Finite-time Regional Verification of Stochastic Nonlinear Systems

Jacob Steinhardt
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307
Email: jsteinha@mit.edu

Russ Tedrake
Computer Science and Aritificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307
Email: russt@mit.edu

September 19, 2011

Abstract

Recent trends pushing robots into unstructured environments with limited sensors have motivated considerable work on planning under uncertainty and stochastic optimal control, but these methods typically do not provide guaranteed performance. Here we consider the problem of bounding the probability of failure (defined as leaving a finite region of state space) over a finite time for stochastic nonlinear systems with continuous state. Our approach searches for exponential barrier functions that provide bounds using a variant of the classical supermartingale result. We provide a relaxation of this search to a semidefinite program, yielding an efficient algorithm that provides rigorous upper bounds on the probability of failure for the original nonlinear system. We give a number of numerical examples in both discrete and continuous time that demonstrate the effectiveness of the approach.

1 Introduction

Consider the problem of a legged robot quickly traversing unknown rough terrain, a vision-based autonomous vehicle flying through a dense forest at high speeds, or a mobile manipulator fetching a beer out of the refrigerator. Each of these robots will be subject to many sources of uncertainty — including uncertainty from imperfect perception, imperfect models of robot and environment, and any unexpected disturbances. At the same time, we hope that our robots are able to accomplish their tasks by executing high-speed dynamic maneuevers, which demands that a high-performance control system will have to reason about the nonlinear dynamics of the machine. While there has been considerable progress recently in designing impressive control systems for this class of machine (e.g., [2, 9, 20, 31, 27, 26]), there is relatively little work on guaranteeing that these systems can achieve their goals in the presence of significant uncertainty.

Most notions of stability that are used in deterministic stability analysis do not apply directly to stochastic stability analysis. For example, a deterministic system that is stable to a fixed point in the sense of Lyapunov (i.s.L.) quickly becomes unstable i.s.L. if it is subjected to even a small amount of Gaussian noise; in most cases these systems will eventually leave any finite region around the fixed point with probability 1. In fact, since Gaussian noise is unbounded, worst-case robustness analysis and approaches, for instance, based on common Lyapunov functions also do not apply. Instead, here we attempt to analyze the stochastic stability of the nonlinear system over a finite time horizon — a framework considered in [15], which is also a special case of planning with chance constraints as formulated in [8]. In particular, we would like to verify an *activation set* — a set of intial conditions from which a closed-loop system will provably achieve its goal with a desired probability. In addition to certifying performance, efficient algorithms for verifying this stochastic stability will lend themselves naturally to improved methods for feedback design and planning algorithms under chance constraints.

A common approach to evaluating stochastic stability of nonlinear systems is by converting it to a finite Markov chain via a finite interpolation of the state-space — either by direct discretization, or through some more sophisticated technique like volumetric interpolation. However, this approach has multiple shortcomings — first, such approximations can end up having large effects on the result, even for relatively large numbers of interpolating functions and for well-behaved systems. Second, for more than a few dimensions there will not be enough memory on the computer to store even coarse approximations to the continuous-state dynamics (for instance, a discretization-based approach in a recent paper hits computation limits around 5 to 8 dimensions [1]; the methods presented here can solve a similar problem in 10 dimensions). For both of these reasons we have been led to consider continuous-state verification. In other words, we would like to perform verification directly on the original system instead of first making a finite-dimensional approximation.

Relatively little progress has been made on the problem of continuous-state, nonlinear, stochastic verification, although many special cases have been studied. If we eliminate stochasticity, then we can perform sum-of-squares verification on Lyapunov functions [25, 29, 30, 19, 21]. If we eliminate continuous-state, then exact solutions can be found by taking a matrix exponential (in continuous-time) or matrix power (in discrete-time) of the transition matrix for the Markov process, after adding an appropriate absorbing state to capture all of the failed states [6]. If we assume linearity of the system then the problem falls into the risk-sensitive control framework [13], which handles not only verification but control design. Risk-sensitive control also deals with nonlinear systems, but in the nonlinear case typically requires a discretization of the state space, which is problematic for the reasons discussed above.

There has been some progress on dealing with the general case. The main approach is to find supermartingales of the system, which bound the probability of leaving a region [3]. These supermartingales can be thought of as stochastic analogues of Lyapunov functions, and are called barrier functions in [23]. They can alternately be thought of as upper bounds on a certain cost-to-go function. We think of these as *certificates of stability*, in the sense that they certify that the system will be stable with high probability.

However, unless the noise is zero at some point, no supermartingale exists (see Proposition 2.6). This requires a slight variation on the supermartingale criterion, as in Kushner [15] or Pham [22], which gives bounds very similar to our Theorems 2.3 and 2.4. We highly recommend Kushner's paper, as it develops a very complete mathematical theory and also works through many practically relevant examples for various noise models. Kushner also proves Theorems 2.3 and 2.4 of this paper. The drawback of Kushner's work is that it does not provide any general algorithms for finding good supermartingales. We hope to remedy this with our work.

Much of the continuous-state verification research has focused on nonlinear systems with Gaussian and switching noise [23, 32]. In this paper we will focus on just Gaussian noise, although we believe that extending the techniques to include switching systems should not be too difficult. This is because the theory presented here holds for general Markov processes, with the computational results we provide tailored to Gaussian noise.

The results in [23] are as far as we know the first to provide an algorithm for finding supermartingales. However, their approach has a few shortcomings that we address. The first is that their method requires their barrier function to be a true supermartingale, which by Proposition 2.6 cannot exist for systems with Gaussian noise; they circumvent this problem by presupposing stochastic stability for sufficiently small initial conditions, a condition which is difficult to check and not always true. A second issue is that they search over polynomially growing barrier functions, which will not give as strong of guarantees as exponential barrier functions. At the same time, while it is tractable to search over relatively high-degree barrier functions in the CT case, we believe that such a search becomes quickly infeasible in the DT case because the Lyapunov function composed with the dynamics leads to a polynomial whose degree is the product of the degrees of the dynamics and Lyapunov functions; this belief is based mainly on our own efforts to apply the methods of [23] to the DT case, as [23] only considers the CT case. Finally, we present results for time-varying systems as well as high-dimensional systems and systems with complex noise dynamics.

To summarize, we are interested in bounding the probability that a nonlinear, possibly time-varying, system with Gaussian noise leaves a region (either pre-specified or computed as part of the optimization) in a certain time interval. We will do this by using the supermartingale approach discussed in [15], searching over a family of exponentially growing barrier functions. We will use sum-of-squares programming to identify a member of this class that provides a good bound on the failure probability.

We start in Section 2 by presenting Kushner's bounds on failure probability. In Section 2 we also give an overview of sum-of-squares programming, an optimization technique that will be important for finding a good barrier function. Next, in Section 3, we will define the family of barrier functions that we intend to search over, and provide semidefinite constraints that allow us to bound the failure probability. In Section 4, we go over specific algorithms for efficiently finding a good certificate of stability. In Section 5 we provide validation algorithms for our method. We conclude in Section 6 by providing examples of our approach on a variety of systems, including a pendulum in discrete and continuous time, a cart-pole system, a simple walking robot example called the rimless wheel, a planar quadrotor perturbed by wind gusts, and the heating system described in [1].

2 Background

There are two main pieces of background relevant to our approach. The first concerns the bounding of statistics on Markov processes; this may be familiar to readers either in terms of the Bellman equations for dynamic programming or in terms of the theory of supermartingales. We will present both approaches here. This material is presented in Subsections 2.1 through 2.3.

The second piece of background is convex optimization, and, more specifically, sum-of-squares optimization. This is the tool that allows us to turn our mathematical theory into an efficient algorithm for verifying a wide range of systems. We omit a general overview of convex optimization and instead focus on sum-of-squares programming. For more information on convex optimization, we direct the interested reader to Boyd and Vandenberghe's book [5], which is available for free online. The sum-of-squares material is presented in Subsection 2.4.

2.1 Statistics on Markov Chains

In this paper, we are interested in showing that the trajectories of a system remain within some possibly time-varying) "safe" region R_t, where the index t corresponds to time. We will refer to the probability of ever leaving the safe set before some final time T as the **failure probability**, and refer to the problem of upper-bounding the failure probability as **stochastic verification**.

We can formulate stochastic verification for a given Markov process $\mathcal{M}$ as a final value cost problem on a modified Markov process. We do this by defining a random variable over time, τ, that is equal to $\min(T, \inf\{t \mid x(t) \not\in R_t\})$. Now modify $\mathcal{M}$ into a new process $\tilde{\mathcal{M}}$ which stops at time τ and define a final value cost of 1 if $x(\tau) \not\in R_\tau$ and 0 otherwise. Under some mild conditions, $x(\tau) \not\in R_\tau$ exactly captures the trajectories that fail by time T:

Lemma 2.1. *Suppose that $\bigcup_t R_t \times \{t\}$ is an open set and that x is right-continuous with probability 1. Then $x(\tau) \not\in R_\tau$ if and only if $x(t) \not\in R_t$ for some $t \in [0, T]$.*

As long as the conditions of Lemma 2.1 hold, the expected final cost is exactly the probability of leaving the safe set R_t for some $t \leq T$. This observation implies that upper-bounding the failure probability for a Markov process $\mathcal{M}$ is equivalent to upper-bounding the expected cost for the modified Markov process $\tilde{\mathcal{M}}$. There are two major approaches to upper-bounding cost for a Markov process — the Bellman equations, and supermartingales. In both cases, the usual strategy is to find a function of state that is non-increasing in expectation whose value at time T upper-bounds the final cost; such a function is called a barrier function, and it is an upper bound on the expected final cost (which would be constant in expectation). However, as we will show, such functions do not exist for most real dynamical systems (unless the statistics are time-varying), and we instead must find a statistic that is increasing very slowly in expectation. In this case, a version of the expected cost bound still holds. There are also generalizations of these results for the case when there are costs on intermediate states as well as the final state; we will touch on these in the next subsection, but focus on the final value case.

Before continuing, it will be useful to review a few facts about continuous-time Markov processes. Suppose that we have some statistic $J(x, t)$ on a Markov process with state variable x. Then it is natural to consider the "expected derivative" of J:

$$\mathcal{A}\, J(x(t), t) = \lim_{t' \downarrow t} \frac{\mathbb{E}[J(x(t'), t') \mid x(t)] - J(x(t), t)}{t' - t}, \tag{1}$$

where $\downarrow$ means that the limit is taken from the right.

If this limit converges uniformly in x and t, then we will refer to J as *$\mathcal{A}$-differentiable*. The uniform convergence property is necessary to perform any useful analysis, as the statistic defined by $J(x, t) = \left\{ \begin{array}{ccc} 0 & : & t < 1 \\ 1 & : & t \geq 1 \end{array} \right.$ has $\mathcal{A}\, J(x, t) = 0$ for all t, but is not constant. Fortunately, as long as J is A-differentiable, we have the following result, which can be thought of as a stochastic analogue of the fundamental theorem of calculus:

Proposition 2.2 (Dynkin's formula)**.** *Let $J(x, t)$ be $\mathcal{A}$-differentiable. Then $\mathbb{E}[J(x(\tau), \tau) \mid x(t_0), t_0] = J(x(t_0), t_0) + \mathbb{E}\left[\int_{t_0}^{\tau} \mathcal{A}\, J(x(s), s) ds\right]$ for any random time τ.*

This result, as well as a thorough treatment of the technical issues surrounding statistics of Markov processes, is covered in Dynkin's book on the subject [10]. We refer the reader in particular to equations (1.2) and (5.8) and the surrounding exposition. We note here that Dynkin's formula holds for Itō diffusions (differential equations driven by Gaussian noise) as long as J is twice-differentiable.

In the following two subsections, we will establish upper bounds on the expected cost in two different ways, obtaining the same bound in both cases. The first method, presented in Subsection 2.2, uses the Bellman equations in an optimal control deriviation. The second method, presented in Subsection 2.3, uses supermartingales. We state the bound here for convenience, in both discrete-time and continuous-time. The discrete-time and continuous-time bounds appear as Theorems 2 and 1, respectively, of [14]. In the theorem statements below, J is the aforementioned barrier function and δ measures the extent to which J fails to decrease in expectation (this is stated more precisely in the theorems).

Theorem 2.3 (Discrete-time cost bound)**.** *Let $\mathcal{M}$ be a Markov chain over a space X. Let J be a scalar function, let $\delta(n) \geq 0$, and let $B_n = \{x \mid J(x,n) + \sum_{m=n}^{N-1} \delta(m) \leq \rho\}$. Suppose that*

$$\begin{aligned} J(x,n) &\geq \mathbb{E}[J(x,n+1) \mid x(n)] - \delta(n) && \forall x \in B_n && (2)\\ J(x,N) &\geq 0 && \forall x && (3)\\ B_n &\subset R_n. && && (4) \end{aligned}$$

Then the probability of failure given initial conditions (x_0, n_0) is at most

$$\frac{J(x_0,n_0) + \sum_{m=n_0}^{N-1} \delta(m)}{\rho} \tag{5}$$

Theorem 2.4 (Continuous-time cost bound)**.** *Let $\mathcal{M}$ be a strong Markov process over a space X whose trajectories are right-continuous with probability 1. Let J be a scalar function, let $\delta(t) \geq 0$, and let $B_t = \{x \mid J(x,t) + \int_t^T \delta(s)ds \leq \rho\}$. Suppose that J is $\mathcal{A}$-differentiable and that*

$$\begin{aligned} \mathcal{A}\, J(x,t) &\leq \delta(t) && \forall x \in B_t && (6)\\ J(x,T) &\geq 0 && \forall x && (7)\\ B_t &\subset R_t. && && (8) \end{aligned}$$

Then the probability of failure given initial conditions (x_0, t_0) is at most

$$\frac{J(x_0,t_0) + \int_{t_0}^T \delta(s)ds}{\rho}. \tag{9}$$

All of the important content for the sequel is in these two theorems. The reader should feel free to skip either or both of subsections 2.2 and 2.3 if desired.

2.2 Bellman Equations

As before, suppose that we want to evaluate some statistic over a Markov process. If the cost of being in a state x at time n is $h(x,n)$, and the final value cost is $h(x,N)$, then the cost-to-go function $J(x,n)$ is the solution to the *Bellman equations*

$$J(x(n),n) = h(x,n) + \mathbb{E}[J(x(n+1),n+1) \mid x(n)] \tag{10}$$
$$J(x(N),N) = h(x,N). \tag{11}$$

If instead of evaluating the cost exactly, we wanted to compute an upper bound, then it would suffice to find a function J such that

$$J(x(n),n) \geq h(x,n) + \mathbb{E}[J(x(n+1),n+1) \mid x(n)] \tag{12}$$
$$J(x(N),N) \geq h(x,N), \tag{13}$$

and similarly for computing a lower bound. But what if (12) is only approximately satisfied? For instance, suppose that $J(x(n),n) \geq h(x,n) + \mathbb{E}[J(x,n+1)] - \delta(n)$ for some $\delta(n) > 0$. Then the function $\tilde{J}(x,n) = J(x,n) + \sum_{m=n}^{N-1} \delta(m)$ will satisfy the required inequalities, and therefore be an upper bound on the cost. We also note that all of these observations continue to hold even if N is a random time, rather than being deterministic, as long as it is finite in expectation.

With these observation in place, we are ready to prove Theorem 2.3.

Proof of Theorem 2.3. Condition (4) of Theorem 2.3 says that $x \in R_n$ whenever $\tilde{J}(x,n) < \rho$. We can therefore bound the failure probability for the condition $\tilde{J}(x,n) < \rho$, since this is a stronger condition than $x \in R_n$. Recall that the failure probability is equal to the expected cost for the modified Markov chain $\tilde{\mathcal{M}}$, which has $h(x,n) = 0$ if $n < \tau$ or $\tilde{J}(x,n) < \rho$, and $h(x,\tau) = 1$ if $\tilde{J}(x,\tau) \geq \rho$.

Condition (2) of Theorem 2.3 ensures that $\tilde{J}/\rho$ satisfies the Bellman inequality (12) for $\tilde{\mathcal{M}}$. Moreover, condition (3) of Theorem 2.3 ensures that $\tilde{J}/\rho$ satisfies (13) — if $\tilde{J}(x,\tau) < \rho$, then $\tilde{J}/\rho \geq 0 = h(x,\tau)$, and if $\tilde{J}(x,\tau) \geq \rho$, then tautologically $\tilde{J}/\rho \geq 1 = h(x,\tau)$. Thus $\tilde{J}/\rho$ is an upper bound on the expected cost and hence the failure probability, which is the conclusion of the theorem. □

We now move on to the continuous-time case. In this case, we end up with the Hamilton-Jacobi-Bellman inequalities:

$$\mathcal{A}\, J(x(t),t) \leq -h(x(t),t) \tag{14}$$
$$J(x(T),T) \geq h(x(T),T), \tag{15}$$

where T is the (possibly random) stopping time. As before, (14) and (15) imply that J is an upper bound on the cost as long as J is $\mathcal{A}$-differentiable. However, this now requires a proof.

Proposition 2.5. *Let $J_0(x(t),t)$ denote the true expected cost. If (14) and (15) are both satisfied, and J is $\mathcal{A}$-differentiable, $J(x(t),t) \geq J_0(x(t),t)$ for all $t \leq T$.*

Proof. Letting J_0 be the exact cost function, we obtain:

$$\mathcal{A}(J - J_0) \leq 0 \tag{16}$$
$$J(x,T) \geq J_0(x,T). \tag{17}$$

Now suppose for the sake of contradiction that $J(x(t),t) < J_0(x(t),t)$ for some t. Then, applying Proposition 2.2 (Dynkin's formula), condition (16) implies that $\mathbb{E}[(J - J_0)(x(T),T) \mid x(T)] = (J - J_0)(x(t),t) + \mathbb{E}[\int \mathcal{A}(J - J_0)(x(s),s)ds] \leq (J - J_0)(x(t),t) < 0$, so $J(x(T),T) < J_0(x(T),T)$, which contradicts (17). We therefore must have $J(x(t),t) \geq J_0(x(t),t)$, as was to be shown. □

With Proposition 2.5 in place, Theorem 2.4 follows in basically the same way as Theorem 2.3, where we define $\tilde{J}(x,t) := J(x,t) + \int_t^T \delta(s)ds$.

Proof of Theorem 2.4. Condition (8) of Theorem 2.4 says that $x \in R_t$ whenever $\tilde{J}(x,t) < \rho$, so we can again upper bound the probability of leaving R_t by the probability that $\tilde{J}(x,t) \geq \rho$. We again seek to compute the expected cost for the Markov chain $\tilde{\mathcal{M}}$ with $h(x,t) = 0$ for $t < \tau$ and final value cost $h(x,\tau) = \left\{ \begin{array}{lcl} 0 & : & \tilde{J}(x,\tau) < \rho \\ 1 & : & \tilde{J}(x,\tau) \geq \rho \end{array} \right.$.

Now apply Proposition 2.5 with final time τ and cost function $\tilde{J}/\rho$. By condition (6) of Theorem 2.4, $\tilde{J}/\rho$ satisfies the Bellman inequality (16). By condition (7) of Theorem 2.4, $\tilde{J}/\rho \geq 0 = J_0(x,\tau)$ whenever $\tilde{J}(x(\tau),\tau) < \rho$. Finally, $\tilde{J}/\rho \geq 1 = J_0(x,\tau)$ tautologically whenever $\tilde{J}(x(\tau),\tau) \geq \rho$. Hence the conditions of Proposition 2.5 are satisfied, and therefore $\tilde{J}$ is an upper bound on the expected cost, as was to be shown. □

2.3 Supermartingales

We now adopt the supermartingale perspective on stochastic verification. We begin with an extension of the classical result about stability of supermartingales. Recall that a *supermartingale* is a function $J(x,t)$ of a Markov process such that $\mathbb{E}[J(x(t+\Delta t), t+\Delta t) \mid x(t)] \leq J(x(t),t)$ for all $\Delta t \geq 0$. We will instead consider functions that are almost supermartingales, in the sense that $\mathbb{E}[J(x(t+\Delta t), t+\Delta t)] \mid x(t)] \leq J(x(t),t) + \int_t^{t+\Delta t} c(s)ds$ for some function c that depends only on time. We will call such functions *c-martingales.* Note the similarity to the condition $\mathcal{A}\, J(x(t),t) \leq c(t)$. In fact, Proposition 2.2 says that $\mathcal{A}\, J(x(t),t) \leq c(t)$ implies the c-martingale condition. In discrete time, we instead consider the condition $\mathbb{E}[J(x(t+1), t+1)] \mid x(t)] \leq J(x(t),t) + c(n)$.

One might wonder why to bother with extending the martingale condition to the c-martingale condition (or why we only dealt with approximate solutions to the Bellman equations in the previous section). One answer lies in the following:

Proposition 2.6. *Let $dx(t) = f(x)dt + g(x)dw(t)$ be a stochastic differential equation on the real line, where f and g are both continuous. If $g(x) > 0$ for all x, then there is no time-invariant supermartingale $J(x)$ that obtains its global minimum on a bounded set.*

Proof. Suppose for the sake of contradiction that such a J exists, and let S be the set on which J obtains its global minimum. Let $\bar{S}$ be the closure of S. Then $\bar{S}$ is compact, and hence f and g obtain minimum values f_0 and g_0, respectively, on $\bar{S}$, where $g_0 > 0$. Now since J obtains its global minimum on S, and J is a supermartingale, for any trajectory x with $x(t_0) \in S$, we must have $x(t) \in S$ with probability 1 for any $t \geq t_0$. Pick $x(t_0)$ to be any point in S. Now assuming that the trajectory $x(s) \in S$ for all $s \in [t_0, t]$, we can upper-bound $x(t)$ by:

$$\begin{aligned} x(t) &= x(t_0) + \int_{t_0}^t f(x)dt + g(x)dw(t) \\ &\geq x(t_0) + \int_{t_0}^t \inf\{f(x)\}dt + \inf\{g(x)\}dw(t) \\ &\geq x(t_0) + \int_{t_0}^t f_0 dt + g_0 dw(t) \\ &= x(t_0) + f_0 \cdot (t - t_0) + g_0 \cdot [w(t) - w(t_0)]. \end{aligned}$$

But the final equation is a Gaussian distribution, and hence unbounded. Thus there is some nonzero probability that $x(t) \notin S$ (since S is bounded), which is a contradiction, so J cannot exist. $\square$

In particular, unless the noise is equal to zero at some point, no radially unbounded supermartingale can exist. The relaxation of the supermartingale condition to the c-martingale condition is thus crucial in allowing us to consider interesting classes of systems. We note that our relaxation of the supermartingale condition is similar to the approach taken in [22] for contracting systems.

We can also draw an analogy between c-martingales and amortized analysis in computer science — if there is some function of our state that increases slowly, then it will be a long time before it can reach a large value. If we can find a function J of our state that increases slowly in expectation (such as a c-martingale for small c), and J is large outside of a region of state space, then it will take a long time for a trajectory of the system to escape that region.

We will now prove Theorems 2.3 and 2.4 from the perspective of supermartingale theory.

Proof of Theorem 2.3. Note that if J is a c-martingale, $\tilde{J}(x,n) := J(x,n) + \sum_{m=n}^{N-1} \delta(m)$ is a supermartingale in the region where $\tilde{J}(x,n) < \rho$. Define a stopping time τ for $\mathcal{M}$ as the minimum of T and $\min_n \{\tilde{J}(x(n),n) \geq \rho\}$. Then by the optional stopping theorem, the expected value of $\tilde{J}$ at τ is at most $\tilde{J}(x(t_0),t_0)$. Thus by Markov's inequality, the probability that $\tilde{J} \geq \rho$ is at most $\frac{\tilde{J}(x(t_0),t_0)}{\rho}$. Since $\tilde{J}(x(\tau),\tau) \geq \rho$ for all failed trajectories, this means that the probability of failure is at most $\frac{\tilde{J}(x(t_0),t_0)}{\rho}$ as well, as was to be shown. $\square$

Proof of Theorem 2.4. Note that if J is a c-martingale, $\tilde{J}(x,t) := J(x,t) + \int_t^T \delta(t)dt$ is a supermartingale in the region where $\tilde{J}(x,t) < \rho$. Define a stopping time τ for $\mathcal{M}$ as the minimum of T and $\inf_t \{\tilde{J}(x(t),t) \geq \rho\}$. Since $\tilde{J}$ and δ are continuous and trajectories of $\mathcal{M}$ are right-continuous, $\tilde{J}(x(\tau),\tau) \geq \rho$ whenever the minimum is attained by the inf (rather than by T).

By the optional stopping theorem, the expected value of $\tilde{J}$ at τ is at most $\tilde{J}(x(t_0),t_0)$. Thus by Markov's inequality, the probability that $\tilde{J} \geq \rho$ is at most $\frac{\tilde{J}(x(t_0),t_0)}{\rho}$. But as shown in the preceding paragraph, $\tilde{J}(x(\tau),\tau) \geq \rho$ for all failed trajectories. This implies that the probability of failure is at most $\frac{\tilde{J}(x(t_0),t_0)}{\rho}$ as well, as was to be shown. $\square$

In the following sections, we will discuss how to usefully apply Theorems 2.3 and 2.4 to dynamical systems with Gaussian noise.

2.4 Sum-of-squares Programming

Suppose that we want to compute the global minimum of a polynomial $p(x_1,\dots,x_n)$. We could formulate this as the optimization problem

$$\begin{aligned} \underset{\delta}{\text{maximize}} \quad & \delta \\ \text{subject to} \quad & p(x) - \delta \geq 0 \quad \forall x. \end{aligned} \tag{18}$$

This problem is NP-hard in general; however, if we could write $p(x) - \delta = h(x)^T Q h(x)$ for some matrix $Q \succeq 0$, then we would know that $p(x) - \delta \geq 0$ for all x. We can thus replace (18) with

$$\begin{aligned} \underset{Q,\delta}{\text{maximize}} \quad & \delta \\ \text{subject to} \quad & p(x) - \delta = h(x)^T Q h(x) \\ & Q \succeq 0. \end{aligned} \tag{19}$$

This is now a semidefinite program, which can be solved efficiently by an optimizer such as SeDuMi [28]. Typically h will be a vector of monomials, but this is not a necessity. The existence of a Cholesky factorization when $Q \succeq 0$ implies that $p(x) - \delta$ can be written as a sum of squares of other polynomials, which is how sum-of-squares programming gets its name.

We can more generally consider programs with several positivity constraints, e.g.

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & h^T \alpha \\ \text{subject to} \quad & \alpha^T p_i(x) \geq 0, \quad i = 1, \ldots, m, \end{aligned} \tag{20}$$

which are then replaced with

$$\begin{aligned} \underset{\alpha,Q}{\text{maximize}} \quad & h^T \alpha \\ \text{subject to} \quad & \alpha^T p_i(x) = h_i(x)^T Q_i h_i(x), \quad && i = 1, \ldots, m \\ & Q_i \succeq 0, && i = 1, \ldots, m. \end{aligned} \tag{21}$$

The α are referred to as *decision variables* and the x are referred to as *free variables.* The expression $h^T \alpha$ is called the *objective* of the program, since it is the quantity that we are trying to maximize.

An extension of this approach is to *matrix sum-of-squares*, where one includes in (20) constraints of the form $P_i(x) \succeq 0$, where P_i is a matrix of polynomials with coefficients linear in α. Since $P_i(x) \succeq 0$ if and only if $y^T P_i(x) y \geq 0$ for all y, we can encode such polynomial semidefiniteness constraints by adding additional free variables to the program.

We may also wish to only enforce a constraint $p_i(x) \geq 0$ in some region described by $q_i(x) \leq 0$. In this case, we can introduce a *Lagrange multiplier* $\lambda(x)$ and impose the constraints $p_i(x) + \lambda(x) q_i(x) \geq 0$ and $\lambda(x) \geq 0$. Note that only one of λ and q_i can be optimized over at once; the other one must be constant to avoid decision variables appearing nonlinearly in the constraints. If we wish to optimize over both λ and q_i, then we generally fix one and optimize over the other, then alternate and repeat until convergence.

We finally make note of the method of *Schur complements*, which allow us to re-formulate certain nonlinear constraints as semidefinite constraints. More precisely, we have that $A \succeq 0$ and $B^T A^{-1} B \preceq C$ if and only if $\left[\begin{array}{cc} A & B \\ B^T & C \end{array} \right] \succeq 0$.

Sum-of-squares programs can be formulated using the MATLAB package YALMIP [16]. YALMIP is a modeling language for both convex and non-convex programs. YALMIP has built-in support for several optimizers; we used SeDuMi [28] for our work. SeDuMi is a software package for optimizing over symmetric cones. While the final version of our code uses YALMIP, we also used CVX [11, 12] and SOSTOOLS [24] during development. CVX is a modeling language for convex programs; SOSTOOLS is a MATLAB toolbox for sum-of-squares programs. All of the software mentioned here is freely available online.

3 Certificates of Stability

Theorems 2.3 and 2.4 show us how to obtain true certificates of stability from approximate certificates. In order to usefully apply these theorems, we need to pick a suitable barrier function for a given system. Using the machinery described in Subsection 2.4, we will find the best barrier function over an entire parameterized family.

For now, the systems we are interested in have polynomial dynamics and (possibly state-dependent) Gaussian noise. Note that in practice, we can Taylor expand most systems in order to obtain polynomial dynamics. In the DT case, the family of systems we consider are of the form $x_{n+1} = f(x_n) + g(x_n)w_n$, where w_n is unit covariance white noise and f and g are polynomials. In the CT case, we consider systems of the form $dx(t) = f(x)dt + g(x)dw(t)$, where w is a vector of independent unit-variance Wiener processes, and f and g are again polynomials. All of the following results also hold for time-varying f and g, but we will sometimes omit the possible dependence on t to keep the equations more readable.

We will consider barrier functions of the form $J_S(x, t) = e^{x^T S(t)x} - 1$. According to the bounds in Theorems 2.3 and 2.4, functions which grow quickly will yield tighter bounds. Functions of the form considered here can grow more quickly than the polynomials used in [23]. Note that, in the DT case, including cubic or higher terms in the exponent would make the expected value of J_S infinite with respect to Gaussian noise (this is not true for the CT case).

Our general goal in the next few sections will be to first explicitly compute the conditions of Theorems 2.3 and 2.4 for our choice of system and barrier function. Then, we will use various inequalities to strengthen the conditions to conditions that are polynomial in x and linear in the various decision variables (which will in this case be S, ρ, and a few other variables to be defined below). Unfortunately, we will not quite achieve this goal; the conditions will instead be bilinear in the decision variables, a problem we will address in Section 4.

3.1 Discrete-time

In discrete-time, we can compute

$$\mathbb{E}[J_S(x(t+1)) \mid x(t)] = \det(I - 2g^T S g)^{-\frac{1}{2}} e^{f(x)^T S\left(S - 2Sgg^T S\right)^{-1} S f(x)} - 1.$$

Applying Theorem 2.3 to J_S lets us bound the failure probability by

$$\frac{e^{x(0)^T S(0) x(0)} - 1 + \sum_{n=0}^{N-1} \delta(n-1)}{e^{\rho} - 1} \tag{22}$$

as long as $x^T S(n)x \geq \rho$ for all $x \notin R_n$ and

$$\delta(n-1) \geq \det(I - 2g^T S(n) g)^{-\frac{1}{2}} e^{f^T S(n)(S(n) - 2S(n)gg^T S(n))^{-1} S(n) f} - e^{x^T S(n-1)x} \tag{23}$$

whenever $x^T S(n)x < \rho$. The expression for $\delta(n-1)$ is cumbersome, as it involves a determinant as well as the difference of two exponential functions. The following two lemmas let us relax the expression to a condition on polynomials.

Lemma 3.1. $\det(I - M) \geq 1 - \mathrm{Tr}(M)$ *when* $0 \preceq M \preceq I$.

Lemma 3.2. *Let p_0, q_0, and r_0 be real numbers with $r_0 < 1$, and let $M := (1-r_0)^{-\frac{1}{2}}e^{p_0} - e^{q_0}$. Suppose that $M \geq 0$ and*

$$\begin{aligned} (1-r_0)^{-\frac{1}{2}}e^{p_0}(p-p_0) - e^{q_0}(q-q_0) &\leq \delta && (24)\\ r &\leq r_0. && (25) \end{aligned}$$

Then

$$(1-r)^{-\frac{1}{2}}e^p - e^q \leq Me^{\frac{\delta}{M}}. \tag{26}$$

Proofs for these lemmas may be found in the appendix. Applying the two lemmas and the Schur complement, we obtain the following proposition:

Proposition 3.3. *Consider a system of the form $x_{n+1} = f(x_n) + g(x_n)w_n$, and let $J_S(x,n) := e^{x^TS(n)x} - 1$. Suppose that for all n and all x with $x^TS(n-1)x < \rho$, we have*

$$\begin{aligned} S(n) &\succeq 0 && (27)\\ b(n) &\leq 1 && (28)\\ \begin{bmatrix} 1 & b(n-1) \\ b(n-1) & 2b(n-1) - 2\operatorname{Tr}(g(x,n-1)^TS(n)g(x,n-1)) \end{bmatrix} &\succeq 0 && (29)\\ \begin{bmatrix} S(n) - 2P(n) & S(n)f(x,n-1) \\ f(x,n-1)^TS(n) & (1-b(n-1))x^TS(n-1)x \end{bmatrix} &\succeq 0 && (30)\\ \begin{bmatrix} I & g(x,n-1)^TS(n) \\ S(n)g(x,n-1) & P(n) \end{bmatrix} &\succeq 0. && (31) \end{aligned}$$

Then the probability that $J_S(x,n) \geq \rho$ for any $0 \leq n \leq N$ is at most

$$\frac{e^{x_0^TS(t_0)x_0} - 1 + \sum_{n=t_0}^{N-1}(1-b(n))^{-1} - 1}{e^\rho - 1}. \tag{32}$$

The derivation may be found in the appendix.

Remark If the noise g is 0, we can set b to 0. It is easy to check that the constraints then reduce to the Lyapunov equation $f(x)^TS(t)f(x) \leq x^TS(t-1)x$.

3.2 Continuous-time

We now turn to the continuous-time case. Recall that we are interested in the infinitesimal operator $\mathcal{A}\,J(x,t)$ defined in Equation 1. For systems of the form $dx(t) = f(x)dt + g(x)dw(t)$, we can compute [32]

$$\mathcal{A}\,J(x,t) = \frac{\partial J}{\partial t} + \frac{\partial J}{\partial x}f(x) + \frac{1}{2}\operatorname{Tr}\left(g(x)^T\frac{\partial^2 J}{\partial x^2}g(x)\right). \tag{33}$$

For functions of the form $J_S(x) = e^{x^TS(t)x} - 1$, (33) becomes

$$\mathcal{A}\,J_S(x,t) = e^{x^TSx} \times \left[x^T\dot{S}x + 2x^TSf + \operatorname{Tr}\left(g^TSg\right) + 2x^TSgg^TSx\right]. \tag{34}$$

Then Theorem 2.4 implies that the failure probability is bounded by

$$\frac{e^{x(0)^TS(0)x(0)} - 1 + \int_0^T \delta(t)dt}{e^\rho - 1} \tag{35}$$

as long as (i) $x^T Sx \geq \rho$ for all $x \notin R_t$ and (ii) $\delta(t) \geq e^{x^T Sx} \left[x^T \dot{S}x + 2x^T Sf + \mathrm{Tr}(g^T Sg) + 2x^T Sgg^T Sx\right]$ whenever $x^T Sx < \rho$. We would therefore like an analog of Lemma 3.2 for functions of the form $p(x)e^{q(x)}$. The following will suffice:

Lemma 3.4. *Suppose that $p(x) \leq p_0(1 + q_0 - q(x))$ and $p_0 \geq 0$. Then $p(x)e^{q(x)} \leq p_0 e^{q_0}$.*

The proof is in the appendix. Using Lemma 3.4 and the Schur complement, we obtain:

Proposition 3.5. *Consider a system of the form $dx(t) = f(x)dt + g(x)dw(t)$, and let $J_S(x, t) = e^{x^T S(t)x} - 1$, where S is a continuous matrix-valued function of time. Suppose that for all t and all x with $x^T S(t)x < \rho$, we have*

$$S(t) \succeq 0 \tag{36}$$

$$\left[\begin{array}{cc} \frac{1}{2}I & g^T Sx \\ x^T Sg & b(1 - x^T Sx) - x^T \dot{S}x - 2x^T Sf - \mathrm{Tr}(g^T Sg) \end{array}\right] \succeq 0. \tag{37}$$

Then the probability that $J_S(x, t) \geq \rho$ for any $0 \leq t \leq T$ is at most

$$\frac{e^{x_0^T S(t_0)x_0} - 1 + \int_{t_0}^{T} b(s)ds}{e^{\rho} - 1}. \tag{38}$$

The derivation may be found in the appendix.

4 Optimizing over S

Propositions 3.3 and 3.5 give us regional polynomial conditions to check in order to get upper bounds on the failure probability. This can be accomplished by sum-of-squares programming with Lagrange multipliers (see Subsection 2.4). However, instead of checking a fixed set of constraints, we would like to be able to optimize the decision variables to minimize the value of our upper bound. If we let λ denote the added Lagrange multipliers, then constraints (27-31) and (36-37) are linear in S and ρ, and also linear in λ and b. Unfortunately, they are not linear in all of S, ρ, λ, and b jointly. As a result, we will need to do some additional work to reduce the constraints to a sum-of-squares program. We also need to choose a good objective function for the program, since the probability bounds (32) and (38) are too complex to optimize directly.

Another practical issue is where in the equations to place Lagrange multipliers. In the algorithms presented below, we present potential examples of places to add them, but in any specific case the efficiency of the method will depend greatly on careful choices of where to put Lagrange multipliers and what degree of multiplier to use. In instances where the Lagrange multiplier is an entire matrix of polynomials, we recommend a strategy such as constraining the polynomial to be a multiple of the identity, or leaving it out altogether if possible.

We will describe below two different approaches for optimizing the decision variables. The first works for both DT and CT systems, as long as they have time-invariant dynamics. The second approach only works for CT systems, but can handle time-varying dynamics (the ideas extend to the DT case in principle, but we have been unable to actually get the method to work for DT in practice, partly due to numerical issues).

For now, when the dynamics are time-varying, we will only check the constraints at a finite (but ideally finely-spaced) set of times $t_1, \ldots, t_k$, following [30]. We also need to choose how to parameterize S as a function of time. The easiest would be to make it piecewise constant, but we need S to be continuous. We therefore choose an S that is piecewise linear on each $[t_i, t_{i+1}]$. Note that this means that S is not differentiable at the t_i, and so we need to check the constraints for both the left- and right-derivatives of S. We also note that there exist more sophisticated approaches for dealing with time-varying constraints, as discussed in [30].

Finally, note that all of the algorithms described below are also provided as pseudocode in the appendix. The pseudocode for Subsection 4.1 is in Algorithms 1-4. The pseudocode for Subsection 4.2 is in Algorithm 5.

4.1 Method 1: Linearize and Binary Search

Our first method begins by considering the system linearized about a given fixed point and finding values S_0 and b_0 that work well for the linearized system. The linear case allows us to satisfy the constraints globally, and hence we can ignore λ and ρ. Once we have found S_0 and b_0, we move back to the original system, but only consider multiples cS_0 of S_0. For any fixed ρ, we can binary search over c, and this turns out to leave us with a sum-of-squares program. We expand on these ideas below.

Without loss of generality, we can assume that we are linearizing about the origin. Then let $F := \nabla f(0)$ be the dynamics linearized about the origin. We will also approximate the noise as having a constant value of $g(0)$.

DT case: If we apply Proposition 3.3 to a time-invariant system with linear dynamics and constant noise, we get the constraints:

$$S_0 \succeq 0 \tag{39}$$

$$b_0 \leq 1 \tag{40}$$

$$\begin{bmatrix} 1 & b_0 \\ b_0 & 2b_0 - 2\operatorname{Tr}(g(0)^T S_0 g(0)) \end{bmatrix} \succeq 0 \tag{41}$$

$$\begin{bmatrix} S_0 - 2P_0 & S_0 F \\ F^T S_0 & (1-b_0)S_0 \end{bmatrix} \succeq 0 \tag{42}$$

$$\begin{bmatrix} I & g(0)^T S_0 \\ S_0 g(0) & P_0 \end{bmatrix} \succeq 0 \tag{43}$$

Note that (42) is equivalent to asking that $\begin{bmatrix} S_0 - 2P_0 & S_0 F x \\ x^T F^T S_0 & (1-b)x^T S_0 x \end{bmatrix} \geq 0$ for all x, which is what (30) would have asked for. For any fixed b_0, constraints (39-43) form a semidefinite program in S_0. We can thus perform a line search over b_0, solving a semidefinite program over S_0 for each value. However, we need a good objective to optimize against. We choose to maximize α such that $S_0 \succeq \alpha M$, for some well-chosen matrix M (this is equivalent to requiring that $x^T S_0 x \geq \alpha$ whenever $x^T M x \geq 1$). There are two reasons to use this objective. First, if $x^T M x$ gives an indication of how nonlinear the system is at x, then we want $x^T S_0 x$ to be large whenever $x^T M x$ is large; this makes it more likely that S_0 will work well for the nonlinear system. Second, if M defines some safety constraint (i.e. the system is safe if $x^T M x < 1$), then we would like S_0 to be large relative to M in order to minimize the failure probability.

With S_0 computed in this way, we turn to the original nonlinear system. Set S to be cS_0, apply Proposition 3.3, and add Lagrange multipliers $\lambda(x)$ to check the constraints in the region

$x^T S_0 x < \rho$. Then we get the constraints:

$$c \succeq 0 \tag{44}$$

$$\begin{bmatrix} 1 & b \\ b & 2b - 2c\,\mathrm{Tr}(g^T S_0 g) \end{bmatrix} \succeq 0 \tag{45}$$

$$\lambda_1(x), \lambda_2(x), \lambda_3(x) \geq 0 \tag{46}$$

$$\begin{bmatrix} cS_0 - 2P & cS_0 f \\ cf^T S_0 & (1-b)cx^T S_0 x + \lambda_1(x)(x^T S_0 x - \rho) \end{bmatrix} \succeq 0 \tag{47}$$

$$\begin{bmatrix} I + \lambda_2(x)(x^T S_0 x - \rho) & cg^T S_0 \\ cS_0 g & P + \lambda_3(x)(x^T S_0 x - \rho) \end{bmatrix} \succeq 0. \tag{48}$$

We may also care about the constraint $S_0 \succeq \rho M$ in the case that $x^T M x < 1$ is a safety condition. This last constraint merely constrains ρ to lie in some interval and hence can be easily dealt with.

Note that the Lagrange multipliers ask for the constraints to hold in the region $x^T S_0 x < \rho$, which is the same as $x^T (cS_0) x > c\rho$. Since the dominant term in the probability bound (32) is then $e^{c\rho}$, we would like to make $c\rho$ as large as possible, which for a fixed ρ is the same as making c as large as possible. It is easy to check that making c larger makes (45-48) harder to satisfy. Our strategy is thus to perform a line search over ρ and then binary search on c. In practice, we find that the maximum possible value of c is fairly flat up to some critical value of ρ, after which it decays sharply.

CT case: If we apply Proposition 3.5, we get the constraints:

$$S \succeq 0 \tag{49}$$

$$\begin{bmatrix} \frac{1}{2}I & g(0)^T S_0 x \\ x^T S_0 g(0) & b_0 - \mathrm{Tr}(g(0)^T S_0 g(0)) - x^T(b_0 S_0 + S_0 F + (S_0 F)^T)x \end{bmatrix} \succeq 0. \tag{50}$$

We can re-write (50) as

$$\begin{bmatrix} I & 0 \\ 0 & x \end{bmatrix}^T \begin{bmatrix} \frac{1}{2}I & g(0)^T S_0 \\ S_0 g(0) & -(b_0 S_0 + S_0 F + (S_0 F)^T) \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & x \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & b_0 - \mathrm{Tr}(g(0)^T S_0 g(0)) \end{bmatrix} \succeq 0,$$

so that we can replace (50) with the two constraints

$$\begin{bmatrix} \frac{1}{2}I & g(0)^T S_0 \\ S_0 g(0) & -(b_0 S_0 + S_0 F + (S_0 F)^T) \end{bmatrix} \succeq 0 \tag{51}$$

$$\mathrm{Tr}(g(0)^T S_0 g(0)) \leq b_0. \tag{52}$$

The system (49,51,52) is again semidefinite for a fixed value of b_0, so we can apply the same strategy as before of line searching over b_0 and then optimizing α against S_0.

For the nonlinear system, we again set S to be cS_0. Then after applying Proposition 3.5 and adding Lagrange multipliers, we get the constraint

$$\begin{bmatrix} \frac{1}{2}I + \lambda_1(x)(x^T S_0 x - \rho) & g^T c S_0 x \\ cx^T S_0 g & b(1 - cx^T S_0 x) - 2cx^T S_0 f - c\,\mathrm{Tr}(g^T S_0 g) + \lambda_2(x)(x^T S_0 x - \rho) \end{bmatrix} \succeq 0 \tag{53}$$

$$\lambda_1(x), \lambda_2(x) \geq 0 \tag{54}$$

As in the DT case, our goal is to maximize $c\rho$. We line search over ρ and then binary search over c to find the largest possible c for a fixed ρ.

4.2 Method 2: Bilinear Optimization

Our next method works for time-varying systems. However, we have only been able to successfully implement it for continuous-time systems. As such, we will only describe it in the continuous-time setting.

After adding Lagrange multipliers, the constraints from Proposition 3.5 become

$$S(t) \succeq 0 \tag{55}$$

$$\left[\begin{array}{cc} \frac{1}{2}I & g^T Sx \\ x^T Sg & b(1 - x^T Sx) - x^T \dot{S}x - 2x^T Sf - \mathrm{Tr}(g^T Sg) + \lambda(x)(x^T Sx - \rho) \end{array} \right] \succeq 0 \tag{56}$$

$$\lambda_1, \lambda_2 \geq 0. \tag{57}$$

These are linear in S and ρ for fixed values of b and λ, and they are linear in b and λ for fixed values of S and ρ. Our strategy is thus to alternate between optimizing (S, ρ) for fixed (b, λ), and optimizing (b, λ) for fixed (S, ρ); we then repeat until convergence. However, we need a good objective to optimize against in each case. Since the dominant term in the probability bound (38) is the e^ρ in the denominator, a good proxy for minimizing the probability bound is to make ρ as large as possible. We will use this when optimizing against (S, ρ). However, we cannot use this when optimizing against (b, λ), as ρ is no longer a decision variable in this case! Instead, we will try to choose values of b and λ that leave as much room for future growth of ρ as possible (in other words, we want to choose b and λ that are well within the interior of the constraint polytope). We do this by first maximizing b, then minimizing b, then taking the average of the two results. By convexity, this point still satisfies the constraints, and since it is the average of two opposite extremes it is likely to be far away from the boundary of the constraint polytope.

A final issue is how to actually initialize the bilinear search, as we require at the very least a feasible initial point (S, ρ, b, λ). Also, since the bilinear search is not necessarily finding a global optimum, the choice of initial point will potentially affect the performance of the algorithm. We outline our current approach to initialization here, although this is one part of our algorithm that could use improvement. We first remove the Lagrange multipliers and arbitrarily set b to be 1, which results in an optimization over S only (ρ is irrelevant now that the Lagrange multipliers are gone). However, the only feasible value of S will be 0 unless the system is globally stable, so we also Taylor expand all the constraints to second order (this roughly corresponds to linearizing the system). To ensure that S is not too close to the boundary of the feasible region, we then optimize $\mathrm{Tr}(H^T S)$ for several randomly chosen matrices H and take the average value.

A strictly feasible point for the linearized constraints will be feasible for the actual constraints as long as ρ is small enough, so we can then decrease ρ until S is in the feasible region.

The pseudocode for the bilinear search (without the initialization step) can be found in Algorithm 5.

5 Validation Methods

In this section we discuss approaches for validating the output of our algorithm. First, as noted by Johan Löfberg[16], it is important to realize that due to termination conditions and numerical tolerances, the output of the SDP solver may not be a valid sum-of-squares certificate for our problem. When accuracy is a premium, we make use of the techniques presented in [17] to attempt to obtain a true certificate.

We can also verify the certificate empirically. Note that if the certificate is valid, then $\mathbb{E}[J_S(x(t+\Delta t)) \mid x(t)] \leq J_S(x(t)) + \int_t^{t+\Delta t} b(s)ds$. Thus for any time t, $\mathbb{P}[J_S(x(t+\Delta t)) > 2(J_S(x(t)) + \int_t^{t+\Delta t} b(s)ds)] < \frac{1}{2}$ by Markov's inequality. We can thus simulate N different trajectories, and for a specific choice of t and Δt look at the number of trajectories where $J_S(x(t+\Delta t)) > 2(J_S(x(t)) + \int_t^{t+\Delta t} b(s)ds)$. If this is significantly greater than $\frac{N}{2}$, then our certificate is invalid. If this is less than or equal to $\frac{N}{2}$, then it provides evidence in favor of the certificate being valid (although is not a proof of validity). We can quantify what "significantly greater than $\frac{N}{2}$" means in terms of the P-value of a one-sided binomial test (a test to see whether the probability of an event is greater than a given number θ; in this case, $\theta = \frac{1}{2}$). Of course, there is nothing special about the number 2. We could replace it with any constant greater than 1 and get a similar test.

Finally, the certificates can always be evaluated through Monte-Carlo; e.g., by simulating many trajectories of the stochastic system and simply counting how many leave the ρ-sublevel set of J_S. In practice, we hope that our bounds are relatively tight over-estimates of this probability of failure. This exhaustive simulation approach is only suitable for relatively low-dimensional examples.

6 Examples

Now that we have covered the theoretical underpinnings of our method, we will demonstrate its effectiveness with several examples. For each example, we first describe the system, then indicate which M matrix we used (see Section 4), the values of S_0 and b_0, the values of c and ρ, and the final probability bound.

6.1 Example 1: Linear 1D Systems

Our first example is a test of how tight our bounds are. We considered the system

$$dx(t) = -xdt + \sigma dw(t) \tag{58}$$

for varying values of σ in the range $[0, 2]$, and computed the probability that $|x(1)| \geq 1$ given that $x(0) = 0$. We did this both by exhaustive Monte Carlo simulation and by using Algorithm 3. We similarly considered the system

$$x(n+1) = (1 - \Delta t) \cdot x(n) + \sqrt{\Delta t}\sigma w(n) \tag{59}$$

for $\Delta t = 0.01$. This is the discrete-time approximation to the continuous-time system defined in (58). The results are given in Figure 1. Our bounds are reasonably tight for σ close to 1. In particular, our answer is within a factor of 2 of the true answer for $\sigma \in [0.7, 1.2]$. In addition, our bound outputs negligibly small failure probabilities (less than 10^{-8}) for $\sigma \leq 0.2$, whereas the true failure probability falls below 10^{-8} somewhere around $\sigma = 0.3$ (this answer is approximate, as we did not have the computational resources to simulate 10^8 trajectories). This suggests that our method performs well relative to the truth as long as the noise is about the same size as the signal, and will output very small probabilities when the signal-to-noise ratio is at least 5.

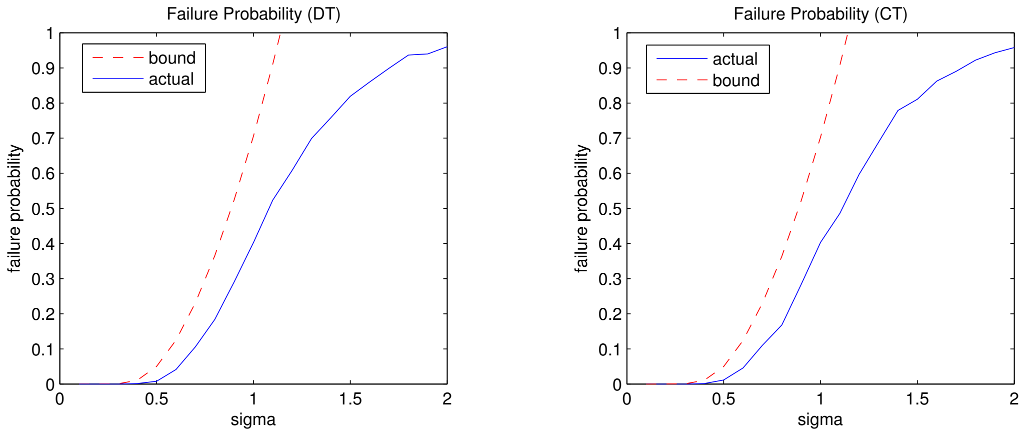

Figure 1: True probability of failure (blue) versus computed probability bound (red) as a function of the noise variance. Left: discrete-time. Right: continuous-time.

6.2 Example 2: Simple Pendulum, Discrete Time

Our second example is a pendulum stabilized about the upright with a time step of $\Delta t = 0.01$. We use the following equations for the pendulum dynamics (the sin term has been Taylor expanded to third order):

$$\begin{bmatrix} \theta_{n+1} \\ \dot{\theta}_{n+1} \end{bmatrix} = \begin{bmatrix} \theta_n + 0.01\dot{\theta}_n \\ -0.0167\theta_n^3 - 0.3\theta_n + 0.97\dot{\theta}_n \end{bmatrix} + \begin{bmatrix} 0.01w_{1,n} \\ 0.05w_{2,n} \end{bmatrix}.$$

We want to bound the probability that θ leaves the region $\left(-\frac{\pi}{6}, \frac{\pi}{6}\right)$ after 3600 seconds. We thus set M to $\begin{bmatrix} \left(\frac{6}{\pi}\right)^2 & 0 \\ 0 & 0 \end{bmatrix}$, as then $x^T M x > 1 \iff |\theta| > \frac{\pi}{6}$.

For $b_0 = 0.0136$, we get $S_0 = \begin{bmatrix} 71.36 & 3.75 \\ 3.75 & 2.55 \end{bmatrix}$ with $\alpha = 18.05$. When we verify on the nonlinear system, we get $c = 0.955$, $\rho = 17.24$ (ρ is equal to $c\alpha$ because the constraint $S \succeq \rho M$, which ensures that the region $x^T S x < \rho$ satisfies the constraint on θ, was the first to become tight). Figure 2 shows the log of the failure probability plotted against initial conditions.

Note that we get strong bounds (failure probabilities less than 10^{-3}) for a large region around the origin. For the sake of comparison, we estimated the actual failure probability using a Kalman filter for the linearized system, also included in Figure 2. While the true probabilities are much smaller than verified (10^{-10} vs. 10^{-4}), the verified region of stability is not much smaller than the actual region of stability. For most robotics applications we are more interested in the operating region where we have a high success probability than in how small the failure probability is for zero initial conditions. In this respect our verification method is close to the true answer.

6.3 Example 3: Simple Pendulum, Continuous Time

We perform the same optimization as before, checking against the continuous-time version of the constraints. For $b_0 = 1.51$, we get $S_0 = \begin{bmatrix} 78.45 & 4.17 \\ 4.17 & 2.82 \end{bmatrix}$ with a corresponding α value of 19.82. When we verify on the nonlinear system, we get $c = 1.0$, $\rho = 19.82$. The failure probability is

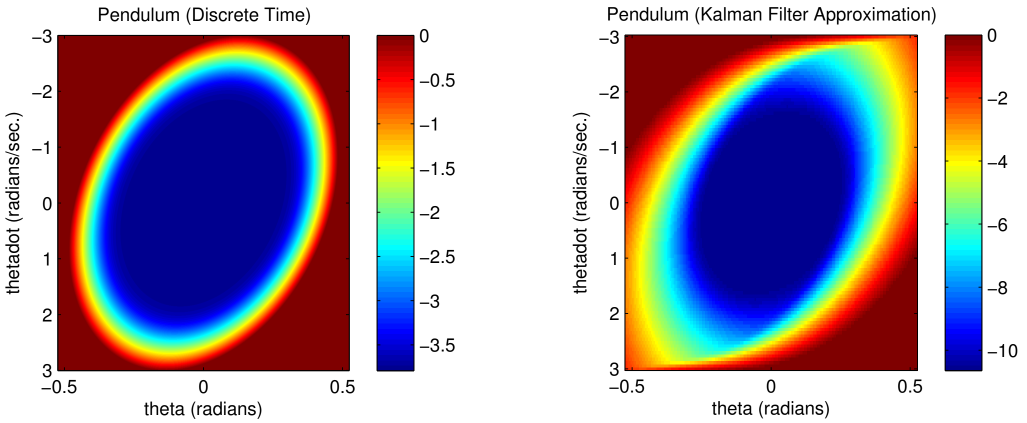

Figure 2: The log-base-10 of the failure probability for the discrete-time pendulum after one hour. Left: failure probability plotted against initial conditions, verified with our algorithm. Right: estimated failure probability for the linearized discrete-time pendulum, computed with a Kalman filter.

plotted in Figure 3.

6.4 Example 4: Cart-Pole Balancing, Continuous Time

The next example demonstrates that our approach is scalable to more complicated systems. It is also an example of including observation noise in the model. The cart and pole system is a pendulum with length L and mass m_p attached to a cart with mass m_c. The system is actuated by a force u on the center of mass of the cart. Letting $\theta = 0$ when the pendulum is pointing straight up, the equations of motion are

$$\ddot{x} = \frac{u - m_p \sin(\theta)(L\dot{\theta}^2 - g\cos(\theta))}{m_c + m_p \sin(\theta)^2},$$

$$\ddot{\theta} = \frac{u\cos(\theta) - m_p L\dot{\theta}^2 \cos(\theta)\sin(\theta) + (m_c + m_p) g \sin(\theta)}{L(m_c + m_p \sin(\theta)^2)}.$$

We set $m_p = 1.0$, $m_c = 10.0$, $L = 0.5$, $g = 9.8$, and take a third-order Taylor expansion to get the following dynamics:

$$\begin{bmatrix} dx \\ d\theta \\ d\dot{x} \\ d\dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ -.75\theta^3 - .01\theta^2 u - .05\theta\dot{\theta}^2 + .98\theta + .1u \\ -5.75\theta^3 - .12\theta^2 u - .1\theta\dot{\theta}^2 + 21.56\theta + .2u \end{bmatrix} dt$$
$$+ \operatorname{diag}\left(\begin{bmatrix} 0.03 & 0.03 & 0.1 & 0.1 \end{bmatrix}\right) dw(t).$$

To stabilize this system, we apply LQR control to the linear system with cost matrices $Q = \operatorname{diag}([10, 10, 1, 1])$, $R = 0.1$ to get a gain matrix of $K = \begin{bmatrix} -10.0 & 289.83 & -19.53 & 63.25 \end{bmatrix}$.

Let us suppose that we also have independent measurement noise on x, θ, $\dot{x}$, and $\dot{\theta}$, with standard deviations of 0.01, 0.01, 0.03, and 0.03, respectively. Our feedback law will push this noise back into the dynamics, adding 4 extra noise channels that end up being functions of θ.

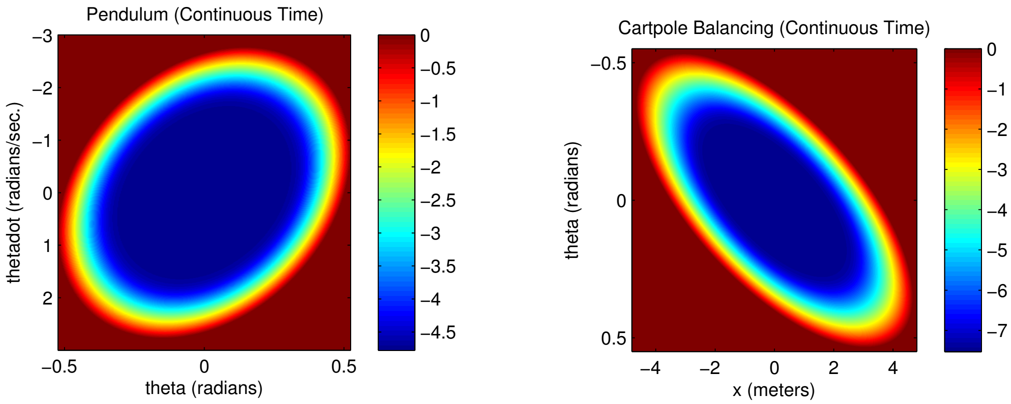

Figure 3: The log-base-10 failure probabilities for two continuous-time systems. Left: balancing for the pendulum, as a function of initial conditions. Right: balancing for the cartpole, as a function of initial conditions in x and θ (the initial conditions for $\dot{x}$ and $\dot{\theta}$ are fixed to 0).

Because the major source of nonlinearity is θ, want $x^T S_0 x$ to grow quickly with θ. We will therefore set M to $\mathrm{diag}\left(\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}\right)$. For $b_0 = 0.728$, we get $S_0 = \begin{bmatrix} 1.23 & -6.01 & 1.29 & -0.93 \\ -6.01 & 115.71 & -10.86 & 11.36 \\ 1.29 & -10.86 & 2.98 & -2.23 \\ -0.93 & 11.36 & -2.23 & 2.63 \end{bmatrix}$, with $\alpha = 62.18$. When we verify on the nonlinear system, we get $c = 0.9023$, $\rho = 10.38$. Figure 3 contains a visualization of the failure probability after one hour.

6.5 Example 5: Rimless Wheel

The rimless wheel is a common model for walking first introduced in [18]. It is a wheel consisting of n_s spokes, each of length L, connected at a point. The angle between consecutive spokes is $\theta = \frac{2\pi}{n_s}$. The spokes are massless; the central point has a mass of M. The rimless wheel typically rolls down a hill, say with slope angle γ. When a spoke impacts the ground, the collision is inelastic, conserves angular momentum, and immediately transfers support to the next spoke. Because of the impacts, the rimless wheel is an inherently discrete-time system. One way to compute its dynamics across several collisions is via the *Poincaré return map*, which gives the angular velocity at the point where the stance leg is vertical. If we let ω_n denote this angular velocity between the nth and $(n+1)$st impacts, and let $x_n = \omega_n^2$, then [7]

$$x_{n+1} = \cos^2(\theta)\left(x_n + \frac{2g}{L}(1 - \cos\beta_1)\right) - \frac{2g}{L}(1 - \cos\beta_2),$$

where $\beta_1 = \frac{\theta}{2} + \gamma$ and $\beta_2 = \frac{\theta}{2} - \gamma$. As in [7], we model γ as Gaussian with mean $\gamma_0 = 8°$ and standard deviation $\sigma = 1.5°$. This means that the actual noise to the system is non-Gaussian since it is filtered through a cosine. The system is locally stable to some value $\bar{x} > 0$ as well as to the state where both stance legs are on the ground and the wheel stops moving. We will consider this second stable point a failure state, which corresponds to $x_n \leq 0$.

We will compare the following approaches to bounding the time until the wheel enters this failure state:

One-step slope bound (nonlinear)	313 impacts
One-step slope bound (linear)	428 impacts
Noise as state variable	50 impacts
Linearized noise	12647 impacts
Discrete-state	643600 impacts

Table 1: Expected failure time/50% failure probability thresholds for the rimless wheel. The first, second, and last bounds compute expected failure times, while the second and third bounds compute the time with a 50% failure probability.

1. Find the smallest slope γ_s such that the rimless wheel would roll forever with a constant slope of γ_s. Then compute the probability that $\gamma < \gamma_s$. The expected time to failure is at least the reciprocal of this probability.

2. Let v_n denote $\gamma - \gamma_0$ for time $n + 1$. Then v_n is Gaussian, and it is okay that it affects the dynamics in a nonlinear way because it is a state variable. We can then apply the techniques of this paper to find a time that has at most a 50% probability of failure.

3. Approximate the noise as an appropriate Gaussian by linearizing around γ_0, then apply the techniques of this paper.

4. Discretize the state space and compute the expected time to failure exactly (up to the discretization) by solving a system of equations, as in [7].

In order to make the point of stability the origin, we make the change of coordinates $x \mapsto x - \bar{x}$.

In the first approach, solving for γ_s yields 3.91° in the nonlinear noise case and 3.76° in the linearized case. The respective bounds on expected time to failure are 313.08 and 427.74 impacts, respectively.

In the second approach, we set M to $\begin{bmatrix} \frac{1}{\bar{x}^2} & 0 \\ 0 & 0 \end{bmatrix}$. On the nonlinear system, we obtain $c = 0.972$, $\rho = 3.73$, leading to a bound of $0.4057T$ for initial conditions at the origin. We thus hit 50% failure at $T = \frac{40.49}{2 \times 0.4057} = 49.90$ impacts. This compares poorly to the first approach, which may imply that dealing with non-Gaussian noise by filtering it through nonlinear dynamics does not work well in practice.

In the third approach, we set M to $\frac{1}{\bar{x}^2}$. We get $c = 1$, $\rho = 9.60$, and a 50% failure rate at $T = 12646.90$ impacts, a significant improvement on both of the first two approaches.

Finally, as computed in [7], the actual expected failure time is 643600. These results are summarized in Table 1.

6.6 Example 6: Quadrotor

We now go over an example based on a physically inspired noise model — the Dryden Wind Turbulence model — for a quadrotor. To simplify the dynamics, we assume that yaw and pitch are fixed and thus there is only roll. This means that the quadrotor is constrained to move only in the x and z directions, and can rotate via an angle θ.

Because there is only one rotational degree of freedom, the four rotors can be separated into two pairs that each act identically. If u_1 represents the total force exerted by the two rotors on the left, and u_2 represents the total force exerted by the two rotors on the right, then we have

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{z} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \frac{L}{I}(u_2 - u_1) \\ -g + \frac{(u_1+u_2)\cos(\theta)}{m} \\ -\frac{(u_1+u_2)\sin(\theta)}{m} \end{bmatrix},$$

where L is the length of a rotor arm, m is the quadrotor mass, I is the moment of inertia, and g is the force due to gravity.

We first apply a nominal control signal of $u_1 = u_2 = \frac{mg}{2\cos(\theta)}$ so that $\ddot{z}$ is nominally zero. Then we stabilize the resulting system using LQR control. Following [4], we set $L = 0.25$, $m = 0.486$, $I = 0.00383$, and $g = 9.81$.

For our noise model, we use the Dryden Wind Turbulence model [33], and assume that a wind gust at a given velocity causes $\dot{x}$ to be shifted by that velocity. This model takes the output of a second-order linear-Gaussian system and multiplies it by a nonlinear term $\sqrt{\dfrac{V_0 + \dot{x}}{L_w}}$, where L_w is a length parameter and the model is assumed to hold in the region $\dot{x} \ll V_0$. We take V_0 to be 60.0 and L_w to be 5.0. There is also a turbulence intensity term σ, which we vary from 0.1 to 1.0 for our experiments.

The final model for the system is:

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \\ \ddot{v} \end{bmatrix} = \begin{bmatrix} -g\tan(\theta) - \frac{\sin(\theta)}{m}(u_1 + u_2) + \sigma\sqrt{\frac{V_0}{L_w}}\left(1 + \frac{\dot{x}}{2V_0}\right) \cdot \left(\sqrt{3}\dot{v} + v\frac{V_0}{L_w}\right) \\ \frac{\cos(\theta)}{m}(u_1 + u_2) \\ \frac{L}{I}(-u_1 + u_2) \\ -v\frac{V_0^2}{L_w^2} - 2\dot{v}\frac{V_0}{L_w} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} dw(t).$$

Here v is an extra state-variable added to capture the fact that the noise is itself a second-order system.

We include a plot to demonstrate the results of the verification method — it gives the time until the verified probability is 0.25 as a function of the turbulence intensity σ. This can be found in Figure 4. So, for example, at a turbulence intensity of $\sigma = 0.15$, the bound on the probability of failure is equal to 0.25 at approximately 10^3 seconds. Note that this is the probability of failure given 0 initial conditions, which means that our probability bound scales linearly with time. So, for instance, this also means that the probability of failure is equal to 0.0025 at 10 seconds.

6.7 Example 7: Room Heating

Our final example evaluates the scalability of our approach. We compare our algorithm to the algorithm presented in Abate et al [1]. The experiment presented in [1] concerns bounding the probability that a heating system allows any of h rooms to leave given temperature ranges. For a heating system with h rooms, we represent the temperature of the h rooms as a vector $x = (x_1, x_2, \ldots, x_h)$, and consider the discrete-time system $x_{n+1} = f(x_n) + g(x_n)w_n$ with

$$f(x)_i = \quad x_i + b(x_0 - x_i) + a\left(\sum_{j\neq i} x_j - x_i\right) + c\sigma\left(\frac{x_i}{\alpha} - 1\right) \tag{60}$$

$$g(x) = \quad \nu I_{h\times h}, \tag{61}$$

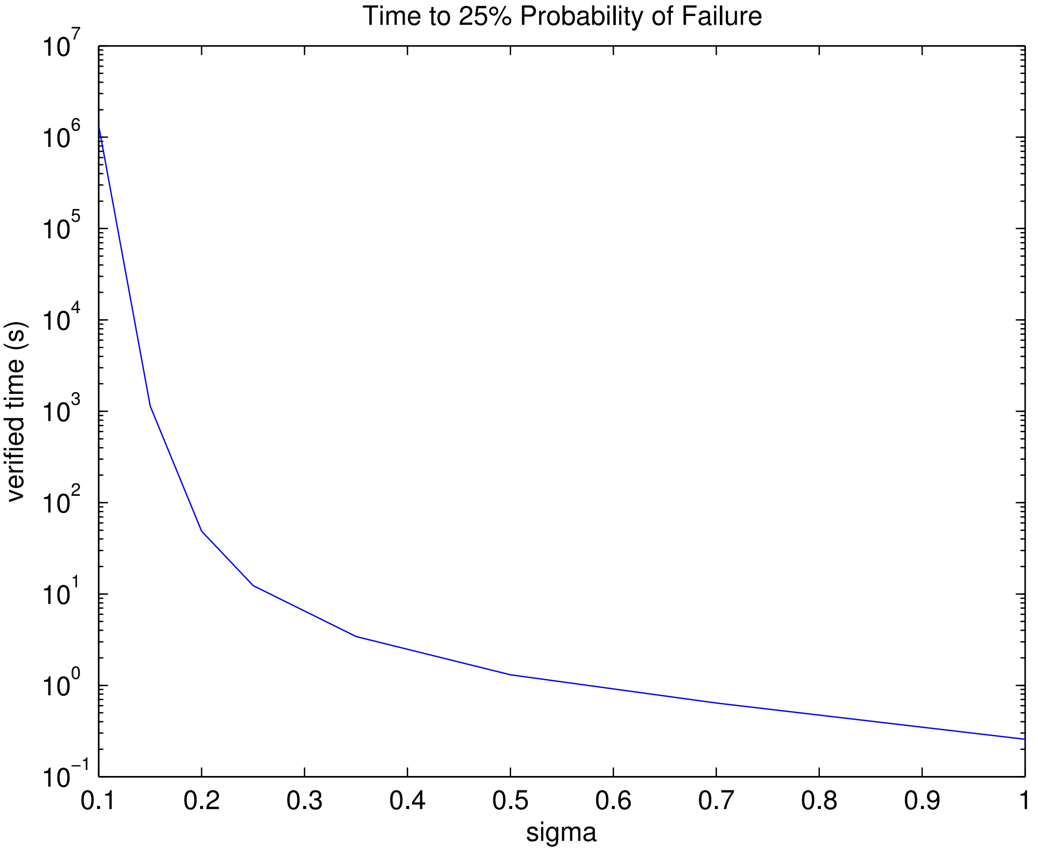

Figure 4: Verified time to failure versus σ (turbulence intensity) for the quadrotor.

where σ is a sigmoidal function rising from 0 to 1, which we approximated as $\sigma(y) = 0.5 - 2.5y + 1.25y^2 + 20y^3$. For our experiment we took $a = 0.0625$, $b = 0.025$, $c = 0.6$, $x_0 = 6.0$, $\alpha = 19.5$, and $\nu = 0.25$. The goal was to bound the probability of leaving the temperature region defined by $[17, 22] \times [16, 23]^{h-1}$. The numbers given above are based on Abate et al.'s paper, although we make a few simplifying assumptions to the dynamics — first, we replace a certain Bernoulli noise source by its expectation; second, we assume symmetric between-room interactions so that there will be an easily identifiable fixed point about which to verify stability. We also remove a one-step lag on noise, which increases the discretization mesh of [1] by a factor of 2 per dimension.

We observe that Abate et al. are able to (using 5 bins per dimension) verify a 5-room heating system in 11 hours on a 3.4GHz PC with 1GB of RAM. Because of the factor of 2 per dimension that they incur, a fair comparison of runtime would be to test our SOS verification on a 7-room heating system (the mesh size in [1] would decrease by a factor of 32 by ignoring lag, then gain a factor of 25 when going from 5 to 7 dimensions, so that their 7-room times without lags would be 6-7 hours, as their runtime scales about quadratically with mesh size).

In this case a single SOS verification runs in an average of 17.2 seconds (our algorithm performs several such verifications). We used a 3.4GHz PC with 24GB of RAM; we note that our PC had 12 cores, with CPU diagnostics indicating that only 4 cores were actually utilized by our computation. We furthermore note that for a fixed degree of Taylor approximation our method scales polynomially with dimension, whereas discretization methods scale exponentially with dimension. Our method is therefore not only more scalable currently, it will also continue to scale well with increased computing power.

7 Conclusion

We have presented a method for verifying stochastic nonlinear systems. However, the results here are by no means a complete theory; there is much work left to be done. Our hope is that the successful examples in this paper will convince others that the methods first presented in [23] can extend usefully to complex systems for suitable choices of barrier functions. We chose exponentials of quadratic barrier functions because the systems we had in mind were locally well-approximated by linear systems and the noise model was Gaussian. Other applications will require different families of barrier functions; hopefully the convex relaxations given in (3.2) and (3.4) will provide inspiration for similar relaxations for those other families. It seems that usually one can obtain such relaxations from simple analytical properties of the expressions in question, but the authors do not yet have a way to make this observation rigorous.

Some interesting modifications to the dynamics would be to consider mixtures of Gaussians, as well as switching processes, in the noise model; also to consider verification about stabilized trajectories. A final case of interest is Gaussian noise passed through a nonlinear filter; as discussed in the Rimless Wheel section, our method handles this case in principal, but performs poorly in practice.

Acknowledgments

This work was supported in part by ONR MURI under grant N00014-09-1-1051.

References

[1] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16:624641, dec 2010.

[2] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of the Neural Information Processing Systems (NIPS '07)*, volume 19, December 2006.

[3] Frederick J. Beutler. On two discrete-time system stability concepts and supermartingales. *Journal of Mathematical Analysis and Applications*, 44(2):464 – 471, 1973.

[4] Bouadi, H., Bouchoucha, M., Tadjine, and M. Sliding mode control based on backstepping approach for an uav type-quadrotor. *International Journal of Applied Mathematics and Computer Sciences*, 4(1):12–17, 2008.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 9780521833783.

[6] Katie Byl and Russ Tedrake. Metastable walking on stochastically rough terrain. In *Proceedings of Robotics: Science and Systems IV*, 2008.

[7] Katie Byl and Russ Tedrake. Metastable walking machines. *International Journal of Robotics Research*, 28(8):1040–1064, August 1 2009.

[8] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6 (1):pp. 73–79, 1959.

[9] Rick Cory and Russ Tedrake. Experiments in fixed-wing UAV perching. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. AIAA, 2008.

[10] E. B. Dynkin. *Markov Processes*, volume 1. Academic Press, 1965.

[11] M Grant and S Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control (tribute to M. Vidyasagar), Lecture Notes in Control and Information Sciences*, pages 95–110. Springer, 2008.

[12] M Grant and S Boyd. CVX: Matlab software for disciplined convex programming - version 1.1beta. `http://cvxr.com/cvx`, January 2011.

[13] Matthew R. James. Asymptotic analysis of nonlinear stochastic risk-sensitive control and differential games. *Mathematics of Control, Signals, and Systems (MCSS)*, 5:401–417, 1992. 10.1007/BF02134013.

[14] H. J. Kushner. On the stability of stochastic dynamical systems. *PNAS*, 53(1):8–12, Jan. 15 1965.

[15] HJ Kushner. Finite time stochastic stability and the analysis of tracking systems. *IEEE Transactions on Automatic Control*, pages 219–227, April 1966.

[16] Johan Lofberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions On Automatic Control*, 54(5):1007–, May 2009.

[17] Johan Lofberg. Strictly feasible sum-of-squares solutions, Feb 2011. http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Blog.Strictly-feasible-SOS-solutions.

[18] Tad McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2): 62–82, April 1990.

[19] A. Megretski. Positivity of trigonometric polynomials. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 4, pages 3814–3817 vol.4, Dec. 2003.

[20] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010)*, 2010.

[21] Antonis Papachristodoulou and Stephen Prajna. Analysis of non-polynomial systems using the sum of squares decomposition. *Positive Polynomials in Control*, 312/2005:23–43, 2005.

[22] Quang-Cuong Pham, Tabareau, N., Slotine, and J.-J. A contraction theory approach to stochastic incremental stability. *Automatic Control, IEEE Transactions on*, 54(4):816 –820, Apr 2009.

[23] S Prajna, A. Jadbabaie, and GJ Pappas. Stochastic safety verification using barrier certificates. *43rd IEEE Conference on Decision and Control*, pages 929–934, 2004.

[24] 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.

[25] Stephen Prajna, Antonis Papachristodoulou, and Fen Wu. Nonlinear control synthesis by sum of squares optimization: A Lyapunov-based approach. In *Proceedings of the ASCC 2004*, 2004.

[26] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, and the BigDog Team. Bigdog, the rough-terrain quadruped robot. *Proceedings of the 17th World Congress, The International Federation of Automatic Control*, 2008.

[27] Alexander Shkolnik, Michael Levashov, Ian R. Manchester, and Russ Tedrake. Bounding on rough terrain with the littledog robot. *Under review*, 2010.

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

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

[30] Mark M. Tobenkin, Ian R. Manchester, and Russ Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *arXiv:1010.3013 [math.DS]*, 2010.

[31] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion.* CRC Press, Boca Raton, FL, 2007.

[32] Y Yang, J Li, and G Chen. Finite-time stability and stabilization of nonlinear stochastic hybrid systems. *Journal of Mathematical Analysis and Applications*, 356:338–345, 2009.

[33] Jesse C. Yeager. Implementation and testing of turbulence models for the f18-harv simulation. Technical report, 1998.

A Proofs and Derivations

A.1 Proof of Lemma 3.1

Lemma. $\det(I - M) \geq 1 - \mathrm{Tr}(M)$ when $0 \preceq M \preceq I$.

Proof. By considering the eigenvalues of M, this is the same as showing that $\prod_{i=1}^{n}(1 - \lambda_i) \geq 1 - \sum_{i=1}^{n} \lambda_i$ whenever $0 \leq \lambda_i \leq 1$. Since $A(1-\lambda) = A - A\lambda \geq A - \lambda \geq B - \lambda$ whenever $B \leq A \leq 1$, the lemma follows by induction on n (with $A = \prod_{i=1}^{n-1}(1 - \lambda_i)$, $B = 1 - \sum_{i=1}^{n-1} \lambda_i$, and $\lambda = \lambda_n$). □

A.2 Proof of Lemma 3.2

Lemma. Let p_0, q_0, and r_0 be real numbers with $r_0 < 1$, and let $M := (1 - r_0)^{-\frac{1}{2}} e^{p_0} - e^{q_0}$. Suppose that $M \geq 0$ and

$$\begin{aligned}(1 - r_0)^{-\frac{1}{2}} e^{p_0}(p - p_0) - e^{q_0}(q - q_0) &\leq \delta \\ r &\leq r_0.\end{aligned}$$

Then

$$(1 - r)^{-\frac{1}{2}} e^{p} - e^{q} \leq M e^{\frac{\delta}{M}}.$$

Proof. Since the left-hand side of (26) is increasing with r, by condition (25) it suffices to consider the case $r = r_0$. We can then maximize $(1 - r_0)^{-\frac{1}{2}} e^{p} - e^{q}$ against (24) using Lagrange multipliers. If we do this, we get the equations

$$\begin{aligned}(1 - r_0)^{-\frac{1}{2}} e^{p_0} &= \lambda(1 - r_0)^{-\frac{1}{2}} e^{p} \\ -e^{q_0} &= -\lambda e^{q}.\end{aligned}$$

Dividing the first equation by the second, we get the equality

$$-(1 - r_0)^{-\frac{1}{2}} e^{p_0 - q_0} = -(1 - r_0)^{-\frac{1}{2}} e^{p - q},$$

which implies that $p - p_0 = q - q_0$. If we let $p - p_0 = q - q_0 = d$, then (24) reduces to $Md \leq \delta$, and the left-hand-side of (26) reduces to Me^{d}. As long as $M \geq 0$, the left-hand-side of (26) is then maximized by making d as large as possible, i.e. setting it to $d = \frac{\delta}{M}$, at which point it is equal to $Me^{\frac{\delta}{M}}$, thus proving the result. □

A.3 Derivation of Proposition 3.3

Applying Lemma 3.1 with $M = 2g^T S(n)g$, we get

$$\begin{aligned}\det(I - 2g^T S(n)g)^{-\frac{1}{2}} e^{f^T S(n)(S(n)-2S(n)gg^T S(n))^{-1}S(n)f} - e^{x^T S(n-1)x} \\ \leq (1 - 2\operatorname{Tr}(g^T S(n)g))^{-\frac{1}{2}} e^{f^T S(n)(S(n)-2S(n)gg^T S(n))^{-1}S(n)f} - e^{x^T S(n-1)x}\end{aligned}$$

as long as $0 \preceq 2g^T S(n)g \preceq I$. Next, applying Lemma 3.2 with $p_0 = q_0 = \delta = 0$ and $r_0 = 2b - b^2$, we get

$$\begin{aligned}(1 - 2\operatorname{Tr}(g^T S(n)g))^{-\frac{1}{2}} e^{f^T S(n)(S(n)-2S(n)gg^T S(n))^{-1}S(n)f} - e^{x^T S(n-1)x} \\ \leq (1 - 2b + b^2)^{-\frac{1}{2}} - 1 \\ = (1 - b)^{-1} - 1\end{aligned}$$

as long as $(1 - b)^{-1} f^T S(n)(S(n) - 2S(n)gg^T S(n))^{-1}S(n)f - x^T S(n-1)x \leq 0$, $2\operatorname{Tr}(g^T S(n)g) \leq 2b - b^2$, and $b \leq 1$. We can thus take $\delta(n-1)$ to be $(1 - b)^{-1} - 1$ as long as the following set of constraints are satisfied:

$$0 \preceq 2g^T S(n)g \preceq I \tag{62}$$

$$f^T S(n)(S(n) - 2S(n)gg^T S(n))^{-1}S(n)f \leq (1 - b)x^T S(n-1)x \tag{63}$$

$$2\operatorname{Tr}(g^T S(n)g) \leq 2b - b^2 \tag{64}$$

$$b \leq 1. \tag{65}$$

We will now manipulate these constraints into a more manageable form. First, note that $b \leq 1$ and $2\operatorname{Tr}(g^T S(n)g) \leq 2b - b^2$ together imply that $2g^T S(n)g \preceq I$. In addition, $S(n) \succeq 0$ implies that $2g^T S(n)g \succeq 0$, so we can drop the first constraint in favor of the simpler constraint $S(n) \succeq 0$. Second, note that, by Schur complements, (63) can be replaced with

$$\begin{bmatrix} S(n) - 2S(n)gg^T S(n) & S(n)f \\ f^T S(n) & (1-b)x^T S(n-1)x \end{bmatrix} \succeq 0 \tag{66}$$

Next, let $P(n)$ be a matrix such that $S(n)gg^T S(n) \preceq P(n)$. Then we can replace (66) with the constraint

$$\begin{bmatrix} S(n) - 2P(n) & S(n)f \\ f^T S(n) & (1-b)x^T S(n-1)x \end{bmatrix} \succeq 0 \tag{67}$$

On the other hand, the constraint $S(n)gg^T S(n) \preceq P(n)$ is equivalent by Schur complements to $\begin{bmatrix} I & g^T S(n) \\ S(n)g & P(n) \end{bmatrix} \succeq 0$. Putting this all together, we can replace (63) with the set of constraints

$$\begin{aligned}\begin{bmatrix} S(n) - 2P(n) & S(n)f \\ f^T S(n) & (1-b)x^T S(n-1)x \end{bmatrix} &\succeq 0 \\ \begin{bmatrix} I & g^T S(n) \\ S(n)g & P(n) \end{bmatrix} &\succeq 0 \\ P(n) &\succeq 0.\end{aligned}$$

Finally, we note that $2\operatorname{Tr}(g^T S(n)g) \leq 2b - b^2 \iff b^2 \leq 2b - 2\operatorname{Tr}(g^T S(n)g) \iff \begin{bmatrix} 1 & b \\ b & 2b - 2\operatorname{Tr}(g^T S(n)g) \end{bmatrix}$, where the last equivalence is by Schur complements. Putting this all back into constraints (63-65), we get Proposition 3.3.

A.4 Proof of Lemma 3.4

Lemma. Suppose that $p(x) \leq p_0(1 + q_0 - q(x))$ and $p_0 \geq 0$. Then $p(x)e^{q(x)} \leq p_0 e^{q_0}$.

Proof. Since $1 - x \leq e^{-x}$, $1 + q_0 - q(x) \leq e^{q_0 - q(x)}$, so $p(x) \leq p_0(1 + q_0 - q(x)) \leq p_0 e^{q_0 - q(x)}$. Multiplying both sides by $e^{q(x)}$ yields $p(x)e^{q(x)} \leq p_0 e^{q_0}$. □

A.5 Derivation of Proposition 3.5

Applying Lemma 3.4 with $p_0 = b$ and $q_0 = 0$ allows us to upper-bound $\mathcal{A}\, J_S(x,t)$ by b as long as

$$x^T \dot{S} x + 2x^T S f + \mathrm{Tr}(g^T S g) + 2x^T S g g^T S x \leq b(1 - x^T S x). \tag{68}$$

We can move terms around in (68) to get the equivalent condition $2x^T S g g^T S x \leq b(1 - x^T S x) - x^T \dot{S} x - 2x^T S f - \mathrm{Tr}(g^T S g)$, which by Schur complements is equivalent to

$$\begin{bmatrix} \frac{1}{2} I & g^T S x \\ x^T S g & b(1 - x^T S x) - x^T \dot{S} x - 2x^T S f - \mathrm{Tr}(g^T S g) \end{bmatrix}.$$

This yields Proposition 3.5.

B Pseudocode

In this section, we provide pseudocode for the algorithms presenting in the main text.

Algorithm 1 Initialization algorithm in DT

```
procedure COMPUTE-$S_0$-DT($M$,$\epsilon$,TOL)    ▷ perform line search over $b_0$, optimizing $S_0$ at each point
                                                   ▷ $\epsilon$ and TOL control the accuracy of the line search
    $l \leftarrow 0$
    $r \leftarrow 1$
    $d \leftarrow \epsilon$
    $S_0^*, b_0^* \leftarrow \emptyset$
    $\alpha^* \leftarrow -\infty$
    while $r - l >$ TOL do    ▷ we narrow the search range until it is smaller than some tolerance
        $S \leftarrow$ RANGE($l, r, d$)    ▷ returns numbers from $l$ to $r$ spaced $d$ apart
        for $b_0 \in S$ do
            $\underset{S_0,P_0,\alpha}{\text{maximize}} \quad \alpha$
            subject to $\quad S_0 \succeq \alpha M$
                $\begin{bmatrix} 1 & b_0 \\ b_0 & 2b_0 - 2\,\mathrm{Tr}(g(0)^T S_0 g(0)) \end{bmatrix} \succeq 0$
                $\begin{bmatrix} S_0 - 2P_0 & S_0 F \\ F^T S_0 & (1 - b_0) S_0 \end{bmatrix} \succeq 0$
                $\begin{bmatrix} I & g(0)^T S_0 \\ S_0 g(0) & P_0 \end{bmatrix} \succeq 0$
            if $\alpha > \alpha^*$ then
                $\alpha^* \leftarrow \alpha$
                $S_0^* \leftarrow S_0$
                $b_0^* \leftarrow b_0$
            end if
        end for
        $l \leftarrow \max(l, b_0^* - d)$    ▷ narrow the search window and increase the search granularity
        $r \leftarrow \min(r, b_0^* + d)$
        $d \leftarrow \epsilon d$
    end while
return $S_0^*$
end procedure
```

Algorithm 2 Binary search algorithm in DT

```
procedure COMPUTE-$S$-DT($M$,$\epsilon$,TOL1,TOL2)
    $S_0 \leftarrow$ COMPUTE-$S_0$-DT$(M, \epsilon, \text{TOL1})$                ▷ see Algorithm 1
    $\rho^* \leftarrow 0$
    $S^*, b^*, \lambda^* \leftarrow \emptyset$
    for $\rho \in R$ do                      ▷ $R$ is a suitable candidate set of $\rho$ values
        $c_l \leftarrow 0$
        $c_r \leftarrow 1$
        while $c_r - c_l > \text{TOL2}$ do                                  ▷ binary search
            $c_m \leftarrow \frac{c_l + c_r}{2}$                           ▷ check feasibility for $c_m$
                   ▷ there is no objective to maximize since we are just checking feasibility
                $\begin{aligned} \text{maximize} \quad & \emptyset \\ \text{subject to} \quad & \begin{bmatrix} 1 & b \\ b & 2b - 2c_m \operatorname{Tr}(g^T S_0 g) \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} c_m S_0 - 2P & c_m S_0 f \\ c_m f^T S_0 & (1-b) c_m x^T S_0 x + \lambda_1(x)(x^T S_0 x - \rho) \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} I + \lambda_2(x)(x^T S_0 x - \rho) & c_m g^T S_0 \\ c_m S_0 g & P + \lambda_3(x)(x^T S_0 x - \rho) \end{bmatrix} \succeq 0 \\ & \lambda_1(x), \lambda_2(x), \lambda_3(x) \geq 0 \end{aligned}$
            if feasible then
                $c_l \leftarrow c_m$
                if $c_l \rho > \rho^*$ then
                    $\rho^* \leftarrow c_l \rho$
                    $(S^*, b^*, \lambda^*) \leftarrow (c_l S_0, b, [\lambda_1, \lambda_2, \lambda_3])$
                end if
            else
                $c_r \leftarrow c_m$
            end if
        end while
    end for
return $(S^*, \rho^*, b^*, \lambda^*)$
end procedure
```

Algorithm 3 Initialization algorithm in CT

```
procedure COMPUTE-$S_0$-CT($M$,$\epsilon$,TOL)    ▷ perform line search over $b_0$, optimizing $S_0$ at each point
                                                   ▷ $\epsilon$ and TOL control the accuracy of the line search
    $l \leftarrow 0$
    $r \leftarrow 1$
    $d \leftarrow \epsilon$
    $S_0^*, b_0^* \leftarrow \emptyset$
    $\alpha^* \leftarrow -\infty$
    while $r - l >$ TOL do    ▷ we narrow the search range until it is smaller than some tolerance
        $S \leftarrow$ RANGE$(l, r, d)$    ▷ returns numbers from $l$ to $r$ spaced $d$ apart
        for $b_0 \in S$ do
            $$\begin{aligned} \underset{S_0, P_0, \alpha}{\text{maximize}} \quad & \alpha \\ \text{subject to} \quad & S_0 \succeq \alpha M \\ & \begin{bmatrix} \frac{1}{2}I & g(0)^T S_0 \\ S_0 g(0) & -(b_0 S_0 + S_0 F + (S_0 F)^T) \end{bmatrix} \succeq 0 \\ & \text{Tr}(g(0)^T S_0 g(0)) \leq b_0 \end{aligned}$$
            if $\alpha > \alpha^*$ then
                $\alpha^* \leftarrow \alpha$
                $S_0^* \leftarrow S_0$
                $b_0^* \leftarrow b_0$
            end if
        end for
        $l \leftarrow \max(l, b_0^* - d)$    ▷ narrow the search window and increase search granularity
        $r \leftarrow \min(r, b_0^* + d)$
        $d \leftarrow \epsilon d$
    end while
return $S_0^*$
end procedure
```

Algorithm 4 Binary search algorithm in CT

```
procedure COMPUTE-$S$-CT($M$,$\epsilon$,TOL1,TOL2)
    $S_0 \leftarrow$ COMPUTE-$S_0 - CT(M, \epsilon, \text{TOL1})$    ▷ see Algorithm 3
    $\rho^* \leftarrow 0$
    $S^*, b^*, \lambda^* \leftarrow \emptyset$
    for $\rho \in R$ do    ▷ $R$ is a suitable candidate set of $\rho$ values
        $c_l \leftarrow 0$
        $c_r \leftarrow 1$
        while $c_r - c_l > \text{TOL2}$ do    ▷ binary search
            $c_m \leftarrow \frac{c_l + c_r}{2}$    ▷ check feasibility for $c_m$
                ▷ there is no objective to maximize since we are just checking feasibility

            maximize    $\emptyset$
            subject to  $\begin{bmatrix} \frac{1}{2}I + \lambda_1(x)(x^T S_0 x - \rho) & c_m g^T S_0 x \\ c_m x^T S_0 g & b(1 - c_m x^T S_0 x) - 2c_m x^T S_0 f \\ & -c_m \operatorname{Tr}(g^T S_0 g) \\ & +\lambda_2(x)(x^T S x - \rho) \end{bmatrix} \succeq 0$
                        $\lambda_1(x), \lambda_2(x) \geq 0$

            if feasible then
                $c_l \leftarrow c_m$
                if $c_l \rho > \rho^*$ then
                    $\rho^* \leftarrow c_l \rho$
                    $(S^*, b^*, \lambda^*) \leftarrow (c_l S_0, b, [\lambda_1, \lambda_2])$
                end if
            else
                $c_r \leftarrow c_m$
            end if
        end while
    end for
return $(S^*, \rho^*, b^*, \lambda^*)$
end procedure
```

Algorithm 5 Bilinear optimization algorithm (CT only)

```
procedure OPTIMIZEBILINEAR(TOL)
    $(S, \rho, \lambda, b) \leftarrow$ INITIALIZE()                ▷ find some initial feasible point
    repeat
        $\rho_0 \leftarrow \rho$

                    $\underset{\lambda, b}{\text{maximize}}$   $b$
                    subject to   $\lambda(x) \geq 0$
                    $\begin{bmatrix} \frac{1}{2}I+ & g^T S x \\ x^T S g & b(1 - x^T S x) - x^T \dot{S} x - 2x^T S f \\ & - \mathrm{Tr}(g^T S g) + \lambda(x)(x^T S x - \rho) \end{bmatrix} \succeq 0$

        $(b_R, \lambda_R) \leftarrow (b, \lambda)$

                    $\underset{\lambda, b}{\text{maximize}}$   $- b$
                    subject to   $\lambda(x) \geq 0$
                    $\begin{bmatrix} \frac{1}{2}I & g^T S x \\ x^T S g & b(1 - x^T S x) - x^T \dot{S} x - 2x^T S f \\ & - \mathrm{Tr}(g^T S g) + \lambda(x)(x^T S x - \rho) \end{bmatrix} \succeq 0$

        $(b_L, \lambda_L) \leftarrow (b, \lambda)$
        $(b, \lambda) \leftarrow \left(\frac{b_L + b_R}{2}, \frac{\lambda_L + \lambda_R}{2}\right)$

                    $\underset{S, \rho}{\text{maximize}}$   $\rho$
                    subject to   $S \succeq 0$
                    $\begin{bmatrix} \frac{1}{2}I & g^T S x \\ x^T S g & b(1 - x^T S x) - x^T \dot{S} x - 2x^T S f \\ & - \mathrm{Tr}(g^T S g) + \lambda(x)(x^T S x - \rho) \end{bmatrix} \succeq 0$

    until $\frac{\rho - \rho_0}{\rho_0} <$ TOL
return $(S, \rho, \lambda, b)$
end procedure
```