

Towards Planning and Control of Hybrid Systems with Limit Cycle using LQR Trees

Ramkumar Natarajan^{1,2}

Siddharthan Rajasekaran^{1,2}

Jonathan D. Taylor³

Abstract—We present a multi-query recovery policy for a hybrid system with goal limit cycle. The sample trajectories and the hybrid limit cycle of the dynamical system are stabilized using locally valid Time Varying LQR controller policies which probabilistically cover a bounded region of state space. The original LQR Tree algorithm builds such trees for non-linear static and non-hybrid systems like a pendulum or a cart-pole. We leverage the idea of LQR trees to plan with a continuous control set, unlike methods that rely on discretization like dynamic programming to plan for hybrid dynamical systems where it is hard to capture the exact event of discrete transition. We test the algorithm on a compass gait model by stabilizing a dynamic walking hybrid limit cycle with point foot contact from random initial conditions. We show results from the simulation where the system comes back to a stable behavior with initial position or velocity perturbation and noise.

I. INTRODUCTION

Bipedal walking robots can easily access man-made environments and enable friendly interaction with humans, but such systems are highly nonlinear with hybrid dynamics posing a challenge to conventional control designs. Controllers designed to track a trajectory for highly nonlinear systems provide only a reflex policy and can at most succeed only in a local region of attraction around the target trajectory being tracked. To find a policy from any given initial state one can discretize the state space and use dynamic programming. However in the problem of stabilizing a hybrid system, which has continuous dynamics punctuated by discrete transitions, one must discretize the system using a fine resolution to capture the jump event with reasonable tolerance. An approach to control a compass gait where many approximations are made to adapt the existing methods to hybrid systems is described in [10].

Also, there is a large variety of robots that consist of a logical discrete event decision-making system interacting with a continuous time process. The action capabilities of such robots cannot be captured with a single controller. These type of hybrid systems have multiple system dynamics governing their behavior and can therefore have multiple discontinuities in any of their trajectory in the state space. With the increase in the development of highly sophisticated robots, the need for a hybrid controller is imperative. Our work focuses on extending a relatively new and a continuous method of controller design to such hybrid systems.

This approach is inspired by randomized feedback motion planning methods, which cover a bounded region of the state space with locally valid linear quadratic regulator (LQR) policies [11] that lead to a target trajectory. Tedrake [11] has mentioned LQR trees as a scalable algorithm that could avoid the pitfall of discretization in dynamic programming and also use continuous control actions. However, the original algorithm did not consider a hybrid system in its formulation and was tested only on a relatively simpler platform like a pendulum. Also, the original method only stabilizes the region of attraction of LQR controller around a stabilizable fixed point. We focus on extending the LQR trees to stabilize a periodic limit cycle of a hybrid system. That is, we would cover a bounded region of state space with funnels around sample trajectories that lead to the region of attraction around the goal limit cycle of a hybrid system. The funnels are regions of attraction around a trajectory created by a controller that act as a vacuum tube and sucks any state inside the tube so that they do not leave the tube while driving the state towards the end of the trajectory. These regions of attraction can be found using Lyapunov function verification. Tedrake [17] shows that recent methods that use Sums-of-Squares (SoS) optimization to verify Lyapunov functions allow very fast determination of these regions of attraction.

Reist *et. al*, [16] provide a very similar algorithm which verifies the funnels empirically through simulation instead of using SoS optimization. This method mainly aims at simplifying the implementation of the verification process at the cost of time complexity. It is tested on a more complex, 2 DoF cart-pole system. Here the size of a funnel is initialized to a very high value and is trimmed for every sample that fails to reach the goal under the time varying LQR policy of the nominal trajectory. Hence the method requires a huge number of samples, both to build the tree and to verify funnels. However, the number of nodes in the tree and several other results will be unchanged even while using the formal verification. Hence we will be comparing our results to this implementation.¹

The original LQR tree algorithm ensures that any sequence of such funnels ends at the region of attraction of the final controller that stabilizes the system at a fixed point. In this paper, we design sequences of funnels that end at the goal limit cycle. For this, we ensure that the size of each funnel, at their end, which directly connects to the limit cycle is

¹This work was equally contributed by these two authors.

² Ramkumar Natarajan <rnatarajan@wpi.edu> and Siddharthan Rajasekaran <sperundurairajas@wpi.edu>, Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA 01609.

³Dr. Jonathan D. Taylor <jdtaylor@andrew.cmu.edu>, Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

¹The original implementation only shows results tested on a pendulum. A 2-DoF cart-pole system would be better to compare with since a compass gait is a 2 DoF system too.

less than the funnel of the limit cycle itself. Also, we use trajectory optimization to generate hybrid trajectories instead of a single continuous trajectory which enables our algorithm to work for hybrid systems.

II. RELATED WORK

Stabilizing the gait has been a great interest among robotics researchers and falls into two broad classes: 1) Based on the Zero Moment Pole (ZMP) principle 2) Passive dynamic and limit cycle walkers. Robots like Honda ASIMO uses the ZMP technique [7] to keep the dynamical degrees of freedom fully actuated. The criteria in ZMP is to keep the center of pressure within the support polygon of the stance foot. However, the motions resulting from these techniques are very unnatural and inefficient.

There is a large body of work that aims to find the most stable limit cycle. Dai *et al.* [18] formulates an optimization problem to find the limit cycle that maximizes its robustness against external disturbances. They do this by minimizing the mean controller costs of limit cycle states encountering collision so that the reset states post impact lie close to the limit cycle. But our method is indifferent to any such goal limit cycle and brings the system back to a given goal limit cycle trajectory from far outside the region of attraction.

To deal with states that are far outside this region of attraction, [5] and [12] plan trajectories beforehand which can act as look-up table that provides a policy for any given initial state. The recent trend in verifying a Lyapunov function using convex optimization techniques gave birth to several effective methods like [11] and [16] that did not exist before due to the time overhead in verifying Lyapunov functions. Numerical methods for computing regions of finite-time invariance [15] (“verification of funnels”) around solutions of polynomial differential equations is extensively used in this paper. In fact, the idea is to cover the state space with the funnels around the sample trajectories that lead to the region of attraction of the goal trajectory. These funnels have non-zero volumes in state space and hence can effectively cover a bounded region filled with nominal trajectories. Moreover, it takes relatively less number of funnels to cover the entire region of state space compared to the number of nodes in other methods like Probabilistic Road Map (PRM) [2] to cover a given region.

There are several previous works that describe the usage of sample paths as fundamental representation of policies. Initial attempts to use sampling based planner to control nonlinear hybrid systems uses Rapidly-exploring Random Tree (RRT) [5]. Here, the nearest state in the tree to a state sampled at random is forward simulated with a random control input. Thus, due to forward simulation, every edge in the tree is a feasible trajectory with which we have a policy to go from the start node to any other node in the tree (which may include goal if a path is feasible). Most of the work in this field following [5] can be categorized as improving the 1) Sampling distribution [12] 2) Distance metric [11], [16], [21] and 3) Extend operation [11], [16]. LQR Tree algorithm [11] uses the controller cost function as

distance metric which improves the success rate of finding a trajectory using direct collocation [1] from the nearest point on the tree to the sampled point (extend operation). Our work focuses on extending the capability of LQR Trees to stabilize a hybrid trajectory.

The remainder of this paper is outlined as follows: Section III sets the mathematical premise for the proposed extension. Section IV explains the proposed method in detail, viz., the basic principle of Time Varying Linear Quadratic Regulator (TVLQR) and how it is used to stabilize a goal limit cycle, the estimation of the limit cycle itself, the details involved in using the direct collocation trajectory optimization for hybrid trajectories and putting it together to cover the bounded region of state space with recovery policies. The testbed and the collision dynamics that is responsible for the discrete jump in the state space are described in section V. The algorithm is experimentally evaluated using the simulation of compass gait in section VI. Section VII and VIII offer a discussion and concluding remarks.

III. PROBLEM STATEMENT

Consider a hybrid system with continuous dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ and discrete transitions at \mathbf{x}_{guard} with stabilizable goal limit cycle given by $\{\mathbf{x}_0^l(t), \mathbf{u}_0^l(t)\}$ where $\mathbf{x}_0^l(t)$ is the limit cycle trajectory and $\mathbf{u}_0^l(t)$ is the open loop control law. Let \mathbb{X} be the entire state space of this system and \mathbb{X}_S be the set of stabilizable states of this space. The LQR tree algorithm has a number of feedback stabilized sample trajectories which we’ll denote using $\{\mathbf{x}^i(t), \mathbf{u}^i(t)\}$ for i^{th} trajectory in the tree. Let each trajectory start at time t_0^i and end at time t_f^i . We use a controller c^i to stabilize the system around every i^{th} trajectory which results in a region of attraction F^i around this trajectory such that $F^i \in \mathbb{X}_S$. From the LQR trees for any i we have, $\mathbf{x}^i(t_f^i) \in F^j$ for some j . We design the system such that, the same holds for every trajectory and the system always ends in the respective parent’s funnel at the end of each child’s trajectory ultimately leading to the funnel of goal limit cycle, F^l . Our primary objective is that, as the number of sample trajectories increase, the union of all the funnels cover the entire stabilizable state space by also accounting for the discrete state transitions *i.e.*, $\left(\lim_{n \rightarrow \infty} \bigcup_{i=1}^n F^i\right) \setminus \mathbb{X}_S = \emptyset$.

IV. METHOD

A. TVLQR feedback stabilization for hybrid systems

We use a TVLQR to stabilize the system around a given trajectory. We first explain the theory behind TVLQR controller which is necessary to estimate the funnels. Let us consider the sub-problem of designing a time-varying LQR feedback based on a time-varying linearization along a nominal trajectory. Consider a smoothly differentiable, nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ with stabilizable limit cycle trajectory, $\mathbf{x}_0^l(t)$ and $\mathbf{u}_0^l(t)$. Let $\mathbf{A}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \Big|_l$, $\mathbf{B}(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \Big|_l$ be the linearization of system dynamics with respect to state and input respectively. For now, assume we have the optimal cost-to-go matrix of TVLQR controller (around the limit cycle)

$\mathbf{S}^l(t)$ for the limit cycle trajectory. Determination of $\mathbf{S}^l(t)$ is discussed later in this subsection.

The optimal cost-to-go for any nominal trajectory $\{\mathbf{x}_0(t), \mathbf{u}_0(t)\}$ of the controller is given by [11] as,

$$J^*(\bar{\mathbf{x}}, t) = \bar{\mathbf{x}}^T(t) \mathbf{S}(t) \bar{\mathbf{x}}(t), \quad \mathbf{S}(t) = \mathbf{S}^T(t) \quad (1)$$

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

$$-\dot{\mathbf{S}} = \mathbf{Q} - \mathbf{SBR}^{-1}\mathbf{B}^T\mathbf{S} + \mathbf{SA} + \mathbf{A}^T\mathbf{S} \quad (2)$$

and the optimal feedback policy of TVLQR controller is

$$\bar{\mathbf{u}}^*(t) = -\mathbf{R}^{-1}\mathbf{B}^T(t)\mathbf{S}(t)\bar{\mathbf{x}}(t) = -\mathbf{K}(t)\bar{\mathbf{x}}(t) \quad (3)$$

where $\bar{\mathbf{x}}(t)$, $\bar{\mathbf{u}}(t)$ are the state and input deviations from the nominal values $\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$, \mathbf{Q} and \mathbf{R} penalize $\bar{\mathbf{x}}(t)$ and $\bar{\mathbf{u}}(t)$ respectively in the LQR cost function. From the above description of obtaining $\mathbf{S}(t)$ for a trajectory we can conclude - 1) We need the cost to go matrix at final time step, $\mathbf{Q}_f = \mathbf{S}(t_f)$ from where we can integrate backwards to obtain $\mathbf{S}(t)$. 2) The controller thus obtained promises only to put the system finally within an ellipsoid $\bar{\mathbf{x}}^T(t_f)\mathbf{S}(t_f)\bar{\mathbf{x}}(t_f) < \rho(t_f)$ (intuitively this means the cost-to-go is less than some threshold $\rho(t_f)$) for some $\rho(t_f)$. The nominal trajectory $\{\mathbf{x}_0^i(t), \mathbf{u}_0^i(t)\}$ stabilized by a controller c^i terminates at a new nominal trajectory $\{\mathbf{x}_0^{i+1}(t), \mathbf{u}_0^{i+1}(t)\}$ stabilized by a controller c^{i+1} . It is required by condition 2) that c^{i+1} must be able to stabilize any state in the ellipsoid resulted by applying c^i . The same holds for trajectories following $\{\mathbf{x}_0^{i+1}(t), \mathbf{u}_0^{i+1}(t)\}$. As we keep transitioning the system from one trajectory to another, the system eventually must find itself in a stabilizable state or the limit cycle.

Consider a limit cycle that starts at t_0 and ends at t_f . Since the limit cycle ends at its start *i.e.*, $\mathbf{x}_0^l(t_0) = \mathbf{x}_0^l(t_f)$, we also have $\mathbf{S}^l(t_0) = \mathbf{S}^l(t_f)$. One of the methods to find $\mathbf{S}^l(t)$ is to initialize $\mathbf{S}^l(t_f)$ to some arbitrary $\mathbf{S}_0(t_f)$ ² and integrate backwards to get $\mathbf{S}_0(t_0)$. In the next iteration, we re-initialize $\mathbf{S}_1(t_f)$ to $\mathbf{S}_0(t_0)$ and integrate backwards to get $\mathbf{S}_1(t_0)$. This process is repeated till convergence *i.e.*, till $\mathbf{S}_k(t_0) = \mathbf{S}_k(t_f)$ ³ for some k . The value of time varying cost to go matrix $\mathbf{S}^l(t)$ is initialized to this $\mathbf{S}_k(t)$. This is explained in Alg. 1.

For a hybrid system, such as the compass gait, we have continuous dynamics and discrete transitions. Hence we cannot obtain $\mathbf{S}(t)$ from a single Riccati equation since a solution to a first order quadratic differential equation must be smoothly differentiable. Therefore we have different Riccati equations for different modes the system is operating in. And we have a discrete 'jump' event in the Riccati equation where we jump from one equation to another. This is called the jump Riccati equation [14]. The collision dynamics of the system is a function $\mathbf{CD}^{pq}(\mathbf{x})$ that maps states in mode p just before the collision event (guards that cause discrete transitions) to states in mode q just after the collision. That is, $\mathbf{x}(t^+) = \mathbf{CD}^{pq}(\mathbf{x}(t^-))$, where t^- and t^+

²In $\mathbf{S}_k(t_f)$, k is the number of iterations. We can initialize it to penalize the state dimensions we care about. A good choice would be \mathbf{Q} from TVLQR cost function

³In practice we would want them to be close under some tolerance.

are the instances just before and after the collision event. We linearize the collision dynamics of the system to get $\mathbf{A}_{cd} = \frac{\partial \mathbf{CD}^{pq}}{\partial \mathbf{x}}$. The cost to go matrix during the jump event is given by [14],

$$\mathbf{S}(t^-) = \mathbf{A}_{cd}^T \mathbf{S}(t^+) \mathbf{A}_{cd} \quad (4)$$

B. Funnel around a trajectory

Consider a system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ with a closed loop limit cycle $\mathbf{x}_0^l(t), \mathbf{u}_0^l(t)$ whose region of attraction is given by $\bar{\mathbf{x}}^T(t)\mathbf{S}^l(t)\bar{\mathbf{x}}(t) < \rho^l(t)$, where $\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_0^l(t)$. $\mathbf{S}^l(t)$ can be determined empirically as described in subsection IV-A. Let ζ be a trajectory that takes the system from an arbitrary start state to the state $\mathbf{x}_0^l(t_e)$ of the limit cycle. The closed loop limit cycle requires ζ to end at a state \mathbf{x}_e such that $(\mathbf{x}_e - \mathbf{x}_0^l(t_e))^T \mathbf{S}^l(t_e)(\mathbf{x}_e - \mathbf{x}_0^l(t_e)) < \rho^l(t_e)$. We can view this as the allowed uncertainty at the tail end of the trajectory ζ . Hence for a time varying system, instead of defining a discrete region of attraction at every time step, we define funnel⁴ as the region around the trajectory where any point is guaranteed to be led by the closed loop system to the region of allowed uncertainty at the end of the trajectory. By stabilizing the system around a open loop trajectory $\mathbf{x}_0(t), \mathbf{u}_0(t)$, the TVLQR control design would give us the time varying controller $\mathbf{u}^*(t) = \mathbf{u}_0(t) - \mathbf{K}(t)(\mathbf{x}(t) - \mathbf{x}_0(t))$ and also the cost-to-go function $J^*(\bar{\mathbf{x}}, t) = \bar{\mathbf{x}}^T(t)\mathbf{S}(t)\bar{\mathbf{x}}(t)$. This cost-to-go function is a candidate Lyapunov function for our system locally.

Mathematically, we can define the funnel as the time varying region \mathcal{B} , where

$$\mathcal{B}(t) = \{\mathbf{x} | F(\mathbf{x}, t) \in \beta^l\} \quad (5)$$

where $F(\mathbf{x}, t)$ is the function that forward simulates the closed loop trajectory from t to t_f and β^l is the region of attraction around the goal trajectory.

For any such trajectory ζ , we use the cost-to-go, which is time varying here, as the Lyapunov candidate and find the largest $\rho(t)$ in the interval $[t_0, t_f]$ using SoS programming and binary search for $\rho(t)$ as in [11], which gives us the region

$$\mathcal{B}(\rho(\cdot), t) = \{\mathbf{x} | 0 \leq V(\mathbf{x}, t) \leq \rho(t)\} \quad (6)$$

where V , the value function of the closed loop system, is nothing but J^* , the optimal cost-to-go from Eq. 1.

This must satisfy Eq. 5. Similarly for the goal region, the region of attraction is

$$\mathcal{B}^l(\rho^l(\cdot), t) = \{\mathbf{x} | 0 \leq V(\mathbf{x}, t) \leq \rho^l(t)\} \quad (7)$$

where $\rho^l(t)$ represents a constraint on the final value, $\rho(t_f)$ such that $\rho(t_f) \leq \rho^l(t)$. For a time varying system, it is not reasonable to talk about asymptotic stability as this can only be defined for the system as time goes to infinity. However, we can still say that the cost to go function is going downhill and the system is converging to the trajectory for the duration

⁴We use the term 'funnel' for a trajectory which is analogous to basin of attraction for a stabilizable state.

of the trajectory. The bounded final value condition can be verified by proving that $\mathcal{B}(\rho(\cdot), t)$ is closed over $t \in [t_0, t_f]$. The set is closed if $\forall t \in [t_0, t_f]$ we have,

$$V(\mathbf{x}, t) \geq 0, \quad \forall \mathbf{x} \in \mathcal{B}(\rho(\cdot), t) \quad (8)$$

$$\dot{V}(\mathbf{x}, t) \leq \dot{\rho}(t), \quad \forall \mathbf{x} \in \mathcal{B}^\#(\rho(\cdot), t) \quad (9)$$

where $\mathcal{B}^\#$ is the boundary of the funnel \mathcal{B} ,

$$\mathcal{B}^\#(\rho(\cdot), t) = \{\mathbf{x} | V(\mathbf{x}, t) = \rho(t)\} \quad (10)$$

The first condition (Eq. 8) is satisfied by the LQR derivation which makes sure that $\mathbf{S}(t)$ is positive definite. The time derivative of the Lyapunov function is given by,

$$J^*(\bar{\mathbf{x}}, t) = 2\bar{\mathbf{x}}^T \mathbf{S}(t) f(\mathbf{x}_0(t) + \bar{\mathbf{x}}, \mathbf{u}_0(t) - \mathbf{K}(t)\bar{\mathbf{x}}) + \bar{\mathbf{x}}^T \dot{\mathbf{S}}(t)\bar{\mathbf{x}} \quad (11)$$

Tedrake verifies the second condition (Eq. 9) by formulating a series of sums-of-squares feasibility programs just as in original LQR Tree algorithm [11].

Building a funnel around the required goal trajectory gives the system a little breathing area. It is impossible for a system to track the trajectory obtained from direct collocation, as it will be comprised of cubic splines. It is however reasonable to ask for the dynamics of the system to evolve in such a manner that it lies within the volume defined by the LQR funnels.

Algorithm 1 CostToGoLimitCycle($\mathbf{x}_0^l, \mathbf{u}_0^l, \mathbf{Q}, \mathbf{R}$) **Sec. IV.A**

```

1:  $\mathbf{Q}_f = \mathbf{Q}$ 
2: converged  $\leftarrow$  false
3: while not converged do
4:    $[c, \mathbf{S}] \leftarrow \text{tvLQR}^*(\mathbf{x}_0^l, \mathbf{u}_0^l, \mathbf{Q}, \mathbf{R}, \mathbf{Q}_f)$  {Eqn.2}
5:    $\mathbf{Q}_{fv} \leftarrow \mathbf{Q}_f$ 
6:    $\mathbf{S}(t_f^+) \leftarrow \mathbf{S}(0)$ 
7:    $\mathbf{x}_f \leftarrow \mathbf{x}_0^l(t_f)$ 
8:    $\mathbf{A}_{cd} \leftarrow \frac{\partial \mathbf{CD}^{pq}}{\partial \mathbf{x}} \Big|_{\mathbf{x}_f}$ 
9:    $\mathbf{S}(t_f^-) \leftarrow \mathbf{A}_{cd}^T \mathbf{S}(t_f^+) \mathbf{A}_{cd}$  {Eqn. 4}
10:   $\mathbf{Q}_f \leftarrow \mathbf{S}(t_f^-)$ 
11:  if  $\|\mathbf{Q}_f - \mathbf{Q}_{fv}\|_F < \text{threshold}$  then
12:    converged  $\leftarrow$  true
13: return  $[c, \mathbf{S}]$ 

```

C. Estimation of the nominal limit cycle

The hybrid system under our consideration has a stabilizable limit cycle behavior executing the trajectory $\{\mathbf{x}_0^l(t), \mathbf{u}_0^l(t)\}$. That is, for some bounded perturbation $< \rho^l(t)$, the system stays inside the bounded region around the limit cycle trajectory. Dynamics of a hybrid system can be factored into modes and the discrete transition is only due to resetting the state governed by the dynamics of current mode to an initial state governed by dynamics of the next mode. That is, a

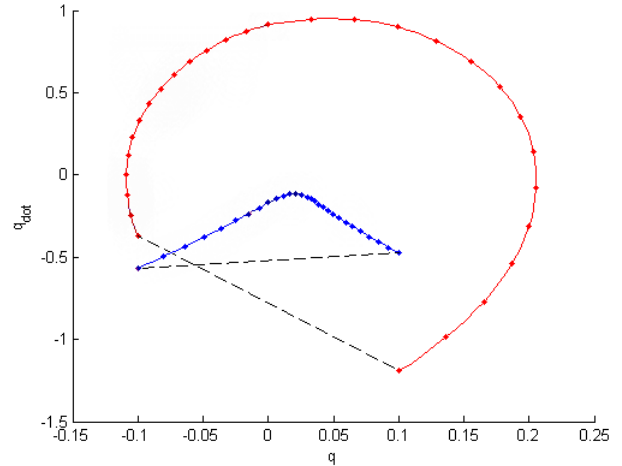


Fig. 1: The limit cycle, $x_0^l(t)$, is estimated using direct collocation for hybrid systems. The dotted points show the knot points used by direct collocation between which it fits cubic splines. The red line represents the swing leg and the blue line the stance leg [20].

guard event only resets the system in the current mode (in dynamics sense) with another configuration.

The time instances just before and after this transition or collision event are t^- and t^+ and the states are $\mathbf{x}(t^-)$ and $\mathbf{x}(t^+)$ respectively. Let the collision dynamics function \mathbf{CD}^{pq} which takes you from mode p to q be such that $\mathbf{x}(t^+) = \mathbf{CD}^{pq}(\mathbf{x}(t^-))$ (Eqn. 17 & 18). To estimate the periodic limit cycle trajectory of the hybrid system we perform point to point trajectory optimization with k knot points, $\mathbf{x}_1, \dots, \mathbf{x}_k$, with the constraint that $\mathbf{x}_1 = \mathbf{CD}^{pq}(\mathbf{x}_k)$ which gives a set of continuous trajectories.

A hybrid system might have more than one stable limit cycle. This is attributed to the range of different possible inputs – all of which achieve limit cycle stability. We choose the limit cycle that results in a local optimum given the cost function $\int_0^\infty \mathbf{u}^T(t) \mathbf{R} \mathbf{u}(t) dt$. Although we can design an LQR-tree algorithm which finds a policy from any initial state to the region of attraction of this open loop limit cycle, it is important to design a TVLQR controller which stabilizes the system around this limit cycle itself to increase the region of attraction of the open loop system. This region of attraction is defined by $\bar{\mathbf{x}}^T(t) \mathbf{S}^l(t) \bar{\mathbf{x}}(t) \leq \rho^l(t)$ as discussed in section IV-A and IV-B

D. Direct collocation for hybrid systems

We use direct collocation to give locally optimal trajectories from a given pose to the goal position. This is required as we need a trajectory that connects a newly sampled state to the nearest funnel in the TVLQR cost function sense. For the hybrid trajectory optimization, we add the mode transition constraint in the optimization problem. The transition can be easily described using the collision event which occurs in the hybrid dynamical system. We search for this collision event between knot points so that the system dynamics at

Algorithm 2 LQRTreesAroundLimitCycle(**Q,R**)

```
1:  $[\mathbf{x}_0^l(t), \mathbf{u}_0^l(t)] \leftarrow \text{PeriodicDirCollocation}^*(\ )$ 
2:  $[c, \mathbf{S}] \leftarrow \text{CostToGoLimitCycle}(\mathbf{x}_0^l, \mathbf{u}_0^l)$ 
3:  $[V, \rho] \leftarrow \text{FTV}^*(\mathbf{x}_0^l, \mathbf{u}_0^l, \mathbf{S}, c)$ 
4:  $T.\text{init}(\{\mathbf{x}_0^l, \mathbf{u}_0^l, V, \rho, \text{NULL}\})$ 
5:  $\text{converged} \leftarrow \text{false}$ 
6:  $\text{iter} \leftarrow 0$ 
7: while not converged do
8:    $\mathbf{x}_s \leftarrow \text{Uniform Random Sample from State Space}$ 
9:    $\mathbf{x}_{\text{near}} \leftarrow \text{Nearest Neighbor with CostToGo metric}$ 
10:   $\text{parent} \leftarrow \text{Pointer to node containing } \mathbf{x}_{\text{near}}$ 
11:  if  $\mathbf{x}_s$  in  $\text{RegionOfAttraction}^*(\mathbf{x}_{\text{near}})$  then
12:     $\text{iter} \leftarrow \text{iter} + 1$ 
13:    continue
14:   $[\mathbf{x}_0, \mathbf{u}_0, \text{Success}] \leftarrow \text{DirCollocation}^*(\mathbf{x}_{\text{near}}, \mathbf{x}_s)$ 
15:  if Success then
16:     $\text{iter} \leftarrow 0$ 
17:     $V_{\text{near}} \leftarrow \text{region of attraction of } \text{parent}$ 
18:     $[c, \mathbf{S}] \leftarrow \text{tvLQR}^*(\mathbf{x}_0, \mathbf{u}_0, \mathbf{Q}, \mathbf{R}, V_{\text{near}})$  Sec. IV.A
19:     $[V, \rho] \leftarrow \text{FTV}^*(\mathbf{x}_0, \mathbf{u}_0, \mathbf{S}, c, V_{\text{near}})$  Sec. IV.B
20:     $T.\text{add}(\{\mathbf{x}_0, \mathbf{u}_0, V, \rho, n\})$ 
21:  else
22:     $\text{iter} \leftarrow \text{iter} + 1$ 
23:  if  $\text{iter} \geq \text{MAXITER}$  then
24:     $\text{converged} \leftarrow \text{true}$ 
```

this event is accounted for. Once this transition is added into the dynamic model, a normal use of the collocation gives us the required nominal trajectory, $\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$. Another problem to be addressed here is the mode sequence that is to be given to the optimizer. We assume that we know the mode sequence of our hybrid system and input it with an initial guess for the time when the transitions occur. However, for systems with several modes, the number of possible mode sequences become enormously high. For such systems, we can let the solver figure out the order of contacts using the implicit trajectory optimization method proposed by Posa *et al.* [19].

We cannot directly use the trajectory generated by the optimizer (line 14 in Alg. 2) method as a sample policy since the probability of a given initial state being one of the states in this nominal trajectory, $\mathbf{x}_0(t)$ and $\mathbf{u}_0(t)$, is zero (a trajectory in state space has zero volume). The TVLQR controller however stabilizes the system around this trajectory and finds the funnel of attraction which has a non-zero volume in state space (and hence has a non-zero probability of occurrence). For a hybrid trajectory, we have piecewise continuous sub-trajectories punctuated by discrete transitions. We apply

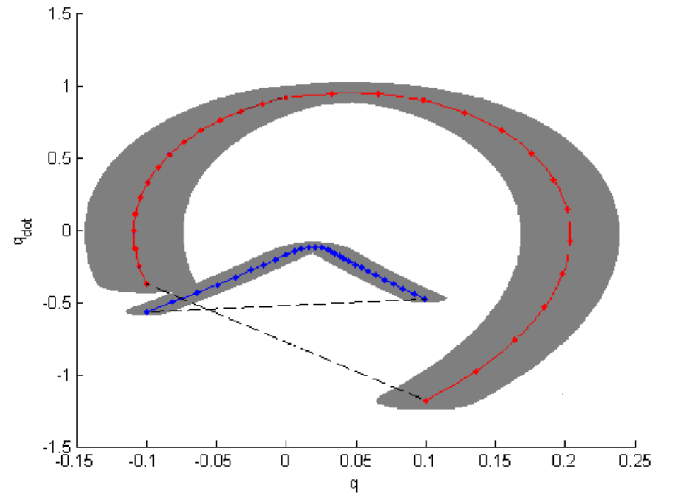


Fig. 2: The limit cycle of compass gait. The red and the blue curves are the nominal trajectories of the swing and the stance leg respectively. The grey region shows the verified funnel the nominal limit cycle trajectory trajectory $\mathbf{x}_0^l(t)$ under the action of TVLQR controller.

TVLQR controller to stabilize the system around every piece of continuous trajectories and find the funnel around each trajectory separately. The jump Riccati equation finds a cost to go $S(t^-)$ such that every state inside $\mathbf{x}^T(t^-)\mathbf{S}(t^-)\mathbf{x}(t^-) < \rho(t^-)$ lands in $\mathbf{x}^T(t^+)\mathbf{S}(t^+)\mathbf{x}(t^+) < \rho(t^+)$. This ensures that the system is stable throughout the hybrid trajectory.

E. Growing the LQR Trees

The LQR tree is initialized with the goal funnel of the nominal limit cycle $\mathbf{x}_0^l(t)$ and $\mathbf{u}_0^l(t)$. Next, a state \mathbf{x}_s is randomly sampled uniformly from the state space. If the newly sampled point falls inside the funnel then it is discarded (Alg.2 line 11). Otherwise, we find the nearest point in the existing tree by using the LQR cost-to-go distance metric[11]. This in a way means that the controller requires minimum effort to bring the sampled point to this nearest point. Once the closest node in the tree is identified we perform trajectory optimization like direct collocation connecting the sampled point and the nearest point (Alg.2 line 14). Then we design the TVLQR controller for the trajectory. Following this, the funnel around this trajectory under the stabilization of TVLQR is verified using section IV-B. We add the newly verified funnel to the existing tree. Our algorithm terminates when a predetermined number of consecutive sample points returns failure or fall in existing funnels.

One can find the TVLQR controller for the given trajectory using Eq. 3. We feed the piecewise continuous trajectory of the hybrid trajectory and design separate TVLQR controllers for each mode. Then we verify every controller separately and determine the Lyapunov function $V(t)$ and the size of the funnel ρ . Hence the region of attraction is given by $V(t) \leq \rho(t)$ with which we check if a sampled point falls inside a funnel or not.

V. EXPERIMENTAL SETUP

The dynamical system in which we investigate our algorithm is a minimalistic version of a compass gait. The compass gait is modeled as a two link (L_1 and L_2) manipulator with masses m concentrated at their center Fig.3.

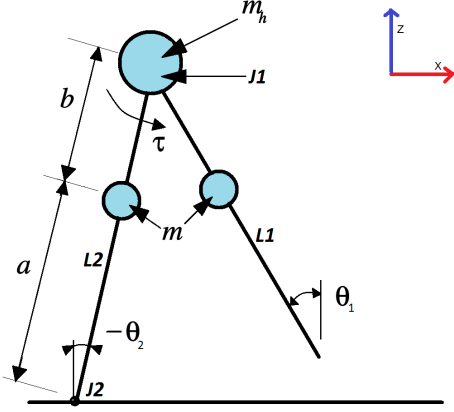


Fig. 3: The compass gait. Link L_2 is attached to the ground and L_1 is attached to L_2 by pin joints J_2 and J_1 respectively. The model has a control input u at joint J_1 that exerts torque between the two legs and a mass m_h is attached at this joint.

We define the state of the system *i.e.*, θ_1 for swing (L_1) and θ_2 for stance (L_2) and their derivatives with respect to the world frame attached to the ground as shown in Fig.3. The state of the system is given by, $\mathbf{x} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]^T$. The dynamics governing the compass gait model can be described as $\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{G}(\mathbf{x}) = \mathbf{B}\mathbf{u}$, where the matrices $\mathbf{M}(\mathbf{x})$, $\mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})$ and $\mathbf{G}(\mathbf{x})$ contain the inertial, Coriolis and the gravity terms as given in [13].

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{G}(\mathbf{x}) = \mathbf{B}\mathbf{u} \quad (12)$$

where,

$$\mathbf{M} = \begin{bmatrix} mb^2 & -mlb \cos(\theta_2 - \theta_1) \\ -mlb \cos(\theta_2 - \theta_1) & (m_h + m)l^2 + ma^2 \end{bmatrix} \quad (13)$$

$$\mathbf{C} = \begin{bmatrix} 0 & mlb \sin(\theta_2 - \theta_1) \dot{\theta}_2 \\ mlb \sin(\theta_2 - \theta_1) \dot{\theta}_1 & 0 \end{bmatrix} \quad (14)$$

$$\mathbf{G} = \begin{bmatrix} mgb \sin(\theta_1) \\ -(m_h l + ma + ml)g \sin(\theta_2) \end{bmatrix} \quad (15)$$

where $l = a + b$. The compass gait is placed on a flat terrain, *i.e.*, the acceleration due to gravity g is in the $-\hat{z}$ direction. During the walking cycle, when the swing leg collides with the ground we generally assume conservation of angular momentum⁵ about the hip joint and toe of the swing leg. The pre-impact and post-impact state parameters can be linearly

⁵We do not consider the change in the mechanical energy of the system during this conservation. In [3], they prove that ΔE is always negative.

related using the collision dynamics as shown below [13],

$$\begin{bmatrix} \dot{\theta}_1^+ \\ \dot{\theta}_2^+ \end{bmatrix} = (\mathbf{T}^+)^{-1} \mathbf{T}^- \begin{bmatrix} \dot{\theta}_1^- \\ \dot{\theta}_2^- \end{bmatrix} \quad (16)$$

$$\mathbf{T}^+ = \begin{bmatrix} mb(b-l \cos \gamma) & ml(l-b \cos \gamma) + ma^2 + m_h l^2 \\ mb^2 & -mbl \cos \gamma \end{bmatrix} \quad (17)$$

$$\mathbf{T}^- = \begin{bmatrix} -mab & -mab + (m_h l^2 + 2mla) \cos(\gamma) \\ 0 & -mab \end{bmatrix} \quad (18)$$

where $\dot{\theta}_1^-, \dot{\theta}_2^-$ and $\dot{\theta}_1^+, \dot{\theta}_2^+$ are the joint angular velocities just before and after the collision, $\gamma = \theta_1 - \theta_2$, \mathbf{T}^- and \mathbf{T}^+ are the transition matrices that contain the coefficients of conservation of angular momentum. The size of \mathbf{T}^- and \mathbf{T}^+ matrices are 2×2 because angular momentum is conserved about two points and for two angular velocities. During collision event, the mapping between joint angles before and after collision is obtained by merely interchanging them *i.e.*, $\theta_1^+ = \theta_2^-$ and $\theta_2^+ = \theta_1^-$. The complete collision dynamics $\mathbf{CD}(\mathbf{x})$ is found using this and velocity mappings in Eq. 16.

VI. EXPERIMENTS AND RESULTS

A. Compass Gait Model

The compass gait model whose dynamics are described above is simulated with the following parameters. The mass at the hip $m_h = 10\text{kg}$, link masses $m = 5\text{kg}$, link lengths $a = b = 0.5\text{m}$ and $g = 9.8\text{m/s}^2$. Here $\mathbf{x} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$ and $\mathbf{u} = [\tau]$ at the hip. The joints are assumed to be frictionless and the collisions are inelastic.

B. Direct Collocation and LQR Trees

The direct collocation procedure is a nonlinear trajectory optimization problem which requires an initial guess for $\mathbf{x}(t)$ and $\mathbf{u}(t)$. The initial guess for $\mathbf{x}(t)$ is a straight line connecting the random sample and the nearest point in the existing tree based on the TVLQR cost function. The initial guess for input trajectory $\mathbf{u}(t)$ is a random value. The mode transition constraint is given by the collision dynamics (Eq. 16) which occurs in the compass gait model when the distance between the swing leg and the ground is zero. The direct collocation function has very sparse gradients and the constraints depend upon the values at knot points or adjacent knot points. Solvers such as SNOPT [8] can very efficiently solve such nonlinear programs with sparse gradients. Every iteration of direct collocation is terminated and a new state is sampled if there is no result after 40 seconds or if the solver fails.

The parameters of the LQR tree algorithm are $\mathbf{Q} = \text{diag}([10, 10, 1, 1])$, $\mathbf{R} = [15]$. We terminate after 500 consecutive samples (MAXITER om Alg. 2) fall in the existing tree or fail to find a trajectory to the tree. The LQR tree is considered to have reasonably covered the entire region of stabilizable state space for compass gait if 500 consecutive samples either fall inside the funnels or outside and the direct collocation fails.

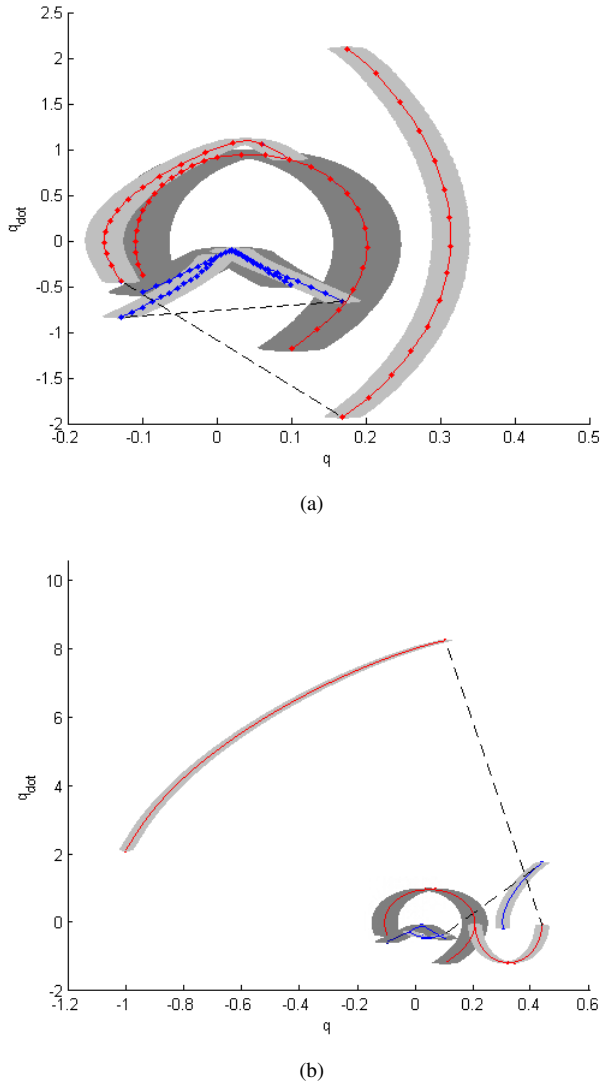


Fig. 4: Phase plane plot showing the recovery of the compass gait with (a) velocity perturbation only (b) both velocity and position perturbation. The curves in red and blue show the nominal trajectory of the swing and the stance legs respectively. The gray region shows the verified funnel around each trajectory under the action of TVLQR controller.

System	Pendulum	Cartpole	Compass gait
No. of Samples	146 6068	- 5821	252 -
No. of nodes	146 477	- 881	192 -
Time taken for planning	0h 2m 0.5h	- 2h	6h 21m -

TABLE I: Results for pendulum and cart pole systems obtained from [11] and [16]. The results of verification by SoS optimization and simulation are shown in blue and red fonts respectively. **Note:** A “-” denotes unavailability of data from [11] or [16].

The goal limit cycle trajectory $\mathbf{x}_0^l(t)$ and $\mathbf{u}_0^l(t)$ are obtained directly from the `PeriodicDirCollocation()` method. Also, a TVLQR controller given a nominal trajectory can be found using `tvqlqr()`. The `sampledFiniteTimeVerification()` (aka FTV) finds the ratio of value of Lyapunov function $V(\mathbf{x})$ to the size of the funnel $\rho(t)$. So to check if a point \mathbf{x} is in a given funnel, we just need to check $V(\mathbf{x})/\rho(t) \leq 1$.

All the function implementations denoted by the `typewriter` typeface and asterisk (*) superscripts in the Alg. 1 & 2 are available in DRAKE [20].

C. Experiments with Initial Perturbation and Noise

Once the state space is filled with LQR trees the compass gait has a recovery policy to stabilize itself back to the limit cycle’s region of attraction from any arbitrary and stabilizable initial condition. It is to be noted that not all states of the compass gait are stabilizable no matter what input we apply. For example, a compass gait lying down cannot pump energy to get itself up in any way to get back up on two legs. We tested our algorithm by perturbing both the position and velocity states of the system. The figure 4(a) shows the phase portrait of the case in which there was an initial velocity perturbation and figure 4(b) shows the phase portrait of the case in which there was both velocity and position perturbation. Both the above cases are simulated by considering a white Gaussian noise in the forward simulation with 0.05 as standard deviation.

D. Inferences

TABLE I provides a comparison of the two methods to generate LQR trees and the dynamical systems they were applied to. It took around 381 minutes to completely cover the state space⁶ with funnels for the compass gait system being evaluated. The results reflect our intuition in that the simulation methods take longer to build the LQR trees. For the compass gait, 23% of the time was used up for direct collocation and 58% for funnel verification. Trajectory verification takes a big chunk of the total time because the trajectories being verified are hybrid trajectories. There are discrete jumps in the trajectories as the mode of the system changes with time. The start of the next mode trajectory is obtained using the jump Riccati equation.

VII. FUTURE WORK

One of the interesting future works would be to extend the algorithm for the case of a high DoF compass gait (Compass Gait with knees[9] – 3 DoFs). We have a more efficient walking limit cycle with knees than keeping the knees straight. To find a policy for kneed gait with LQR trees of kneeless gait, a set of approximations could be made to map a given state in kneed gait to a state in simple gait and follow its policy given by the LQR-tree. Also the number of possible mode sequences becomes intractable for high DoF systems. For such systems (kneed compass gait) mode

⁶It was evaluated on a Intel Core i7-6700K 4 GHz Quad-Core Processor with 32 GB RAM

sequence specification could be circumvented by using the algorithm laid by [19].

When a perturbation causes the kneed compass gait to go off the region of attraction of its limit cycle we can approximate the instantaneous configuration by 1) making the links move to a configuration with equal angles at the knees and by locking the knee angles 2) stretching the knee to obtain an equivalent low DoF configuration. In both cases, we go to the described configurations using direct collocation with the constraint that the knee angles are equal (or zero for straight knees). Once we stabilize the system with the knees straight or locked, the system will exhibit a stable kneeless limit cycle. We just need one trajectory from one of the points of this limit cycle to a point in the more efficient kneed limit cycle. Moreover, we can use these trajectories obtained from such mappings as seed trajectories to build trees in the state space of kneed compass gait.

VIII. CONCLUSIONS

We presented a method to use LQR trees to control hybrid systems like compass gait. We also devised a method (Algorithm.1) to find the region of attraction of TVLQR controller around a dynamic limit cycle which is important to find the funnel around every other trajectory of LQR Tree. This method effectively covers the stabilizable regions in a bounded state space with a series of feedback stabilized sample trajectories that lead to the goal trajectory. The algorithm was tested for a compass gait modeled as a two-link manipulator. We also discuss how an LQR Tree built for a lower DoF robot may be reused on a higher DoF systems. While we probabilistically cover stabilizable regions of state space to provide a policy beyond a controller's region of attraction, it takes more than six hours to do this for one goal trajectory. For a new goal trajectory, one can reuse the tree if a trajectory can be found between any two points – one in the old and one in the new goal trajectory. Otherwise, it could potentially take another six hours of planning.

ACKNOWLEDGMENT

We specially thank Prof. Russ Tedrake for his quick responses to our queries regarding DRAKE simulator [20]. We also thank Prof. Dmitry Berenson for his motivation behind exploring this area of research.

REFERENCES

- [1] Von Stryk, Dipl Math Oskar. "Numerical solution of optimal control problems by direct collocation." *Optimal Control*. Birkhuser Basel, 1993. 129-143.
- [2] Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation* 12.4 (1996): 566-580.
- [3] Goswami, Ambarish, Benoit Thuilot, and Bernard Espiau. "Compass-like biped robot part I: Stability and bifurcation of passive gaits." (1996).
- [4] Lavalle, Steven M. "Rapidly-Exploring Random Trees: A New Tool for Path Planning." (1998).
- [5] Branicky, Michael S., and Michael M. Curtiss. "Nonlinear and hybrid control via RRTs." *Proc. Intl. Symp. on Mathematical Theory of Networks and Systems*. Vol. 750. 2002.
- [6] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, SOS-TOOLS: Sum of squares optimization toolbox for MATLAB, 2004.
- [7] Vukobratovi, Miomir, and Branislav Borovac. "Zero-moment point—thirty five years of its life." *International Journal of Humanoid Robotics* 1.01 (2004): 157-173.
- [8] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99131, 2005.
- [9] Chen, Vanessa F. Hsu. *Passive dynamic walking with knees: A point foot model*. Diss. Massachusetts Institute of Technology, 2007.
- [10] Byl, Katie, and Russ Tedrake. "Approximate optimal control of the compass gait on rough terrain." *Robotics and Automation*, 2008. ICRA 2008. IEEE International Conference on. IEEE, 208.
- [11] Tedrake, Russ. "LQR-Trees: Feedback motion planning on sparse randomized trees," in *Proc. of Robotics: Science and Systems*, Seattle, WA, June 2009.
- [12] Shkolnik, Alexander, Matthew Walter, and Russ Tedrake. "Reachability-guided sampling for planning under differential constraints." *Robotics and Automation*, 2009. ICRA'09. IEEE International Conference on. IEEE, 2009.
- [13] Tedrake, Russ. "Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832." Working draft edition (2009).
- [14] Manchester, Ian R., et al. "Regions of attraction for hybrid limit cycles of walking robots." *arXiv preprint arXiv:1010.2247* (2010).
- [15] Tobenkin, Mark M., Ian R. Manchester, and Russ Tedrake. "Invariant funnels around trajectories using sum-of-squares programming." *arXiv preprint arXiv:1010.3013* (2010).
- [16] Reist, Philipp, and Russ Tedrake. "Simulation-based LQR-trees with input and state constraints." *Robotics and Automation (ICRA)*, 2010 IEEE International Conference on. IEEE, 2010.
- [17] Tedrake, Russ, et al. "LQR-trees: Feedback motion planning via sums-of-squares verification." *The International Journal of Robotics Research (IJRR)*, 2010.
- [18] Dai, Hongkai, and Russ Tedrake. "Optimizing robust limit cycles for legged locomotion on unknown terrain." *Decision and Control (CDC)*, 2012 IEEE 51st Annual Conference on. IEEE, 2012.
- [19] Posa, Michael, and Russ Tedrake. "Direct trajectory optimization of rigid body dynamical systems through contact." *Algorithmic foundations of robotics X*. Springer Berlin Heidelberg, 2013. 527-542.
- [20] Russ Tedrake. *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*. <http://drake.mit.edu>, 2014
- [21] Bharatheesha, Mukunda, et al. "Distance metric approximation for state-space rrts using supervised learning." *Intelligent Robots and Systems (IROS 2014)*, 2014 IEEE/RSJ International Conference on. IEEE, 2014.