Controller Verification for Nonlinear Systems: A computational approach using phase-space geometric models*

Feng Zhao
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
zhao@parc.xerox.com

Jeff A. May
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210

Abstract

This paper presents an algorithm for verifying control laws using phase-space geometric modeling of dynamical systems. The algorithm evolves a hierarchically-refined bound of system nonlinear dynamics and can address practical concerns such as sensor, actuator, and modeling uncertainties in a systematic manner. The algorithm has been applied to verifying a control law for a magnetic levitation system, and the computational results are compared against the performance of the actual physical system.

Keyword. Control systems, Verification, Phase space, Geometric approaches, Computational methods.

Introduction

Control verification ensures correct behaviors for controlled physical systems. Important applications range from safety-critical systems such as aircraft controllers, where improper behavior can result in loss of life, to cost-critical systems such as factory controllers, where a faulty controller can result in costly inefficiency. Unfortunately, obtaining a closed-form analytic solution to the verification problem is often impractical. The nonlinear nature of many man-made systems requires that approximations be made to apply most verification techniques. Uncertainties such as modeling and sensing error make it difficult to express the range of possible behaviors of the system in a tractable form. Thus, verifying a controlled system frequently requires that linear approximations be made, and that considerations for factors such as modeling error and sensing error be omitted.

This paper describes a computational verification algorithm that relies on evolving phase-space geometric models of system dynamics. The contributions of this paper are threefold: (1) It introduces a novel hierarchical refinement of bounds on system dynamics to avoid unnecessary over-approximation of nonlinear behavior; (2) The phase-space representation it uses can model nonlinearity and uncertainty in a systematic, intuitive manner; (3) The algorithm has been applied to verify a nonlinear maglev control system prototyped in our laboratory. Although the phase space of the example used in the paper is two dimensional, the algorithm is applicable to higher-order control systems as well as those whose dynamics does not admit a closed-form analytic description but whose states can be fully observed via experimental means.

Our algorithm evolves from earlier work in phase-space control synthesis (Bradley & Zhao 1993; Zhao, Loh, & May 1997).

* An earlier version of the paper appeared in Proc. of the 7th IFAC Symposium on Artificial Intelligence in Real-Time Control, Grand Canyon, Arizona, October 1998.

Computational methods for automated phase-space analysis

Researchers in qualitative reasoning, hybrid systems, and control engineering have developed a number of algorithms and computer programs that automatically perform phase-space analysis of a dynamical system:

- Sacks developed the PLR program for analyzing dynamical systems using inequality reasoning of phase space trajectories in two dimensions (Sacks 1991). The algorithm constructs piecewise linear approximations to nonlinear terms in a 2nd-order differential equation, examines how trajectories intersect these piecewise linear approximation boundaries in phase space, and partitions the space into monotonic regions. A transition graph is then constructed to encode how the system moves from region to region and hence captures the more global dynamics of the system. For instance, the oscillatory behavior of a van del Pol oscillator is graphically represented as a cycle of transitions among four phase-space regions. Yip's KAM program analyzes the orbit structure of phase space by embedding the orbits in a neighborhood graph and uses techniques of computer vision to recognize the types of orbits (Yip 1991). For instance, the branching structure in a minimum-spanning tree embedding states of a Hamiltonian system is used to determine if the orbit is a separatrix or a quasi-periodic orbit. The KAM program has achieved an impressive level of performance in solving an open problem in fluid dynamics: it performed an extensive search in the parameter space for the dynamics of a wave tank problem and identified the critical parameter values for the onset of chaotic motion (Tsai, Yue, & Yip 1990).

- Using techniques of computational geometry and graph theory, Zhao developed the MAPS algorithm for constructing phase-space geometric models for equivalence classes of trajectories (flow pipes) that can be efficiently manipulated and searched for control reference trajectories (Zhao 1995). A phase space is first tessellated with a Delaunay neighborhood graph. The cells in the graph are then aggregated to form flow pipes for homotopy equivalence classes of trajectories. A reachability graph for the dynamics under different control signals is built by intersecting the flow pipes. Other practical considerations such as measurement errors or parametric uncertainties can be conveniently represented by geometric volumes in phase spaces. The MAPS algorithm applies to two or higher-dimensional phase spaces of nonlinear systems. The work of Nishida et al. has proven that mappings of flows in phase space can be accurately constructed using a fine grain representation of flow patterns (Nishida & et al. 1991). The PSX2NL algorithm analyzes topological flow patterns of a two-dimensional system and encodes flow mappings between well-defined surfaces in phase space in a symbolic structure called flow grammar. Recently, Greenstreet and Mitchell developed a similar polyhedral representation for efficiently projecting phase-space dynamics in two dimensions (Greenstreet & Mitchell 1998).
- A number of approaches adopted a grid-based representation for phase space behaviors. Hsu pioneered the idea of cell-to-cell mapping and introduced a basic algorithm for constructing such a map computationally (Hsu 1987). Bradley developed one of the first programs that automatically builds a cell map for a phase space (Bradley 1995). The cell partition is adaptive in that the resolution depends on the complexity of the behaviors in a local region. The cell map is used to recursively find control trajectories that satisfy a set of performance criteria and has been applied to chaotic dynamical systems. Dang and Maler recently used the grid-based representation for phase-space regions of a hybrid system during verification (Dang & Maler 1998).
- Several recently developed verification methods rely on symbolic techniques such as model checking in temporal or modal logic. The work of Alur et al. (Alur *et al.* 1993) uses phase-space polyhedral approximation to bound dynamics; so does the differential inclusion of Puri et al. (Puri, Borkar, & Varaiya 1996). Others do not yet explicitly exploit phase spaces but their models have clear phase-space interpretations. For instance, Kuipers and Astrom expressed behaviors of a heterogeneous controller, qualitatively modelled as a transition graph derived from the so-called qualitative differential equation, as propositions in temporal logic and uses logical inference to verify properties of the system (Kuipers & Astrom 1994). The expressiveness of these analytic languages is often restricted in order to make the computation tractable (e.g. propositional vs. predicate logic). It remains as an open research problem to explore how phase-space geometric models can be exploited to significantly constrain the search space in symbolic verification.

In summary, the phase-space model of dynamical behaviors serves as a convenient vehicle for *behavioral programming* of physical systems: desired behaviors can be computationally explored, selected, mixed, and modified according to the goals and constraints (Branicky 1995).

Phase-Space Verification Algorithm

The phase-space verification algorithm is used to verify proper regions of operation that have the desired limit behaviors for stabilization control systems (i.e. control systems that are designed to stabilize the plant in a goal region). Other properties such as overshoot or convergence rate can also be verified with only minor changes to the algorithm. The algorithm is applicable to discrete-time control systems with a fixed sampling frequency. The underlying dynamics of the plant may be continuous, discrete, or hybrid. It is assumed that the system is designed to operate within a bounded region of the phase space. Let the system dynamics be described by: $\dot{x} = F(x, u)$, where x is the system state, and u is the control input. Since the system is discretely sampled, the dynamics of the controlled system can be written as $x_{n+1} = f(x_n)$, where x_i denotes the system state at time t_i and $f(x_i) = \int F(x_i, o(x_i))$ after one time period ($o(x_i)$ is the controller output at x_i).

The algorithm proceeds as follows:

1. Partition the phase-space region of interest into a finite set of cells C.
2. Determine the initial controllable region R_{cont}.
3. For each cell c in $C - R_{cont}$,
   (a) Find a polytope p_c bounding the image of c under f.
   (b) Compute the "escape polytope" $e_c = p_c - c$.
   (c) If e_c is contained within R_{cont}, and f generates no cycles within c, mark c as verified, and set $R_{cont} = R_{cont} \cup c$.
4. If any new cells were added to R_{cont} in step 3, repeat step 3.
5. If the region of interest has been verified, or if a pre-specified number of steps has been taken, quit. Otherwise, form a new set C' by subdividing the unmarked cells in C. Set $C = C'$, and go to step 3.

The partitioning of the phase space in step 1 is arbitrary; however, regular partitions are often used, and certain types of control suggest preferred partitions.

For example, control based on cell maps suggests an initial partition identical to the one used to generate the cell maps (Hsu 1987).

Determination of the initial controllable region requires a bit more effort. There are two basic approaches. If the controller has already been verified for a certain region R_1 (e.g. using analytic techniques), and the verification algorithm is being used to extend that region, R_{cont} is set to R_1, and all cells that are fully contained within R_1 are marked. This approach is taken, for example, when a local controller (e.g. one based on linear techniques) is being augmented by a global controller.

If no "pre-verified" region is available, R_{cont} cannot just be set to the goal region, because it is possible that for the given controller, the plant can start out in the goal region, but later exit it and never return. Thus, in this approach, a set R_{cont} of "core cells" must be found. The "core cells" have the following two properties:

1. Every $c \in R_{cont}$ is in the goal region.
2. For every $c \in R_{cont}$, the image of c under f is contained in R_{cont}.

A maximal set of core cells (for a given phase space partition) can be generated by selecting all cells contained in the goal region, and then iteratively eliminating cells whose image bound lies outside the set of selected cells.

Finding a polytope p_c bounding $f(c)$ can be achieved in many ways. One of the simplest and most efficient ways to find a suitable p_c is to compute the minimum and maximum values for each component of F over the cell c and form a bounding box for $f(c)$ using these values. In hybrid systems terms (see e.g. (Branicky 1995)), each cell can be thought of as a discrete state, and the bounding polytope is determined by approximating system dynamics with a rectangular differential inclusion.

In step 3, we are checking for two properties:

1. $\forall x \in c : f^n(x) \notin c$ for some $n \in \mathbf{N}$. That is, all trajectories of the system starting within c eventually exit c.
2. $\forall x \in c, n \in \mathbf{N} : f_n(x) \notin c \wedge f_{n-1}(x) \in c \Rightarrow f_n(x) \in R_{cont}$. That is, when a trajectory exits c, it reaches a cell that has already been marked as verified.

Checking the second property is straightforward. To check the first property we intersect $f(c)$ with c to form a polytope p'_c. This process is repeated with p'_c until the intersection is empty, or a pre-specified number of iterations, i, is exceeded. Thus, we replace the first property with a stronger condition—that all trajectories leave c within i time steps.

Assuming that a regular, rectangular initial partition is used, and that subdivision is done uniformly, the space requirement of the algorithm is $O((2^d)^s n)$ where d is the dimension of the phase space, s is the level of subdivision, and n is the number of cells in the initial partition. Thus, the memory requirements depend linearly on the size of the initial partition, and exponentially on the level of subdivision.

Step 1 typically has time complexity linear in n, although a complex partition may require more time. Step 2 requires at most $O(n^2)$ time if a core set is being determined. Each iteration of step 3 takes $O(n)$ time (for the comparison with R_{cont} in 3c), so the entire algorithm requires $O(((2^d)^s n)^3)$. As with the space complexity, the time complexity has an exponential dependence on the level of subdivision, and a polynomial dependence on the size of the initial partition. With a regular partition, the comparison in step 3c depends only on the cells intersected by the image bound, which is typically far less than n. Similarly, the order of selection in step 3 can have a large impact on the efficiency of the algorithm. In practice, the verification of a cell far from the initial controllable region depends on the verification of cells nearer the region. An intelligent ordering, that starts from the cells nearest the controllable region and works outward, often results in more cells being verified for each iteration of step 3, thus reducing the number of times step 3 must be iterated to a number far less than n.

Proof of Soundness

In this section, it is shown that if the system starts in a cell that has been marked as verified, the system will eventually progress to a state within the goal region.

Proof: Number the cells of the phase space partition in the order that they are marked, with all initially verified cells (or core cells) numbered zero. Consider cell number i, with $i > 0$. Since the cells are numbered by order of marking, all trajectories starting in cell i eventually flow into a lower numbered cell (since cell i will be marked only if its image bound lies in cells that have already been marked). Thus, by induction, all trajectories starting in a cell with a positive number will eventually flow into a cell numbered zero. By assumption (or by the definition of core cell), no trajectories starting from a zero numbered cell will leave the set of zero numbered cells, so all trajectories will eventually progress to a state within the goal region.

Note that this only guarantees that marked cells exhibit proper behavior—there is no guarantee that all cells that exhibit proper behavior will be marked (i.e. the algorithm is sound but not necessarily complete).

Enhancements and Optimizations

In the above description, several practical issues, such as measurement error, controller output error, and modeling error are not mentioned. However, because of the geometric nature of the computation, such considerations can be incorporated in a straightforward fashion. Measurement error can be accounted for by expanding the cell when the bounding polytope is determined. Controller output error and model error can be dealt with (assuming the error is bounded) by expanding the image bounding polytope corresponding to the potential range of the function f describing the dynamics.

Continuous time systems can be verified by selecting a time period, and examining the evolution of the system over this time period (i.e. treating the system as a discretely-sampled system, and using the base verification algorithm). If necessary, this time period can be iteratively reduced to provide a less conservative bound on system behavior (as is done with the phase-space partition in the base algorithm).

Certain other properties of a system can be verified with minor modifications to the base algorithm. For example, suppose it is necessary to verify not only that the system reaches the goal region, but also that the percent overshoot is limited. In this case, when a cell is being checked, its image bound must fall within marked cells and each of these intersected cells must be annotated as having trajectories with a tolerable maximum distance to the goal region. The newly marked cell is then annotated with its maximum distance as well (computed as the maximum of the annotated values of the intersected cells and the distance of the cell itself).

In the base algorithm, only the behavior after one sampling period is considered. This is because the bounding polytope of the image of a cell increases in size exponentially with time, thus making the bound less accurate the longer the time period considered. However, when the system dynamics are "slow" in comparison to the partition granularity and the sampling period, a cell's image bound will often overlap with a neighbor, resulting in a dependence from the cell to its neighbor (Since the neighbor must be verified before the cell in consideration can be; see figure 1). If the system has "spiraling" trajectories (figure 2), a cycle of dependence (figure 3) between a set of unverified cells can occur. In this case, the partition must be subdivided several times to properly verify the system. This subdivision is costly—both memory usage and computation time scale exponentially with the level of subdivision in the worst case. In these cases, the algorithm can be optimized by continuing to iterate the function f when doing so is beneficial. If the escape polytope e_c does not lie entirely within verified cells, the polytope is clipped against those unverified cells it intersects, and the resulting polytopes are recursively checked. This process continues until either the cell is verified (i.e. the iteration produces an image bound that is contained within the verified region), or no further "progress" is made. The current implementation considers "progress" to be made as long as the unverified area (volume) occupied by the bounding polytope is decreasing in size (see figure 4). Other criteria may also be useful.

Algorithmically, this optimization replaces steps 3a - c with a call to the following function on cell c:

```
bool CheckRegion(region r)
1. Find a polytope $p_r$ bounding the image of $r$ under $f$.
2. Compute the "escape polytope" $e_r = p_r - r$.
3. Set $r_{unverified} = e_r - R_{cont}$.
4. If $r_{unverified} = \phi$ return true.
```

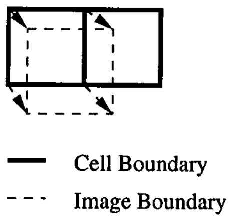

Figure 1: Slow dynamics with respect to the partition granularity and the sampling period results in dependence on an adjacent cell.

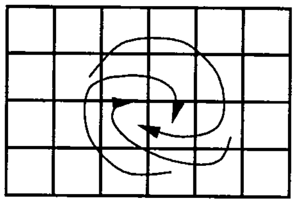

Figure 2: "Spiraling" trajectories in phase space.

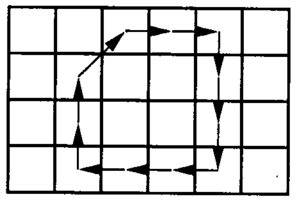

Figure 3: Cyclic dependence between cells created by spiral trajectories above.

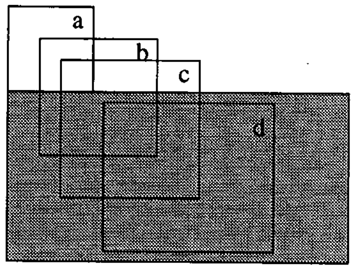

Figure 4: Sequence of images with decreasing unverified area. The shaded area represents the previous verified region. The transparent boxes represent a sequence of image bounds (b is the image bound of a, c is the image bound of $b - R_{cont}$, et cetera).

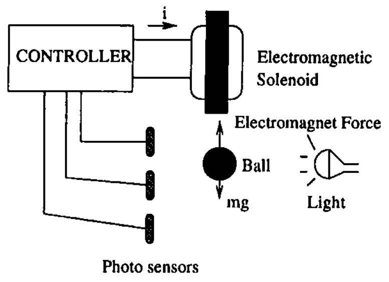

Figure 5: Diagram of magnetic levitation system.

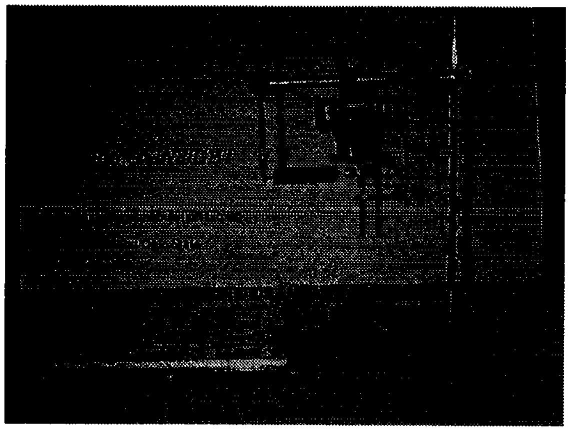

Figure 6: Photo of the actual magnetic levitation testbed prototyped at Ohio State University.

5. Otherwise, if $volume(r_{unverified}) \leq volume(r)$, return $CheckRegion(r_{unverified})$, else return false.

Results and Analysis

The verification procedure has been tested on the controller for a magnetic levitation system (figures 5 and 6). This system is a useful testbed since it is inherently unstable and nonlinear (due to the inverse square law of magnetic attraction).

The initial verifiable region is obtained using a local controller, generated using a linearized model of the original nonlinear system. This linear controller is augmented by a nonlinear, phase space based global controller (as in (Loh 1997; Zhao, Loh, & May 1997)). The equilibrium point was set to 11.6 millimeters from the bottom of the solenoid to the center of mass of the steel ball. The coordinate system is chosen such that the displacement vector points downwards from the solenoid to the steel ball. The local controller was used when the ball was within one millimeter of the equilibrium point with a velocity having absolute value less than 0.05 meters per second. Outside this region, the global controller was invoked.

The optimized algorithm was used to generate a predicted region of stability for the system — no considerations were made for uncertainties due to the difficultly in measuring these on the simple hardware used. The theoretical region of stability produced by the verification algorithm is shown in figure 7.

The performance of the control law on the real physical system was also measured. This allows for the comparison of the predicted region of stability with the region of stability measured on the actual physical system. However, due to hardware limitations, only a limited portion of the phase space can be explored. In particular, the stability region of the actual system is measured by introducing short disturbances in the form of an increased or decreased input current to the system. This causes the ball to achieve a positive velocity and drop below the desired equilibrium point (reduced input current), or to achieve a negative velocity and rise above the equilibrium point. The region of stability can be measured by observing the system behavior after the disturbance.

Unfortunately, this method of collecting experimental data on the actual system has the drawback that certain regions of the phase space cannot be explored. In particular, regions above the equilibrium point with positive velocity, and regions below the equilibrium point with negative velocity cannot be reached. Another difficulty that occurs is that when an increased input current is applied, the steel ball will often bounce off the solenoid into a region of phase space near the equilibrium point. Thus, for these experiments, only a decreased current type of disturbance is used. Finally, this method enables only coarse grain control of the regions of phase space explored, so often the data obtained will contain "holes" for areas where insufficient samples were obtained.

The measurable region of stability for the actual system is shown in figure 8. These results were obtained by applying a variety of disturbances, and recording the trajectories of the system when it recovered correctly from the disturbance. Outliers were removed by filtering out data points with few nearby samples.

The measurable region obtained is quite a bit smaller than the predicted region of stability; however, the primary reason for this is the limited area of phase space the hardware enables one to explore, as discussed above. Notice that the top of the measured region corresponds closely with the boundary of the theoretical controllable region. This suggests that a larger disturbance would cause the system to enter an uncontrollable region, and fail to return to the equilibrium point. Thus, for the limited region of phase space explored, the theoretical and actual results are similar.

Related Work

The HyTech system (Alur, Henzinger, & Ho 1996) uses convex polyhedra to represent regions of a hybrid system and shares the concept of verification through exploration of the pre-images of the desired goal states. HyTech allows the expression of properties to be verified as explicit mathematical formulas. For many properties, representation as a simple formula is more flexible than the implicit representation of properties through

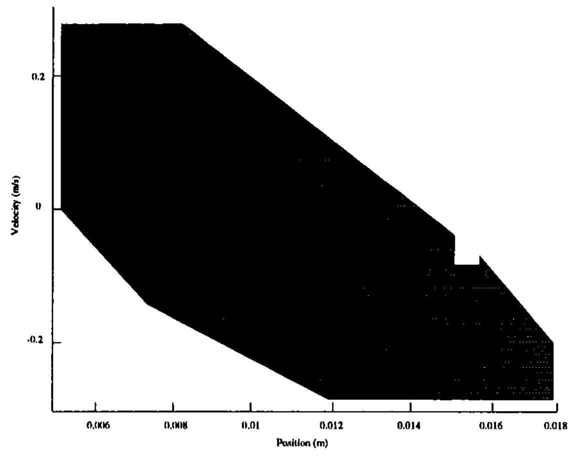

Figure 7: Region of the magnetic levitation system's phase space that is marked as verified by the verification algorithm.

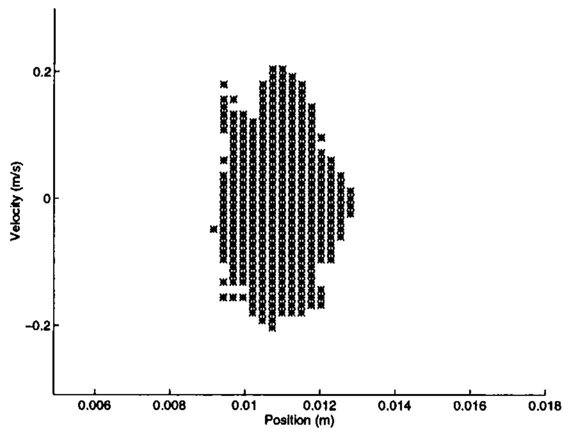

Figure 8: Measurable region of stability for the actual magnetic levitation system. An asterisk is plotted at the center of each cell that is marked as verified.

cell annotation. HyTech is designed to easily represent hybrid systems, which must be represented by an augmented phase space in our approach. However, the geometric approach used here allows for a more straightforward representation of real-world uncertainties. Furthermore, our algorithm is guaranteed to terminate, and allows for a systematic refinement of approximation of dynamics.

Recently, several researchers suggested verification techniques based on projecting phase-space regions. In (Greenstreet & Mitchell 1998), the authors described a method for computing projection polyhedra obtained from integrating initial regions. While an efficient and exact algorithm exists for two-dimensional linear systems with convex spaces, a general higher-dimensional polyhedra have to be reconstructed from a series of two-dimensional subspace projections. Because the paper did not provide implementation details and computational results for the higher-dimensional case, it is difficult to evaluate the effectiveness of the algorithm. The approaches in (Puri, Borkar, & Varaiya 1996; Dang & Maler 1998) share a grid-based representation of phase space with the algorithm developed in this paper. In comparison, our algorithm introduces a hierarchically refined bound of system dynamics to avoid unnecessary over-approximation.

Conclusions

An algorithm for verification of control laws using phase-space geometric modeling was presented. This algorithm is applicable to a wide range of control systems that are continuous, discrete, or hybrid, and can be used with a variety of forms of control laws. Once a bound for measurement, controller output, and modeling uncertainty is obtained, considerations for these types of uncertainties can easily be incorporated into the verification algorithm. The algorithm was applied to a nonlinear controller for a magnetic levitation system, and the resulting region of stability was compared to that of the actual physical system.

Future avenues of research include exploring the applicability of the algorithm to other real systems, and investigating optimal initial partitions for different control laws. Furthermore, more accurate and flexible techniques for measuring the region of stability of the physical maglev system need to be developed so that the results of the verification algorithm can more readily be compared to those of the actual system.

Acknowledgment

This work is conducted at Ohio State University, supported in part by ONR YI grant N00014-97-1-0599, NSF NYI grant CCR-9457802, and a grant from the Xerox Foundation. The authors thank Chris Bailey-Kellogg and Ivan Ordonez for valuable discussions, and especially Andy Berlin for encouraging FZ to work on control performance prediction problems.

References

Alur, R.; Courcoubetis, C.; Henzinger, T.; and Ho, P. 1993. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems I, Springer-Verlag Lecture Notes in Computer Science* 736.

Alur, R.; Henzinger, T.; and Ho, P. 1996. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering* 22.

Bradley, E., and Zhao, F. 1993. Phase-space control system design. *IEEE Control Systems* 13(2):39–47.

Bradley, E. 1995. Autonomous exploration and control of chaotic systems. *Cybernetics and Systems* 26.

Branicky, M. 1995. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* Ph.D. Dissertation, Massachusetts Institute of Technology.

Dang, T., and Maler, O. 1998. Reachability analysis via face lifting. *Hybrid Systems: computation and control, Springer-Verlag Lecture Notes in Computer Science* 1386.

Greenstreet, M., and Mitchell, I. 1998. Integrating projections. *Hybrid Systems: computation and control, Springer-Verlag Lecture Notes in Computer Science* 1386.

Hsu, C. S. 1987. *Cell-to-Cell Mapping.* Springer-Verlag, New York.

Kuipers, B., and Astrom, K. 1994. The composition and validation of heterogeneous control laws. *Automatica* 30(2):233–249.

Loh, S. 1997. An experimental environment for designing and evaluating nonlinear phase space based control laws with application to magnetic levitation. Master's thesis, The Ohio State University.

Nishida, T., and et al. 1991. Automated phase portrait analysis by integrating qualitative and quantitative analysis. In *Proceedings of AAAI.*

Puri, A.; Borkar, V.; and Varaiya, P. 1996. ϵ-approximation of differential inclusions. *Hybrid Systems III, Springer-Verlag Lecture Notes in Computer Science* 1066.

Sacks, E. 1991. Automatic analysis of one-parameter planar ordinary differential equations by intelligent numerical simulation. *Artificial Intelligence* 51:27–56.

Tsai, W.; Yue, D.; and Yip, K. 1990. Resonantly excited regular and chaotic motions in a rectangular wave tank. *J. of Fluid Mechanics* 216.

Yip, K. M. 1991. *KAM: A system for intelligently guiding numerical experimentation by computer.* MIT Press.

Zhao, F.; Loh, S.; and May, J. 1997. Phase-space nonlinear control toolbox: The maglev experience. *Hybrid Systems V, Springer-Verlag Lecture Notes in Computer Science.*

Zhao, F. 1995. Intelligent simulation in designing complex dynamical control systems. In Tzafestas, and Verbruggen., eds., *Artificial intelligence in industrial decision making, control, and automation.* Kluwer Academic Publishers.