Algebraic Verification for Parameterized Motion Planning Libraries

Anirudha Majumdar, Mark Tobenkin, and Russ Tedrake

Abstract— **Recent progress in algorithms for estimating regions of attraction and invariant sets of nonlinear systems has led to the application of these techniques to motion planning in complex environments. In most instances, the verification occurs offline as the algorithms are still too computationally demanding for realtime implementation; as a result any online planner is restricted to applying the finite set of motion plans that were verified offline. In this paper we attempt to present a partial remedy by algebraically verifying families of parameterized feedback controllers. We provide a specific example using LQR controllers parameterized by their goal or nominal motion. We formulate this verification using robust region of attraction techniques in sums-of-squares optimization, and show that perturbations of a Lyapunov or Riccati equation can be used to provide algebraically parameterized Lyapunov candidates. The resulting verified "funnels" then provide a parameterized motion library that can be used efficiently in online planning. We present a number of numerical examples to demonstrate the effectiveness of our approach.**

I. Introduction

Imagine an unmanned aerial vehicle (UAV) flying at high speeds through an uncharted forest, or a bipedal robot walking on rough terrain with limited information about the terrain ahead, or a manipulator arm trying to grasp an object without exact prior knowledge of where the object has been placed. Each of these scenarios requires motion planning in complex environments with dynamic goal specifications. This area of research has been the subject of significant focus, and has enjoyed much success over the last few decades. Traditional approaches to the problem considered the planning task purely in the kinematic domain, ignoring constraints and complexities arising from the potentially nonlinear dynamics of the system ([5], [7]). However, recognition of the challenges that arise from underactuation and input saturations have spurred research in *feedback motion planning* algorithms, which explicitly take into account the dynamics of the system during the offline planning stage ([6], [14], [3]).

These advances have in part been made possible by recent progress in the direct algebraic verification of regions of attraction and invariant sets ("funnels") for nonlinear systems [10]. For example, the LQR-Trees algorithm [14] constructs a family of linear controllers around locally optimal trajectories of the system with corresponding "funnels", composed together in a tree structure. The "funnels" are computed using sums-of-squares programming that check the conditions for local Lyapunov stability by verifying positive definiteness of polynomials by expressing them as "sums of squares".

Since the sums-of-squares verification procedure is still too expensive for online implementation, in order to adapt to dynamic task specifications one is restricted to a set of motion plans that are verified offline. In this paper, we address this problem by computing regions of attraction that are parametrized by the goal state. Thus, in the offline precomputation stage, we compute an entire family of "funnels" that an online planner can use in order to adapt to changing goal states.

An important aspect of our approach is that we avoid discretizing the goal set in any way, and instead compute regions of attraction for the entire *continuum* of goal states via a single sums-of-squares program. There are multiple advantages to this approach. First, solving separate sums-of-squares programs for a discrete set of goal states is prohibitively expensive, and scales exponentially with the dimension of the goal set. Second, we avoid issues that arise when choosing how finely to discretize the set. Finally, an online planner that uses our parameterized regions of attraction has the ability to drive the system to *any* goal state in the continuum of goals.

II. Related Work

The use of sums-of-squares (SOS) programming to automate Lyapunov analysis of polynomial vector fields has seen many recent successes (see [10] and [4]). These techniques provide sufficient conditions to verify Lyapunov inequalities using semidefinite programming which can be addressed via modern interior point optimization techniques [9]. Our work analyzes vector fields depending on an uncertain parameter using parameter dependent Lyapunov functions similar to [1]. We also provide extensions of finite-time invariance conditions explored in [14] and [15] to the parameter dependent case. An alternative branch-and-bound based method for addressing parameter dependent vector fields using SOS programming has been suggested in [17].

This paper uses formal power series expansions to approximate the solution of Riccati equations to find parameter dependent stabilizing control laws. Similar perturbation type techniques have been applied in the state-dependent Riccati equation (SDRE) literature (see [2]) to derive approximately optimal nonlinear state feedback control laws for a fixed nonlinear vector field. By contrast the approximations in this work are used to find parameter-dependent linear control laws to stabilize a vector field at a parameter-varying equilibrium.

The authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL) at the Massachusetts Institute of Technology, Cambridge, MA, USA {anirudha,mmt,russt}@csail.mit.edu. This work was supported by ONR MURI grant N00014-09-1-1051. Anirudha Majumdar is supported by the MIT Intelligence Initiative.

We anticipate applications of our work to feedback motion planning algorithms such as the Maneuver Automaton [3] and the LQR-Trees algorithm [14]. In particular, the goal-parameterized control laws we compute can serve as "maneuvers" for the Maneuver Automaton while our parameterized ROAs can be used by LQR-Trees for online planning.

III. Parameterized family of Candidate Lyapunov Functions

A. Parameterized Lyapunov equations

We consider a parametric family of differential equations:

$$\dot{x} = f_\delta(x) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system and $\delta \in P \subset \mathbb{R}^p$ is a parameter which varies the vector field. We assume the dependence of $f_\delta(x)$ on δ and x is polynomial. We further assume we are given a polynomial map $x_g : P \mapsto \mathbb{R}^n$ such that $x_g(\delta)$ is an exponentially stable equilibrium of $f_\delta(x)$. We take $X_g = x_g(P)$, the set of such equilibria. Finally, for simplicity, we assume $0 \in P$.

Our goal is to simultaneously estimate the region of attraction of each $x_g(\delta)$ for $\delta \in P$. To do so, we search for a family of Lyapunov functions parameterized by δ. A locally valid quadratic Lyapunov function of the form:

$$V_\delta(x) = (x - x_g(\delta))^T S_\delta^\star (x - x_g(\delta))$$

is guaranteed to exist. A candidate $S_\delta^\star$ is given by the solution of the Lyapunov equation:

$$0 = Q + A_\delta^T S_\delta^\star + S_\delta^\star A_\delta \tag{2}$$

where:

$$A_\delta = \frac{\partial f_\delta}{\partial x}(x_g(\delta))$$

i.e. the Jacobian of the dynamics at the equilibrium, and Q is an arbitrary positive definite, symmetric matrix in $\mathbb{R}^{n\times n}$.

For our class of $f_\delta(\cdot)$, the dependence of A_δ on δ is polynomial, however the dependence of the solution $S_\delta^\star$ on A_δ can be non-polynomial. To find Lyapunov functions amenable to SOS programming we seek to find a polynomial approximation of $S_\delta^\star$, which we arrive at through a formal power series expansion [1] about A_0.

For $k \in \mathbb{N}^{n^2}$ we use the notation:

$$a^k := a_1^{k_1} \dots a_{n^2}^{k_{n^2}}. \tag{3}$$

For $k, \bar{k} \in \mathbb{N}^{n^2}$ we write $k \leq \bar{k}$ if $k_i \leq \bar{k}_i$ for all $i \in \{1, \dots, n^2\}$. Finally we define e_i to be those vectors with their i-th component being one and the remaining components being zero and let $\mathcal{E} = \{e_i\}_{i=1}^{n^2}$.

We examine variations in A in affine coordinates about A_0:

$$A(a) = A_0 + \sum_{i=1}^{n^2} a_i E_{e_i} = A_0 + \sum_{e\in\mathcal{E}} a^e E_e \tag{4}$$

[1] An alternative is to take the expansion of the (2) equation directly with respect to the parameters δ.

where $a \in \mathbb{R}^{n^2}$ and $\{E_{e_i}\}_{i=1}^{n^2}$ are a set of linearly independent matrices spanning all of $\mathbb{R}^{n\times n}$. We now examine the solutions $S(a)$ defined by the Lyapunov equation:

$$0 = Q + A(a)^T S + S A(a) \tag{5}$$

in the neighborhood of the symmetric positive definite solution S_0 of (5) with $a = 0$.

To compute a local approximation of $S(a)$ about $a = 0$ we write $S(a)$ as a formal power series:

$$S(a) := S_0 + \sum_{k\in\mathbb{N}^{n^2}} a^k S_k \tag{6}$$

We substitute $S(a)$ into (5) and for each $k \in \mathbb{N}^{n^2}$ take the term which multiplies a^k to be zero:

$$0 = Q + S_0 A_0 + A_0^T S_0, \tag{7}$$

$$0 = S_k A_0 + A_0^T S_k + \sum_{\substack{e\in\mathcal{E}\\ e\leq k}} E_e^T S_{k-e} + S_{k-e} E_e. \tag{8}$$

We see that (7) is the Lyapunov equation for $a = 0$. Each equation (8) is a Lyapunov equation in S_k which depends only on $S_{k'}$ where k' has smaller total degree, defined as $\sum_i k_i$. As a result, these equations can be efficiently solved recursively for increasing total degree.

To complete the analysis we compose the function $S(a)$ with the map from δ to A_δ by finding the unique function $a : P \mapsto \mathbb{R}^{n^2}$ such that:

$$A_\delta = A_0 + \sum_{i=1}^{n^2} a_i(\delta) E_{e_i}. \tag{9}$$

Thus, we have achieved our goal of deriving a power series representation of the Lyapunov function candidate as a function of the goal state parametrization δ. For each goal state,

$$V_\delta(x) = (x - x_g(\delta))^T S(a(\delta))(x - x_g(\delta))$$

is a locally valid Lyapunov function candidate. In practice, one must use a trunction of the map $S(a)$ by fixing some maximal degrees $\bar{k} \in \mathbb{N}^{n^2}$ and ignoring higher power terms. We denote by S_δ the map:

$$S_\delta = S_0 + \sum_{k\leq\bar{k}} a(\delta)^k S_k$$

i.e. the truncation of the map $S(\cdot)$ composed with the map $a(\delta)$. The terms in the truncation can be computed offline using equations (7) and (8). This provides us with a compact algebraic approximation of the candidate Lyapunov equation for each goal state parametrized by $\delta \in P$.

B. Parameterized Riccati Equations

A very similar approach can be used to approximate the solution of a family of Riccati equations parameterized by δ. We restrict our attention here to the design of a Linear Quadratic Regulator (LQR) solution for each $\delta \in P$, though similar approaches could be applied to robust state

feedback design [11]. Suppose that we have a nonlinear control system:

$$\dot{x} = f_\delta(x, u)$$

where $u \in \mathbb{R}^m$ is the control input. We assume now the existence of two polynomial maps, $u_g : P \mapsto \mathbb{R}^m$ and $x_g : P \mapsto \mathbb{R}^n$ such that for each $\delta \in P$, $x_g(\delta)$ is an exponentially stabilizable equilibrium of $f_\delta(x, u_g(\delta) + \bar{u})$ with the input $\bar{u} \in \mathbb{R}^m$. For each δ we can calculate a locally stabilizing controller and Lyapunov function for the closed loop system by examining the linearization:

$$A_\delta = \frac{\partial f_\delta}{\partial x}(x_g(\delta), u_g(\delta)), \qquad B_\delta = \frac{\partial f_\delta}{\partial u}(x_g(\delta), u_g(\delta)).$$

For any particular choice of cost matrices $Q = Q' \geq 0$ and $R = R' > 0$ with $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ we can compute an affine feedback law $u(x) = u_g(\delta) - K^\star_\delta(x - x_g(\delta))$ via the positive definite solution of the Riccati equation:

$$0 = Q - S^\star_\delta B_\delta R^{-1} B_\delta^T S^\star_\delta + S^\star_\delta A_\delta + A_\delta^T S^\star_\delta \qquad (10)$$

and the relationship:

$$K^\star_\delta = R^{-1} B_\delta^T S^\star_\delta.$$

Both $S^\star_\delta$ and $K^\star_\delta$ have a potentially complicated, though smooth, dependence on δ. Our objective again is to find a polynomial approximation of $S^\star_\delta$ (which in turn gives us an approximation of $K^\star_\delta$) and we again turn to a formal power series. By assuming $f_\delta(x, u)$ has polynomial dependence on δ, x and u we again are concerned primarily how the solutions of (12) vary as a function of A_δ and B_δ directly. Let $A(a)$ be as in equation (4) and define:

$$G(b) = B_0 R^{-1} B_0^T + \sum_{\bar{e}\in\mathcal{F}} b^{\bar{e}} F_{\bar{e}}, \qquad (11)$$

where $b \in \mathbb{R}^{n(n+1)/2}$, $\mathcal{F}$ is the set of all standard basis vectors for $\mathbb{R}^{(n+1)n/2}$, and $\{F_{\bar{e}}\}_{\bar{e}\in\mathcal{F}}$ are linearly independent matrices spanning the space of symmetric matrices in $\mathbb{R}^{n\times n}$ (c.f. the definition of $A(a)$ in (4)). Then we would like to examine the solutions $S(a, b)$ of the equation:

$$0 = Q - SA(a) + A(a)^T S + SG(b)S, \qquad (12)$$

in the neighborhood of the positive definite symmetric solution S_0 found with $a = 0$ and $b = 0$.

To approximate the map $S(a, b)$ we again write it as a formal power series:

$$S(a, b) \quad = \quad S_0 + \sum_{k\in\mathbb{N}^{n^2}, \ell\in\mathbb{N}^{n(n+1)/2}} a^k b^\ell S_{k,\ell} \qquad (13)$$

where a^k and b^ℓ are defined as in (3).

Substituting this expansion of $S(a, b)$ into the equation (12) and again setting terms multiplied by $(k, \ell) \in \mathbb{N}^{n^2} \times \mathbb{N}^{(n+1)n/2}$ to zero we arrive at the following equations:

$$0 = Q + S_0 A_0 + A_0^T S_0 - S_0 B_0 R^{-1} B_0^T S_0 \qquad (14)$$

$$\begin{aligned} 0 =& S_{k,\ell} A_0 + A_0^T S_{k,\ell} + \sum_{\substack{e\in\mathcal{E}\\ e\leq k}} S_{k-e,\ell} E_e + E_e^T S_{k-e,\ell} \qquad (15)\\ &- \sum_{0\leq\kappa\leq k} \Bigg[\sum_{0\leq\lambda\leq\ell} S_{\kappa,\lambda} B_0 R^{-1} B_0^T S_{k-\kappa,\ell-\lambda} \\ &+ \sum_{\bar{e}\in\mathcal{F}} \sum_{\bar{e}\leq\lambda\leq\ell} S_{\kappa,\lambda-\bar{e}} F_{\bar{e}} S_{k-\kappa,\ell-\lambda} + S_{\kappa,\lambda} F_{\bar{e}} S_{k-\kappa,\ell-\lambda-\bar{e}} \Bigg]. \end{aligned}$$

We see that (14) is the exact Riccati equation for $(a, b) = (0, 0)$. Further each (15) is affine in $S_{k,\ell}$ and the linear term of each such equation is given by:

$$S_{k,\ell}(A_0 - B_0 R^{-1} B_0^T S_0) + (A_0 - B_0 R^{-1} B_0^T S_0)^T S_{k,\ell},$$

and the constant term depends only on $S_{k',\ell'}$ for (k', ℓ') of smaller total degree (here $\sum_i k'_i + \sum_j \ell'_j < \sum_i k_i + \sum_j \ell_j$). Thus these equations can be solved in order of increasing total degree. As $A_0 - B_0 R^{-1} B_0^T S_0$ is Hurwitz, each such equation will have a unique solution.

To complete the mapping from δ to S we find the unique functions $a : P \mapsto \mathbb{R}^{n^2}$ as in (9) and $b : P \mapsto \mathbb{R}^{(n+1)n/2}$ such that:

$$B_\delta R^{-1} B_\delta^T = B_0 R^{-1} B_0^T + \sum_{i=1}^{(n+1)n/2} b_i(\delta) F_{\bar{e}_i},$$

with $\bar{e}_i$ being the vector in $\mathbb{R}^{(n+1)n/2}$ with one in its i-th component and zero otherwise.

The approximate Lyapunov candidate and control law are then given by:

$$K(\delta) = R^{-1} B_\delta^T S(a(\delta), b(\delta))$$

and:

$$V_\delta(x) = (x - x_g(\delta))^T S(a(\delta), b(\delta))(x - x_g(\delta))$$

respectively. Just as in the previous section, we define S_δ in this context to be a fixed order truncation of the map $S(a, b)$ composed with the map $(a(\delta), b(\delta))$. Similarly, we denote K_δ to be the fixed order truncation of $K(\delta)$.

Note that S_δ may not be positive definite for all δ. As mentioned in Section IV, the positive definiteness condition should be checked via a sums-of-squares program for all δ of interest.

IV. Computation of Regions of Attraction

Using the algebraic representation of the parameterized Lyapunov function candidate (derived in Section III), we now provide conservative estimates of the region of attraction associated with each $x_g(\delta)$, using techniques from verification with sums-of-squares optimization [16].

For the remainder of the paper, when examining controlled vector fields we use $f_\delta(x)$ to denote the closed loop dynamics using the state feedback law from Section III-B. We use the

parameterized candidate Lyapunov functions from Section III-B:

$$V_\delta(x) = (x - x_g(\delta))^T S_\delta (x - x_g(\delta))$$

with the derivative along solutions given by:

$$\dot{V}_\delta(x) = 2(x - x_g(\delta))^T S_\delta f_\delta(x).$$

For each $\delta \in P$, we aim to find a maximal $\rho : P \mapsto \mathbb{R}_+$ such that:

$$V_\delta(x) \leq \rho(\delta) \implies \dot{V}_\delta(x) \leq -cV_\delta(x)$$

which verifies exponential stability (here $c > 0$ is a fixed positive constant).

By construction S_δ and K_δ are polynomials in δ. When $f_\delta(x)$ and x_g are polynomials and P is a semialgebraic set defined by polynomial equations and inequalities:

$$P = \{\delta \mid \max_i g_i(\delta) \leq 0, h(\delta) = 0\},$$

with $g : \mathbb{R}^p \mapsto \mathbb{R}^{p_g}$ and $h : \mathbb{R}^p \mapsto \mathbb{R}^{p_h}$, we can directly address this problem using sums-of-squares programming. Section VI-A gives further examples of classes of systems which can be addressed using SOS programming.

We parameterize $\rho(\delta)$ as a fixed degree polynomial in δ. We additionally search over a "multiplier" polynomial $\nu : \mathbb{R}^n \times \mathbb{R}^p \mapsto \mathbb{R}$ of fixed degree. The ideal form of the SOS program is:

$$\begin{aligned} \underset{\rho,\nu}{\text{maximize}} \quad & \int_P \rho(\delta)\, d\delta \\ \text{subject to} \quad & \|x - x_g(\delta)\|^2 (V_\delta(x) - \rho(\delta)) \\ & - \nu(\boldsymbol{x}, \delta) \left(\dot{V}_\delta(x) + cV_\delta(x) \right) \geq 0 \\ & \rho(\delta) \geq 0 \qquad \forall \delta \in P, x \in \mathbb{R}^n \end{aligned} \tag{16}$$

Enforcing these constraints only for $\delta \in P$ is handled by introducing further multipliers [10]. Depending on the complexity of P, the integral may need to be numerically approximated.

One can verify that any feasible solution of the above satisfies that for $\delta \in P$ we have $\rho(\delta) \geq 0$ and $V_\delta(x) \geq \rho(\delta)$ whenever $\dot{V}_\delta(x) + cV_\delta(x) = 0$ and $x \neq x_g(\delta)$. Combined with the knowledge that S_δ is positive definite $\forall \delta \in P$, and the Hessian of $\dot{V}_\delta(x) + cV_\delta(x)$ is negative definite in a neighborhood about $x_g(\delta)$, this certifies the exponential stability of the closed loop system. Similar SOS programs can be used to easily verify these conditions. If either program is infeasible, P must be reduced in size.

Note that although the feedback and Lyapunov candidates are approximations of the local optimal control solutions, the verification procedure here is exact. Thus, issues such as convergence of the Taylor series used to represent the Lyapunov functions do not affect the *validity* of the guarantees obtained from the SOS program. However, it should be noted that the size of the regions of attraction could be adversely affected by the approximate nature of the Lyapunov candidate. This is further explored in Section V.

V. Numerical Results

For the results in the current work, we use the SPOT toolbox [8] to cast SOS programs a semidefinite programs which we solve using SeDuMi [13]. In this section, we present results of the application of the theoretical framework presented above to a simple numerical example: the Van der Pol oscillator. The dynamics of the oscillator are described by the following equation:

$$\ddot{x} = \dot{x}(1 - x^2) + x + u.$$

Using the notation from Sections III and IV, we take $P = [-0.5, 0.5]$, $x_g(\delta) = [\delta, 0]^T$ and $u_g(\delta) = -\delta$. Thus, our goal is to compute LQR controllers and corresponding regions of attraction (ROAs) for the set represented by the black line segment along the x-axis in Figure 1. Figure 1 compares the ROAs obtained using the techniques described in this paper (green ellipses) with ROAs obtained from standard SOS-verification procedures ([16]) (red ellipses). Note that while our verification program provides us with ROA estimates for the entire *continuum* of goal states in X_g by solving a *single* SOS program, standard SOS procedures require that a separate SOS program be solved for each goal state considered in the plot (thus providing only a finite set of ROAs at the sampled points). Hence, for the sake of comparison, we sample a discrete set of points along the goal set and compare the ROAs at these points.

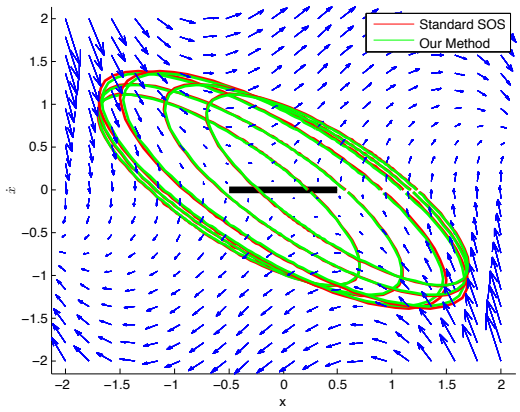

Fig. 1: Comparison of regions of attraction for the Van der Pol oscillator obtained from standard SOS verification and our method.

Figure 2 provides a numerical comparison of the volume of the ROAs obtained using our method versus standard SOS verification. The set of discrete red points represent the volumes of the ROAs using standard SOS verification. The blue curve represents the ROA volumes obtained from a 0^{th} order expansion of the power series (13), i.e. $S_\delta = S_0$. The green curve represents a 1^{st} order expansion. As is evident from the plot, the performance (as measured by the volume of the ROAs) is better for the higher order (1^{st} order) expansion than it is for the 0^{th} order expansion. Note that this may not be true in general since a better approximation

of the LQR controller is not guaranteed to produce larger ROAs.

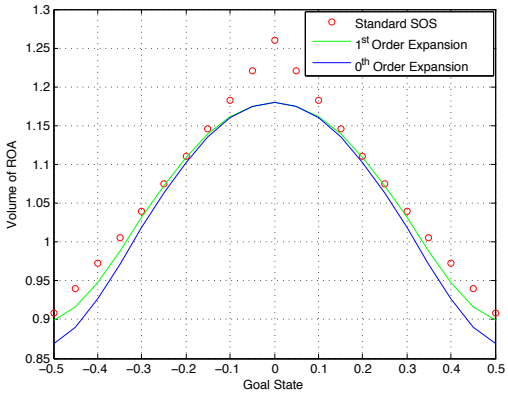

Fig. 2: Comparison of volumes of regions of attraction for the Van der Pol oscillator

Note that it is important to distinguish between two sources of the approximation errors we observe in the size of the ROAs that result from using our method: (**i**) the approximation of the LQR controller via the power series in (13), and (**ii**) the requirement that $\rho(\delta)$ be a polynomial (see Section IV). The relatively poor approximation of the ROA around the middle of the goal set is due to approximation (**ii**) ($\rho(\delta)$ was chosen to be a 4^{th} order polynomial). Using a polynomial of higher degree results in a better approximation, but results in a longer running time for the SOS verification program.

Approximation (**i**) is illustrated in Figure 3. Here, standard SOS verification was used to compute ROAs for a discrete set of goal states in the goal set. The optimal LQR controller (calculated individually for each goal state) was used for the red points, while a 1^{st} order power series approximation for the controller was used for the green points and a 0^{th} order expansion was used for the blue points. As is clear, the controller is optimal for the nominal goal state, the origin, since this is the point about which the power series is expanded. The controller approximation gets poorer as one moves away from this point, resulting in smaller regions of attraction.

VI. Extensions to Basic Framework

A. Goal Set Parametrization

In the derivations provided in Sections III and IV, it was assumed that a goal set parametrization was available. Although this is often the case, a simple parametrization may not always suffice in describing the set of stabilizable goal states that one wants to compute regions of attraction for. In such cases, additional constraints can often be introduced into the SOS verification program in order to ensure that all the points in the goal set are in fact stabilizable.

For example, consider the following non-linear control

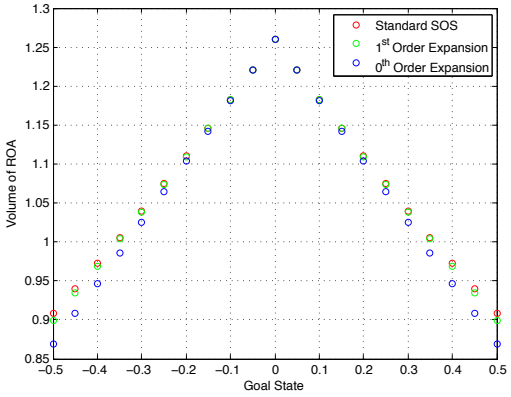

Fig. 3: Volumes of regions of attraction for the Van der Pol oscillator for various order expansions of the control. All verifications are exact.

system:

$$\begin{aligned} \dot{x_1} &= x_2 - x_1^2 \\ \dot{x_2} &= (x_2 - x_1^2)x_2 + x_1^2 + u. \end{aligned} \tag{17}$$

The set of stabilizable points of this system is $S = \{(x_1, x_2) | x_2 - x_1^2 = 0\}$. Defining the perturbation parameter δ as the vector $[\delta_1, \delta_2]^T$, we take $x_g(\delta) = \delta$. We can then let δ vary freely in some range, for example $\delta_1 \in [-0.2, 0.2]$, $d_2 \in [0, 0.04]$. However, this parametrization of the goal set will lead to the SOS conditions in (16) being checked outside the one-dimensional manifold S. In order to check the Lyapunov conditions only for the states in S, we can apply the S-procedure verify the conditions only when $\delta_2 = \delta_1^2$. The SOS program then becomes:

$$\begin{aligned} \underset{\rho,\nu,L}{\text{maximize}} \quad & \int_P \rho(\delta)\, d\delta \\ \text{subject to} \quad & \|x - x_g(\delta)\|^2 (V_\delta(,x) - \rho(\delta)) \\ & - \nu(\boldsymbol{x}, \delta) \left(\frac{d}{dt} V_\delta(,x) + c V_\delta(,x) \right) \\ & + L(\delta)(\delta_2 - \delta_1^2) \geq 0 \\ & \rho(\delta) \geq 0 \qquad \forall \delta \in P, x \in \mathbb{R}^n \end{aligned} \tag{18}$$

where $L(\delta)$ is a polynomial multiplier. Results from this SOS program are shown in Figure 4, which plots the ROAs for the goal set depicted by the black curve.

For many mechanical and robotic systems, the set of stabilizable points is, in general, a complicated, disconnected manifold in the configuration space (derivative variables are always zero at a stabilizable point of a mechanical system), defined by $0 = f(\boldsymbol{x}, u)$, plus any input and state constraints. As in the example above, this constraint can be simplified to a tractable form. For (potentially over- or under-actuated) robots defined by the manipulator equations:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Bu,$$

the constraint reduces to

$$Bu = G(q), \quad \dot{q} = 0.$$

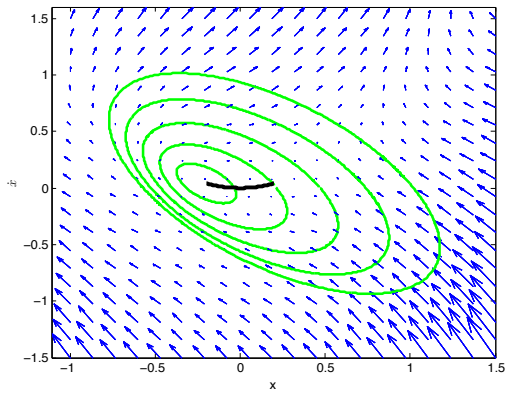

Fig. 4: Regions of attraction for the non-linear system (17) obtained by constraining the goal set to the stabilizable manifold using a SOS equality constraint.

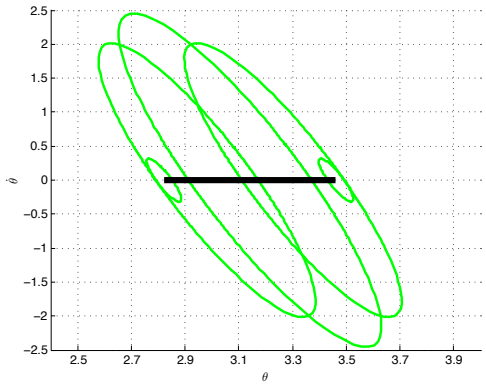

Fig. 5: Regions of attraction for a torque-saturated pendulum.

These can be folded into the SOS verification program by adding equality constraints via multipliers exactly as was done for the system (17).

B. Input Constraints

The basic SOS program presented in Section IV can be augmented in order to handle systems with input saturations. Although we examine the single-input case in this section, this framework is very easily extended to handle multiple inputs.

Let the control law $u(x) = u_0 - Kx$ be mapped through the following control saturation function:

$$s(u(x)) = \begin{cases} u_+ & \text{if } u(x) \geq u_+ \\ u_- & \text{if } u(x) \leq u_- \\ u(x) & \text{o.w.} \end{cases}$$

Then, [14] provides a solution to this problem in the case where one is trying to compute a ROA for a single goal state. This solution involves a simple application of the S-procedure for a piecewise-polynomial analysis of $\dot{V}$, and can be used to augment the SOS program (16) in order to obtain ROAs $\forall \delta \in P$ with the given input saturations.

A plot of ROAs obtained from applying this procedure to a pendulum with torque limits is shown in Figure 5. The goal states, shown in the figure as a black line segment, were in the range $[\pi - 0.3, \pi + 0.3]$. As one would expect, the ROAs get smaller as one moves away from the upright position (i.e. $\theta = \pi$) and become vanishingly small when the maximum torque that the pendulum can exert exceeds the torque due to gravity.

C. Time-Varying Finite-time Invariance Around Trajectories

So far in this paper, we have restricted our attention to computing regions of attraction around stabilizable goal states for time-invariant systems. However, an important extension to this problem is that of computing regions of finite-time invariance (funnels) around time-varying trajectories of a system. [14] uses SOS verification to compute time-varying regions, $\mathcal{B}(\rho(t), t)$ for which the closed-loop system obeys:

$$x(t) \in \mathcal{B}(\rho(t), t) \implies x(t_f) \in \mathcal{B}_G.$$

In a manner similar to the results presented previously in the present work, we extend the results in [14] in order to compute funnels for a *set* of trajectories around some nominal trajectory. We constrain our perturbations to "shifts" of the nominal trajectory. i.e. If $x_0(t) : [t_0, t_f] \mapsto \mathbb{R}^n$ is the nominal trajectory, $x(t; \delta) = x_0(t) + \delta$ is a shifted trajectory (with $\delta \in P$ a constant, and $P \subset \mathbb{R}^n$). We see that generally $x(t; \delta)$ will not be a solution of dynamical equations but, as shown in [15], one can still make guarantees of invariance over finite time intervals.

Further, note that it is considerably more difficult to approximate the optimal time-varying LQR controller for a "shifted" trajectory as a function of δ than it was for the stabilizable goals in Section III. Thus, in this section, we simply use the LQR controller computed for the nominal trajectory for computing funnels for the set of "shifted" trajectories. Hence, our candidate Lyapunov function for each trajectory in the set is given by:

$$V(x, t; \delta) = (x - x(t; \delta))^T S(t)(x - x(t; \delta)$$

where $S(t)$ is the time-varying solution to the differential Riccati equation:

$$-\dot{S}(t) = Q - S(t)B(t)R^{-1}B^T S(t) + S(t)A(t) + A(t)^T S(t)$$

with final value conditions $S(t) = S_0$. (Here $A(t), B(t)$ describe the time-varying linearization of the dynamics around the nominal trajectory).

As described in [14], we want to compute $\rho(t; \delta)$ such that:

$$\dot{V}(x, t; \delta) \leq \frac{d}{dt}\rho(t; \delta).$$

This will provide us with funnels for each δ (of course δ must be constrained to lie in some bounded set). As in Section IV, this can be easily checked using SOS programming. Figure 6 shows the results of the SOS verification on the Van der Pol oscillator. A nominal trajectory (shown in blue), $x(t; [0, 0]^T)$,

of the system was computed, and a family of funnels were computed for "shifted" trajectories around it. The blue funnel was obtained by evaluating $V(x;t;[0,0]^T)$ (i.e. the funnel for the nominal trajectory), while the green funnel was obtained from $V(x;t;[0.07,-0.07]^T)$.

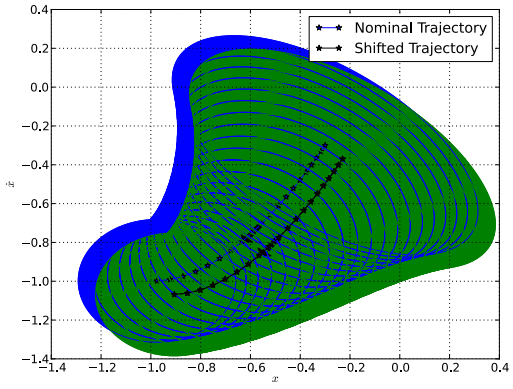

Fig. 6: Regions of Finite-Time Invariance for two trajectories of the Van der Pol oscillator. The blue trajectory is the nominal one, while the black one is the "shifted" one.

Figure 7 compares the funnel for the shifted trajectory $x(t;[0.07,0.07]^T)$ obtained using our method (i.e. by evaluating $V(x;t;\delta)$ at $\delta=[0.07,0.07]^T$) with a funnel computed directly on the "shifted" trajectory using standard SOS methods. As the plot illustrates, our funnel is slightly conservative, though the results are quite close. Of course, as in Section IV, it should be noted that our method provides us with funnels for a *continuum* of "shifted" trajectories, while the standard SOS method only gives us a single funnel.

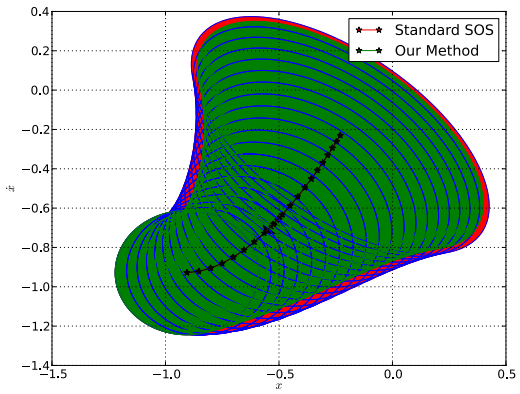

Fig. 7: Comparison of funnels computed on a "shifted" trajectory ($\delta=[0.07,0.07]^T$). The red funnel was computed using Standard SOS methods directly on the shifted trajectory. The green funnel was computed by evaluating our family of funnels at $\delta=[0.07,0.07]^T$.

VII. Discussion

A. Planning Under Goal State Uncertainty

Our method for computing regions of attraction (and possibly LQR controllers) for a *continuum* of goal states allows for compact representation and efficient computation. Further, since the goal state is considered a continuous variable in our method, we avoid discretization and computation issues that arise if one attempts to apply standard region-of-attraction SOS optimization. Instead of having to sample a discrete set of points in the goal set and solving a separate SOS optimization program for each point - a computationally expensive operation - we can simply solve a single SOS program that provides us with ROAs for *all* points in the goal set. This provides us with a powerful tool for feedback motion planning when dealing with situations in which the desired goal state is not determined before runtime. For example, imagine a scenario in which the start state, x_s of the system is known before runtime, but the goal state is declared only at runtime, but is known to lie in a certain goal set, X_g. Using our approach, we can compute regions of attraction for each state in X_g in the pre-computation stage (i.e. before runtime). If X_g is "large" or disconnected, we could divide it into several subsets and compute parameterized ROAs for each subset separately. Further, using trajectory optimization methods [12], we can compute a trajectory, $x(t)$ that starts at x_s and passes through each region of attraction (corresponding to each goal state in X_g). At runtime, when the desired goal state, x_g, is declared, we can simply follow our pre-computed trajectory $x(t)$ and switch to the LQR controller corresponding to x_g when we enter its ROA. This will drive our system to x_g.

These ideas can also be used in order to extend the capabilities of LQR-Trees, a feedback motion planning algorithm, presented in [14]. This algorithm computes a randomized tree of trajectories and stabilizing controllers that grab initial conditions from a bounded set in state space and drive them to the desired goal state. Using our approach, the algorithm can be extended to handle cases where the goal state is not declared before runtime, thus allowing for a *multi-query* version of the algorithm. Such an algorithm would be extremely important in robotics applications such as walking over rough terrain, or flying through a dense forest - scenarios in which the exact geometry of the environment is not known till runtime.

B. Future Considerations

There are still a few outstanding issues that need to be addressed in future work. First, there is the question of convergence of the power series (13). Results from [18], written in the context of the State Dependent Riccati Equation (SDRE) approach, prove local convergence of the power series (13) when the matrices $A(a)$ and $B(b)$ (in Section III) are continuous functions of a and b respectively. The exact radius of convergence and the rate of convergence of the series will, however, depend on how non-linear the system is, thus varying from system to system.

As was observed in Figure 3, the performance of the controller becomes worse as moves away from the nominal state. Clearly, at some distance from the nominal state, the controllers and resulting ROAs will become unacceptably sub-optimal. Thus, the size of the goal set for which the

controllers and ROAs are computed will need to be curtailed accordingly. This needs to be done on a system by system basis. One general heuristic could be to curtail the size of the goal set (or equivalently, the size of the parameter set P in Section III-B) by setting a threshold for the sub-optimality of the controllers as measured by the matrix norm of the difference between the optimal controller and the sub-optimal power series approximation. This will require solving the LQR problem exactly for a discrete set of points and evaluating the matrix norm of the difference at these points. However, this approach still does not guarantee that the resulting ROAs will be acceptably close to optimal, since the relationship between sub-optimality of the controller and sub-optimality of the ROAs is not a simple linear one.

Having mentioned these points, however, it is important to note that although the radius of convergence and sub-optimality of our approximated controller affect the *size* of the ROAs we obtain, they do not affect the *validity* of the ROAs. The computed results are still valid estimates of regions of attraction for the sub-optimal controllers.

VIII. Conclusion

In this paper, we have presented a method for applying sums-of-squares optimization in order to compute parameterized regions of attraction (ROAs) for a set of stabilizable goal states and regions of finite time invariance parameterized by "shifts" to some nominal time-varying trajectory. We are able to compute ROAs for an entire *continuum* of goal states or trajectories, using a single sums-of-squares program. Hence, we avoid issues that arise if one tries to discretize the space of goal states/trajectories. We expect this framework to be useful for online planning in exciting application domains such as UAV flight through cluttered environments and legged locomotion over rough terrain.

IX. Acknowledgements

This work was partially supported by ONR MURI grant N00014-09-1-1051 and the MIT Intelligence Initiative.

References

[1] G. Chesi. Estimating the domain of attraction for uncertain polynomial systems. *Automatica*, 40(11):1981 – 1986, 2004.

[2] T. Cimen. State-dependent riccati equation (sdre) control: A survey. In *Proceedings of the 17th IFAC World Congress*, pages 3761–3775, 2008.

[3] Frazzoli and E. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, June 2001.

[4] D. Henrion and A. Garulli, editors. *Positive Polynomials in Control*. Lecture Notes in Control and Information Sciences. Springer-Verlag, 2005.

[5] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[6] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[7] Lozano-Perez and Tomas. Automatic planning of manipulator transfer movements. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(10):681–698, Oct. 1981.

[8] A. Megretski. Systems polynomial optimization tools (SPOT), available online: `http://web.mit.edu/ameg/www/`. 2010.

[9] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied Mathematics. Society for Industrial Mathematics, 1995.

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

[11] Petersen, I.R., Ugrinovskii, V.A., Savkin, and A.V. *Robust control design using H-8 methods*. Communications and control engineering series. Springer, 2000.

[12] Schultz, G., Mombaur, and K. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, 15(5):783 –792, Oct. 2010.

[13] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625 – 653, 1999.

[14] R. Tedrake, I. R. Manchester, M. M. Tobenkin, and J. W. Roberts. LQR-Trees: Feedback motion planning via sums of squares verification. *International Journal of Robotics Research*, 29:1038–1052, July 2010.

[15] M. M. Tobenkin, I. R. Manchester, and R. Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *Proceedings of the 18th IFAC World Congress, extended version available online: arXiv:1010.3013 [math.DS]*, 2011.

[16] Topcu, U., Packard, A.K., Seiler, P., Balas, and G.J. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137 –142, Jan 2010.

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

[18] A. Wernli and G. Cook. Suboptimal control for the nonlinear quadratic regulator problem. *Automatica*, 11(1):75–84, jan 1975.