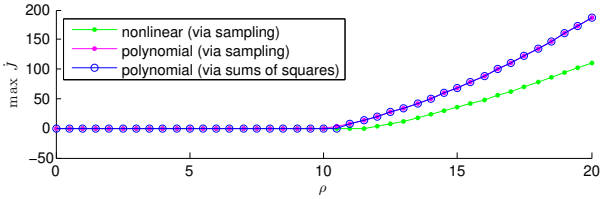


Fig. 2: Polynomial verification of LTI feedback on the damped simple pendulum ($m = 1kg, l = .5m, b = .1m^2kg/s, g = 9.8m/s^2, \mathbf{Q} = diag([10, 1]), \mathbf{R} = 15, N_f = 3, N_m = 2$).

- $\dot{V}(\mathbf{x}) < 0$ in $\mathcal{B}(\rho)$.

Furthermore, all initial conditions in $\mathcal{B}(\rho)$ will converge to 0[22].

Here we use $V(\mathbf{x}) = J^*(\mathbf{x})$; the linear optimal cost-to-go function is (locally) a Lyapunov function for the nonlinear system. The first condition is satisfied by the LQR design. For the second condition, first observe that

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

In the case where $\mathbf{f}$ is polynomial in $\mathbf{x}$ and $\mathbf{u}$, we can verify this condition exactly by specifying a sums-of-squares (SOS) feasibility program[17]:

$$\dot{J}^*(\bar{\mathbf{x}}) + h(\bar{\mathbf{x}})\left(\rho - J^*(\bar{\mathbf{x}})\right) < 0$$
$$h(\bar{\mathbf{x}}) = \mathbf{m}^T(\bar{\mathbf{x}})\mathbf{H}\mathbf{m}(\bar{\mathbf{x}}), \quad \mathbf{H} > 0,$$

where $\mathbf{m}$ is a vector of monomials of order N_m. Note that some care must be taken because $J^*(0) = 0$; we use a slack variable approach and search for solutions were $\dot{J}^*$ is uniformly less than some numerical tolerance above zero.

In many cases (including the manipulator dynamics considered in this paper), even if $\mathbf{f}$ is not polynomial it is still possible to perform the verification algebraically through a change of coordinates. However, for simplicity and generality, in the algorithm presented here we simply approximate the stability condition using a Taylor expansion of $\mathbf{f}$, with order N_f greater than one. We use $\hat{\mathbf{f}}$ to denote the Taylor expansion of $\mathbf{f}$ and $\hat{\dot{J}}^*$ for the resulting approximation of $\dot{J}^*$.

Finally, we estimate the basin of attraction by formulating a convex optimization to find find the largest region $\mathcal{B}(\rho)$ over which the second condition is also satisfied:

$$\max \rho \quad \text{subject to}$$
$$\hat{\dot{J}}^*(\bar{\mathbf{x}}) + \mathbf{m}^T(\bar{\mathbf{x}})\mathbf{H}\mathbf{m}(\bar{\mathbf{x}})\left(\rho - J^*(\bar{\mathbf{x}})\right) < 0$$
$$\rho > 0, \quad \mathbf{H} > 0.$$

The estimated basin of attraction is a conservative approximation of the true basin of attraction in every way, except that the nonlinear dynamics are approximated by the polynomial expansion. This limits our analysis to smooth nonlinear systems, and restricts our strict claims of verification in this paper to truly polynomial systems. In practice, the algorithm acquires conservative, but impressively tight approximations of the basin of attraction for the system in detailed tests with the pendulum, as illustrated in Figure 2, and the cart-pole.

3) LTV verification: Next we attempt to verify the performance of the linear time-varying feedback over the time $t \in [0, t_G]$. Rather than stability, we specify a bounded region of state space, $\mathcal{B}_f$, (the outlet of the funnel) and search for a time-varying region, $\mathcal{B}(t)$, (the funnel) where

$$\mathcal{B}(t) : \{\mathbf{x} | \mathbf{F}(\mathbf{x}, t) \in \mathcal{B}_f\}, \tag{3}$$

and $\mathbf{F}(\mathbf{x}, t)$ is defined as the simulation function which integrates the closed-loop dynamics from t to t_f. When B_f is chosen as the LTI basin of attraction from the previous section, this funnel becomes the basin of attraction of the infinite-horizon trajectory. As before, we will use the cost-to-go as a (now time-varying) storage function, $V(\mathbf{x}, t)$, and search for the largest positive time-varying level-set, $\rho(t)$, over the interval $[t_0, t_f]$, which defines a region,

$$\mathcal{B}(\rho(\cdot), t) : \{\mathbf{x} | 0 \leq V(\mathbf{x}, t) \leq \rho(t)\},$$

satisfying condition 3. Similarly, we use

$$\mathcal{B}_f : \{\mathbf{x} | 0 \leq V(\mathbf{x}, t_f) \leq \rho_f\},$$

where ρ_f is a positive constant representing the constraint on final values (specified by the task). Note that this naturally implies that $\rho(t_f) \leq \rho_f$.

A sufficient, but conservative, verification of our bounded final value condition can be accomplished by verifying that $B(\rho(\cdot), t)$ is a closed set over $t \in [t_0, t_f]$. The set is closed if $\forall t \in [t_0, t_f]$ we have

- $V(\mathbf{x}, t) \geq 0$ in $\mathcal{B}(\rho(\cdot), t)$,
- $\dot{V}(\mathbf{x}, t) \leq \dot{\rho}(t)$ in $\mathcal{B}^\sharp(\rho(\cdot), t)$,

where $\mathcal{B}^\sharp$ is the boundary of the region $\mathcal{B}$,

$$\mathcal{B}^\sharp(\rho(\cdot), t) : \{\mathbf{x} | V(\mathbf{x}, t) = \rho(t)\}.$$

Again, we choose here to use $V(\mathbf{x}, t) = J^*(\mathbf{x}, t)$; the first condition is again satisfied by the LQR derivation which ensures $\mathbf{S}(t)$ is uniformly positive definite. Now we have

$$\dot{J}^*(\bar{\mathbf{x}}, t) = 2\bar{\mathbf{x}}^T \mathbf{S}(t) \mathbf{f}\left(\mathbf{x}_0(t) + \bar{\mathbf{x}}, \mathbf{u}_0(t) - \mathbf{K}(t)\bar{\mathbf{x}}\right) + \bar{\mathbf{x}}^T \dot{\mathbf{S}}(t) \bar{\mathbf{x}}. \tag{4}$$

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 are the result of numerical integration (e.g., with `ode45` in Matlab). We will approximate this temporal dependence with (elementwise) piecewise polynomials using splines of order N_t, where N_t is often chosen to be 3 (cubic splines), with the knot points at the timesteps output by the variable step integration, which we denote $t_0, t_1, ..., t_N$, with $t_N = t_f$, e.g.:

$$\forall t \in [t_k, t_k + 1], \quad S_{ij}(t) \approx \sum_{m=0}^{N_t} \alpha_{ijm} (t - t_k)^m = \hat{S}_{ij}(t),$$
$$\hat{J}^*(\bar{\mathbf{x}}, t) = \bar{\mathbf{x}}^T \hat{\mathbf{S}} \bar{\mathbf{x}}.$$

Once again, we substitute a Taylor expansion of the dynamics to obtain the estimate $\hat{\dot{J}}^*$.

Now we approximately verify the second condition by formulating a series of sums-of-squares feasibility programs

$$\hat{\dot{J}}^*(\bar{\mathbf{x}}, t) - \dot{\rho}(t) + h_1(\bar{\mathbf{x}}, t) \left(\rho(t) - \hat{J}^*(\bar{\mathbf{x}}, t)\right)$$
$$+ h_2(\bar{\mathbf{x}}, t)(t - t_k) + h_3(\bar{\mathbf{x}}, t)\,(t_{k+1} - t) \leq 0, \tag{5}$$
$$h_1(\bar{\mathbf{x}}, t) = \mathbf{h}_1^T \mathbf{m}(\bar{\mathbf{x}}, t), \tag{6}$$
$$h_2(\bar{\mathbf{x}}, t) = \mathbf{m}^T(\bar{\mathbf{x}}, t) \mathbf{H}_2 \mathbf{m}(\bar{\mathbf{x}}, t), \quad \mathbf{H}_2 = \mathbf{H}_2^T > 0, \tag{7}$$
$$h_3(\bar{\mathbf{x}}, t) = \mathbf{m}^T(\bar{\mathbf{x}}, t) \mathbf{H}_3 \mathbf{m}(\bar{\mathbf{x}}, t), \quad \mathbf{H}_3 = \mathbf{H}_3^T > 0, \tag{8}$$

for $k = N - 1, ..., 1$.

We attempt to find the largest $\rho(t)$ satisfying the verification test above by defining a piecewise-polynomial of order N_ρ given by

$$\rho_k(t) = \sum_{m=0}^{N_\rho} \beta_{km} (t - t_k)^m,$$
$$\rho(t) = \begin{cases} \rho_k(t), & \forall t \in [t_k, t_{k+1}) \\ \rho_f, & t = t_f, \end{cases}$$

and we formulate the optimization:

$$\max_\beta \int_{t_k}^{t_{k+1}} \rho_k(t) dt, \quad \text{subject to}$$
$$\rho_k(t_{k+1}) \leq \rho_{k+1}(t_{k+1}), \text{ equations (5) - (8),}$$

for all $k = N - 1, ..., 1$.

4) Growing the tree: Another essential component of the LQR-tree algorithm is the method by which the backwards tree is extended. Following the RRT approach, we select a sample at random from some distribution over the state space, and attempt to grow the tree towards that sample. Unfortunately, RRTs typically do not grow very efficiently in differentially constrained (e.g., underactuated) systems, because simple distance metrics like the Euclidean distance are inefficient in determining which node in the tree to extend from. Further embracing LQR as a tool for motion planning, in this section we develop an affine quadratic regulator around the sample point, then use the resulting cost-to-go function to determine which node to extend from, and use the open-loop optimal policy to extend the tree.

Choose a random sample (not necessarily a fixed point) in state space, $\mathbf{x}_s$ and a default $\mathbf{u}_0$, and use $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}_s$, $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}_0$.

$$\begin{aligned} \dot{\bar{\mathbf{x}}} =& \frac{d}{dt}(\mathbf{x}(t) - \mathbf{x}_s) = \dot{\mathbf{x}}(t) \\ \approx& \mathbf{f}(\mathbf{x}_s, \mathbf{u}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}(t) - \mathbf{x}_s) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u} - \mathbf{u}_0) \\ =& \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}} + \mathbf{c}. \end{aligned}$$

Now define an affine quadratic regulator problem with a hard constraint on the final state, but with the final time, t_f, left as a free variable[13]:

$$J(\bar{\mathbf{x}}_0, t_0, t_f) = \int_{t_0}^{t_f} \left[1 + \frac{1}{2} \bar{\mathbf{u}}^T(t) \mathbf{R} \bar{\mathbf{u}}(t)\right] dt,$$
$$\text{s.t.} \quad \bar{\mathbf{x}}(t_f) = \mathbf{0}, \ \bar{\mathbf{x}}(t_0) = \bar{\mathbf{x}}_0, \ \dot{\bar{\mathbf{x}}} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\bar{\mathbf{u}} + \mathbf{c}.$$

Without loss of generality (since the dynamics are autonomous), we will use $J(\bar{\mathbf{x}}_0, t_f - t_0)$ as a shorthand for $J(\bar{\mathbf{x}}_0, t_0, t_f)$. It can be shown that the optimal (open-loop) control is

$$\bar{\mathbf{u}}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^T e^{\mathbf{A}^T(t_f - t)}\mathbf{P}^{-1}(t_f)\mathbf{d}(\bar{\mathbf{x}}(t_0), t_f),$$

where

$$\dot{\mathbf{P}}(t) = \mathbf{A}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{A}^T + \mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T, \quad \mathbf{P}(t_0) = \mathbf{0}$$
$$\mathbf{d}(\bar{\mathbf{x}}, t) = \mathbf{r}(t) + e^{\mathbf{A}t}\bar{\mathbf{x}}, \quad \dot{\mathbf{r}}(t) = \mathbf{A}\mathbf{r}(t) + \mathbf{c}, \quad \mathbf{r}(\bar{\mathbf{x}}, t_0) = \mathbf{0}$$

and the resulting cost-to-go is

$$J^*(\bar{\mathbf{x}}, t_f) = t_f + \frac{1}{2}\mathbf{d}^T(\bar{\mathbf{x}}, t_f)\mathbf{P}^{-1}(t_f)\mathbf{d}(\bar{\mathbf{x}}, t_f).$$

Thanks to the structure of this equation, it is surprisingly efficient to compute the cost-to-go from many initial conditions (here the existing vertices in the tree) simultaneously. For each $\bar{\mathbf{x}}$ the horizon time, $t_f^* = \text{argmin}_{t_f} J^*(\bar{\mathbf{x}}, t_f)$, is found by selecting the minimum after integrating $\mathbf{P}(t)$ and $\mathbf{r}(t)$ over a fixed horizon. This cost-to-go function provides a relatively efficient *dynamic* distance metric[2] for the RRT expansion which performs much better than Euclidean metrics for underactuated systems[6].

Once the "closest" node in the existing tree is identified, by this LQR distance metric, the tree is extended by applying a series of actions backwards in time from the closest node. The initial guess for this series of actions is given by $\bar{\mathbf{u}}^*(t)$ from the LQR distance metric, but this estimate (which is only accurate in the neighborhood of the sample point) can be further refined by a fast, local, nonlinear trajectory optimization routine. In the current results, we use a direct collocation[24, 2] implementation using the formulation from equation III-A.4, but with the nonlinear dynamics. If the direct collocation method cannot satisfy the final value constraint, then the point is considered (temporarily) unreachable, and is discarded. Interestingly, using the LQR open-loop control to initialize the nonlinear optimization appears to help overcome many of the local minima in the nonlinear optimization process.

5) A sampling heuristic: Finally, we take advantage of the Lyapunov verification by changing the sampling distribution. Adding branches of the tree that will be contained by the existing basin of attraction has little value. The sampling heuristic used here is implemented by sampling uniformly over the desired subset of state space, then rejecting any sample which are already in the basin of attraction of any of the tree branches. This "collision checking" is very inexpensive; it is far more expensive to add a useless node into the tree. Other sampling distributions are possible, too. One interesting alternative is sampling from states that are just at the edges of the basin of attraction, e.g, $\forall_i J^*(\mathbf{x} - \mathbf{x}_0^i, t) > \rho_i(t), \exists_j J^*(\mathbf{x} - \mathbf{x}_0^i, t) \leq 1.5\rho_j(t)$.

[2]Note that it is not technically a distance metric, since it is not symmetric, but the RRT does not require symmetry.

B. 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 $J^*(\bar{\mathbf{x}}, t) = \bar{\mathbf{x}}^T\mathbf{S}\bar{\mathbf{x}}$ 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, $\rho(t)$ is described by the vector of polynomial coefficients ρ_c, and i is a pointer to the parent node.

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

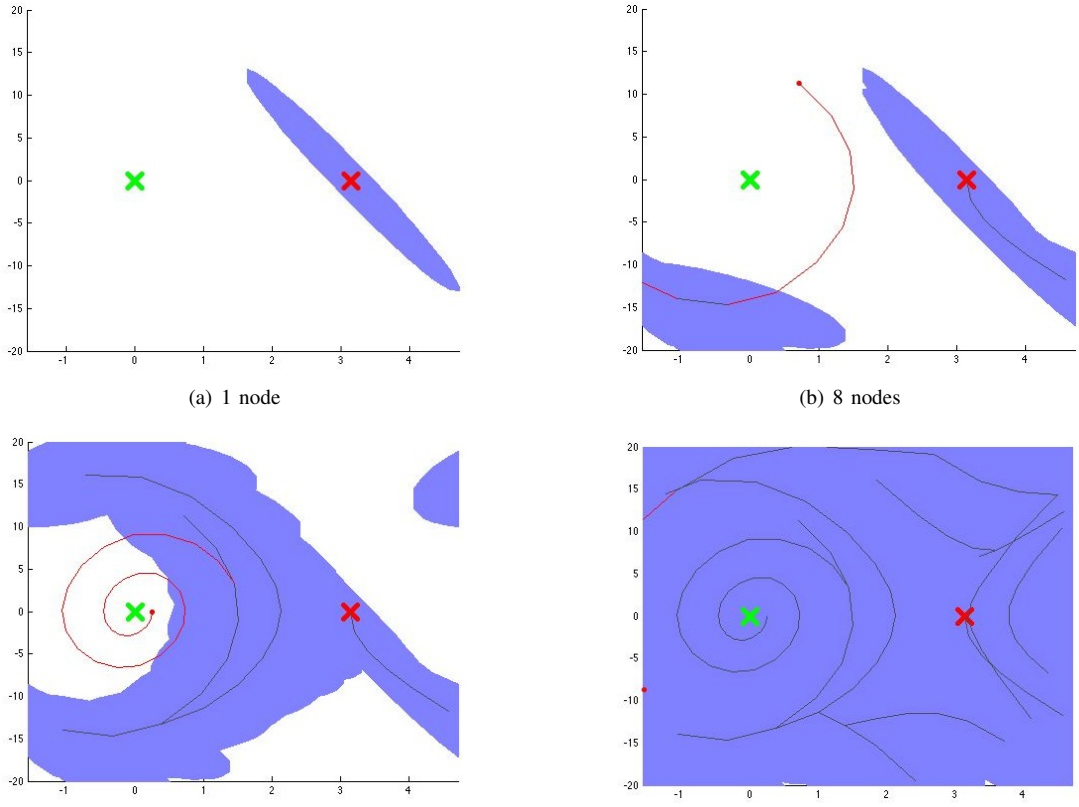
```
[A, B] ⇐ linearization of f(x, u) around x_G, u_G
[K, S] ⇐ LQR(A, B, Q, R)
ρ_c ⇐ level-set computed as described in section III-A.2
T.init({x_g, u_g, S, K, ρ_c, NULL})
for k = 1 to K do
    x_rand ⇐ random sample as described in section III-A.5; if no samples are found, then FINISH
    x_near from cost-to-go distance metric described in section III-A.4
    u_tape from extend operation described in section III-A.4
    for each u in u_tape do
        x ⇐ Integrate backwards from x_near with action u
        [K, S] from LQR derivation in section III-A.1
        ρ_c ⇐ level-set computed as in section III-A.3
        i ⇐ pointer to node containing x_near
        T.add-node(x, u, S, K, ρ_c, i)
        x_near ⇐ x
    end for
end for
```

Execution of the LQR-tree policy is accomplished by selecting any node in the tree with a basin of attraction which contains the initial conditions, $\mathbf{x}(0)$, and following the time-varying feedback policy along that branch all of the way to the goal.

IV. Simulations

Simulation experiments on a two-dimensional toy problem have proven very useful for understanding the dynamics of the algorithm. Figure 3 tells the story fairly succinctly. The algorithm was tested on a simple pendulum, $I\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = \tau$, with $m = 1, l = .5, b = .1, I = ml^2, g = 9.8$. Here $\mathbf{x} = [\theta, \dot{\theta}]^T$ and $\mathbf{u} = \tau$. The parameters of the LQR-tree algorithm were $\mathbf{x}_G = [\pi, 0]^T$, $\mathbf{u}_G = 0$, $\mathbf{Q} = diag([10, 1])$, $\mathbf{R} = 15$, $N_f = 3$, $N_m = 2$, $N_x = 3$, $N_S = 3$.

Figure 3(a) shows the basin of attraction (blue oval) after computing the linear time-invariant (LTI) LQR solution around the unstable equilibrium. Figure 3(b) shows the entire trajectory to the first random sample point (red dot), and the funnels that have been computed so far for the second-half of the trajectory. Note that the state-space of the pendulum lives on a cylinder, and that the trajectory (and basin of attraction) wraps around from the left to the right. Plots (c-d) show the basin of attraction as it grows to fill the state space. The final tree in Figure 3(d) also reveals three instances where



(a) 1 node (b) 8 nodes

(c) 24 nodes (d) 104 nodes

Fig. 3: An LQR-tree for the simple pendulum. The x-axis is $\theta \in [-\pi/2, 3\pi/2]$ (note that the state wraps around this axis), and the y-axis is $\dot{\theta} \in [-20, 20]$. The green X (on the left) represents the stable fixed point; the red X (on the right) represents the unstable (upright) fixed point. The blue ovals represent the "funnels," sampled at every node.

the trajectories on the tree cross - this is a result of having an imperfect distance metric.

Note that state $\mathbf{x} = [0, 0]^T$, corresponding to the stable fixed-point of the unactuated pendulum, is covered by the basin of attraction after 32 nodes have been added. The algorithm was not biased in any way towards this state, but this bias can be added easily. The entire space is probabilistically covered (1000 random points chosen sequentially were all in the basin of attraction) after the tree contained just 104 nodes. On average, the algorithm terminates after 146 nodes for the simple pendulum with these parameters. For contrast, [3] shows a well-tuned single-directional RRT for the simple pendulum which has 5600 nodes. However the cost of adding each node is considerably greater here than in the traditional RRT, dominated by the line search used to maximize the estimated region of stability. The entire algorithm runs in about two minutes on a laptop, without any attempt to optimize the code.

V. Discussion

A. Properties of the algorithm

Recall that for nonlinear systems described by a polynomial of degree $\leq N_f$, the verification procedures used here are conservative; the true basin of attraction completely contains the estimated stability region. In practice, this is often (but not provably) the case for more general smooth nonlinear systems.

Proposition 1: For nonlinear systems described by a polynomial of degree $\leq N_f$, the LQR-tree algorithm probabilistically covers the sampled portion of the reachable state space with a stabilizing controller and a Lyapunov function, thereby guaranteeing that all initial conditions which are capable of reaching the goal will stabilize to the goal.

Proving proposition 1 carefully requires a proof that the local trajectory optimizer is always capable of solving a trajectory to a reachable point in the state space that is in an ϵ-region outside the existing basin of attraction. This is likely the case, seeing as the nonlinear optimizer is seeded by a linear optimal control result which will be accurate over some region of similar size to the basin of attraction ellipse. However, the full proof is left for future work.

Perhaps even more exciting is the fact that, in the model explored, this coverage appears to happen rapidly and allow for fast termination of the algorithm. The pendulum is a surprisingly rich test system - for example, as key parameters such as $\mathbf{R}$ or b change, the size of the funnels can change dramatically, resulting in quite different feedback policy coverings of the state space, and always resulting in rapid coverage.

It is also worth noting that the trajectories out of a more standard RRT are typically smoothed. Trajectories of the closed-loop system which result from the LQR algorithm are (qualitatively) quite smooth, despite coming from a randomized algorithm. The LQR stabilizing controller effectively smoothes the trajectory throughout state space.

B. Straight-forward variations in the algorithm

- **Compatible with optimal trajectories**. The LQR-tree algorithm provides a relatively efficient way to fill the reachable state space with funnels, but does not stake any claim on 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 (e.g., using [1]), then use the random exploration to fill in any missing regions.
- **Early termination**. For higher dimensional problems, covering the reachable 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* defined by the basin of attraction of the backwards tree.
- **Finite-horizon trajectories**. The LQR stabilization derived in section III-A.1 was based on infinite horizon trajectories. This point was necessary in order to use the language of basins of attraction and asymptotic stabilization. Finite-horizon problems can use all of the same tools (though perhaps not the same language), but must define success as being inside some finite volume around the goal state at t_G. Funnels connecting to this volume are then computed using the same Riccati backup.

C. Controlling walking robots

A feedback motion planning algorithm like the LQR-tree algorithm could be a very natural control solution for walking robots, or other periodic control systems. In this case, rather than the goal of the tree being specified as a point, the goal would be a periodic (limit cycle) trajectory. This could be implemented in the tree as a set of goal states, which happen to be connected, and the basin of attraction of this goal would emerge from the periodic steady-state solution of the Riccati equation and verification process on the limit cycle. Limit cycles for walking systems in particular are often described as a hybrid dynamics punctuated by discrete impacts. These discrete jump events must be handled with care in the feedback design and verification, but are not fundamentally incompatible with the approach[20].

Figure 4 cartoons the vision of how the algorithm would play out for the well-known compass gait biped[7]. On the left is a plot of the (passively stable) limit cycle generated by the compass gait model walking down a small incline. This trajectory can be stabilized using a (periodic) time-varying linearization and LQR feedback, and the resulting basin of attraction might look something like the shaded region in Figure 4(a). The goal of the LQR-tree algorithm would then be to fill the remaining portion of state space with transient "maneuvers" to return the system to the nominal limit cycle. A potential solution after a few iterations of the algorithm is cartooned in Figure 4(b). This work would naturally build on previous work on planning in hybrid systems (e.g.,[3]).

D. Multi-query algorithms

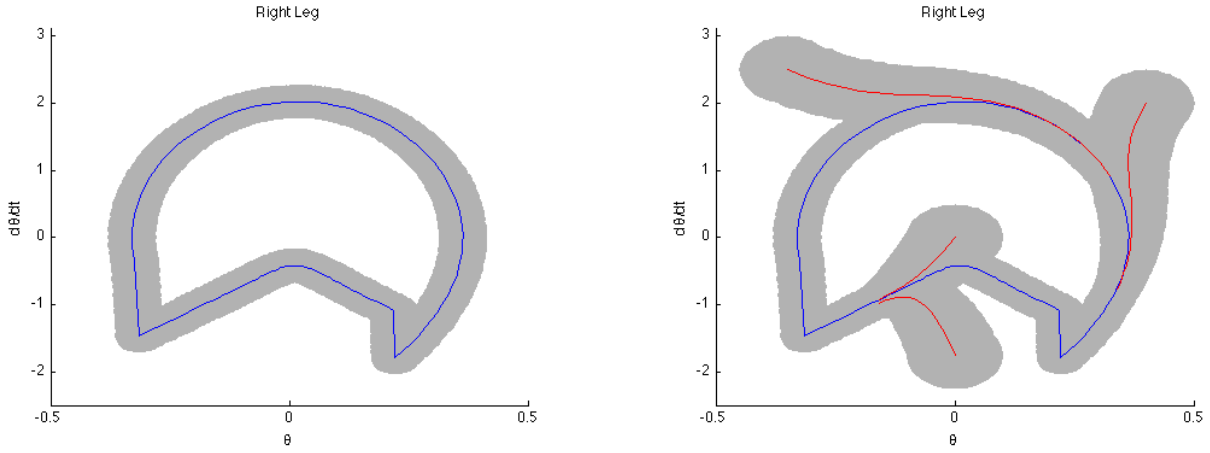
Another very interesting question is the question of reusing the previous computational work when the goal state is changed. In the pendulum example, consider having a new goal state, $\mathbf{x}_G = [\pi + 0.1, 0]^T$ - this would of course require a non-zero torque to stabilize. To what extent could the tree generated for stabilizing $\mathbf{x}_G = [\pi, 0]^T$ 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 can be preserved, but naively, one would think that all of the stabilizing controllers and the verification would have to be re-calculated. Interestingly, there is also a middle-road, in which 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 could be accomplished very efficiently, by just propagating a new ρ_{max} through the tree, but might come at the cost of losing coverage. One reason why this multi-query question is so exciting is that the problem of controlling a robot to walk on rough terrain could be nicely formulated as a multi-query stabilization of the limit cycle dynamics from Figure 4.

E. A software distribution

A MATLAB toolbox implementing the LQR-Tree algorithm is available at `http://groups.csail.mit.edu/locomotion/software.html`.

VI. Summary and Conclusions

Recent advances in direct computation of Lyapunov functions have enabled a new class of feedback motion planning algorithms for complicated dynamical systems. This paper presented the LQR-Tree algorithm which uses Lyapunov computations to evaluate the basins of attraction of randomized trees stabilized with LQR feedback. Careful investigations on a torque-limited simple pendulum revealed that, by modifying the sampling distribution to only accept samples outside of the computed basin of attraction of the existing tree, the result was a very sparse tree which covered the state space with a basin of attraction.



(a) The initial tree

(b) After expanding four random nodes

Fig. 4: Sketch of LQR-trees on the compass gait biped.

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

ACKNOWLEDGMENT

The author gratefully acknowledges Alexandre Megretski for introducing me to sums of squares optimization and for many helpful discussions.

REFERENCES

[1] Christopher G. Atkeson and Benjamin Stephens. Random sampling of states in dynamic programming. In *Advances in Neural Information Processing Systems*, 2008.

[2] John T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Advances in Design and Control. Society for Industrial and Applied Mathematics, 2001.

[3] Michael Branicky and Michael Curtiss. Nonlinear and hybrid control via RRTs. *Proc. Intl. Symp. on Mathematical Theory of Networks and Systems*, 2002.

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

[5] Katie Byl and Russ Tedrake. Approximate optimal control of the compass gait on rough terrain. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[6] Elena Glassman and Russ Tedrake. Rapidly exploring state space. *In Progress*, 2009.

[7] A. Goswami, B. Espiau, and A. Keramane. Limit cycles and their stability in a passive bipedal gait. pages 246–251. IEEE International Conference on Robotics and Automation (ICRA), 1996.

[8] David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., 1970.

[9] Tor A. Johansen. Computation of lyapunov functions for smooth nonlinear systems using convex optimization. *Automatica*, 36(11):1617 – 1626, 2000.

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

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

[12] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[13] Frank L. Lewis. *Applied Optimal Control and Estimation*. Digital Signal Processing Series. Prentice Hall and Texas Instruments, 1992.

[14] Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. Technical Report TR-2006-35, University of Massachusetts, Department of Computer Science, July 2006.

[15] Ian R. Manchester, Uwe Mettin, Fumiya Iida, and Russ Tedrake. Stable dynamic walking over rough terrain: Theory and experiment. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2009.

[16] M.T. Mason. The mechanics of manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 544–548. IEEE, 1985.

[17] Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, May 18 2000.

[18] Stephen Prajna, Antonis Papachristodoulou, Peter Seiler, and Pablo A. Parrilo. *SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB Users guide*, 2.00 edition, June 1 2004.

[19] Khashayar Rohanimanesh, Nicholas Roy, and Russ Tedrake. Towards feature selection in actor-critic algorithms. Technical report, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, 2007.

[20] A.S. Shiriaev, L.B. Freidovich, and I.R. Manchester. Can we make a robot ballerina perform a pirouette? orbital stabilization of periodic motions of underactuated mechanical systems. *Annual Reviews in Control*, 2008.

[21] Athanasios Sideris and James E. Bobrow. A fast sequential linear quadratic algorithm for solving unconstrained nonlinear optimal control problems, February 2005.

[22] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, October 1990.

[23] Emanuel Todorov and Weiwei Li. Iterative linear-quadratic regulator design for nonlinear biological movement systems. volume 1, pages 222–229. International Conference on Informatics in Control, Automation and Robotics, 2004.

[24] Oskar von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control, (International Series in Numerical Mathematics 111)*, pages 129–143, 1993.

[25] E. R. Westervelt, B. Morris, and K. D. Farrell. Analysis results and tools for the control of planar bipedal gaits using hybrid zero dynamics. *Autonomous Robots*, 23:131–145, Jul 2007.