Computation and Reasoning in Automating Control Design*

Feng Zhao†
MIT Artificial Intelligence Laboratory
545 Technology Square, Room 438
Cambridge, MA 02139
fz@ai.mit.edu

Abstract

I describe how intelligent scientific computing techniques are used in automating the difficult task of analyzing and synthesizing complex control systems. My research has developed a novel design methodology for the synthesis of automatic controllers, together with a suite of computational tools that automatically analyze and design controllers for high-performance, global control of nonlinear systems. These programs combine powerful numerical and symbolic computations with artificial intelligence representation and reasoning mechanisms. They embody deep knowledge of nonlinear dynamics and control theory and actively exploit special properties of the domains to attain otherwise impossible performance. They formalize implicit working knowledge of professional control engineers in computational terms and use the formalized knowledge to autonomously explore the design space. The two major programs in the suite of tools—MAPS and Phase Space Navigator—work together to visualize and model the phase-space geometry and topology of a given system, use a novel technique of "flow pipes" to plan global reference trajectories in phase space, and navigate the system along the planned trajectories. The flow-pipe technique parses a continuous phase space of a dynamical system, consisting of an infinite number of individual trajectories, into a discrete collection of equivalence classes that a computer can efficiently reason about. The programs have been demonstrated on a real engineering problem—the automatic design of a high-quality controller for a magnetic levitation system.

*This working notes paper is prepared for the AAAI 1992 Fall Symposium on Intelligent Scientific Computation.

†Currently at: Dept. of Computer and Information Science, Ohio State Univ., 2036 Neil Ave., Columbus, OH 43210. Tel. 614-292-1553.

Introduction

My research is concerned with the mechanization of control design and analysis tasks by autonomous computer programs. We have developed computational means to represent and manipulate dynamics of control systems. Specifically, this research has developed a flow-pipe based phase-space method for designing nonlinear controllers and a qualitative representation for encoding dynamics. It has formulated the task of control design and analysis as a computational one in which computation and reasoning about dynamics are essential. It has constructed a Control Engineer's Workbench comprising programs MAPS[1] and Phase Space Navigator that automates a significant portion of a control engineer's design task. The Workbench has been applied to the design of a nonlinear controller for a magnetic levitation vehicle.

Design of complex control systems is difficult to mechanize. The difficulties arise from the lack of design methodologies that actively exploit and efficiently represent the special nature of nonlinear dynamics, and from the lack of high-level computational tools that effectively use the representation to guide and perform the control design. Complex nonlinear systems rarely admit closed-form solutions. Consequently, many engineering applications rely on extensive numerical experiments. A numerical simulation typically generates an immense amount of quantitative information about a complex system. To interpret the numerical result and to use the information for engineering designs, it is essential to develop qualitative methods that automatically analyze the system, extract the qualitative features, and represent them in a high-level description sensible to human beings and manipulable by other programs. This qualitative representation needs to be parsimonious and yet capture the essential properties of the dynamical system under study. The task of control design requires that the representation facilitate manipulation for synthesizing new dynamical behaviors.

[1]MAPS stands for Modeler and Analyzer for Phase Spaces.

The modern geometric theory of dynamical systems pioneered by Poincaré provides a qualitative way to describe the rich dynamical behaviors of nonlinear systems. Abelson *et al.* described a collection of computer programs that analyze dynamical systems in phase space at the level of expert dynamicists [Abelson, 1989]. The task of control design, however, imposes stronger requirements on the form and use of the representation of the dynamics: the phase-space geometry and topology of a dynamical system should be represented in a way that allows for efficient computational manipulations for the purpose of control design. Nonlinear systems can have extremely convoluted phase-space geometries; the complexity becomes much worse as the dimensionality increases. Humans can comfortably picture and manipulate two and three-dimensional objects with the aid of graphic, geometric modeling techniques. For higher-dimensional systems, however, few visualization and manipulation tools exist. Automatic modeling, analysis, and design tools are necessary to identify, extract, and reason about the spatial properties of phase space.

Research Summary

The work reported in [Zhao, 1991a; Zhao, 1991b; Zhao, 1992a] demonstrates that the difficult control design and analysis tasks can be automated, using a suite of computer programs that actively exploit knowledge of nonlinear dynamics and phase space. These programs combine numerical and symbolic computations with spatial-reasoning techniques. The research is summarized as follows:

- This work has developed a phase-space **qualitative representation** for complex behaviors of dynamical systems and a **design language** for computationally expressing and manipulating these behaviors. The 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 language describes a control design task in terms of well-defined geometric, combinatorial operations on the flow pipes. This language helps formalize aspects of implicit expert reasoning of control engineers in solving control design problems. The representation and the language are developed independently of the orders of systems, *i.e.*, the dimensionality of phase spaces.

  The qualitative representation and the design language enable us to develop a phase-space design methodology for the synthesis of control systems. The methodology designs a prespecified control law—control reference trajectories—for a system by synthesizing the desired shape for phase-space geometry dictating trajectory flows. It uses the flow pipes to group infinite numbers of distinct behaviors into a manageable discrete set that becomes the basis for establishing reference trajectories, and navigates the system along the planned reference trajectories. The phase-space design approach requires powerful computational tools that are able to identify, extract, represent, and manipulate qualitative features of phase space.

- This work has 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. These programs combine powerful techniques from numerical and symbolic computations with novel representation and reasoning mechanisms of artificial intelligence. The two major components in the Workbench—MAPS and Phase Space Navigator—work together to visualize and model the phase-space geometry and topology of a given system. They reason about and manipulate the phase-space geometry and topology and search for optimal control paths connecting initial state and the desired state for the system. The Workbench represents the result of design and analysis in a symbolic form manipulable by other programs, and produces a high-level summary meaningful to professional engineers. It also presents the result in a graphical form.

  The Workbench embodies domain knowledge from control engineering and dynamical systems theory. It understands concepts like asymptotic and transient behaviors, stability regions, reachable sets, convergence, overshooting, etc. The Workbench organizes the modules of analysis, design, and graphics presentation around the control task and encourages incremental changes to the Workbench, for example, incorporating programs tailored to particular applications and encoding knowledge of specific domains. Suppose we want to use the Workbench to design electric power control systems. In addition to their differential equation models, the power systems have special structural properties and model formulations that can be exploited to reduce design complexity and improve design quality. Special program fragments exploiting these properties can be integrated into the Workbench. The integration is made possible and easier by the underlying Scheme implementation as a substrate. The Scheme programming language facilitates composition and abstraction of procedures.

- This work has demonstrated the Workbench in an **application** of great practical interest: the Workbench helped design a high-quality controller for a magnetic levitation system—the German Transrapid system. The controller synthesized by the Workbench outperforms the one manually designed for the same system by professional control engineers: our controller stabilizes the maglev vehicle with much larger initial displacements than those allowed in the manual design using classical linear feedback technique [Zhao and Thornton, 1992b].

Reasoning about Control Design

How does a control expert reason about a control design task? A professional control engineer uses a body of specialized knowledge to carry out the design. The engineer develops insight through an analysis of the physical system, uses the insight to explore the design space constrained by control requirements, and makes engineering judgment about design choices and trade-offs. A particularly intuitive design method is the phase-plane method for analyzing and designing a second-order nonlinear control system in a phase plane.

The engineer goes through the following steps with the phase-plane method:

- simulate the system extensively with different initial conditions
- plot the behaviors in trajectories on a piece of paper
- interpret the result with visual inspection
- design a control law to obtain desired behavior.

The phase-plane method is a useful tool for analyzing qualitative responses of a control system. It is, however, manual, prohibitively expensive, and confined to two-dimensional planes. Although a diagrammatical sketch of a phase plane illustrates just the qualitative aspects of the system, to obtain the phase-plane sketch means lots of human effort in preparing numerical simulations, collecting the numbers, sketching the results on papers, analyzing the trajectory plot, and interpreting it in a qualitative picture. Worse, this method becomes useless in cases when the order of a system is greater than two and the nonlinearity results in convoluted phase-space geometry. The mechanization of the task with autonomous computer programs would alleviate many of these restrictions.

The Control Engineer's Workbench formulates in computational terms the informal working knowledge of professional control engineers in analyzing complex control systems, in particular, in the form of the phase-plane method.

A Real Scenario with the Workbench

The Control Engineers' Workbench automates a significant portion of the control engineer's analysis and design tasks. In the following scenario, the Workbench autonomously analyzes the buckling motion of a steel column under compression.

The Engineer types in the model for the elastic column buckling under axial compressive force and asks the Workbench to analyze the system for the given parameter values.

`name:`	`buckling_column`
`equation_of_motion:`	$\begin{cases} dx_1/dt = x_2 \\ dx_2/dt = -p_1x_1 - p_2x_1^3 - p_3x_2 + u \end{cases}$
`state_variable:`	x_1
`state_variable:`	x_2
`parameter:`	$p_1 =$ -2.0
`parameter:`	$p_2 =$ 1.0
`parameter:`	$p_3 =$ 0.2
`parameter:`	$u =$ 0.0
`bounding_box:`	$x_1 \in$ [-3.0,3.0], $x_2 \in$ [-4.0,4.0]

The Workbench models the trajectory flows of the system and reports the following summary:

```
flow-pipes:
 1. flow-pipe from *infinity* to (1.41 0.):
   flow-pipe-boundary:
     trajectory 1:(from *infinity* to (0. 0.))
     trajectory 2:(from *infinity* to (0. 0.))
 2. flow-pipe from *infinity* to (-1.41 0.):
   flow-pipe-boundary:
     trajectory 1:(from *infinity* to (0. 0.))
     trajectory 2:(from *infinity* to (0. 0.))
```

The Workbench also displays the phase portrait of the system showing flow pipe 2, consisting of all the trajectories that end at the left-hand attractor (see Figure 1).

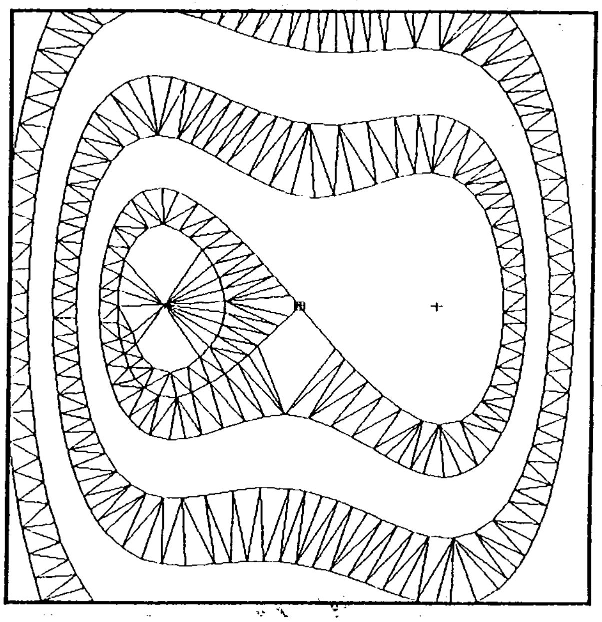

Figure 1: The phase portrait of the buckling column, automatically generated by the Control Engineer's Workbench.

Like the phase-plane analysis discussed earlier, the Workbench is able to reason about dynamics in terms of phase-space geometries. It partitions a phase space into discrete regions. It generates a high-level picture of the phase portrait in Figure 1. The picture contains essentially the same kind of information as one a professional would produce.

But unlike the phase-plane method, the Workbench provides computational means for modeling the dynamics. It decomposes the phase space into subregions that can even be globally nonlinear. The Workbench internally represents the critical points and geometries of the regions in a data structure that allows other programs to manipulate, visualize, and communicate with human users. Because the geometry of a system's phase space is modeled with a simplicial structure, the representation and reasoning mechanisms for this structure are independent of the dimensionality of phase space. Reference [Zhao, 1992a] details a synthesis algorithm for designing control systems using the phase-space modeling.

Critical Issues

We have identified the following issues that are critical in mechanizing the control design:

- The mechanization of control design calls for a concise representation that captures essential features of a control system. The representation should be meaningful to professional control engineers and manipulable by other programs. The Workbench needs to present the result of analysis and design to human designers and to encourage the designers to interact with the design in a direct way. This communication requires a high-level, intuitive presentation of the result. Other programs in the Workbench need to efficiently access and manipulate the representation. Instead of encapsulating everything about the system, the representation should only contain information that is useful for the control synthesis task. We have chosen an equivalence-class based qualitative representation for this purpose.
- The mechanization needs modeling algorithms to efficiently construct the representation from simulations. The algorithms should identify and extract implicit dynamical properties from numerical explorations and summarize the result in a qualitative form. In the implementation, the Workbench internally uses a hierarchy of intermediate representations. The qualitative information about the system is extracted in a step-by-step fashion, from local descriptions to global ones.
- The mechanization needs a reasoning mechanism to efficiently manipulate the representation for synthesizing a control law. The program searches through the representation to find feasible control trajectories. The Workbench has used a graph mechanism that manipulates a discrete collection of flow pipes.
- The domain knowledge and techniques from symbolic, numerical, and geometric computing have been proven essential. In order to fully exploit the dynamics and to build programs to imitate human control designers, the Workbench uses whatever knowledge and techniques that are necessary: geometric theory of dynamical systems, control theory, and techniques from artificial intelligence, vision, computational geometry, numerical analysis, and graph search.
- The complexities of the control design task necessitate the need for automatic modeling and analysis tools. Autonomous programs like those in the Workbench have extended the capabilities of the "eyes" and "hands" of control engineers in seeing and manipulating objects. They have enlarged the design space engineers can explore. In certain cases, programs can even outperform human experts, as in the maglev controller design.

Conclusions

Novel computational representation and reasoning mechanisms can be developed in the context of automating challenging engineering tasks. The dynamics of nonlinear systems is difficult to describe and manipulate. The qualitative representation developed in this research provides a way to computationally describe the qualitative aspects of the dynamics. With this representation, the difficult control design is translated into a computational task: the flow-pipe based mechanism manipulates a system's natural dynamics and synthesizes the desired dynamics for the system. The Control Engineer's Workbench is a prototype of a new class of intelligent computational tools that combine numerical and symbolic computations with AI reasoning techniques and automatically model, analyze, and design complex physical systems.

References

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

Zhao, F. 1991a. Extracting and Representing Qualitative Behaviors of Complex Systems in Phase Spaces. In *Proc. IJCAI-91.*

Zhao, F. 1991b. Phase Space Navigator: Towards Automating Control Synthesis in Phase Spaces for Nonlinear Control Systems. In *Proc. 3rd IFAC Int'l Workshop on AI in Real Time Control*, Pergaman Press.

Zhao, F. 1992a. Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase Space Knowledge. PhD Thesis, Dept. of Elec. Eng. & Comp. Sci., MIT.

Zhao, F. & Thornton, R. 1992b. Automatic Design of a Maglev Controller in State Space. In *Proc. 31st IEEE Conf. on Decision and Control.* to appear.