Parameter Synthesis for Hybrid Systems with an Application to Simulink Models

Alexandre Donzé[1], Bruce Krogh[2], and Akshay Rajhans[2]

[1] Dept. of Computer Science
[2] Dept. of Electrical and Computer Engineering,
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
adonze@cs.cmu.edu, {krogh,arahjans}@ece.cmu.edu

Abstract. This paper addresses a parameter synthesis problem for nonlinear hybrid systems. Considering a set of uncertain parameters and a safety property, we give an algorithm that returns a partition of the set of parameters into subsets classified as safe, unsafe, or uncertain, depending on whether respectively all, none, or some of their behaviors satisfy the safety property. We make use of sensitivity analysis to compute approximations of reachable sets and an error control mechanism to determine the size of the partition elements in order to obtain the desired precision. We apply the technique to Simulink models by combining generated code with a numerical solver that can compute sensitivities to parameter variations. We present experimental results on a non-trivial Simulink model of a quadrotor helicopter.

1 Introduction

A standard problem in model-based analysis and design is to find the ranges of parameters (including initial states) for which the system behavior will be acceptable [HWT96, FJK08]. We call this the *parameter synthesis problem.* One approach to this problem is to run simulations of the system for a set of parameter values that covers the range of values of interest. This approach is attractive because of its generality: one can simulate almost any system. It can take a very large number of simulations to cover the parameter space at a sufficient level of granularity, however.

Reachability analysis offers an alternative to simulation [ADF+06]. By computing reachable sets rather than simulating single trajectories, it may be possible to explore the design space more efficiently. Although this is the case for low-dimensional systems, the ability to perform reachability computations for higher-dimensional systems remains an elusive goal, even for so-called linear hybrid automata [HHWT97, Fre05]. The parameter synthesis problem for nonlinear and higher-dimensional systems remains intractable using reachability tools.

This paper proposes an approach to the parameter synthesis problem that offers the strength of reachability analysis while using only numerical simulations. This approach is in the spirit of other work on methods for obtaining

R. Majumdar and P. Tabuada (Eds.): HSCC 2009, LNCS 5469, pp. 165–179, 2009.

reachable set information from single simulation runs [GP06, DM07, LKCK08]. Most of this work has focused on using simulations to propagate representations of reachable sets that are guaranteed to be conservative [GP06, LKCK08]. We use a different approach that leverages the simplicity of sensitivity analysis to generate approximations to reachable sets very efficiently [DM07]. Speed is achieved with a slight sacrifice in accuracy–the reachable set approximations are not guaranteed to be conservative. We are able to estimate the error in the approximations, however, providing a mechanism for gaining some assurance that the final estimation of the set of good parameters is reasonable.

The paper is organized as follows. The following section introduces notation and the basic algorithm for simulating hybrid dynamic systems. Section 3 recalls the method for generating sensitivity matrices with only a slight increase in computation during the simulation run. Section 4 presents the formulation and solution of the parameter synthesis problem using sensitivity-based reachability. We describe an implementation of the approach in Section 5 that solves the parameter synthesis problem for hybrid systems modeled in MATLAB Simulink and illustrate its application to the design of a supervisory safety control algorithm for a quadrotor helicopter. The concluding section summarizes the contributions of the paper and identifies directions for future research.

2 Hybrid Model and Simulation

The set $\mathbb{R}^n$ is equipped with the infinity norm, noted $\|\mathbf{x}\| = \max_i |x_i|$. It is extended to $n \times n$ matrices as usual. We define the diameter of a compact set $\mathcal{P}$ to be $\|\mathcal{P}\| = \sup_{(\mathbf{p},\mathbf{p}')\in\mathcal{P}^2} \|\mathbf{p}-\mathbf{p}'\|$. The distance from $\mathbf{x}$ to a set $\mathcal{R}$ is $d(\mathbf{x},\mathcal{R}) = \inf_{\mathbf{y}\in\mathcal{R}} \|\mathbf{x}-\mathbf{y}\|$. The Hausdorff distance between two sets $\mathcal{R}_1$ and $\mathcal{R}_2$ is

$$d_H(\mathcal{R}_1,\mathcal{R}_2) = \max\left(\sup_{\mathbf{x}_1\in\mathcal{R}_1} d(\mathbf{x}_1,\mathcal{R}_2), \sup_{\mathbf{x}_2\in\mathcal{R}_2} d(\mathbf{x}_2,\mathcal{R}_1)\right).$$

Given a matrix S and a set $\mathcal{P}$, $S\mathcal{P}$ represents the set $\{S\mathbf{p}, \mathbf{p}\in\mathcal{P}\}$. Given two sets $\mathcal{R}_1$ and $\mathcal{R}_2$, $\mathcal{R}_1 \oplus \mathcal{R}_2$ is the Minkowski sum of $\mathcal{R}_1$ and $\mathcal{R}_2$, i.e., $\mathcal{R}_1 \oplus \mathcal{R}_2 = \{\mathbf{x}_1+\mathbf{x}_2, \mathbf{x}_1\in\mathcal{R}_1, \mathbf{x}_2\in\mathcal{R}_2\}$.

2.1 Dynamics

We consider a dynamical system $\mathscr{S} = (\mathcal{Q}, f, e, g)$ with evolutions described by

$$\begin{cases} \dot{\mathbf{x}} = f(q,\mathbf{x},\mathbf{p}),\ \mathbf{x}(0) = \mathbf{x}_0 \\ q^+ = e(q^-,\lambda),\ q(0) = q_0 \\ \lambda = \text{sign}(g(\mathbf{x})) \end{cases} \tag{1}$$

where

- $\mathbf{x}\in\mathbb{R}^n$ is the *continuous state*, $\mathbf{p}$ is the *parameter vector* lying in a compact set $\mathcal{P}\subset\mathbb{R}^{n_p}$, $q\in\mathcal{Q}$ is the *discrete state*,

- λ is a vector in $\{-1, 0, +1\}^{n_g}$,
- g is the *guard function* mapping $\mathbb{R}^n$ to $\mathbb{R}^{n_g}$,
- sign is the usual sign function extended to vectors, i.e., if $\lambda = \text{sign}(g(\mathbf{x}))$, then $\lambda_i = 1$ if $g_i(\mathbf{x}) > 0$, $\lambda_i = -1$ if $g_i(\mathbf{x}) < 0$ and $\lambda_i = 0$ if $g_i(\mathbf{x}) = 0$.
- e is the *event function* which updates the discrete state when a component of the guard function g changes its sign. At each time t, q^+ (respectively q^-) represents the value of q immediatly after t (respectively immediatly before t). It is assumed that $q^+ = e(q^-, \lambda) = q^-$ if $\lambda_i \neq 0$ for all i. In words, q^+ may differ from q^- only when one component of the guard function g is zero.

Let $\mathcal{T} = \mathbb{R}^+$ be the *time set.* Given $\mathbf{p} \in \mathcal{P}$, a trajectory $\xi_\mathbf{p}$ is a function from $\mathcal{T}$ to $\mathbb{R}^n$ which satisfies (1), i.e., for all t in $\mathcal{T}$, we have

$$\dot{\xi}_\mathbf{p}(t) = f(q(t), \xi_\mathbf{p}(t), \mathbf{p}),\ q(t^+) = e(q(t^-), \lambda(t)) \text{ and } \lambda(t) = \text{sign}(g(\xi_\mathbf{p}(t)))$$

For convenience, the initial state $\mathbf{x}_0$ is included in the parameter vector $\mathbf{p}$. The dimension n_g of $\mathcal{P}$ is thus greater than n and we have $\xi_\mathbf{p}(0) = \mathbf{x}_0 = (x_{0_1}, x_{0_2}, \ldots, x_{0_n})$, where for all $i \leq n$, $x_{0_i} = p_i$.

In this work, we assume that a trajectory can always be computed by appending solutions of (1) on successive time intervals of the form $[t_k, t_{k+1}]$. This is possible if for all i, there is a neighborhood of $(t_k, \xi_\mathbf{p}(t_k))$ where $(t, \mathbf{x}) \mapsto f(q, \mathbf{x}, \mathbf{p})$ is continuously differentiable (C^1). In this case, we know by the Cauchy-Lipshitz theorem that there exists $h_k > 0$ such that a solution of the $\dot{\mathbf{x}} = f(q, \mathbf{x}, \mathbf{p})$ can be uniquely continued on the interval $(t_k, t_k + h_k]$. Thus t_{k+1} can be defined as $t_{k+1} = t_k + h_k$ and the process can be repeated indefinitely to form a unique trajectory on the whole time set $\mathcal{T}$ given a parameter vector $\mathbf{p}$.

2.2 Event Detection

In the model (1) above, the event function triggered by the guard function makes it possible to introduce discontinuities in the evolution. The function f can be discontinuous in a state $\mathbf{x}$ where some component of g is zero. Assume that $g_i(\xi_\mathbf{p}(\tau)) = 0$ for some i and $\tau > t_k$. It can thus be that

$$f(q(\tau^-), t^-, \xi_\mathbf{p}(\tau^-), \mathbf{p}) \neq f(q(\tau^+), t^+, \xi_\mathbf{p}(\tau^+), \mathbf{p})$$

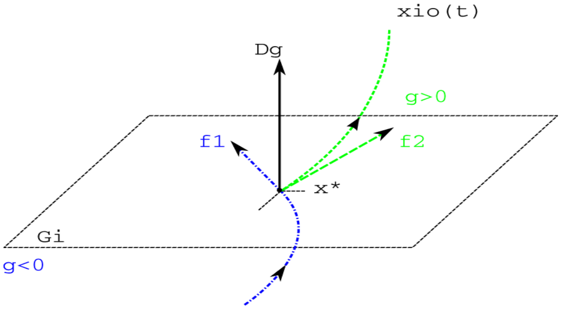
This means that at time τ, the system switches from one continuous dynamics to another continuous dynamics, which is called a *switching event.* In such a situation, the Cauchy-Lipshtz theorem does not apply in $(\tau, \xi_\mathbf{p}(\tau))$ and standard numerical schemes may have problem to provide an accurate result. A solution is to integrate the dynamics until the time event τ, set $t_{k+1} = \tau$ and then continue from t_{k+1} with the new dynamics. Thus τ needs to be detected as precisely as possible, which can be done through *discontinuity locking* and *zero-crossing* detection [EKP01]. The idea is to fix (or *lock*) the value of q to $q(t_k)$ in order to prevent the occurrence of a switching event, and to integrate the equation $\dot{\mathbf{x}} = f_k(\mathbf{x}, \mathbf{p})$, where $f_k(\mathbf{x}, \mathbf{p}) = f(q(t_k), \mathbf{x}, \mathbf{p})$, on an interval $[t_k, t_k + h_k[$. Then check whether there is a time $t \in]t_k, t_k + h_k[$ such that the sign of g changed, i.e., $\mathrm{sign}(g(\mathbf{x}(t_k))) \neq \mathrm{sign}(g(\mathbf{x}(t)))$, in which case, by continuity, it is guaranteed that g has at least one zero on the interval $]t_k, t[$. A bisection procedure can be applied to determine the first time τ when a component of g is zero.

This is summarized in the following algorithm to compute $\xi_{\mathbf{p}}(t_{k+1})$ knowing $\xi_{\mathbf{p}}(t_k)$.

```
Compute $\xi_{\mathbf{p}}$ solution of $\dot{\mathbf{x}} = f_k(\mathbf{x}, \mathbf{p})$ on $[t_k, t_k + h_k]$
if $\forall t \in [t_k, t_k + h_k]$, $\mathrm{sign}(g(\xi_{\mathbf{p}}(t)) = \mathrm{sign}(g(\xi_{\mathbf{p}}(t_k))$ then
    Return $t_{k+1} = t_k + h_k$ and $\xi_{\mathbf{p}}(t_{k+1})$
else
    Find the minimum time $\tau > t_k$ such that $g_i(\tau) = 0$ for some $i$
    Return $t_{k+1} = \tau$ and $\xi_{\mathbf{p}}(t_{k+1}) = \xi_{\mathbf{p}}(\tau)$
end if
```

For the above algorithm to be correct, we make the following assumption: at the time τ of an event,

$$\langle \nabla g_i(\mathbf{x}(\tau)),\ f(q^-, \mathbf{x}(\tau^-), \mathbf{p}) \rangle \ \langle \nabla g_i(\mathbf{x}(\tau)),\ f(q^+, \mathbf{x}(\tau^+), \mathbf{p}) \rangle > 0. \qquad (2)$$



This assumption, illustrated in the figure above, guarantees that when a transition occurs, the dynamics of the systems leans strictly toward the guard before the switch and strictly away from it after the transition. Thus we do not allow *sliding*, i.e., when a trajectory remains on the transition surface, nor *grazing*, i.e., when a trajectory hits the surface tangentially. This assumption holds for many real physical systems. Condition (2) can often be checked for all possible states in the model.

3 Sensitivity Analysis

The concept of *sensitivity to parameters* is a classical topic in the theory of dynamical systems. It is concerned with the question of the influence of a parameter change $\delta\mathbf{p}$ on a trajectory $\xi_{\mathbf{p}}$. A first order approximation of this influence can be obtained by a Taylor expansion of $\xi_{\mathbf{p}}(t)$ with respect to $\mathbf{p}$. For $\delta\mathbf{p} \in \mathbb{R}^{n_p}$, we have:

$$\xi_{\mathbf{p}+\delta\mathbf{p}}(t) = \xi_{\mathbf{p}}(t) + \frac{\partial \xi_{\mathbf{p}}}{\partial \mathbf{p}}(t)\ \delta\mathbf{p} + \varphi(t, \delta\mathbf{p}) \text{ where } \varphi(t, \delta\mathbf{p}) = \mathcal{O}\big(\|\delta\mathbf{p}\|^2\big) \qquad (3)$$

The second term in the right hand side of (3) is the derivative of the trajectory with respect to the parameter $\mathbf{p}$. Since $\mathbf{p}$ is a vector, this derivative is a matrix called the *sensitivity matrix*, denoted as $S_{\mathbf{p}}(t) = \frac{\partial \xi_{\mathbf{p}}}{\partial \mathbf{p}}(t)$. By applying the chain rule to the time derivative of $\frac{\partial \xi_{\mathbf{p}}}{\partial \mathbf{p}}(t)$ we get

$$\dot{S}_{\mathbf{p}}(t) = \frac{\partial f}{\partial \mathbf{x}}(q, \xi_{\mathbf{p}}(t), \mathbf{p})\ S_{\mathbf{p}}(t) + \frac{\partial f}{\partial \mathbf{p}}(q, \xi_{\mathbf{p}}(t), \mathbf{p}) \qquad (4)$$

Here $\frac{\partial f}{\partial \mathbf{x}}(q, \xi_{\mathbf{p}}(t), \mathbf{p})$ is the Jacobian matrix of f at the trajectory point at time t. This equation is thus an affine, time-varying ODE that, in the absence of discontinuity, can be solved in parallel with the ODE defining the dynamics (1).

When a trajectory switches from a mode q_1 to a mode q_2 due to the crossing of a surface given by $g_i(\mathbf{x}) = 0$, the dynamics of the system changes from $\dot{\mathbf{x}} = f_1(\mathbf{x}, \mathbf{p}) \triangleq f(q_1, \mathbf{x}, \mathbf{p})$ to $\dot{\mathbf{x}} = f_2(\mathbf{x}, \mathbf{p}) \triangleq f(q_2, \mathbf{x}, \mathbf{p})$. It follows that the evolution of the sensitivity matrix also changes from $\dot{S}_{\mathbf{p}} = \frac{\partial f_1}{\partial \mathbf{x}} S_{\mathbf{p}} + \frac{\partial f_1}{\partial \mathbf{p}}$ to $\dot{S}_{\mathbf{p}} = \frac{\partial f_2}{\partial \mathbf{x}} S_{\mathbf{p}} + \frac{\partial f_2}{\partial \mathbf{p}}$. Even though we do not consider resets in our models, i.e., the continuous state remains unaffected by the switching ($\mathbf{x}(\tau^-) = \mathbf{x}(\tau^+)$), the sensitivity matrix $S_{\mathbf{p}}$ can be discontinuous in τ. It can be shown that the jump condition, i.e. the difference between τ^- and τ^+ is given by [HP00]

$$S_{\mathbf{p}}(\tau^+) - S_{\mathbf{p}}(\tau^-) = \frac{d\tau}{d\mathbf{p}}\big(f_2(\tau, \mathbf{x}^*, \mathbf{p}) - f_1(\tau, \mathbf{x}^*, \mathbf{p})\big), \qquad (5)$$

$$\text{where } \frac{d\tau}{d\mathbf{p}} = \frac{\langle \nabla_{\mathbf{x}} g_i(\mathbf{x}^*), S_{\mathbf{p}}(\tau)\rangle}{\langle \nabla_{\mathbf{x}} g_i(\mathbf{x}^*), f_1(\tau, \mathbf{x}^*, \mathbf{p})\rangle}. \qquad (6)$$

In [HP00], conditions for the computation of sensitivity matrices are given for hybrid models more general than ours. They include evolutions given by differential algebraic equations, state resets, etc. Since our technique relies on the ability to compute numerical simulations and sensitivity matrices, it means that it can be straigthfowardly extended to handle these systems.

4 Parameter Synthesis Algorithm

In this section, we consider an hybrid system $\mathscr{S} = (\mathcal{Q}, f, e, g)$, a compact set of parameters $\mathcal{P}$ and a set of so-called "bad" states, $\mathcal{B} \in \mathbb{R}^n$. Our goal is to partition $\mathcal{P}$ into *safe*, *unsafe* and *uncertain* subsets, defined as follows.

Definition 1 (Parameter Synthesis Problem)

- *A* parameter synthesis problem *is a 4-uple* $(\mathscr{S}, \mathcal{P}, \mathcal{B}, T)$ *where* $\mathscr{S}$ *is an hybrid system,* $\mathcal{P}$ *a compact set,* $\mathcal{B}$ *a set and* T *a non-negative real number;*
- *A* solution *of the parameter synthesis problem* $(\mathscr{S}, \mathcal{P}, \mathcal{B}, T)$ *is a partition of* $\mathcal{P}$ *into three sets* $(\mathcal{P}_{saf}, \mathcal{P}_{unc}, \mathcal{P}_{bad})$ *such that: for all* $\mathbf{p} \in \mathcal{P}_{bad}$*,* $\xi_{\mathbf{p}}(t) \in \mathcal{B}$ *for some* $0 \leq t \leq T$*; for all* $\mathbf{p} \in \mathcal{P}_{saf}$*,* $\xi_{\mathbf{p}}(t) \notin \mathcal{B}$ *for all* $0 \leq t \leq T$*; and* $\mathcal{P}_{unc} = \mathcal{P} - \mathcal{P}_{saf} \cup \mathcal{P}_{bad}$.

Solutions to the parameter synthesis problem can be defined in terms of *reachable sets*, which we define next.

Definition 2 (Reachable Set). *The reachable set induced by a set of parameters $\mathcal{P}$ at time t is $\mathcal{R}_t(\mathcal{P}) = \bigcup_{\mathbf{p}\in\mathcal{P}} \xi_{\mathbf{p}}(t)$*

It is clear that $(\mathcal{P}_{\text{saf}}, \mathcal{P}_{\text{unc}}, \mathcal{P}_{\text{bad}})$ is a solution of $(\mathscr{S}, \mathcal{P}, \mathcal{B}, T)$ if and only if $\mathcal{R}_t(\mathcal{P}_{\text{saf}}) \cap \mathcal{B} = \emptyset\ \forall 0 \leq t \leq T$ and $\mathcal{P}_{\text{bad}} = \cup_l \mathcal{P}_l$ where for each l, $\mathcal{R}_t(\mathcal{P}_l) \subset \mathcal{B}$ for some $0 \leq t \leq T$. To characterize the precision of a solution, we use the following definition:

Definition 3 (δp-precise solution). *A solution $(\mathcal{P}_{saf}, \mathcal{P}_{unc}, \mathcal{P}_{bad})$ is said to be $\delta\mathbf{p}$-precise either if $\mathcal{P}_{unc}$ is empty or if it can be decomposed into a finite number of sets $\mathcal{P}_{unc} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_l$ such that for all j,*

- *The diameter of $\mathcal{S}_j$ is smaller than $\delta\mathbf{p}$, i.e., $\|\mathcal{S}_j\| \leq \delta\mathbf{p}$,*
- *The reachable set induced by $\mathcal{S}_j$ intersects with $\mathcal{B}$ for some $0 \leq t \leq T$, i.e., $\mathcal{R}_t(\mathcal{S}_j) \cap \mathcal{B} \neq \emptyset$,*
- *The reachable set induced by $\mathcal{S}_j$ is not a subset of $\mathcal{B}$ for any $0 \leq t \leq T$, i.e., $\mathcal{R}_t(\mathcal{S}_j) \nsubseteq \mathcal{B}$.*

Intuitively a $\delta\mathbf{p}$-precise solution covers the boundary between safe and unsafe parameters with a finite number of sets whose sizes are at most $\delta\mathbf{p}$. Also, by this definition, a solution for which the uncertain set is empty is $\delta\mathbf{p}$-precise for any $\delta\mathbf{p}$. In the remainder of this section, we present an algorithm that aims at computing a $\delta\mathbf{p}$-precise solution. The method is based on an iterative partitioning of the parameter space, the computation of reachable set estimates and their intersections with the bad set.

4.1 Reachable Set Estimation Using Sensitivity

For some subset $\mathcal{S}$ of $\mathcal{P}$, set $\mathcal{R}_t(\mathcal{S})$ can be approximated by using sensitivity analysis. Let $\mathbf{p}$ and $\mathbf{p}'$ be two parameter vectors in $\mathcal{S}$ and assume that we computed the trajectory $\xi_{\mathbf{p}}$ and the sensitivity matrix $S_{\mathbf{p}}$ at time t. Then we can use $\xi_{\mathbf{p}}(t)$ and $S_{\mathbf{p}}(t)$ to estimate $\xi_{\mathbf{p}'}(t)$. We denote this estimate by $\hat{\xi}^{\mathbf{p}}_{\mathbf{p}'}(t)$. The idea is to drop higher order terms in the Taylor expansion (3), which gives

$$\hat{\xi}^{\mathbf{p}}_{\mathbf{p}'}(t) = \xi_{\mathbf{p}}(t) + S_{\mathbf{p}}(t)(\mathbf{p}' - \mathbf{p}). \tag{7}$$

If we extend this estimate to all parameters $\mathbf{p}'$ in $\mathcal{S}$, we get the following estimate for the reachable set $\mathcal{R}_t(\mathcal{S})$:

$$\hat{\mathcal{R}}^{\mathbf{p}}_t(\mathcal{S}) = \bigcup_{\mathbf{p}'\in\mathcal{S}} \hat{\xi}_{\mathbf{p}'}(t) = \{\xi_{\mathbf{p}} - S_{\mathbf{p}}(t)\mathbf{p}\} \oplus S_{\mathbf{p}}(t)\mathcal{S} \tag{8}$$

Note that this is an affine transformation of the initial set $\mathcal{S}$ (see Fig. 1). As a particular situation, if the dynamics of the system is affine, the estimate is exact as there are no higher order terms in the Taylor expansion.

When the dynamics is nonlinear, $\hat{\mathcal{R}}^{\mathbf{p}}_t(\mathcal{S})$ is different from $\mathcal{R}_t(\mathcal{S})$. For instance, we have the following lemma.

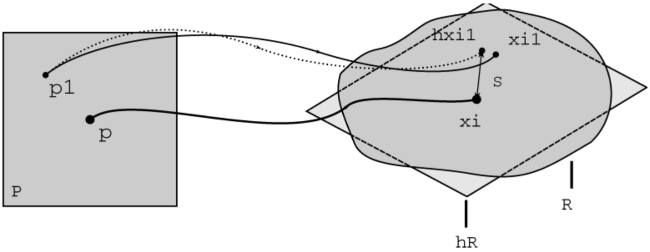


Fig. 1. Approximation of the reachable set using one trajectory and the corresponding sensitivity matrix

Lemma 1. *There exists a real number $K > 0$ such that*
$$d_H(\hat{\mathcal{R}}_t^{\mathbf{p}}(\mathcal{S}), \mathcal{R}_t(\mathcal{S})) \;\le\; K\;\|\mathcal{S}\|^2.$$

Proof. Let $\mathbf{y}$ be in $\hat{\mathcal{R}}_t^p(\mathcal{S})$, $\mathbf{p_y} \in \mathcal{S}$ be such that $\mathbf{y} = \hat{\xi}^p_{\mathbf{p}_y}(t)$ and $\mathbf{x} = \xi_{\mathbf{p}_y}(t) \in \mathcal{R}_t(\mathcal{S})$. From (3) we have $\mathbf{x} - \mathbf{y} = \xi_{\mathbf{p}_y}(t) - \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_y}(t) \;=\; \varphi(t, \mathbf{p}_y - \mathbf{p})$ where φ is a function such that $\varphi(t, \mathbf{p}_y - \mathbf{p}) = \mathcal{O}\big(\|\mathbf{p} - \mathbf{p}_y\|^2\big)$, meaning that we can find $K > 0$ such that $\|\mathbf{y} - \mathbf{x}\| = \|\xi_{\mathbf{p}_y}(t) - \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_y}(t)\| \le K\|\mathbf{p}_y - \mathbf{p}\|^2 \le K\|\mathcal{S}\|^2$. Since this is true for any $\mathbf{y}$ in $\hat{\mathcal{R}}_t^{\mathbf{p}}(\mathcal{S})$, $\sup_{\mathbf{y} \in \hat{\mathcal{R}}_t^{\mathbf{p}}(\mathcal{S})} d(y, \mathcal{R}_t(\mathcal{S})) \le K\|\mathcal{S}\|^2$. Similarly we can prove that $\sup_{\mathbf{x} \in \mathcal{R}_t(\mathcal{S})} d(x, \hat{\mathcal{R}}_t^p(\mathcal{S})) \le K\|\mathcal{S}\|^2$ which implies the result. □

Thus, the error depends on the diameter of $\mathcal{S}$. In order to improve the estimation, we can partition $\mathcal{S}$ into smaller subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_l$ and introduce new parameters, $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_l$ to compute more precise local estimates. Then we need to be able to estimate the benefit of such a refinement. To do so, we can compare for each $\mathcal{S}_j$ the estimate $\hat{\mathcal{R}}_t^{\mathbf{p}}(\mathcal{S}_j)$ that we get using the "global" center $\mathbf{p}$ with the estimate $\hat{\mathcal{R}}_t^{\mathbf{p}_j}(\mathcal{S}_j)$ that we get when using the "local" center $\mathbf{p}_j$. We have the following result:

Proposition 1. *Let $\mathcal{S}_j$ be a subset of a parameter set $\mathcal{S}$. Let $\mathbf{p} \in \mathcal{S}$ and $\mathbf{p}_j \in \mathcal{S}_j$. Then*
$$d_H(\hat{\mathcal{R}}_t^p(\mathcal{S}_j), \hat{\mathcal{R}}_t^{p_j}(\mathcal{S}_j)) \le Err(\mathcal{S}, \mathcal{S}_j) \tag{9}$$
where $Err(\mathcal{S}, \mathcal{S}_j) = \|\xi_{\mathbf{p}_j}(t) - \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j}(t)\| + \|S_{\mathbf{p}_j}(t) - S_{\mathbf{p}}(t)\|\|\mathcal{S}_j\|$.

Proof. Let $\mathbf{y}$ be in $\hat{\mathcal{R}}_t^{\mathbf{p}}(\mathcal{S}_j)$, $\mathbf{p}_y$ in $\mathcal{S}_j$ such that $\mathbf{y} = \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_y}(t)$ and $\mathbf{x} = \hat{\xi}^{\mathbf{p}_j}_{\mathbf{p}_y}(t)$. We need to compare

$$\mathbf{y} = \xi_{\mathbf{p}}(t) + S_{\mathbf{p}}(t)(\mathbf{p}_y - \mathbf{p}) \text{ with } \mathbf{x} = \xi_{\mathbf{p}_j}(t) + S_{\mathbf{p}_j}(t)(\mathbf{p}_y - \mathbf{p}_j). \tag{10}$$

We introduce the quantity $\hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j}(t) \;=\; \xi_{\mathbf{p}_j}(t) + S_{\mathbf{p}_j}(t)(\mathbf{p}'_j - \mathbf{p}_j)$ and with some algebraic manipulations of (10), we get

$$\hat{\xi}^{\mathbf{p}}_{\mathbf{p}'_j}(t) - \hat{\xi}^{\mathbf{p}_j}_{\mathbf{p}'_j}(t) = \xi_{\mathbf{p}_j}(t) - \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j}(t) + (S_{\mathbf{p}_j}(t) - S_{\mathbf{p}}(t))(\mathbf{p}'_j - \mathbf{p}_j)$$

which implies that $\|\mathbf{y} - \mathbf{x}\| \le Err(\mathcal{S}, \mathcal{S}_j)$. The end of the proof is then similar to that of Lemma 1. □

As illustrated in Fig. 2, the difference between the global and the local estimate can thus be decomposed into the error in the estimate $\hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j}(t)$ of the state reached at time t using $\mathbf{p}_j$ and another term involving the difference between the local and the global sensitivity matrices and the distance from local center. The quantity $Err(\mathcal{S}, \mathcal{S}_j)$ can be easily computed knowing the trajectory states $\xi_{\mathbf{p}}(t)$ and $\xi_{\mathbf{p}_j}(t)$ and their corresponding sensitivity matrices and from the diameter of $\mathcal{S}_j$.[1] Err has the following interesting properties:

- If the dynamics is affine, then $Err(\mathcal{S}, \mathcal{S}_j) = 0$. Indeed, in this case, $\hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j} = \xi_{\mathbf{p}_j}$, so the first term vanishes and $\mathcal{S}_{\mathbf{p}} = \mathcal{S}_{\mathbf{p}_j}$ so the second term vanishes as well;
- If limit $\|\mathcal{S}\|$ is 0 then limit $Err(\mathcal{S}, \mathcal{S}_j)$ is also 0. Indeed, as $\|\mathcal{S}\|$ decreases, so does $\|\mathbf{p} - \mathbf{p}_j\|$ and thus $\|\xi_{\mathbf{p}_j}(t) - \hat{\xi}^{\mathbf{p}}_{\mathbf{p}_j}(t)\|$ and $\|\mathcal{S}_j\|$ since $\mathcal{S}_j$ is a subset of $\mathcal{S}$. Moreover, $Err(\mathcal{S}, \mathcal{S}_j) = \mathcal{O}\big(\|\mathcal{S}\|^2\big)$.

Thus we can compute a reachable set $\mathcal{R}_t(\mathcal{S})$ at a given time instant t and estimate the approximation error. To get an estimate $\mathcal{R}_{[0,T]}(\mathcal{S})$ on the interval $[0, T]$, one can do the computation for $t_0 = 0$, t_1, ... and $t_N = T$ and use some form of interpolation between t_k and t_{k+1}. This introduces additional error which depends on $|t_{k+1} - t_k|$ and the order of the interpolation method used.

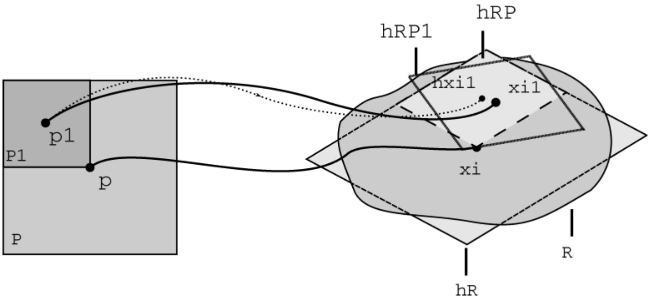


Fig. 2. "global" and "local" estimate of the reachable set $\mathcal{R}_t(\mathcal{S}_j)$

4.2 Algorithm

The key ideas of the algorithm presented below are the following:

- use the estimate $\hat{\mathcal{R}}^{\mathbf{p}}_t$ and its intersection with $\mathcal{B}$ to classify sets as safe, uncertain or unsafe;
- use Err to testify whether $\hat{\mathcal{R}}^{\mathbf{p}}_t$ is a reliable estimate: if it is more than a given tolerance $Tol > 0$ for a set $\mathcal{S}$, we classify $\mathcal{S}$ as uncertain;
- iteratively apply a refining operator on uncertain subsets to produce a finer partitioning from which we deduce more safe or unsafe subsets;
- stop when there are no uncertain subsets left or when all uncertain subsets are smaller in diameter than δp.

[1] Note that the value of Err actually depends not only on $\mathcal{S}$ and $\mathcal{S}_j$ but also on the choice of $\mathbf{p}$ and $\mathbf{p}_j$. We leave it implicit to simplify the notation.

To guarantee that the algorithm always terminates in a finite number of steps, we partition uncertain sets into a set of subsets that are at least γ times smaller. We define these γ-Refining partitions as follows.

Definition 1 (γ-Refining partition). *A γ-refining partition, where $0 < \gamma < 1$, of a set $\mathcal{S}$ is a finite set of sets $\{\mathcal{S}_1,\ \mathcal{S}_2,\ \ldots, \mathcal{S}_l\}$ such that*

$$\mathcal{S} = \bigcup_{j=1}^{l} \mathcal{S}_j \quad \textit{and} \quad \max_{j \in \{1,\ldots,l\}} \|\mathcal{S}_j\| \leq \gamma \|\mathcal{S}\|$$

We assume the existence of a function Refine_γ that maps a set to one of his γ-refining partitions for some $0 < \gamma < 1$ and give the complete algorithm below.

Algorithm 1. Parameter Synthesis Algorithm

```
procedure ParamSynthesis($\mathcal{P}$, $\mathcal{B}$, T, $\delta p$, $Tol$)
    $\mathcal{P}_{\text{saf}} = \mathcal{P}_{\text{bad}} = \emptyset$, $\mathcal{P}_{\text{unc}} = \{\mathcal{P}\}$
    repeat
        Pick and remove $\mathcal{S}$ from $\mathcal{P}_{\text{unc}}$
        for each $\mathcal{S}_j \in \text{Refine}_\gamma(\mathcal{S})$ do
            if $Err(\mathcal{S}, \mathcal{S}_j) \leq Tol$ then                ▷ Reach set estimate is reliable
                if $\hat{\mathcal{R}}^q_{[0,T]}(\mathcal{S}_j) \cap \mathcal{B} = \emptyset$ then            ▷ Reach set away from $\mathcal{B}$
                    $\mathcal{P}_{\text{saf}} = \mathcal{P}_{\text{saf}} \cup \mathcal{S}_j$
                else if $\hat{\mathcal{R}}^q_{[0,T]}(\mathcal{S}_j) \subset \mathcal{B}$ then               ▷ Reach set inside $\mathcal{B}$
                    $\mathcal{P}_{\text{bad}} = \mathcal{P}_{\text{bad}} \cup \mathcal{S}_j$
                else
                    $\mathcal{P}_{\text{unc}} = \mathcal{P}_{\text{unc}} \cup \{\mathcal{S}_j\}$       ▷ Some intersection with the bad set
                end if
            else
                $\mathcal{P}_{\text{unc}} = \mathcal{P}_{\text{unc}} \cup \{\mathcal{S}_j\}$           ▷ Reach set estimate not enough precise
            end if
        end for
    until $\mathcal{P}_{\text{unc}} \neq \emptyset$ and $\max_{\mathcal{P}_j \in \mathcal{P}_{\text{unc}}} \|\mathcal{P}_j\| \leq \delta p$
    return $\mathcal{P}_{\text{saf}}$, $\mathcal{P}_{\text{unc}}$, $\mathcal{P}_{\text{bad}}$
end procedure
```

5 Implementation and Experimentations

We implemented Algorithm 1 within the toolbox `Breach` described in [Don07]. Parameter sets are specified as symmetrical rectangular sets $\mathcal{S}(\mathbf{p}, \epsilon)$ where $\mathbf{p}$ and ϵ are in $\mathbb{R}^{n_p}$ and such that $\mathcal{S}(\mathbf{p}, \epsilon) = \{\mathbf{p}' : \mathbf{p} - \epsilon \leq \mathbf{p}' \leq \mathbf{p} + \epsilon\}$. The procedure uses a simple refinement operator $\text{Refine}_{\frac{1}{2}}$ such that

$$\text{Refine}_{\frac{1}{2}}(\mathcal{S}(\mathbf{p}, \epsilon)) = \{\mathcal{S}(\mathbf{p}^1, \epsilon^1), \mathcal{S}(\mathbf{p}^2, \epsilon^2), \ldots, \mathcal{S}(\mathbf{p}^l, \epsilon^l)\}$$

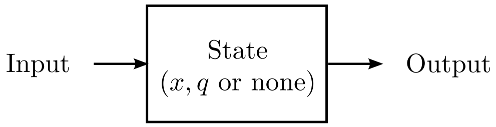
with $\epsilon^k = \epsilon/2$ and $\mathbf{p}^k = \mathbf{p} + (\nu_1^k \frac{\epsilon_1}{2}, \nu_2^k \frac{\epsilon_2}{2}, \ldots, \nu_n^k \frac{\epsilon_n}{2})$ where $\nu_i^k \in \{-1, +1\}$ so that when k ranges over $\{1, \ldots, l\}$ all possible sign combinations are met[2]. The

[2] The use of alternative refinement operators is a direction for further investigation.

toolbox interfaces MATLAB with CVODES [SH05], a numerical solver designed to solve efficiently and accurately ODEs and sensitivity equations of the form Eq. (4).

5.1 Sensitivity Analysis for Simulink Models

A Simulink block diagram model is a graphical representation of a mathematical model of an hybrid dynamic system [Mat]. It is composed of interconnected *time-based* blocks of the form shown below



where the state contained in the block (if present) can be either discrete or discontinuous. At each time step, the Simulink engine:

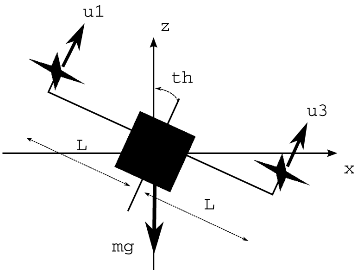
1. Computes each block output;
2. Updates the discrete states;
3. Computes the time derivatives $f(\mathbf{x})$ of continuous states;
4. Updates the continuous states by integrating $\dot{\mathbf{x}} = f(\mathbf{x})$ for one step;
5. Optionally checks for zero-crossing;
6. Updates the time for the next time step.

Thus the simulation scheme of Simulink is similar to the simulation algorithm presented in Section 2. To apply our algorithm to a Simulink model we need to extract from it the function f defining the continuous dynamics, the event function e and the guard function g, and to make them available for `Breach` to compute trajectories and sensitivity matrices. This is done using code generation provided by the Real-Time Workshop Toolbox [Mat]. The generated code implements routines for each of the above steps. For instance f can be obtained from step 3, e can be obtained from the discrete states update in step 2 and g is obtained from step 5. The overall procedure is shown in Fig. 3. A script generates C routines compatible with CVODES from the code generated by the Real-Time Workshop. Then `f()` calls the routines `MdlOutputs()` and `MdlDerivative()` for integration, `e()` calls `MdlUpdate()` to update discrete states and `g()` is used for zero-crossing detection (which is a CVODES built-in feature). Eq. (6) and (7) are used to update the sensitivity matrices at switching times.

5.2 Starmac Model with Navigation Supervisor

We consider a simplified model of a quad-rotor helicopter [HHWT07] where only the altitude z and the axis x are considered. The equations of motions for the quadrotor illustrated below are given by:

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



where $m = 0.5184$, $c = 0.15$, $L = 0.236$, $I_y = 0.04774$. The state vector is then $\mathbf{x} = (\dot{x}, x, \dot{z}, z, \dot{\theta}, \theta)$.

Given a goal state $\mathbf{x}^*$, a standard linear quadratic regulator (LQR) of the form $\mathbf{u} = K\,(\mathbf{x} - \mathbf{x}^*) + \frac{mg}{4}\,\mathbf{1}$ (where $\mathbf{1}$ is the vector $(1, 1, 1, 1)$) was designed to drive the system to $\mathbf{x}^*$ from any state $\mathbf{x}_0$. While doing so, the Starmac needs to avoid collisions with obstacles and maintain a pre-specified minimum safe flying altitude above an unknown terrain. This is monitored using two on-board proximity sensors, one in the horizontal (x) and the other in vertical (z) directions. Using the sensors and the value of the current state, a supervisor implements the following navigation strategy: in absence of proximity warnings, use the LQR control and move towards the target ('GoToTarget' mode); if either of the proximity warnings is active, switch to a constant control $\mathbf{u} = (\bar{u}, \bar{u}, \bar{u}, \bar{u})$, for some $\bar{u} > 0$, in order to go up until being safe ('GoUp' Mode) then resume to GoToTarget mode (see Figure 4).

While crashing of the Starmac into an obstacle is certainly undesirable, it may be desirable for it to be able to hover close to an obstacle. Hence the horizontal proximity warning was made velocity-dependent in the GoToTarget state, i.e., the more the velocity the farther away the system needs to be from the obstacle. The critical distance is set to be the product $\dot{x}\,t_{safe}$, for some $t_{safe} > 0$. In GoUp mode, the supervisor checks for a fixed horizontal distance h_{safe} from the obstacle. In both GoToTarget and GoUp modes, the vertical proximity from ground is a fixed desired vertical distance v_{safe}. This switching of control strategies leads

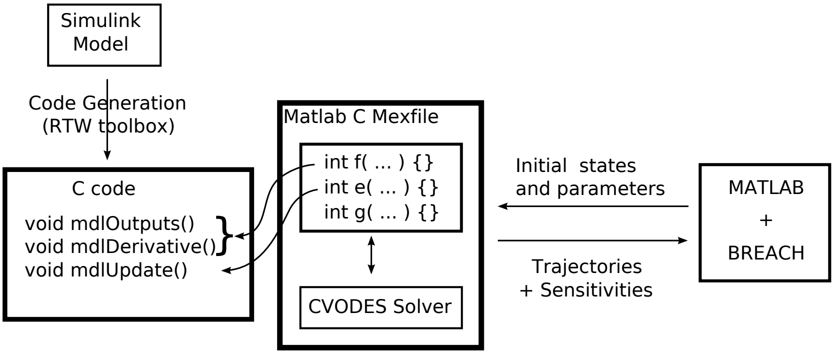


Fig. 3. Implementing sensitivity analysis for Simulink models in `Breach`

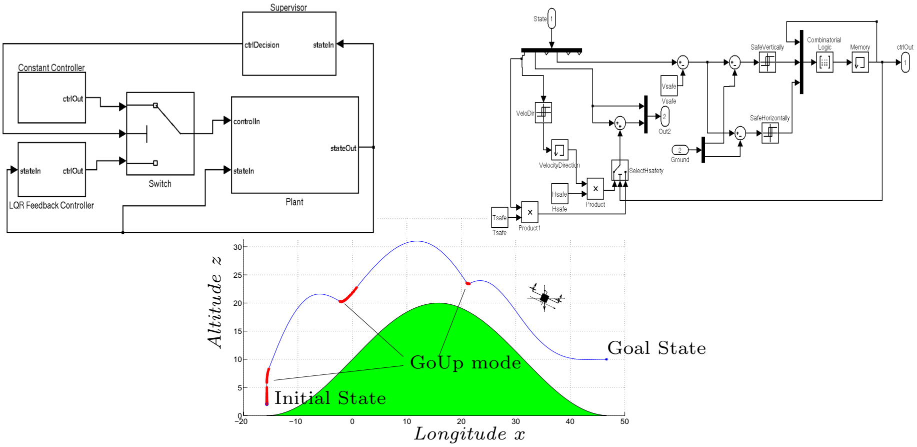


Fig. 4. Simulink diagram of the model (top left) and the supervisor (top right) and a sample (safe) trajectory

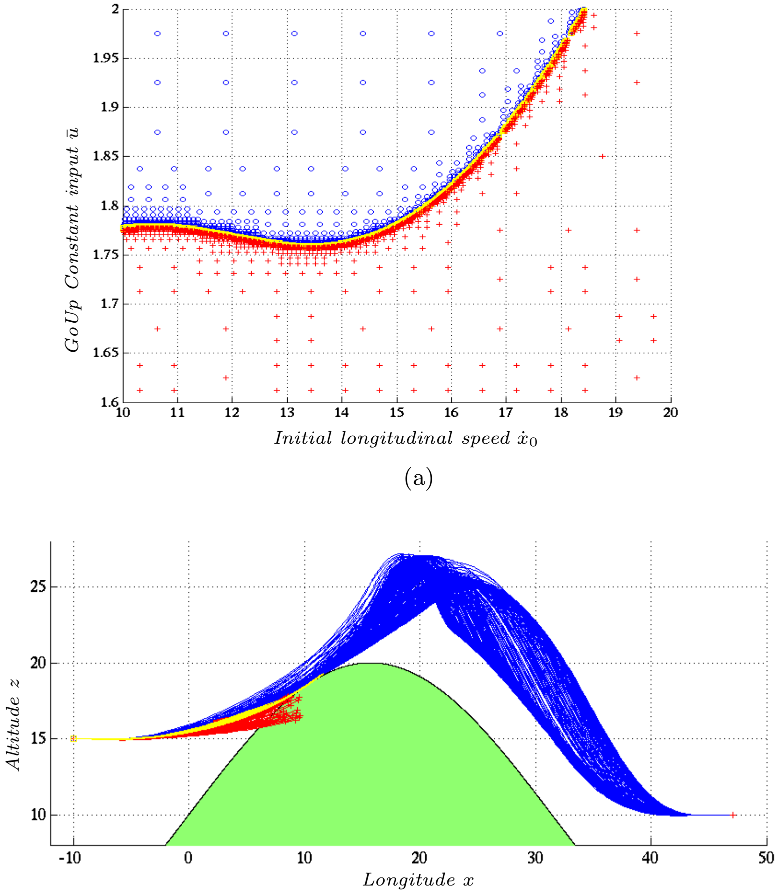
to hybrid dynamics with two discrete states, namely 'GoToTarget' and 'GoUp'. The proximity warning conditions in the two directions serve as the guards for discrete jumps between the two states.

5.3 Experimental Results

The Starmac dynamics, the LQR control and the supervisor were modeled in Simulink (Fig. 4). Proximity detection was modeled using relay blocks. The difference between desired distance from the obstacles and the current distance from obstacles can be fed as the input to these relays. These input signals to the relays are in turn extracted from the generated C code and fed to our sensitivity analysis machinery as the zero-crossing detection function $g()$.

We applied our parameter synthesis to the Starmac and the supervisor for different sets of parameters and a given terrain. The parameters that can vary in this model include the initial state variables (x_0, z_0, θ_0, $\dot{x}_0$, $\dot{z}_0$ and $\dot{\theta}_0$), the supervisor parameters (h_{safe}, v_{safe}, t_{safe}, $\bar{u}$), the Starmac characteristics (m, I_y, b), etc. We present the results we obtained for a situation where an initial position was set on one side of a hill (described by a simple sinusoid) and a goal state on the other side. The varying parameters where chosen to be the initial horizontal speed $\dot{x}_0$ and the constant control input $\bar{u}$ in GoUp mode, so that if we omit other parameters with a fixed value, $\mathcal{P} = \{(\dot{x}_0, \bar{u}) : 10 \leq \dot{x}_0 \leq 20,\ 1.6 \leq \bar{u} \leq 2\}$. The ground was set to be the bad set, given by $\mathcal{B} = \{z \leq \text{Terrain}(x)\}$ where Terrain is a sinusoidal function. The results are presented in Figure 5.

The algorithm performed 3642 simulations for a computational time of 55 seconds on a laptop with a Dual Core 1.8GHz processor. Most simulations stem from the neighborhood of a curve delimiting values of $(\dot{x}_0, \bar{u})$ for which trajectories cannot avoid the ground from values for which the avoidance maneuver



(a)

(b)

Fig. 5. Results varying the initial horizontal speed $\dot{x}_0$ and the GoUp constant input $\bar{u}$. (a) Parameters used for the simulation. Crosses represent values for which trajectories hit the ground while circles represent values for which the goal state is safely reached. (b) Resulting trajectories in the (x, z) plane.

works and the Starmac safely reaches the goal state. The algorithm refined the parameter set until a precision of $\delta\dot{x}_0 = 0.001$ and $\delta\bar{u} = 0.0004$. Note that performing simulations from parameters on a complete grid of this resolution would have required 262,144 simulations, more than 70 times the number of simulations executed by the sensitivity-based algorithm.

6 Conclusion

This paper presents a parameter synthesis algorithm for nonlinear hybrid systems based on numerical simulation and sensitivity analysis. The algorithm is scalable in terms of number of state variables and is implemented in a MATLAB toolbox, `Breach`, that can handle Simulink models directly. The proposed approach is illustrated for a six-dimensional nonlinear Simulink model of the STARMAC quadrotor helicopter with a non-trivial hybrid supervisor.

The primary limitation of the algorithm is that the complexity of the refinement procedure is exponential in the number parameters. We are currently investigating methods for scaling the approach to large numbers of parameters. We are also extending the prototype implementation, which currently handles zero crossing detection for relay blocks, to a larger set of Simulink and Stateflow blocks. Directions for future research include the ability to handle models with uncertain inputs, i.e., dynamics of the form $\dot{\mathbf{x}} = f(q, \mathbf{x}(t), u(t), \mathbf{p})$, and extensions to stochastic systems.

References

[ADF$^+$06] Asarin, E., Dang, T., Frehse, G., Girard, A., Le Guernic, C., Maler, O.: Recent progress in continuous and hybrid reachability analysis. In: Proc. IEEE International Symposium on Computer-Aided Control Systems Design. IEEE Computer Society Press, Los Alamitos (2006)

[DM07] Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 174–189. Springer, Heidelberg (2007)

[Don07] Donzé, A.: Trajectory-Based Verification and Controller Synthesis for Continuous and Hybrid Systems. PhD thesis, University Joseph Fourier (June 2007)

[EKP01] Esposito, J.M., Kumar, V., Pappas, G.J.: Accurate event detection for simulating hybrid systems. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 204–217. Springer, Heidelberg (2001)

[FJK08] Frehse, G., Jha, S.K., Krogh, B.H.: A counterexample-guided approach to parameter synthesis for linear hybrid automata. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 187–200. Springer, Heidelberg (2008)

[Fre05] Frehse, G.: Phaver: Algorithmic verification of hybrid systems past hytech, pp. 258–273. Springer, Heidelberg (2005)

[GP06] Girard, A., Pappas, G.J.: Verification using simulation. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 272–286. Springer, Heidelberg (2006)

[HHWT97] Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: Hytech: A model checker for hybrid systems. Software Tools for Technology Transfer 1, 460–463 (1997)

[HHWT07] Hoffmann, G., Huang, H., Waslander, S., Tomlin, C.J.: Quadrotor helicopter flight dynamics and control: Theory and experiment. In: Proceedings of the AIAA Conference on Guidance, Navigation and Control, Hilton Head, South Carolina (August 2007)

[HP00] Hiskens, I.A., Pai, M.A.: Trajectory sensitivity analysis of hybrid systems. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 47(2), 204–220 (2000)

[HWT96] Henzinger, T.A., Wong-Toi, H.: Using hytech to synthesize control parameters for a steam boiler. In: Abrial, J.-R., Börger, E., Langmaack, H. (eds.) Dagstuhl Seminar 1995. LNCS, vol. 1165, pp. 265–282. Springer, Heidelberg (1996)

[LKCK08] Lerda, F., Kapinski, J., Clarke, E.M., Krogh, B.H.: Verification of supervisory control software using state proximity and merging. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 344–357. Springer, Heidelberg (2008)

[Mat] The Mathworks. Simulink User Guide

[SH05] Serban, R., Hindmarsh, A.C.: Cvodes: the sensitivity-enabled ode solver in sundials. In: Proceedings of IDETC/CIE 2005, Long Beach, CA (September 2005)