# Intelligent Computing About Complex Dynamical Systems*

**Feng Zhao**
Laboratory for Artificial Intelligence Research and
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210 U.S.A.
E-mail: fz@cis.ohio-state.edu

## Abstract

We develop computational mechanisms for intelligently simulating nonlinear control systems. These mechanisms enhance numerical simulations with deep domain knowledge of dynamical systems theory and control theory, a qualitative phase-space representation of dynamical systems, symbolic and geometric manipulation capabilities, and a high-level interface. Programs equipped with these capabilities are able to autonomously simulate a dynamical system, analyze the simulation results, and utilize the analysis to perform design tasks. We demonstrate the mechanisms with an implemented computational environment called the Control Engineer's Workbench.

**Keywords.** Qualitative reasoning, scientific computing, numeric/symbolic processing, control system design.

## Introduction

*The purpose of computing is insight, not numbers.*
— R. W. Hamming

Computationally simulating complex physical systems in engineering design has become a common practice. Yet most of today's simulations rely entirely on extensive numerical computations and laborious human analysis. Human engineers have to shoulder the burden of translating physics and constraints into models, preparing numerical simulations, interpreting consequences of the experiments, and performing design tasks. Moreover, nonlinear systems can exhibit extremely complicated behaviors that defy human analysis and pure numerical simulations. The complexities of these systems are largely due to nonlinearities, high dimensionality, and uncertainties of the systems and environments the systems operate in.

The difficulties in the traditional engineering simulation arise from the lack of (1) parsimonious representations capturing the essence of physical systems and amenable to efficient computations, (2) efficient modeling algorithms for constructing the representations, and (3) effective reasoning methods that can use the representations to compute and synthesize useful properties for the systems. The lack of computable representations for physical dynamical systems hinders the exploitation of the special properties of the systems and the attainment of the maximum performance for the design. The simulation and design of the dynamical systems are limited by the available computational power and the complexities of the systems.

While the traditional numerical computing has successfully attacked many practical problems, we can greatly enhance its effectiveness and significantly expand the scope of what can be done with this style of engineering computing by integrating the numerical computation with advanced artificial intelligence technology and symbolic computing methods. For example, programs equipped with AI reasoning techniques and deep domain knowledge have already helped engineers solve an open problem in hydrodynamics [Yip, 1991], given new insights into behaviors of a heart model in cardiology [Sacks & Widman, 1993], and designed a high-performance nonlinear controller in maglev engineering [Zhao & Thornton, 1992]. Abelson *et al.* described a collection of computer programs that analyze dynamical systems at the level of expert dynamicists [Abelson, 1989]. Other related work is discussed in [Nishida et al., 1991; Bradley, 1992; Kant et al., 1992; Amador, Finkelstein & Weld, 1993].

## Task Domain: Control System Simulation and Design

We study the analysis and design of nonlinear control systems. A real-world control system is a complex closed-loop system with extremely rich dynamics. Computation and reasoning are pervasive in the design and operation of the controller. Sensors collect a large

*This research was supported in part by the National Science Foundation grants CCR-9308639 and MIP-9001651, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-89-J-3202. An early version of this paper appeared in *Mathematics and Computers in Simulation.*

amount of quantitative information. State and parameter estimators infer hidden information about the system from the sensed data. The system is modeled with a representation appropriate for further analysis and design based on available information. The model is then analyzed to extract behaviors that are considered significant for the control objective. To meet the control objective, a control law is synthesized to change the natural dynamics of the system.

The domain of automatic control brings together issues of sensing, estimation, control synthesis, and control execution. The study of their common themes—computation and reasoning—serves as a framework for coherently addressing these issues and makes it possible to employ advanced computational techniques to drastically improve modern control design. We focus on the control synthesis that *maps* a model of a physical system together with some control objective *to* a synthesized control law:

*Control Design*:
`model + control objective` $\longmapsto$ `control law.`

A control engineer goes through the following design steps to synthesize a controller for a given dynamical system:

1. *Analysis:* analyze the model of the dynamical system. The physics and the constraints of the system are often modeled with a quantitative mathematical model, typically a set of differential equations. This step examines the model and analyzes the behaviors of the system.
2. *Design:* design a controller for the system. Based on the analysis, this step arrives at a control design according to the prespecified control constraints.
3. *Verification:* verify the control design. This step ensures that the design meets the control specification.

The steps 1, 2, and 3 of the above design procedure are often iterated before a reasonable control law is synthesized. Except for very few cases in which analytic-form solutions are available, computer simulations are the main tools for analyzing nonlinear systems and for designing and verifying the controllers.

Existing control simulation softwares are inadequate for automatically designing highly complex nonlinear systems. Commercially available programs like MATLAB and SIMULAB [MathWorks, 1989] rely on numerical simulations. These programs are, at their very best, semi-automatic and serve as interactive design aids to human engineers. Although they are equipped with elaborate graphic interfaces, these programs provide only fragmented, limited capabilities such as integration and root finding for performing the simulation task; human users need to prepare the simulation and to interpret the result. The specialized control toolboxes embedded in these programs are "shallow" expert systems; they lack deep domain knowledge and do not have mechanisms for computationally representing and manipulating a control design.

## Smart Simulation of Dynamical Systems

We address the difficulties of traditional numerical computing by developing computer representation and simulation technologies necessary for enabling programs to autonomously perform and interpret numerical simulations of control systems. The design of complex control systems requires powerful computational mechanisms to represent, reason about, and manipulate the dynamics of nonlinear systems. We demonstrate that difficult control synthesis tasks can be automated, using a computational workbench that actively exploits knowledge of nonlinear dynamics and phase space. More specifically, we develop a computer representation for dynamical systems in terms of qualitative, geometric phase-space features and equivalence classes of behaviors. We employ hybrid computation integrating symbolic and numerical methods with AI reasoning and representation techniques and exploiting sophisticated mathematical domain knowledge. We provide a high-level interface for communicating the result of analysis in qualitative terms and for visualizing the phase-space dynamics.

We have constructed a computational environment, the Control Engineer's Workbench, integrating a suite of programs that automatically analyze and design high-performance, global controllers for a large class of nonlinear systems using the qualitative phase-space representation [Zhao, 1992]. Given a model of a physical system and a control objective, the Control Engineer's Workbench analyzes the system and designs a control law achieving the control objective. A user typically interacts with the Workbench in the following way.

The user first tells the Workbench about the system: he inputs a system model in terms of an ordinary differential equation, parameter values, and bounds on state variables for analysis in the form of a phase-space region. The user also tells the Workbench about the requirements on the control design: he specifies the desired state for the system to settle in, the initial states of the system, the allowable control parameter values, and the constraints on the control responses.

The user then asks the Workbench to analyze the system within the parameter ranges of the model. The Workbench visualizes the totality of the behaviors of the system over the parameter ranges; it represents the qualitative aspects of the system in a data structure and reports to the user a high-level, symbolic summary of the system behaviors and, if necessary, a graphic visualization of the phase-space qualitative features.

Next, the user instructs the Workbench to synthesize a control law for the system, subject to the specified design requirements. The Workbench searches for the global control paths that connect the initial

equation_of_motion:

$$\begin{cases} dx_1/dt = x_2 \\ dx_2/dt = -p_1 x_1 - p_2 x_1^3 - p_3 x_2 + u \end{cases}$$

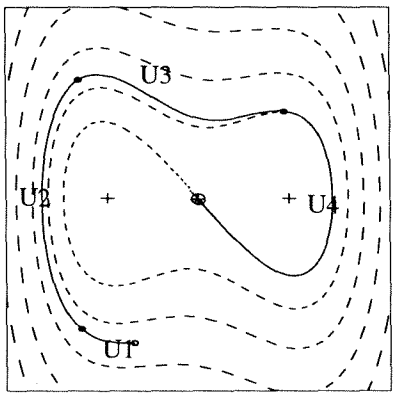
| | |
|---|---|
| state_variable: | $x_1$ |
| state_variable: | $x_2$ |
| parameter: | $p_1 =$ -2.0 |
| parameter: | $p_2 =$ 1.0 |
| parameter: | $p_3 =$ 0.2 |
| bounding_box: | $x_1 \in$ [-3.0,3.0], |
| | $x_2 \in$ [-4.0,4.0] |
| goal_state: | (0.0, 0.0) |
| initial_state: | (-1.0, -3.0) |
| range_of_control: | $u \in$ [-0.2,0.2] |

Figure 1: Workbench Input: the model for a buckling steel column and the control objective of stabilizing the buckling motion.

states of the system and the desired goal state, using the qualitative description about the system. More specifically, the search is conducted in a collection of discrete entities representing trajectory flows in phase space. After the global control paths are established, the Workbench determines the controllable region of the system and the switching surfaces where control parameters should change values. A synthesized control reference trajectory consists of a sequence of trajectory segments, each of which is under a constant control.

The following example illustrates how the Workbench autonomously analyzes the buckling motion of a steel column under compression and synthesizes a control law to stabilize the motion. Figure 1 shows the input to the Workbench—the model and control objective—and Figure 2 reports the synthesized control reference trajectory superimposed on the phase space of the column. The global portion of the reference trajectory shown in Figure 2 consists of four segments, each of which starts at a switching state marked as a small circle; the reference trajectory connects the initial state with the goal state at the origin of the phase space. The control parameter value is held constant for each segment denoted by $U1$, $U2$, $U3$, and $U4$, respectively.

The analysis of the nonlinear dynamics of the buckling column accounts for a large percentage of the computation in arriving at the desired control design. The equation of motion for the column is described as a nonlinear ordinary differential equation. An exhaustive numerical simulation can be very expensive. The Workbench employs a geometric analysis of dynamical systems that uses a phase-space representation to capture the qualitative features of the system, first developed by Poincaré at the beginning of the century.




```
;; The Synthesized Control Law specifying the time
;; instance, switching state, and corresponding
;; control value for each switching:
((time 0.)
 (switching-state #(-1 -3))
 (control .2))
((time .284)
 (switching-state #(-1.82 -2.71))
 (control 0.))
((time 1.06)
 (switching-state #(-1.86 2.49))
 (control -.2))
((time 2.71)
 (switching-state #(1.35 1.82))
 (control 0.))
((time 6.76)
 (switching-state #(-.0023 -.0692))
 (control *local-control*))
```

Figure 2: Workbench Output: the control reference trajectory and the control law for stabilizing the column. In the plot, the reference trajectory is drawn in solid lines, and the phase-space stability boundaries of the uncontrolled column in dashed lines.

## Representation, Simulation, and Interface

The Workbench has demonstrated the following capabilities in designing a control law for the buckling column:

- Deciding what behaviors are significant. The Workbench looks for qualitative features like equilibrium points, stability regions, and trajectory flows.
- Describing the behaviors qualitatively in computational terms. The Workbench models the stability regions and trajectory flows geometrically.
- Reasoning about the geometry of a phase space.
- Performing the control design autonomously.

These capabilities are supported in the Workbench by (1) qualitative representation consisting of dimension-independent geometric constructs, (2) hierarchical extraction of the behaviors, (3) modeling and

manipulation mechanisms for trajectory flows, and (4) algorithms implementing the geometric, combinatorial, and numerical computations.

Our intelligent simulation environment has three generic layers: (1) computer representations of the physical world, (2) simulation technology that makes feasible the computation about the physical world, and (3) high-level interfaces for communications between human users and the computer. For our intended task domain of control design and analysis, we present the following structure for our environment:

1. Computer representation:

   We develop a qualitative representation describing a dynamical system in terms of its phase-space geometric features: the qualitative phase-space structure. This qualitative representation captures asymptotic and transient behaviors of a dynamical system and essential constraints of the system useful for the control synthesis task.

2. Simulation technology:

   Our simulation of a control system comprises numerical and symbolic computation about the system model and geometric modeling of phase space. The Workbench also embodies deep domain knowledge of dynamical systems theory and control theory that can guide quantitative simulation and reduce the amount of computation necessary for analyzing the system. We choose Scheme, a dialect of LISP, as the implementation language for the Workbench [Hanson, 1991] and use extensively the Scheme mathematical library supporting generic numerical and symbolic manipulations. Scheme supports procedural abstraction that facilitates the extraction of common patterns in numerical computation and the composition of numerical procedures.

3. High-level interface:

   The Workbench presents a high-level, qualitative description of the result of its analysis and design to human designers; this level of interaction is more intuitive and direct than that of pure quantitative presentation. Other programs in the Workbench can efficiently access and manipulate the result. The internal data structure for the analysis ensures that the result is sensible to human engineers and manipulable by other programs.

The rest of the paper describes our representation for control systems and an implemented simulation environment.

## Representing and Manipulating Constraints of a Control Design

The complexity of a nonlinear system necessitates the need for a design vocabulary capable of describing implicit constraints of a control design and providing means to manipulate and reason about these constraints and to build abstractions. We present a computational mechanism that allows one to represent and manipulate constraints of a control design in terms of phase-space geometry and topology of a dynamical system. A control design will be specified in terms of the composition of geometric objects in phase space.

### A qualitative representation

We have interpreted a control design as a *mapping* from the model of a physical system to be controlled and the control objective *to* the control law, under the influence of which the physical system will behave in the desired way. The synthesized control law alters natural dynamics of the system through the selection and composition of the natural behaviors.

We describe a qualitative representation for complex behaviors of dynamical systems in phase space and a design vocabulary for computationally expressing and manipulating these behaviors [Zhao, 1991; Zhao, 1993]. A phase space of a dynamical system is an $n$-dimensional geometric space, each dimension of which represents a state variable of the system. We are interested in representing the qualitative behaviors of dynamical systems for control analysis and design. One useful qualitative representation of the phase space of a dynamical system is in terms of equilibrium points and limit cycles, stability regions, trajectory flows exhibiting the same qualitative features, and the spatial arrangement of these geometric objects in phase space. This qualitative representation captures the gross aspects of dynamics in a relational graph of phase-space structure and a set of discrete objects called *flow pipes*—the equivalence classes of behaviors. The design vocabulary describes a control design task in terms of well-defined geometric, combinatorial operations on the flow pipes. This vocabulary formalizes aspects of implicit expert reasoning of control engineers in solving control design problems with the phase-plane method. The representation and the vocabulary are developed independently of the orders of systems, *i.e.*, the dimensionality of phase spaces.

### Designing control by manipulating equivalence classes of trajectories

The flow pipes group infinite numbers of distinct behaviors into a manageable discrete set that becomes the basis for establishing control reference trajectories; each flow pipe models an equivalence class of trajectories exhibiting similar qualitative features.

The geometric modeling of a phase space with flow pipes makes the phase-space control planning and navigation possible. Given a discrete set of possible control actions, the search for a control path from an initial state to a destination is a reachability problem, *i.e.*, the problem of finding a sequence of connected path segments each of which is under a single control action, as schematically illustrated in Figure 3. This point-to-point planning can be naturally described within the

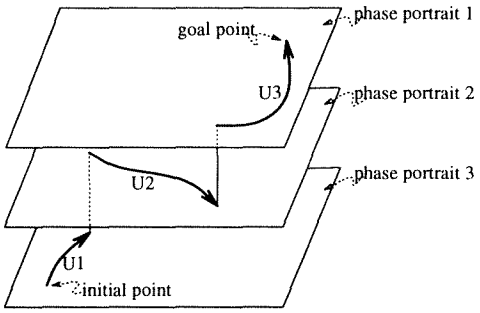



Figure 3: Search for a control path from an initial point to a goal point in a stack of phase spaces.

flow-pipe representation of phase space: a system under control jumps from one flow pipe to another upon the switching of control, eventually arriving at the destination.

To make this approach computationally feasible, the phase spaces of the dynamical system indexed by different control actions are first parsed into a discrete set of trajectory flow pipes. These flow pipes are then aggregated to intersect each other and are pasted together to form a graph, the *flow-pipe graph*. The flow-pipe graph is a directed graph where nodes are intersections of flow pipes and edges are segments of flow pipes. The initial state and the goal state are nodes in the graph. Each edge of the graph is weighed according to traveling time, smoothness, *etc.* With this graph representation, the search for optimal paths is formulated as a search for shortest paths in the directed graph.

## An Environment: the Control Engineer's Workbench

The Control Engineer's Workbench is an implemented system that analyzes and designs control systems and interacts with users qualitatively. The Workbench serves as an intelligent assistant to control engineers. The components of the Workbench are shown in Figure 4:

- MAPS program for simulation and interpretation
- Phase Space Navigator for control synthesis
- A graphic program for visualizing the design
- A user interface for communication with the system.

MAPS, standing for Modeler and Analyzer for Phase Spaces, is an autonomous phase-space analysis and modeling program that extracts and represents qualitative phase-space structures of nonlinear dynamical systems [Zhao, 1991]. MAPS generates a high-level description of the behaviors of a dynamical system, sensible to humans and manipulable by other programs. Phase Space Navigator visualizes the phase-space structure of a given system computed by MAPS,

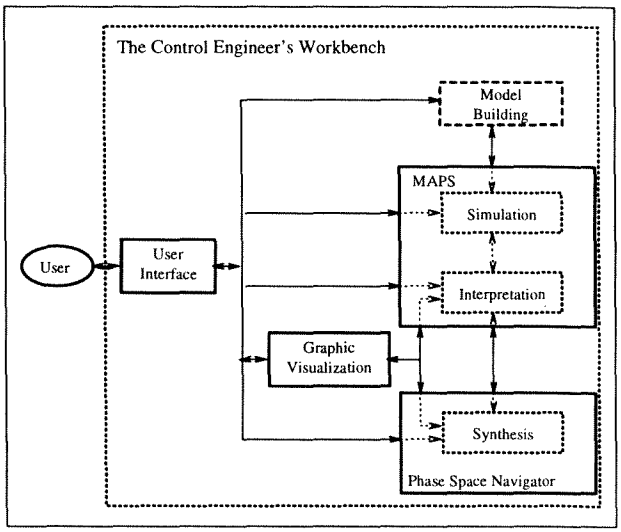



Figure 4: The Control Engineer's Workbench.

plans global control reference trajectories, and navigates the system along the planned trajectories [Zhao, 1992]. The component for model building in the figure has not yet been implemented.

The programs in the Workbench implement various algorithms: symbolic differentiation, numerical algorithms on differential equations, modeling of geometric structures, clustering of equivalence classes, graph algorithms, *etc.* The inference mechanism of the Workbench uses these programs to construct a qualitative phase-space structure for representing a system, to check for the consistency of the structure, and to reason about and manipulate the representation through a graph of flow pipes.

The flow of computation within the Workbench is illustrated in Figure 5. Given a model of a system, a bounded phase-space region of interest, allowable parameter values, and control objectives and constraints, the Workbench performs stability and trajectory flow analysis for the system in phase space and interprets the result in a phase-space graph. The Workbench then explores the control space to synthesize a desired control law subject to the design constraints. It reports the control law specifying reference trajectories and performance properties. The control design for steering towards an equilibrium is performed in this way: for a point-to-point control, the output is a reference trajectory, whose control law consists of a sequence of tuples of time, switching state and the corresponding control value; if an initial operating region is given, the output is a controllable region geometrically represented as a polyhedral structure.

The current implementation of the Workbench takes as input the model for a dynamical system in terms of an ordinary differential equation. Incorporating model formulation capability that constructs models

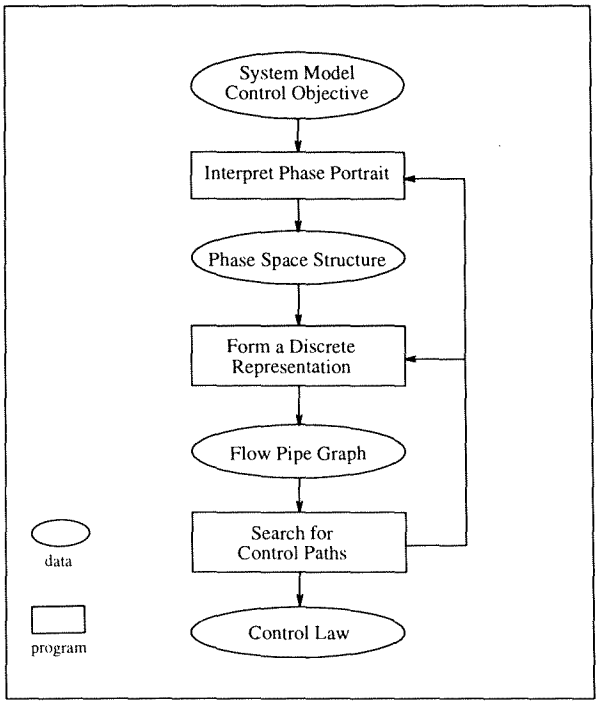



Figure 5: The flow of computation in the Workbench.

from physical principles or measurements can broaden the scope of control systems the Workbench can design. Although the analysis and design algorithm of the Workbench applies to dynamical systems of any order, the computational complexity of high-dimensional systems remains as a future research topic.

## Conclusion

We have developed the Control Engineer's Workbench comprising programs MAPS and Phase Space Navigator that automate a significant portion of a control engineer's simulation and design task. The Workbench employs computational means to represent and manipulate dynamics of control systems, using a qualitative representation for encoding dynamics and a flow-pipe based phase-space method for designing nonlinear controllers; it combines numerical simulation with symbolic techniques and AI reasoning; it provides an interface for visualizing the result of control design and analysis. The Workbench has been applied to the design of a nonlinear controller for a magnetic levitation vehicle. We plan to extend the capabilities of the Workbench to support interactive editing of phase-space geometric representation of dynamical systems for the purpose of experimenting and testing new ideas in control design.

The Control Engineer's Workbench complements and enhances human design activities. By providing manipulation and visualization mechanisms for the design, the Workbench relieves engineers from routine, tedious low-level tasks of simulation and interpretation, allows the engineers to focus on higher-level design issues, and enlarges the design space the engineers can explore.

## References

H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G.J. Sussman, J. Wisdom, and K. Yip, "Intelligence in Scientific Computing." *CACM*, 32(5), May 1989.

F. G. Amador, A. Finkelstein, and D. S. Weld, "Real-Time Self-Explanatory Simulation." *Proc. of QR-93*, 1993.

E. Bradley, "Taming Chaotic Circuits." *Technical Report AI-TR-1388*, MIT Artificial Intelligence Lab, 1992.

J. Guckenheimer and S. Kim, "Kaos: Dynamical System Toolkit with Interactive Graphic Interface." *Technical Report (Draft)*, Cornell University, 1990.

C. Hanson, "MIT Scheme Reference Manual." *AI-TR-1281*, MIT Artificial Intelligence Lab, 1991.

E. Kant et al. (eds.) *Intelligent Scientific Computation.* Working notes of AAAI Fall Symposium Series, 1992.

The MathWorks, SIMULAB, *a program for simulating dynamic systems, a user's guide.* The MathWorks, Inc., 1990 and MATLAB *User's Guide.* The MathWorks, Inc., 1989.

T. Nishida, K. Mizutani, A. Kubota, and S. Doshita, "Automated Phase Portrait Analysis by Integrating Qualitative and Quantitative Analysis." *Proc. of AAAI-91*, July 1991.

M. F. Russo and R. L. Peskin, "Automatically Identifying the Asymptotic Behaviors of Nonlinear Singularly Perturbed Boundary Value Problems." *J. Automated Reasoning*, 8(3), June 1992.

E. Sacks, "Automatic Analysis of One-Parameter Planar Ordinary Differential Equations by Intelligent Numeric Simulation." *Artificial Intelligence*, 48(1):27-56, 1991.

E. Sacks and L. Widman, "Nonlinear Heart Model Predicts the Range of Heart Rates for 2:1 Swinging in Pericardial Effusion." *American Journal of Physiology*, 1993.

K. M. Yip, *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer.* MIT Press, 1991.

F. Zhao, "Extracting and Representing Qualitative Behaviors of Complex Systems in Phase Spaces." *Proc. 12th Int'l Joint Conf. on Artificial Intelligence*, Morgan Kaufmann, 1991. To appear in *Artificial Intelligence* journal.

F. Zhao, "Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase Space

Knowledge." *Technical Report AI-TR-1385*, MIT Artificial Intelligence Lab, 1992.

F. Zhao and R. Thornton, "Automatic Design of a Maglev Controller in State Space." *Proc. of the 31st IEEE Conf. on Decision and Control*, Tucson, Arizona, December 1992.

F. Zhao, "Computational Dynamics: Modeling and Visualizing Trajectory Flows in Phase Space." *Annals of Mathematics and Artificial Intelligence*, 8(3-4), 1993.