Sensitive State-Space Exploration

Thao Dang, Alexandre Donzé, Oded Maler and Noa Shalev

***Abstract*— In this paper, we describe a simulation-based approach to the verification of high dimensional nonlinear systems subject to disturbances and uncertainty in the initial conditions. Standard simulation can only sample finitely many initial states and disturbance signals and cannot verify correctness in an exhaustive manner. The alternative approach of computing all the reachable states of the system using set-based simulation, can provide, in principle, correctness proofs but is computationally expensive especially for high dimensional and nonlinear systems. In this paper we propose an approach that provides a good compromise between set-based computation and simulation by combining guided random exploration of the state space together with sensitivity analysis. The exploration technique is used to choose input signals that guarantee good coverage of the reachable set, while sensitivity information is used to create neighborhoods around explored behaviors that cover the trajectories generated by neighboring input signals.**

I. Introduction

Simulation-based verification of continuous and hybrid systems [11], [8] attempts to demonstrate, as rigorously as possible, the correctness of a system based on a finite number of simulations that sample only finitely-many initial conditions and input signals. The potential advantages of this approach, compared to "traditional" reachability-based techniques [1] are:

- Simulations are closer to the world view of practitioners and can be better understood than the "tubes" of trajectories computed in reachability-based methods;
- Simulation-based methods can be applied to complex nonlinear systems for which no efficient reachability computation technique has been developed to date.
- Simulation can be applied to systems given as a simulator black-box without a decent mathematical description, as is often the case with transistor-level circuit models;
- Heuristics can be used to guide the process to discover property violations quickly;
- The computational burden associated with storing sets of states can sometimes be alleviated.

The present paper combines two recent research threads in simulation-based verification and reachability analysis:
Guided random exploration: In [7], [2], [12], [3], [16] guided random exploration methods originating from robotics path planning [13], are used to generate input signals that induce trajectories that have good coverage of the reachable state space. The g-RRT procedure [14], which is based on the *star discrepancy* coverage measure, generates trees of trajectories such that in the limit, for every $\epsilon > 0$, every reachable state of the system has, with probability 1, an ϵ-close state in an exploration tree. Of course, since the algorithm is run for a finite number of steps, the only information it can provide upon termination is whether one of the simulated trajectories violates the property in question.
Sensitivity analysis: In [6] we have shown that sensitivity information, already available in numerical simulators, can be used to approximate the reachable set based on simulated trajectories. The technique developed in [6] can verify safety properties of a system subject to uncertainty in *initial conditions* using a finite samplings of the initial set. A sampling of the initial set $\mathcal{X}_0$ defines a ball cover with each ball centered around a sample point and the radius is determined by the sample density. Sensitivity information is then used to compute for each ball $\mathcal{N}_x$ a ball $\mathcal{N}_x(t)$ such that all trajectories originating from $\mathcal{N}_x$ end up at time t in $\mathcal{N}_x(t)$. Since $\mathcal{N}_x(t)$ is an *over approximation* of the states reachable from $\mathcal{N}_x$, its intersection with a set of forbidden states does not imply property violation, and the procedure of [6] refines the sampling of $\mathcal{X}_0$ until a safety property is proved or a counter-example is found. This technique cannot, in its present form, handle systems subject to *input uncertainty* where, in addition to sampling the initial set, one has to sample the input space in every step of the simulation.

In the present paper we combine these two ideas. We use g-RRT to generate trajectories that achieve good coverage of the reachable set. If any of these trajectories reaches the bad set we are done and the safety property is violated. Otherwise, when a trajectory gets dangerously close to the bad set we use sensitivity analysis to check whether trajectories in its neighborhood (trajectories generated by neighboring input signals) may intersect the bad set. To this end we extend the technique of [6] to deal with sensitivity to variation in the input and expand the suspicious trajectory with sensitivity balls that over-approximate its neighboring trajectories. If an intersection with the bad set is detected we conclude that the system is not robustly safe, otherwise we continue the exploration in other branches of the tree. We have implemented this technique and used it to verify some complex examples.

The rest of the paper is organized as follows. Section II summarizes the approximate computation reachable states using sensitivity information with respect to initial conditions and input. Section III summarizes the g-RRT procedure while in Section IV we augment the latter by the former, leading to an algorithm that gives a more refined verification answer. The application of the algorithm to some complex examples is described in Section V, followed by concluding remarks.

T. Dang, O. Maler and N. Shalev are with Verimag, 2 Avenue de Vignates, 38610 Gières, France, e-mail: thao.dang@imag.fr, A. Donzé is with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA, e-mail: adonze@cs.cmu.edu

II. Reach Set Estimation using Sensitivity

A. Preliminary Definitions

We are interested in dynamical systems of the form

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

where x ranges over a state space $\mathcal{X} \subseteq \mathbb{R}^n$ and $u(\cdot)$ is an admissible input function of the form $u : \mathbb{R}^+ \to U \subset \mathbb{R}^m$. We assume that f are Lipschitz continuous and that all $u(\cdot)$ are piecewise continuous. We will refer to such systems as $(\mathcal{X}, U, f, \mathcal{X}_0)$ where $\mathcal{X}_0 \subset \mathcal{X}$ is a set of initial states.

Definition 1 (Trajectories): Given a positive real number $h > 0$ and an admissible input function $u(\cdot)$, $x \xrightarrow{u(\cdot),h} x'$ is a trajectory from x to x', iff $x' = \xi_{x,u(\cdot)}(h)$ where $\xi_{x,u(\cdot)}(t)$ is the solution of the differential equation with initial condition x and under the input $u(\cdot)$.

We will be interested by the following question: given a systems $(\mathcal{X}, U, f, \mathcal{X}_0)$ and a set $B \subset \mathcal{X}$ of bad states, is there an initial state $x_0 \in \mathcal{X}_0$, a time h, an input $u(\cdot)$ and a state $x' \in B$ such that $x' = \xi_{x)0,u(\cdot)}(h)$. We use $\oplus$ to denote the Minkowski sum of two sets, that is, $X \oplus Y = \{x + y : x \in X, y \in Y\}$. For a set X and a linear transformation A, we let $AX = \{Ax : x \in X\}$.

B. Autonomous Evolution

In the following, we present a method to approximate reachable sets using sensitivity analysis techniques that was initially proposed in [6]. We first assume that the system is deterministic admitting only one $u(\cdot)$ which is continuous. Consequently, we drop $u(\cdot)$ in the notation ξ_x which denotes the unique trajectory starting from $\xi_x(0) = x$. Let $\mathcal{N}_x$ be a neighborhood of a state x from which we want to estimate the set reachable at time h. We have

$$\mathcal{R}_h(\mathcal{N}_x) = \bigcup_{x' \in \mathcal{N}_x} \xi_{x'}(h).$$

We first explain how this set can be approximated using sensitivity analysis. The concept of *sensitivity to initial conditions* is a classical topic in the theory of dynamical systems. It is concerned with the question of the influence on a trajectory of a perturbation of its initial state. To get a first order approximation of this influence, we use the Taylor expansion of $\xi_x(t + h)$ seen as a function of x. For $\epsilon \in \mathbb{R}^n$ we have:

$$\xi_{x+\epsilon}(h) = \xi_x(h) + \frac{\partial \xi_x}{\partial x}(h)\epsilon + \mathcal{O}\|\epsilon\|^2 \tag{1}$$

The second term in the right hand side of (1) is then the derivative of the trajectory with respect to the initial condition x. Since x is a vector, this derivative is a matrix, known as the *sensitivity matrix*. We note it as

$$S_x(h) = \frac{\partial \xi_x}{\partial x}(h).$$

Given a state x' in $\mathcal{N}_x$, we can use this matrix to get an estimation of $\xi_{x'}(h)$ by dropping higher order terms in (1):

$$\hat{\xi}_{x'}(h) = \xi_x(h) + S_x(h)(x' - x) \tag{2}$$

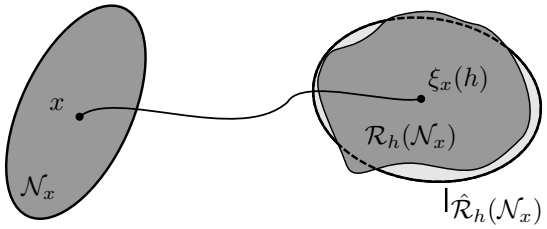


Fig. 1. Approximation of the reachable set using one trajectory and the corresponding sensitivity matrix. In this picture, the initial set is an ellipsoid and thus, by the affine transformation, so is the approximation of $\mathcal{R}_h(\mathcal{N}_x)$

If we extend this estimation to all the states $x' \in \mathcal{N}_x$, we get the following estimation of the reachable set $\mathcal{R}_h(\mathcal{N}_x)$:

$$\hat{R}_h(\mathcal{N}_x) = \bigcup_{x' \in \mathcal{N}_x} \hat{\xi}_{x'}(h) \tag{3}$$

$$= \{\xi_x - S_x(h)x\} \oplus S_x(h)\mathcal{N}_x \tag{4}$$

Note that this estimation is an affine transformation of the initial neighborhood $\mathcal{N}_x$ (see Figure 1) which is exact in the special case of affine dynamics.

To improve the precision of the approximation scheme (3) it has been suggested in [5] to *refine* the sampling of the initial set, that is, to partition it into smaller neighborhoods, to issue one trajectory from each of those, along with their sensitivity matrices, and to approximate $\mathcal{R}_h(\mathcal{N}_x)$ with the union of the sets obtained by the above transformation. Due to (1), the resulting approximation error will then decrease quadratically w.r.t. the size ϵ of the neighborhoods in the initial sampling.

C. Computing Sensitivity Matrices

Computing the sensitivity matrix requires to compute the derivative of ξ_x with respect to x. It is very unlikely in general that we have an explicit formulation of ξ_x hence we cannot obtain its derivative directly. However its value at time $t = 0$ is trivially the identity matrix and we can relate its evolution to that of ξ_x. Indeed, ξ_x satisfies the ordinary differential equation $\dot{x} = f(x, u)$ with $\xi_x(0) = x$ meaning that $\forall t \in [0,\ h]$

$$\dot{\xi_x}(t) = f(\xi_x(t), u(t)). \tag{5}$$

Then, if we take the time derivative of this expression by applying the chain rule to the right hand side term we obtain

$$\dot{S}_x = \partial_x f(\xi_x, u)\ S_x,\ S_x(0) = \mathbf{I}_n \tag{6}$$

where $\partial_x f_q$ is the derivative of f_q w.r.t. x. Then computing the full sensitivity matrix S_x amounts to solving (5) and (6) i.e. to solve $n + 1$ ordinary differential equations of order n. Note that the cost can be made less than that of the resolution of $n + 1$ different systems since the Jacobian f_x can be extensively reused in the computation [15], [9].

The precise computation of $\mathcal{R}_h(\mathcal{N}_x)$ thus relies on the size of $\mathcal{N}_x$ and on the precise computation of the trajectory and the sensitivity matrix at time h. To do this, we use a numerical ODE solver with an error control mechanism [15], [9]. To compute the solutions of (5) and (6) at time

h, the solver typically performs several internal steps to get intermediate values at time instants in $(0, h)$. Each internal step satisfies the tolerances specified to the solver. The method also guarantees that an interpolated solution between two internal steps satisfies these tolerances as well. Thus, by using such an interpolated solution, one can get an estimation of the trajectory and the sensitivities, and thus of the reachable set, at any time instant between 0 and h. The precision of this estimation is controlled by the tolerances specified to the ODE solver.

D. Extension to Non Deterministic Inputs

We now consider the case where the input function $u(\cdot)$ is constant on the interval, $u(t) = u \in U$ for all $t \in [0, h]$. Given an initial state x and an input u, we get a unique trajectory $\xi_{x,u}$ on $[0, h]$. Let $S_u(h)$ be the derivative of $\xi_{x,u}(h)$ w.r.t. u, that is, the sensitivity matrix related to the input parameter u. Then taking a neighboring state x' and a neighboring input u' and applying the Taylor expansion as in (1), we get

$$\xi_{x',u'}(h) = \xi_{x,u}(h) + S_x(h)\ (x' - x) + S_u(h)\ (u' - u) \quad (7)$$
$$+\mathcal{O}\|x' - x\|^2 + \mathcal{O}\|u' - u\|^2 \quad (8)$$

By ignoring higher order terms, we can then build an estimation of $\xi_{x',u'}$ for any x' and u' using $\xi_{x,u}$ and matrices S_x and S_u:

$$\hat{\xi}_{x',u'}(h) = \xi_{x,u}(h) + S_x(h)\ (x' - x) + S_u(h)\ (u' - u) \quad (9)$$

and correspondingly an estimation of the set reachable from initial states in $\mathcal{N}_x$ using inputs in $\mathcal{N}_u$:

$$\hat{\mathcal{R}}_h(\mathcal{N}_x, \mathcal{N}_u) = \bigcup_{(x',u') \in \mathcal{N}_x \times \mathcal{N}_u} \hat{\xi}_{x',u'}(h) \quad (10)$$
$$= \{\xi_x(h)\} \oplus S_x(h)(\mathcal{N}_x - x) \oplus S_u(h)(\mathcal{N}_u - u) \quad (11)$$

Note that S_u can be computed by solving:

$$\dot{S}_u = \partial_x f(\xi_{x,u}, u)\ S_u + \partial_u f(\xi_{x,u}, u) \quad (12)$$
$$S_u(0) = \mathbf{0} \quad (13)$$

where $\partial_u f$ is the partial derivative of f w.r.t. u.

This means that we can compute the set reachable from $\mathcal{N}_x$ using constant inputs in $\mathcal{N}_u$ on the interval $[0, h]$. However, the process of refinement (at least its straightforward realization) is now more costly: if we refine $\mathcal{N}_u$ into two smaller neighborhoods $\mathcal{N}_{u^1}$ and $\mathcal{N}_{u^2}$ then the number of piecewise-constant input signals with which we need to simulate in order to approximate the states reachable at time Nh is 2^N. In the following sections we propose an intermediate approach using an exploration algorithm that computes trajectories with a good coverage property, and estimates the actual reachable sets only around those trajectories that get too close to the bad set B.

III. RRT-BASED EXPLORATION

RRT (Rapidely-exploring Random Trees) [13] is a family of of space exploration techniques, motivated by robotic path planning. In the context of dynamical systems with input, it can be used to generate input signals that induce trajectories with a good coverage of the reachable part of the state space. In what follows we describe g-RRT , a "guided" variant of this technique, first presented in [14].

A. The g-RRT Algorithm

To simplify presentation assume that $\mathcal{X}_0$ is a singleton set $\{x_0\}$ so that we have one tree instead of several. We assume a fixed time-discretization step h during which the input remains constant. By abuse of notation we write $x' = f(x, u)$ if $x' = \xi_{x,u}(h)$, that is, x' is reached from x when the constant input u is applied for duration h.

Definition 2 (Exploration Trees): An exploration tree for a dynamical system $(\mathcal{X}, U, f, \{x_0\})$ is a labeled tree $\mathcal{T} = (S, E, s_0)$ where $S \subseteq \mathcal{X}$ is a set of nodes, $E \subseteq S \times U \times S$ is a set of labeled edges, and $s_0 = x_0$ is the root of the tree. An edge (x, u, x') appears in the tree only if $x' = f(x, u)$.

The g-RRT algorithm (Algorithm 1) works as follows. Initially, the tree has only a root. At every step, a goal state $x_g \in \mathcal{X} - S$ is chosen semi-randomly (we explain later how this choice is made). Then, for the nearest neighbor x_* of x_g in S we find $u_* \in U$ which minimizes the distance between x_g and $f(x_*, u)$. The node $x' = f(x_*, u_*)$ and the edge (x_*, u_*, x') are inserted into the tree. The process is repeated J times, where J is determined according to the system characteristics and the computational budget. We use Euclidean distance while selecting x_* and u_*. When $\mathcal{X}_0$ is not a singleton, we sample it and grow several trees rooted by those initial points.

Algorithm 1 The g-RRT Exploration Algorithm

$\mathcal{T} := (x_0, \emptyset, x_0)$ ▷ initial tree
repeat $j = 1, 2 \ldots$
  $x' :=$ ADDNEW ▷ explore a new node
  if $x' \in B$ **then** ▷ new state is bad
    return unsafe
  end if
until $j = J$
return no violation found

Procedure ADDNEW

$x_g :=$ GUIDEDSAMPLING$(S, \mathcal{X})$
$x_* := \arg\min_{x \in S} |x - x_g|$
$u_* := \arg\min_{u \in U} |f(x_*, u) - x_g|$
$x' := f(x_*, u_*)$
$S := S \cup \{x'\}$
$E := E \cup \{(x_*, u_*, x')\}$
return (x')

B. Coverage-guided Sampling

The only random and heuristic component of Algorithm 1 is the GUIDEDSAMPLING procedure for choosing the goal state x_g toward which the rest of the ADDNEW procedure tries to steer the system. In classical RRT algorithms, x_g is typically sampled *uniformly* over the state space. In contrast, our procedure attempts to improve the sample coverage. To this end we use the *star discrepancy* measure [4], that we explain intuitively below.

For simplicity assume that $\mathcal{X}$ is a box (hyper rectangle) and let $P \subset \mathcal{X}$ be a finite sample whose coverage quality we want to evaluate. The local discrepancy of P with respect to sub-box $\mathcal{Y}$ of $\mathcal{X}$ is

$$D(P, \mathcal{Y}) = |\sharp(P \cap \mathcal{Y})/\sharp(P) - \lambda(\mathcal{Y})/\lambda(\mathcal{X})|$$

where $\sharp$ denotes the number of elements and λ denotes volume. Intuitively, $D(P, \mathcal{Y})$ will be close to 0 when the share of $\mathcal{Y}$ in the sample is more or less proportional to its volume. On the other hand $D(P, \mathcal{Y})$ will approach 1 when either $\mathcal{Y}$ is a low-volume set with a lot of sample points or a high-volume set sampled very sparsely. The star discrepancy of P with respect to $\mathcal{X}$ is defined as $D^*(P, \mathcal{X}) = sup_{\mathcal{Y}} D(P, \mathcal{Y})$. Intuitively, the star discrepancy is a measure for the irregularity of a set of points with large value of $D^*(P, \mathcal{X})$ indicating that points in P are not well equidistributed over $\mathcal{X}$.

To evaluate the coverage of a set of states, we estimate a lower and upper bound of the star discrepancy (exact computation is hard). These bounds, as well as the information obtained from their estimation, are used to decide which parts of the state space have been sufficiently explored and which parts need to be explored further. Our implementation keeps $\mathcal{X}$ partitioned into a finite set Π of elementary boxes, where the partition is refined during execution. At each step we first select a box $\mathcal{Y} \in \Pi$ with a distribution biased toward coverage improvement, that is, we favor an elementary box such that adding a new point to it will reduce the lower and upper bounds on the star discrepancy. Once the box has been selected, a random point is drawn from it uniformly. More details concerning the g-RRT procedure can be found in [14].

IV. COMBINING BOTH APPROACHES

We assume from now on that U is discretized into a finite number of points $\hat{U} = \{u^1, \ldots, u^k\}$ around which neighborhoods $\{\mathcal{N}_{u^1}, \ldots, \mathcal{N}_{u^k}\}$ are defined whose union covers U. The exploration tree is thus labeled by elements of $\hat{U}$.

A. The Algorithm

In Algorithm 2, the function ADDNEW inserts a new node x' to the exploration tree as in Algorithm 1 and behaves likewise if $x' \in B$. If x' is only close to the bad set, the path τ from the root to x' is retrieved. Then, as described below, the function REACHSENS computes an approximation $\mathcal{R}$ of the reachable set along that path using sensitivity analysis. If $\mathcal{R}$ intersects B the algorithm concludes that the system is not robustly safe.

Algorithm 2 sg-RRT : Combining g-RRT with Sensitivity

```
T := (x_0, ∅, x_0)                                  ▷ initial tree
repeat j = 1, 2 ...
    x' := ADDNEW
    if x' ∈ B then                                  ▷ new state is bad
        return unsafe
    end if
    if |x' − B| ≤ ε then                            ▷ new state close to bad
        τ := TRACETO(x')                            ▷ retrieve the path to x'
        R := REACHSENS(τ)                           ▷ compute neighborhood
        if R ∩ B ≠ ∅ then
            return not robustly safe
        end if
    end if
until j = J
return no violation found
```

Assume that TRACETO returns a trace of the form

$$\tau: \quad x_0 \xrightarrow{u_1} x_1 \xrightarrow{u_2} \cdots \xrightarrow{u_N} x_N$$

we would like to approximate all the states reachable at time $T = Nh$ by similar traces, that is, traces of the form

$$\tau': \quad x'_0 \xrightarrow{u'_1} x'_1 \xrightarrow{u'_2} \cdots \xrightarrow{u'_N} x'_N$$

where $x'_0 \in \mathcal{N}_{x_0}$ and $u'_k \in \mathcal{N}_{u_k}$ for every $k = 1..N$. To this end, we first compute the sensitivity matrix $S_{x_0}(T)$ w.r.t. x_0 by solving (6) on $[0, T]$. Then we compute the matrices $S_{u_k}(h)$ for each step from x_k to x_{k+1} as described in Section II-D. Finally, we remark that due to the time-invariance of the dynamics, the sensitivity matrix $S_{u_k}(T)$ w.r.t. the input u_k at time T is the product

$$S_{u_k}(T) = S_{u_k}(h) S_{u_{k+1}}(h) \ldots S_{u_N}(h). \tag{14}$$

From there, we can deduce the estimation for the reachable set at time T that REACHSENS will return:

$$\begin{aligned} \mathcal{R} = \{x_N\} &\oplus S_{x_0}(T)(\mathcal{N}_{x_0} - x_0) \\ &\oplus \bigoplus_{k=1}^{N} S_{u_k}(h)(\mathcal{N}_{u_k} - u_k) \end{aligned} \tag{15}$$

Note that the precision of this approximation depends on the size of the sets $\mathcal{N}_{x_0}$ and $\mathcal{N}_{u_k}$ for all k. The size of the latter depends on the density of the discretization $\hat{U}$ of U that we use. In fact, there is a tradeoff here between accuracy and complexity: the larger is $\hat{U}$ the smaller are the neighborhoods surrounding the trajectories but more of them are needed to cover the reachable set.

B. Bounding boxes

To compute the intersection of $\mathcal{R}$ with B we need to represent it in a form for which intersection is easy to compute. In the following we assume that the sets $\mathcal{N}_{x_0}$ and $\{\mathcal{N}_{u_k}\}_{k=1}^{N}$ are boxes of the form

$$\mathcal{N}_{x_0} = \prod_{i=1}^{n} [\underline{x}_0^i, \overline{x}_0^i] \quad \text{and} \quad \mathcal{N}_{u_k} = \prod_{i=1}^{n} [\underline{u}_k^i, \overline{u}_k^i],$$

and the B is a box as well. We would want to find the smallest box

$$\mathcal{N}_{x_N} = \prod_{i=1}^{n} [\underline{x}^i, \overline{x}^i]$$

containing $\mathcal{R}$. Then for every dimension i, $\overline{x}^i$ is the solution of the following linear optimization problem:

$$\max_{\tilde{x}_0 \in \mathcal{N}_{x_0},\ \tilde{u}_k \in \mathcal{N}_{u_k}} x^i \quad \text{s.t.}$$

$$x = x_N + S_{x_0}(\tilde{x}_0 - x_0) + \sum_{k=0}^{N} S_{u_k}(\tilde{u}_k - u_k)$$

and $\underline{x}^i$ is a solution of the corresponding minimization problem. Thus to find $\mathcal{N}_{x_N}$ we have to solve $2n$ linear programs with $n + Nm$ variables each. Since N is typically large we use an incremental procedure that computes an approximation of $\mathcal{N}_{x_{k+1}}$ from $\mathcal{N}_{x_k}$ and $\mathcal{N}_{u_k}$.

V. Examples

We have applied a preliminary implementation of Algorithm 2 to several examples.

A. Helicopter

We consider a simplified model of a quadri-rotor helicopter [10] (see Figure 2). Only the altitude z and the axis x are considered. The equations of motions are given by:

$$\begin{aligned}
\ddot{x} &= -\frac{b}{m}\dot{x} + \frac{1}{m}(u_1 + u_2 + u_3 + u_4)sin(\theta) \\
\ddot{z} &= -\frac{b}{m}\dot{z} + \frac{1}{m}(u_1 + u_2 + u3 + u_4)cos(\theta) - g \\
\ddot{\theta} &= \frac{L}{I_y}(u_1 - u_3) - \frac{c}{L}\dot{\theta}
\end{aligned}$$

where $m = 0.5184$, $c = 0.15$, $L = 0.236$, $I_y = 0.04774$. The state variable is then $\mathbf{x} = (\dot{x}, x, \dot{z}, z, \dot{\theta}, \theta)$. Given a goal state $\mathbf{x}^*$, a linear quadratic regulator can be used to drive the system to $\mathbf{x}^*$ from any state. It is given by:

$$\mathbf{u} = K(\mathbf{x} - \mathbf{x}^*) + \frac{mg}{4}\mathbf{1}$$

where $\mathbf{1}$ is the vector $(1, 1, 1, 1)$ and

$$K = \begin{bmatrix} -.8521 & .7071 & -.5457 & -.5 & -1.122 & -4.461 \\ 0 & 0 & -.5457 & -.5 & 0 & 0 \\ -.8521 & .7071 & -.5457 & -.5 & 1.1217 & 4.461 \\ 0 & 0 & -.5457 & -.5 & 0 & 0 \end{bmatrix}$$

We assume that due to imperfections in the sensors, the angle velocity $\dot{\theta}$ is estimated with an additional error w by the regulator, meaning that the value of $\mathbf{u}$ it computes uses $\tilde{\dot{\theta}} = \dot{\theta} + w$ instead of $\dot{\theta}$. Our goal is to verify that despite this error, the system still reaches its goals state while the angle θ always remains less than O.15 rad. We present preliminary results on Figure 3.

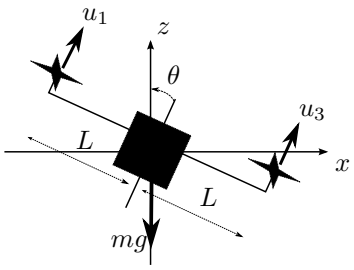
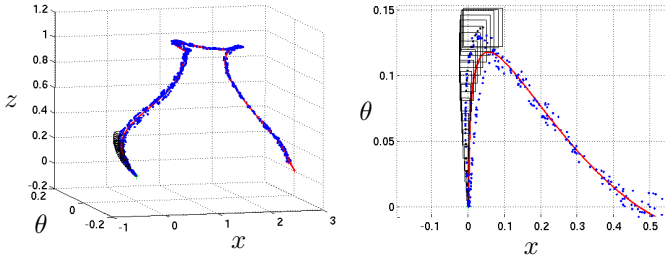


Fig. 2. Simplified model of a quadrirotor helicopter



(a) (b)

Fig. 3. Preliminary results for the quadrirotor helicopter. The vehicule is assigned three way-points: $(x = 1, z = 1, \theta = 0)$, $(x = 2, z = 1, \theta = 0)$ and $(x = 3, z = 0, \theta = 0)$. The solid line represents the ideal trajectory without uncertainties. (a) Simulation states around the ideal trajectory generated by the exploration algorithm. (b) Reachable set estimation around one trajectory. This estimation is found to intersect with $\theta = 0.15$ thus the system is declared not robustly safe.

B. A Robotic Vehicle Benchmark

This example is adapted from the robotic navigation system benchmark of [16]. We consider a car with the following continuous dynamics with 5 variables: $\dot{x} = vcos(\theta)$, $\dot{y} = v\sin(\theta)$, $\dot{\theta} = v\tan(\phi)/L$, $\dot{v} = u_0$, $\dot{\phi} = u_1$ where x, y, θ describe the position and heading of the car, v is its speed and ϕ is its steering angle. The inputs of the system are u_0 and u_1 which are, respectively, the acceleration and steering control inputs. The system uses a hybrid control law with 3 modes. In the first mode, called *RandomDriver*, the control inputs are selected uniformly at random between their lower and upper bounds. In the second mode, called *StudentDrive*, when the speed is low, u_0 is randomly chosen as in first mode; otherwise, the strategy is to reduce the speed. In the third mode, called *HighwayDrive*, the strategy is to reduce the speed when it is high and increase it when it is low. A detailed description of this control law can be found in [16]. In our model, the system selects a control mode randomly and uses it for some fixed amount of time. After that, the system selects again a control mode randomly.[1]

A safety property is specified by a bad set

$$x \in [20.0, 21.0] \ \wedge \ y \in [20.0, 21.0] \ \wedge \ \theta \in [1.2, 1.25],$$

a small rectangle in the state space. We compared the performance of the sg-RRT algorithm (with sensitivity) with

[1]In [16], in addition to the three modes of the controller, the car itself can be in one of three modes. In this work, we consider only one of those.

that of the standard g-RRT algorithm on this example. For the uncertainty $\delta u_1 \leq 0.005$, the sg-RRT algorithm discovers that the system is not robustly safe after 7643 iterations (with 0.56 minutes of computation time), while g-RRT needs 20751 iterations to detect the violation of the property (with about 1 minute of computation time).

Note that since this example is hybrid we use the hybrid version of g-RRT described in [14] as well as a hybrid adaptation of the sensitivity computation which works in this case because the system does not perform jumps while switching between modes.

C. Linear systems

To evaluate the performance of the algorithm, we performed similar experiments on a set of randomly-chosen linear systems in various dimensions. These linear systems are given in the Jordan form with the values on the diagonal chosen from the interval $[-3, 3]$. The results are summarized in Table I showing the number of iterations needed by sg-RRT and g-RRT in order to discover the violation of a safety property. The corresponding computation times are reported in the third column of the table. The fourth column contains the computation time for 10000 iterations of the sg-RRT algorithm. Note that in the versions of the two algorithms used in this experimentation, we did not use a precise coverage estimation in the guiding strategy, which allows shorter computation time (compared to the experimental results reported in [14]) at the price of poorer coverage quality. From these results we can see that sg-RRT detected the violation faster than the g-RRT algorithm. In addition, the sg-RRT algorithm is still scalable to high dimensional systems.

dim n	Nb of Iter until detection		Time		Time for 10000 iter.
	sg-RRT	g-RRT	sg-RRT	g-RRT	
10	7653	11090	7s	12s	23s
20	7720	10025	11s	15s	29s
100	9920	11152	36s	44s	37s
200	3035	11243	21s	1m 16s	1m 27s
200	125	>50000	2s	6m 45s	2m 52s

TABLE I

EXPERIMENTAL RESULTS FOR SOME LINEAR SYSTEMS.

VI. CONCLUSION

We have made some progress in the difficult problem of analyzing complex nonlinear systems subject to disturbances. Our solution was based on a combination of two ideas, the g-RRT algorithm which solves the coverage problem efficienctly at the exepense of formal completeness (in detecting bad behaviors), and sensitivity-based reachability computation which brings back partial completeness around dangerous trajectories. The heuristic nature of g-RRT (and other RRT algorithms) does not leave much room for useful theorems but, neverthelss, completeness can be proved *relative* to the coverage provided by g-RRT. For example, it can be shown that if every reachable state x has an ϵ-close $x' \in S$ then the sg-RRT algorithm is conservative in the sense of not missing an erronous behavior.

We mention some ideas for further research: once a sensitivity ball $\mathcal{N}_{x_N}$ intersects B, we can launch a refinement process in order to either find a concrete disturbance that actually leads to B, or replace $\mathcal{N}_{x_N}$ with a collection of smaller neighborhoods which are safe. As mentioned above, care should be taken so that the refinement process will not lead to exponentially-many trajectories. Alternative guiding heuristics and as well as other represntation and computation scheme for neighborhoods are subject to ongoing research and experimentation.

REFERENCES

[1] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler, Recent Progress in Continuous and Hybrid Reachability Analysis, *CACSD 2006*, 1582-1587, IEEE, 2006.

[2] A. Bhatia and E. Frazzoli, Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems, *HSCC'04*, 142-156, 2004.

[3] M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan, Sampling-based Reachability Algorithms for Control and Verification of Complex Systems. *13th Yale Workshop on Adaptive and Learning Systems*, 2005.

[4] J. Beck and W.W.L. Chen, Irregularities of Distributios Relative to Convex Polygons III, *J. London Mathematical Society* **56**, 222-230, 1997.

[5] A. Donzé. *Trajectory-Based Verification and Controller Synthesys for Continuous and Hybrid Systems*. PhD thesis, University Joseph Fourier, June 2007.

[6] A. Donzé and O. Maler. Systematic simulations using sensitivity analysis. In *HSCC'07*, LNCS, April 2007.

[7] J.M. Esposito, J. W. Kim, and V. Kumar, Adaptive RRTs for Validating Hybrid Robotic Control Systems, *Workshop on Algorithmic Foundations of Robotics*, 2004.

[8] A. Girard and G.J. Pappas: Verification Using Simulation, *HSCC'06*, LNCS 3927, 272-286, 2006.

[9] A.C. Hindmarsh and R. Serban, *User Documentation for CVODES v2.4.0*, March 2006.

[10] G. Hoffmann, H. Huang, S. Waslander, and C.J. Tomlin, Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment, *AIAA Conf. on Guidance, Navigation and Control*, 2007.

[11] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg, On Systematic Simulation of Open Continuous Systems, *HSCC'03*, 283-297, 2003.

[12] J.W. Kim, J.M. Esposito, and V. Kumar, Sampling-based Algorithm for Testing and Validating Robot Controllers, *Int. Journal of Robtics Research* **25**, 1257-1272, 2006.

[13] S. LaValle and J. Kuffner. Rapidly-Exploring Random Trees: Progress and Prospects *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[14] T Nahhal and T. Dang, Test Coverage for Continuous and Hybrid Systems, *CAV'07*, LNCS 4590, 449-462, 2007.

[15] R. Serban and A.C. Hindmarsh, Cvodes: the Sensitivity-enabled ODE Solver in Sundials, *IDETC/CIE'05*, 2005.

[16] E. Plaku, L.E. Kavraki, and M.Y. Vardi, Hybrid Systems: from Verification to Falsification, *CAV'07*, LNCS 4590, 468-481, 2007.